Systems Engineering Technical Short Course for Waters Corporation

This course planned to be conducted at the Waters facilities as follows:

Session 1:	**January 27 – 29, 2015:**	**Overview and Concepts of Systems Engineering**
Session 2:	**February 16 – 17, 2015:**	**Stakeholder Needs**
Session 3:	**March 16 – 17, 2015**	**System Requirements**
Session 4:	**April 13 – 14, 2015**	**System Architecture and Design**
Session 5:	**May 13 – 14, 2015**	**System Integration and Test**
Session 6:	**June 10, 2015**	**Collaboration Training with WLI. (not part of WPI material)**
	June 11, 2015	**Capstone presentations**

Robert B. Wheeler, ESEP, PMP
Thomas F. Gannon, Ph.D.

This course is developed by Worcester Polytechnic Institute (WPI) for Waters Corporation. Course materials remain the property of WPI. WPI and the course developer (Robert Wheeler of Robert B Wheeler & Associates for sessions 1, 2, and 3) retain the right to use, reuse, and modify this content in future course deliveries.



Systems Engineering Technical Short Course

Session 1: Overview and Concepts of Systems Engineering

January 27 through 29, 2015

Robert B. Wheeler, ESEP, PMP

Systems Engineering Technical Short Course

(Session 1)

Program Kick-off

Waters Corporation Management



Systems Engineering Technical Short Course

(Session 1)

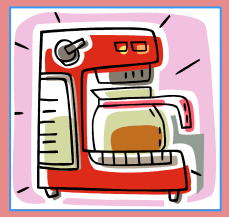
Program Overview

Robert B. Wheeler, Instructor

Welcome and Opening Remarks……

Logistics:

- **Facility**
- **Safety & emergencies**
- **Schedule & breaks**
- **Parking lot**
- **Other?**









Code of Conduct:

- On time
- Mutual respect
- Machines off & closed unless we are using them
- Participation!
- Other?

During this course you will: (1/2)

Pre-class: Be assigned a project suggested by Waters senior management

Sessions 1 through 3 (Bob Wheeler, instructor):

- **Develop and use a common vocabulary**
- **Apply systems thinking and holism to your projects**
- **Articulate the Problem to be Solved by your project**
- **Develop a Technical Risk Management Plan**
- **Elicit and express stakeholders expectations and requirements**
- **Develop User Concept of Operations (CONOPS), solution selection criteria, and acceptance criteria**
- **Conduct system functional and performance analyses**
- **Define and express system requirements**
- **Manage requirements through the system lifecycle**

During this course you will: (2/2)

Sessions 4 & 5 (Tom Gannon, instructor):

- **Develop creative alternative system architectures**
- **Conduct analyses of your architectural alternatives to meet stakeholder and system requirements**
- **Define a candidate architecture and design for your solution**
- **Develop strategies and plans for system integration, validation and verification testing**
- **Manage requirements through the system lifecycle (continued)**
- **Document the results of your work in appropriate artifacts**

Session 6:

- **Present a status report on your project to Waters senior management**

Schedule

Session	Dates	Topic
1.	January 27 – 29	Introduction – Overview and Concepts of SE (US)
2.	February 16 – 17	Stakeholder Needs*
3.	March 16 – 17	Systems Requirements*
4.	April 13 – 14	Systems Architecture and Design (UK)*
5.	May 13 – 14	Systems Integration and Test*
6.	June 10	Collaboration Training with WLI
	June 11	Capstone Presentations (US)

Notes:

- Denotes sessions held for some participants via video-teleconference.
- Class times and room locations are shown later

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

Slide not posted to WPI site

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

Slide not posted to WPI site

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

Slide not posted to WPI site

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

Slide not posted to WPI site

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

Slide not posted to WPI site

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

Slide not posted to WPI site

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

Slide not posted to WPI site

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

Slide not posted to WPI site

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

Slide not posted to WPI site

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

Slide not posted to WPI site

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

Slide not posted to WPI site

©2015 Waters Corporation

This slide intentionally left blank

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

Slide not posted to WPI site

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

Slide not posted to WPI site

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

Slide not posted to WPI site

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

Slide not posted to WPI site

©2015 Waters Corporation

Session 1 Agenda (Jan 27 – 29): Introduction – Overview and Concepts of Systems Engineering

- **Day 1**
  - **Waters management kick-off**
  - **Introduction and overview of the entire program**
    - **Project assignments, team formation, and project overviews**
  - **Presentation and discussion of Concepts of Systems Engineering**
  - **Project working sessions: applying SE concepts to their projects**
  - **Day 1 wrap-up: Takeaways and Q&A**

- **Day 2**
  - **Review of Day 1 working sessions**
  - **Presentation and discussion of Concepts of SE (concluded)**
  - **Project team working sessions (continued)**
  - **Day 2 wrap-up and Q&A**

- **Day 3**
  - **Team working session feedback to the class**
  - **Project team working sessions (concluded)**
  - **Check-in with class on application of SE Concepts to their projects**
  - **Session 1 wrap-up: Q&A and homework due next Session**

Approximate timing is shown on next slide

Session 1 Agenda (Jan 27 – 29): Timing

Day	Overview and Concepts of Systems Engineering	Hours*
1	Waters Management kick-off	1/2
	Introduction and overview of entire program (Lesson 1), including Introduction and overview of the projects (15 minutes each)	2
	Presentation and discussion of Concepts of SE (Lessons 2 – 9)	4
	Project assignments, team formations, and working sessions to apply SE concepts to the projects	1
	Day 1 wrap-up: Takeaways and Q&A	1/2
2	Review of day 1 working sessions	1
	Concepts of Systems Engineering (concluded) (Lesson 10 – 14)	4
	Project team working sessions (continued)	2
	Day 2 wrap up and Q&A	1
3	Team working session feedback to class (informal presentations)	1
	Project team working sessions (concluded)	5
	Check-in with class on application of SE concepts to their projects	1
	Session 1 wrap up: Q&A and homework due next session	1

* Times include breaks, and may be changed based on needs of the class

Sessions 2 through 5 Agenda

- **Day 1**
  - **Review project team assignments from previous Session**
  - **Presentation and discussion of Session topic**
  - **Project team working sessions on the application of the Session topics to their projects**
  - **Day 1 wrap-up and Q&A**

- **Day 2**
  - **Review Day 1 topic and project working sessions**
  - **Presentation and discussion of Session topic (concluded)**
  - **Project team working sessions (concluded)**
  - **Check-in with class on the application of the Session topic to their projects**
  - **Session wrap-up: Q&A and homework due next Session**

Timing will be similar to session 1, and will be announced at beginning of each session

Session 6 (June 11) Capstone Presentations

- **Each team will present a summary status report[1,2] of their project to Waters senior management. Your presentations should:**
  - **Summarize the highlights of all your project work throughout the short course**
    - **Describe your project, studies, analyses, architecture and design efforts, and work products that you performed and/or developed**
    - **Include a "behind the scenes" discussion of how you developed them**
  - **Discuss your lessons learned**
    - **These can be related to your team dynamics, if appropriate, the design process you used, things you learned about your project and about the systems engineering work that you performed**
  - **Discuss how well your team performed against your work plan**
  - **Discuss any recommendations for future project work at Waters**

1. Use of selected results from the homework assignments in the Capstone presentation is encouraged;
2. Class scheduled for 8AM to 1PM; Each group will have 40 minutes for the report, and 20 minutes Q&A

Room Assignments

Milford Rooms

Date	Time	Primary Location	Break-out Rooms			
27-Jan	8:00 - 4:00 pm	C4		N/A	N/A	N/A
28-Jan	8:00 - 4:00 pm	C4		N/A	N/A	N/A
29-Jan	8:00 - 4:00 pm	C4		N/A	N/A	N/A
16-Feb	7:00 - 4:00 pm	Atriums 2 & 3	1:00 - 4:00 pm	mfg conf	Atlantis	Global Ops
17-Feb	7:00 - 4:00 pm	C4	1:00 - 4:00 pm	mfg conf	Atlantis	Acquity
16-Mar	7:00 - 4:00 pm	Atriums 2 & 3	1:00 - 4:00 pm	mfg conf	Atlantis	Global Ops
17-Mar	7:00 - 4:00 pm	C4	1:00 - 4:00 pm	mfg conf	Atlantis	Acquity
13-Apr	7:00 - 4:00 pm	Atriums 2 & 3	1:00 - 4:00 pm	mfg conf	Atlantis	Global Ops
14-Apr	7:00 - 4:00 pm	C4	1:00 - 4:00 pm	mfg conf	Acquity	Atlantis
13-May	7:00 - 4:00 pm	C4	1:00 - 4:00 pm	mfg conf	Acquity	Atlantis
14-May	7:00 - 4:00 pm	C4	1:00 - 4:00 pm	mfg conf	Acquity	Atlantis
10-Jun	8:00 - 4:00 pm	WTC 2&3		N/A	N/A	N/A
11-Jun	8:00 - 1:00 pm	Director's Room		N/A	N/A	N/A

Wilmslow Rooms

Date	Time	Primary Location	Break-out Rooms			
16-Feb	12:00 - 18:00	Hydrogen	14:00 to 17:00	Helium	Neon	Silicon
17-Feb	12:00 - 18:00	Hydrogen	12:30 to 16:30	Helium	Oxygen	Silicon
16-Mar	12:00 –18:00	Hydrogen	14:00 to 17:00	Helium	Oxygen	Silicon
17-Mar	12:00 - 18:00	Hydrogen	12:30 to 16:30	Helium	Oxygen	Silicon
13-Apr	12:00 - 18:00	Hydrogen	14:00 to 17:00	Helium	Oxygen	Silicon
14-Apr	12:00 - 18:00	Hydrogen	12:30 to 16:30	Helium	Oxygen	Silicon
13-May	12:00 - 18:00	Hydrogen	14:00 to 17:00	Helium	Oxygen	Silicon
14-May	12:00 - 18:00	Hydrogen	12:30 to 16:30	Helium	Oxygen	Silicon

What Happens if I Miss a Class?

It is important to your own learning and to the group process to attend all the class sessions.

However, it is understood that some absences cannot be avoided.

If you miss a class, please take the responsibility to:

- **Review the missed lectures**
- **Contact the instructor with any questions you may have as a result of reviewing the lecture materials**
- **Provide a brief outline of the missed material, identifying what you believe to be the key learning points, to the instructor.**
- **Contact your team project leader to ensure that you are aware of any project work completed, or action items assigned, during your absence.**

All lecture material will be posted on the WPI course site:

- **You will be told how to use the WPI course site, and how to access the lecture material, early in session 1**

Video Tutorial for myWPI

What are your Questions before We Introduce Ourselves?



Student Introductions (1 Minute Each)

- Name
- Specialty/Function/Title
- Brief Experience
- Current Project
- One expectation you have for the class
- One non-work Interest (Optional)

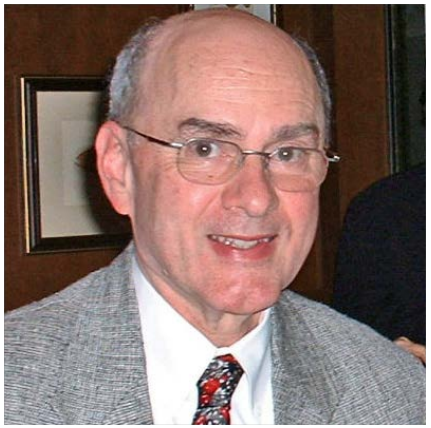


Instructor will collect on flip chart

Please print your name neatly on your tent card and bring it to class each day

Make sure we have your preferred email id and contact information

Your Instructor



- Robert B. Wheeler (Bob), ESEP, PMP
- Education:
  - BSEE (Drexel), MS Physics (Syracuse), MBA (URI)
- Experience:
  - IBM Federal Systems Division (1968 – 1993)
    - Avionics systems, Trident Submarine Program, SE technology development, IBM Enterprise Business Process Reengineering, Finance/Insurance Systems
  - Robert B. Wheeler & Associates, LLC (1993 to present)
    - Training and consulting world-wide: (North America, Europe, China, SE Asia, Australia)
    - Clients: US Intelligence agencies, DOD, NUWC, NASA, FAA, Lockheed Martin, Northrop Grumman, General Dynamics, IBM, AT&T, Lucent Technologies, Nortel Networks, MITRE, Idaho National Labs, Amtrak, Syntell, The Center for Systems Management, Strategy Bridge International, and others
  - Adjunct Faculty: WPI, Salve Regina, Roger Williams
- Portsmouth, RI (Baltimore, Philadelphia, Upstate NY, London)
- Married, three children, four grandchildren
- Travel, barbershop, model trains, genealogy, running “5K”s……

Office: 401-849-0299 Cell: 401-683-8311 mail: bob@rwheeler.net

Your Instructor



- Thomas F. Gannon (Tom), Ph.D.
- Education:
  - BSEE (IIT), MSEE (Purdue), PhD EE/CS (Stevens)
- Experience:
  - Bell Telephone Laboratories (1970 - 1978)
    - SAFEGUARD ABM, real-time digital control systems, development and integration of large-scale (enterprise) information processing systems for the Bell System
  - Digital Equipment Corporation (1978 to 1999)
    - Research and advanced development of high performance information processing systems
    - Established and guided the evolution of several national research consortia
  - The MITRE Corporation (1999 – 2014)
    - Enterprise systems engineering, command and control systems, and (enterprise) information processing systems for the DoD and other US government agencies
  - WPI Professor of Practice, ECE and Systems Engineering (2014 – present)
  - Previously Adjunct Faculty @ WPI, The Moore School, Stevens Institute
- Boylston, MA (Northern NJ, Philadelphia, Maynard and Bedford, MA)
- Married, three children, two grandchildren
- Travel, woodworking, antique home restoration, horse-back riding …

Office: 508-831-5517 Cell: 508-340-8973 mail: tgannon@wpi.edu

Primary Reference Textbooks

- ***Systems Engineering Principles and Practice (2nd ed.)*, Kossiakoff, A., Sweet, W. N., Seymour, S., and Biemer, S. M., John Wiley & Sons, Inc., 2nd Edition (May, 2011), ISBN-13: 9780470405482.**

- ***Discovering Requirements: How to Specify Products and Services*, Alexander, Ian and Beus-Dukic, Ljerka, John Wiley & Sons, Inc., 2009, ISBN-13: 9780470712405.**

- ***The Engineering Design of Systems: Models and Methods*, Buede, Dennis M., John Wiley & Sons, Inc., 2nd Edition (February, 2009), ISBN: 0-470-16402-6.**

- ***The Art of Systems Architecting, Third Edition,* Rechtin, Eberhardt and Maier, Mark W., CRC, 3rd Edition (January, 2009), ISBN: 1-420-07913-1.**

- ***Visualizing Project Management: Models and Frameworks for Mastering Complex Systems*, Forsberg, Mooz, and Cotterman, Wiley, 3rd edition (September 1, 2005), ISBN-13: 978-0471648482**

Learning Objectives and Homework Assignments

Session 1 Introduction: Learning Objectives

- **Develop and use a common vocabulary**
- **Explain systems engineering processes (both hard and soft) in the context of a system lifecycle model**
- **Plan and implement a risk management process**
- **Use a Systems Engineering Management Plan to plan the technical work on the project**

Session 1 Introduction: Homework Assignment: Due at beginning of Session 2*

Develop the following deliverables:

- Communication Plan, including common vocabulary approach and draft glossary
- System Lifecycle Model for your project
- Risk Analysis emphasizing technical risks (consider doing a draft Risk Management Plan)
- Draft Systems Engineering Management Plan (SEMP)
- Draft Configuration Management Plan showing how you will control the evolving baseline

Update the previously developed deliverables:

- Project Description

* Each project team must decide which of, and the extent to which, each of these deliverables will be done

Session 2 Stakeholder Needs: Learning Objectives

- **Identify the “problem to be solved” and the “context”**
- **Elicit and express stakeholders expectations and requirements**
- **Develop a user Concept of Operations (CONOPS)**
- **Develop solution selection criteria and acceptance criteria**

Session 2 Stakeholder Needs: Homework Assignment due at beginning of Session 3*

Develop the following deliverables:

- **Statement of the Problem to be Solved for your project**
- **Stakeholder Analysis**
- **User CONOPS**
- **Stakeholder Expectations**
- **Stakeholder Requirements**
- **Initial cut at Traceability**

Update the previously developed deliverables

* Each project team must decide which of, and the extent to which, each of these deliverables will be done

Session 3 System Requirements: Learning Objectives

- Conduct system functional and performance analysis
- Analyze and express system non-functional requirements
- Flow-down system level requirements to lower level entities
- Manage requirements through the system lifecycle
- Express system requirements in appropriate artifacts

Session 3 System Requirements: Homework Assignment due at beginning of Session 4*

Develop the following deliverables:

- **Functional Architecture (see Buede text)**
- **Analysis of Non-Functional requirements**
- **System Requirements Document or equivalent expression of system requirements**
- **Draft Requirements Verification Traceability Matrix, or equivalent (Should trace to source, and point to verification approach)**

Update the previously developed deliverables

* Each project team must decide which of, and the extent to which, each of these deliverables will be done

Session 4 System Architecture and Design: Learning Objectives

- **Develop creative alternative system architectures which address system operational, functional and physical viewpoints**
- **Conduct analyses of architectural alternatives to meet stakeholder and system requirements**
- **Manage system architecture and design through the system lifecycle**
- **Express system architectures and designs in appropriate artifacts**

Session 4 System Architecture and Design: Homework Assignment due at beginning of Session 5*

Develop the following deliverables:

- **Map of functional to physical architecture**
- **Candidate architecture descriptions**
- **Trade-off study documentation**
- **Selected architecture description**

Update the previously developed deliverables

* Each project team must decide which of, and the extent to which, each of these deliverables will be done

Session 5 System Integration and Test: Learning Objectives

- **Develop strategies and plans for system integration, validation and verification testing**
- **Manage system test strategies and plans through the system lifecycle**
- **Express system test strategies and plans in appropriate artifacts**

Session 5 System Integration and Test: Homework Assignment due at beginning of Session 6

Develop the following deliverables:

- **Test and Evaluation Plan, including plans for all verification, validation, and testing required for the project, and all associated facilities and support needed**
- **Requirements Verification and Test Matrix (RVTM)**

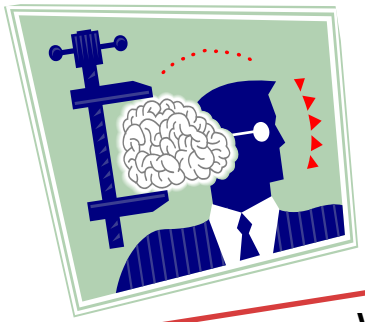
Update the previously developed deliverables

*Notes:
- Each project team must decide which of, and the extent to which, each of these deliverables will be done;
- This assignment will not be separately reported during a class session, but will be reflected in the capstone Presentation

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		

All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?



What's next?

Day	Overview and Concepts of Systems Engineering	Hours*
1	Waters Management kick-off	1/2
	Introduction and overview of entire program (Lesson 1), including Introduction and overview of the projects (15 minutes each)	2
	Presentation and discussion of Concepts of SE (Lessons 2 – 9)	4
	Project assignments, team formations, and working sessions to apply SE concepts to the projects	1
	Day 1 wrap-up: Takeaways and Q&A	1/2
2	Review of day 1 working sessions	1
	Concepts of Systems Engineering (concluded) (Lesson 10 – 14)	4
	Project team working sessions (continued)	2
	Day 2 wrap up and Q&A	1
3	Team working session feedback to class (informal presentations)	1
	Project team working sessions (concluded)	5
	Check-in with class on application of SE concepts to their projects	1
	Session 1 wrap up: Q&A and homework due next session	1

* Times include breaks, and may be changed based on needs of the class



Systems Engineering Technical Short Course

(Session 1)

Concepts of Systems Engineering

Robert B. Wheeler, Instructor

<= Indicates slide change after printing

Session 1 Agenda (Jan 27 – 29): Timing

Day	Overview and Concepts of Systems Engineering	Hours*
1	Waters Management kick-off	1/2
	Introduction and overview of entire program (Lesson 1), including Introduction and overview of the projects (15 minutes each)	2
	Presentation and discussion of Concepts of SE (Lessons 2 – 9)	4
	Project assignments, team formations, and working sessions to apply SE concepts to the projects	1
	Day 1 wrap-up: Takeaways and Q&A	1/2
2	Review of day 1 working sessions	1
	Concepts of Systems Engineering (concluded) (Lesson 10 – 14)	4
	Project team working sessions (continued)	2
	Day 2 wrap up and Q&A	1
3	Team working session feedback to class (informal presentations)	1
	Project team working sessions (concluded)	5
	Check-in with class on application of SE concepts to their projects	1
	Session 1 wrap up: Q&A and homework due next session	1

* Times include breaks, and may be changed based on needs of the class

Concepts of Systems Engineering

1. Program Overview
2. What is Systems Engineering?
3. Structure and Environment of Systems
4. Systems Development Process
5. Systems Engineering Processes Overview
6. Systems Engineering in Needs Assessment
7. Risk Management
8. Concept Exploration
9. Concept Definition
10. Advanced Development Stage
11. Engineering Design Stage
12. Integration and Evaluation Stage
13. Production and Deployment
14. Operations, Support, and Disposal

Plus Appendices (See next slide)

Lesson Content:
- Why are we here?
- What are successful systems?
- What is SE – definitions and characteristics

Exercise: Your SE environment

Appendix A

The course contains interactive lectures, student exercises, quizzes, and takeaways

The course will reference, but will not cover, the Appendices

Lecture Content (Parts 3 and 5)

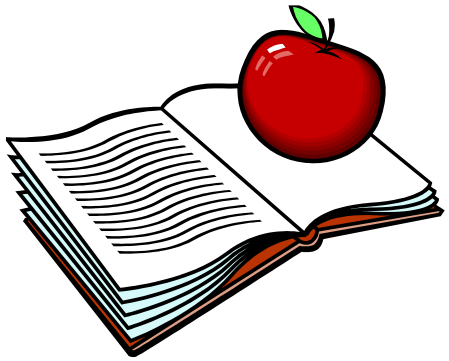
Appendices

A. **Lesson 2 Supplement – What is SE?**

B. **Lesson 4 Supplement – DOD Instruction 5000.02 information**

C. **Lesson 14 Supplement – COTS**

D. **Course Supplement prepared after printing, not in your book**



The course contains interactive lectures, student exercises, quizzes, and takeaways

The course will reference, but will not cover, the Appendices



Systems Engineering Technical Short Course

(Session 1)

Lesson 2
What is Systems Engineering?

Robert B. Wheeler, Instructor

Reference : “Systems Engineering, Principles and Practice,” Kossiakoff et. al – Chapter 1, 2

Why Are We Here? (1/3)

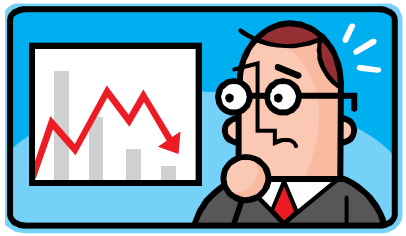
- **To meet your expectations from the "Student Introduction Exercise" just completed……**
- **And to meet the instructor's expectations:**
  - **To have you all learn the material**
  - **To meet your expectations. . . . . .**
  - **To have you be able to use the course concepts to make a positive difference in the performance of your jobs and the success of your projects.**



Good systems engineering lead to better results for customers and for ourselves……..

- **And, we are here because of the state of systems and systems engineering in the World today……..**

Unfortunately, things are not always so happy….



Why Are We Here? (2/3)

- **System Successes:**
  - **Trident Submarine**
  - **IPOD**
  - **Internet**
  - **Lexus**
  - **747**
  - **C-130 Hercules**
  - **IBM PC (original)**
  - **Hybrid Automobiles**
  - **Cellular Communications**
  - **Apollo**
  - **eBay**

- **System Failures?**
  - **Iridium and Globalstar - Bankrupt**
  - **English Channel Tunnel – Can't service debt**
  - **Concorde – Money loser**
  - **Boston Big Dig – 500% overrun to $15B**
  - **Oakland San Francisco Bay Bridge – $Bs overrun**
  - **Betamax**
  - **Denver Airport Baggage system – ultimately cancelled**
  - **Shanghai Maglev Train – Can't service debt**

- Affordable Health Care Act & Website?

Most systems experience failures and successes.

Where do your projects fit on this slide?

Why Are we Here? (3/3)

What are the Characteristics of Successful Systems?

-
-
-
-
-
-

Instructor will facilitate this brainstorming session

Are any of these success criteria related to Systems Engineering?

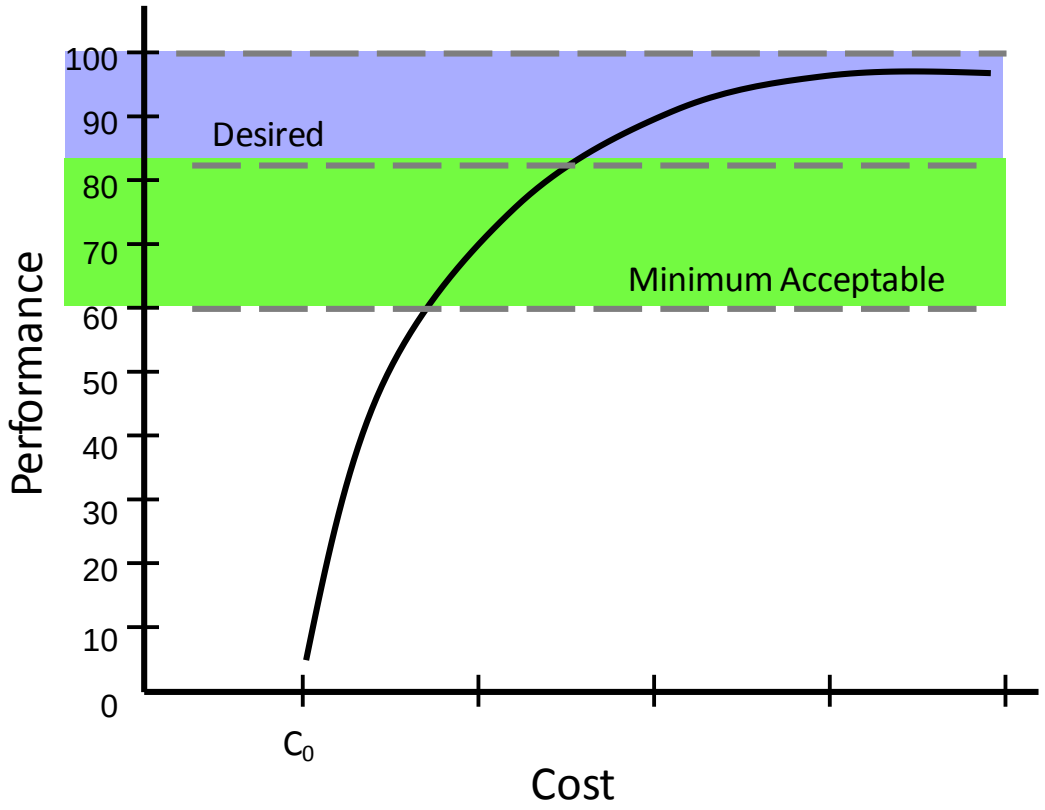
What is a Successful System? (1/4)

- **A Successful System** *… meets the user's requirements, operates successfully in the field, has a long, useful operating life and can be upgraded easily to accommodate new requirements and technological innovation*

- **The "Best" System** *… embodies the "best" balance among critical system attributes of value to the end user (e.g.: cost, performance, ease of maintenance, etc.). It may not have the highest performance.*

What is a Successful System? (2/4)

"Best" System: Performance vs. Cost



What is a Successful System? (3/4)

"Best" System: Performance vs. Cost

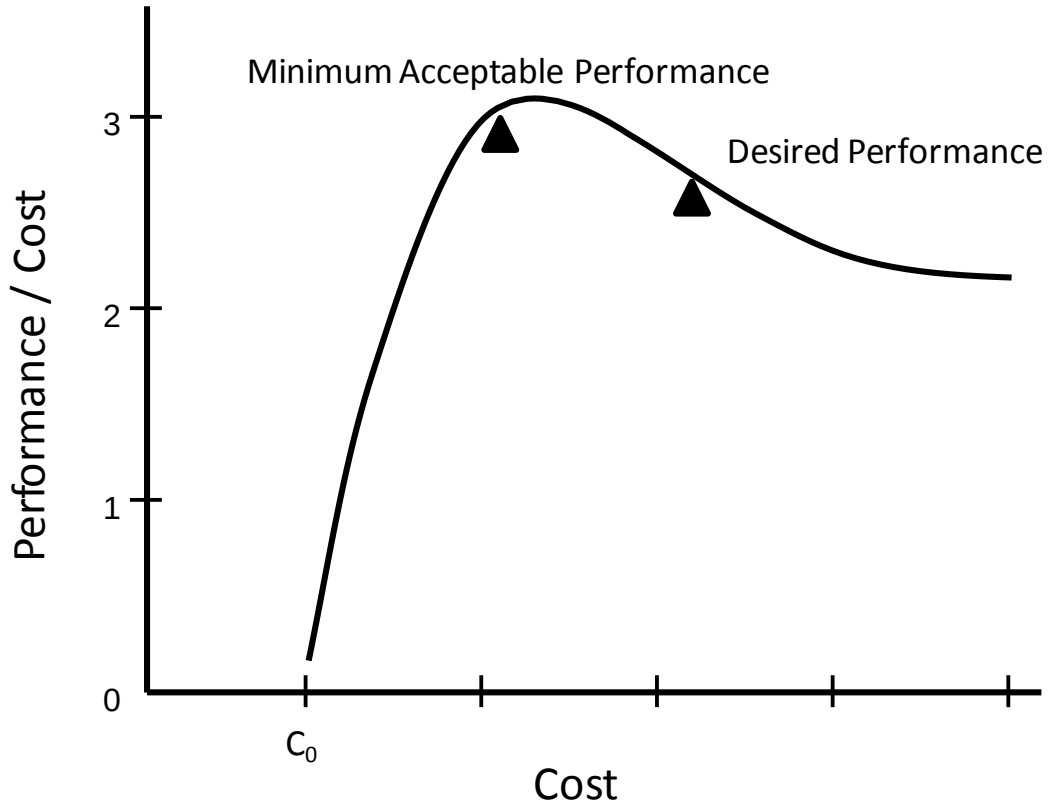
- **A Successful System** *... meets the user's requirements, operates successfully in the field, has a long, useful operating life and can be upgraded easily to accommodate new requirements and technological innovation.*

- **The "Best" System** *... embodies the "best" balance among critical system attributes of value to the end user (e.g.: cost, performance, ease of maintenance, etc.) It may not have the highest performance.*

- **A Balanced System** ***... composed of components which are designed individually to provide the specific functionality while delivering capabilities to the user that are optimal for the best overall system operation.***

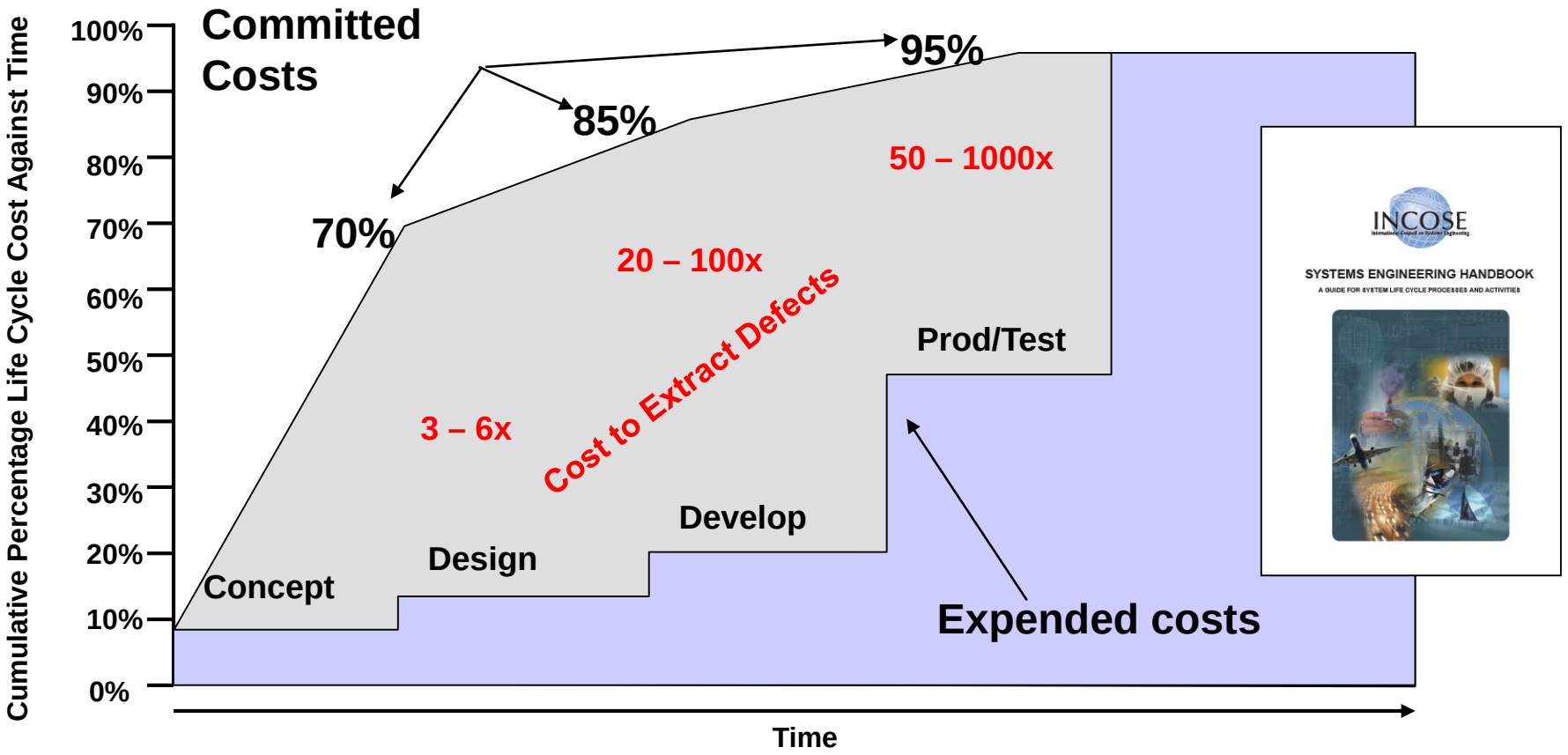
What is a Successful System? (4/4)

Balanced System: Performance/Cost vs. Cost



Systems engineering is key to achieving the right balance

Why Are We Here? (1/2)



Work and decisions made early in the project set the stage for project success or failure. In which of these stages is systems engineering most important?

Why Are We Here? (2/2)

3. When do we define what we build, test, and deploy?

2. When do we build and test what we deploy?

1. What do we do here?

Produce, transition, operate, maintain, support, secure, ……. , dispose

DEFINE	BUILD/TEST	DEPLOY
Concept Development Phase Needs analysis, concept exploration, concept definition, requirements engineering, and architecture	Engineering Design Phase Design and development, system integration, test and evaluation	Post-development Phase System implementation, Operations & Maintenance, and Transition

4. When do we:
- spend the most money and time?
- have the most leverage?

5. What is the focus of:
- Your organization's projects?
- Your current project(s)?

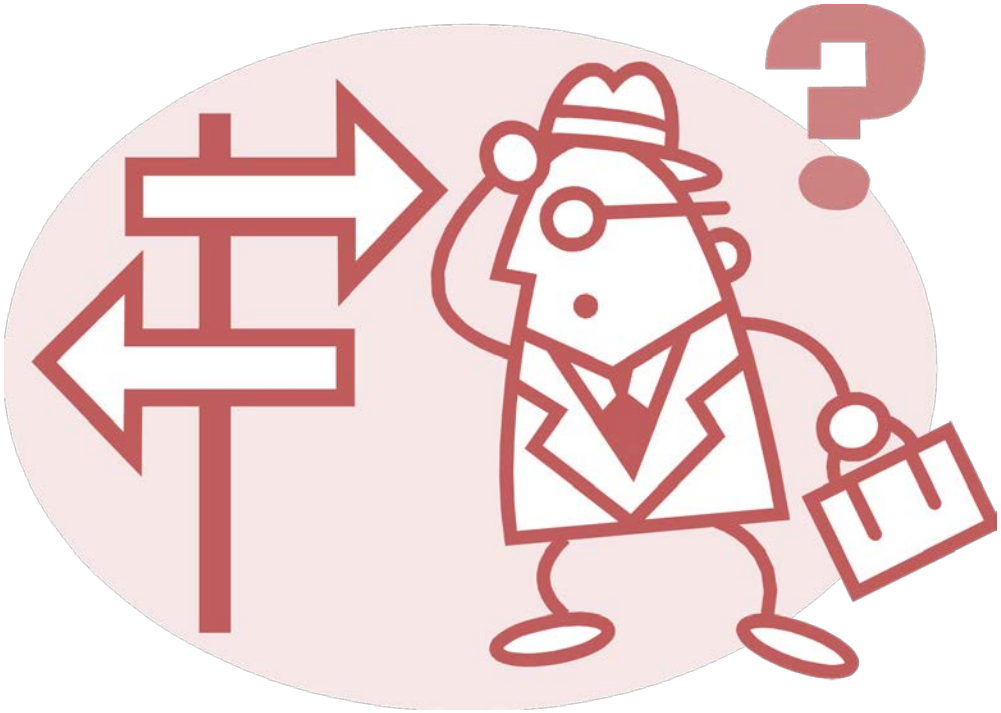
What does this suggest about our systems engineering efforts?

Motivation for Systems Engineering

- There is tremendous potential for wasted effort on large projects, since their development requires that many subsystems be developed in parallel
- Without a clear understanding of what must be done for each subsystem the development team risks having inconsistent designs, conflicting interfaces, or duplication of effort
- Systems engineering provides a systematic, disciplined approach to defining, for each member of the development team, what must be done for success
- Systems engineering can contribute significantly to successful systems, and to achieving the right balance of performance, cost, and schedule through the system lifecycle

How does this apply to Waters Corporation Projects?

What are your Questions before We Define Systems Engineering?



What is Systems Engineering? (1/1)

- Systems Engineering is an interdisciplinary approach and means to enable the realization of successful systems
- It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem, e.g.,
  - Operations
  - Cost & Schedule
  - Performance
  - Training & Support
  - Test
  - Disposal
  - Manufacturing

What is Systems Engineering? (2/2)

- **Systems Engineering integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation**

- **Systems Engineering considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs**

Systems Engineering

Another Definition from Wikipedia

- What makes Systems Engineering (SE) unique, especially in contrast with traditional engineering disciplines, is that SE does not build tangible products
- Whereas civil engineers might design buildings and electrical engineers might design circuits, systems engineers deal with abstract systems, and rely on other engineering disciplines to design and deliver the tangible products that are the realization of those systems
- The systems engineering effort spans the whole system lifecycle
- SE focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem

A Consensus of INCOSE Fellows (1/2)

What is a System?

- **A system is a construct or collection of different elements that together produce results not obtainable by the elements alone**
- **The elements, or parts, can include people, hardware, software, facilities, policies, and documents, that is, all things required to produce systems-level results**
- **The results include system level qualities, properties, characteristics, functions, behavior and performance**
- **The value added by the system as a whole, beyond that contributed independently by the parts, is primarily created by the relationship among the parts, that is, how they are interconnected**

A Consensus of INCOSE Fellows (2/2)

What is Systems Engineering?

- **Systems Engineering is an engineering discipline whose responsibility is creating and executing an interdisciplinary process to ensure that the customer's and stakeholders' needs are satisfied in a high quality, trustworthy, cost efficient, and schedule compliant manner throughout a system's entire life cycle**
- **The process is usually [often?] comprised of these tasks:**
  - **State the problem**
  - **Investigate alternatives**
  - **Model the system**
  - **Integrate**
  - **Launch the system**
  - **Assess performance**
  - **Reevaluate**
- **It is important to note that the Systems Engineering Process is not sequential. The functions are performed in a parallel and iterative manner.**

Variations

Like all processes, the Systems Engineering Process in any organization should be documented, measurable, stable, of low variability, used the same way by all, adaptive, and tailorable!

Now this may seem like a contradiction—and perhaps it is—but one size does not fit all.

The description(s) of the Systems Engineering process contained in this course is(are) just one(some) of many that have been proposed. Some are bigger, some are smaller, but most are similar.

What is Systems Engineering?

> ***"The function of systems engineering is to guide the engineering of complex systems."***

- **Guide** ⇒ ***Lead, manage, direct … to show the way***
- **Engineering** ⇒ ***The application of scientific principles to practical ends***
- **System** ⇒ ***A set of interrelated components working together towards a common objective***
- **Complex** ⇒ ***Elements of the system are diverse and tightly coupled with one another***

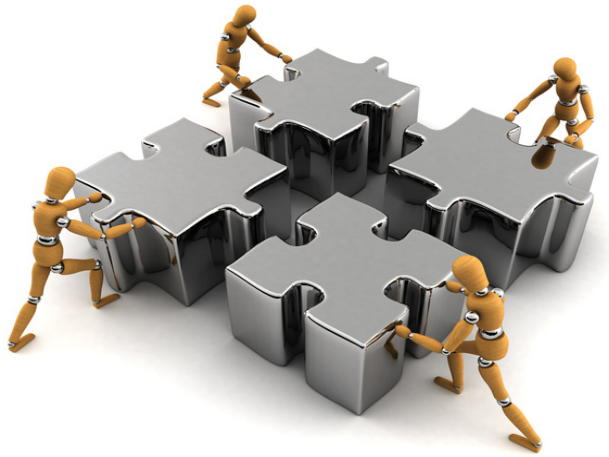
Systems engineering must lead the interdisciplinary team performing the appropriate processes through the systems lifecycle

Systems Engineering ⇔ Traditional Engineering

- Systems engineering is focused on the system as a whole
  - Emphasizes total system operation
  - Includes interactions between the system and its operational environment
- Systems engineers lead the formative (Concept Development) stage of development
  - Making critical design decisions
  - Leading to the system's functional design
- Systems engineering bridges the traditional engineering disciplines
  - Traditional engineering focuses on the development of unique system elements
  - Systems engineering must ensure that all elements must function properly in combination with one another to realize an operational system

How Are You Doing in Your Environment? Exercise: *Knowledge Café* (1/2)

- We will use the Knowledge Café to understand the systems engineering environment in which we work;
- Three rounds of brainstorming, three questions, with a summary presentation at the end.
- Instructor will explain the Knowledge café' process



Depending on the time available, this exercise might be run as a "5 minute manager"

How Are You Doing in Your Environment? Exercise: *Knowledge Café* (2/2)

Café Rules

- Each table has a topic and a table owner who stays with the table and records the results
- Groups stay intact and rotate to each table
  - At the first 3 tables they provide input
  - At the last table, they identify the top 3 items from the previous groups
- Each table owner presents a wrap up to the class



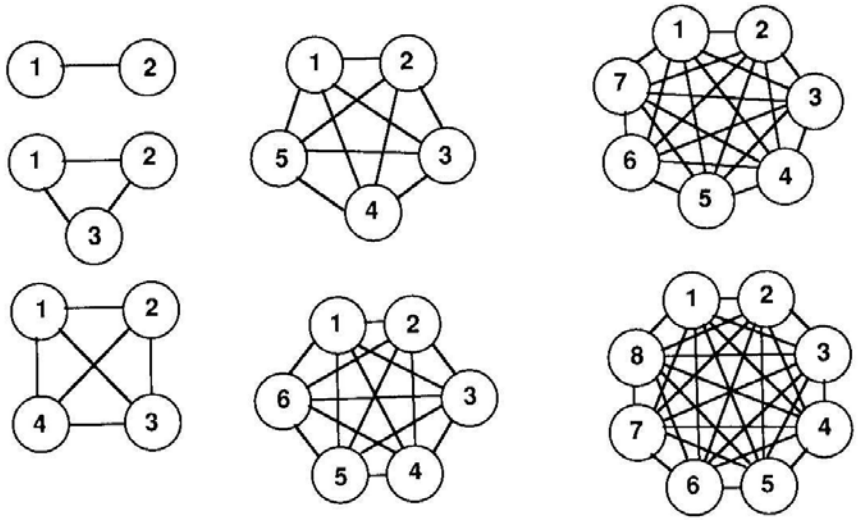
Café Table Topics

1. What are the trends and issues in your Systems Development environment?
2. What is your organization doing well in systems development? What improvements do you need to make?
3. What do you need from others (your customers, management , contractors, and subcontractors) to improve your organization's Systems Engineering?

4. Optional Question: What "tools" are you and your organization using to perform systems engineering?

Systems Engineering as a Discipline

- **Complexity increased orders of magnitude with the creation of coupled mechanical-digital systems, especially in defense**
- **Creation of systems of systems, with interacting users, acquisition, training, service, support, etc. made development and fielding even more complex**



Complex Engineered Systems

- **Basic Characteristics:**
  - **Satisfies a specific need** *… the system provides a useful function*
  - **Composed of diverse components** *… that have well-defined, complex relationships among one another*
  - **Uses advanced technology** *… that is required to realize its primary function and involves development risks and high cost*

Complex Engineered Systems - Examples

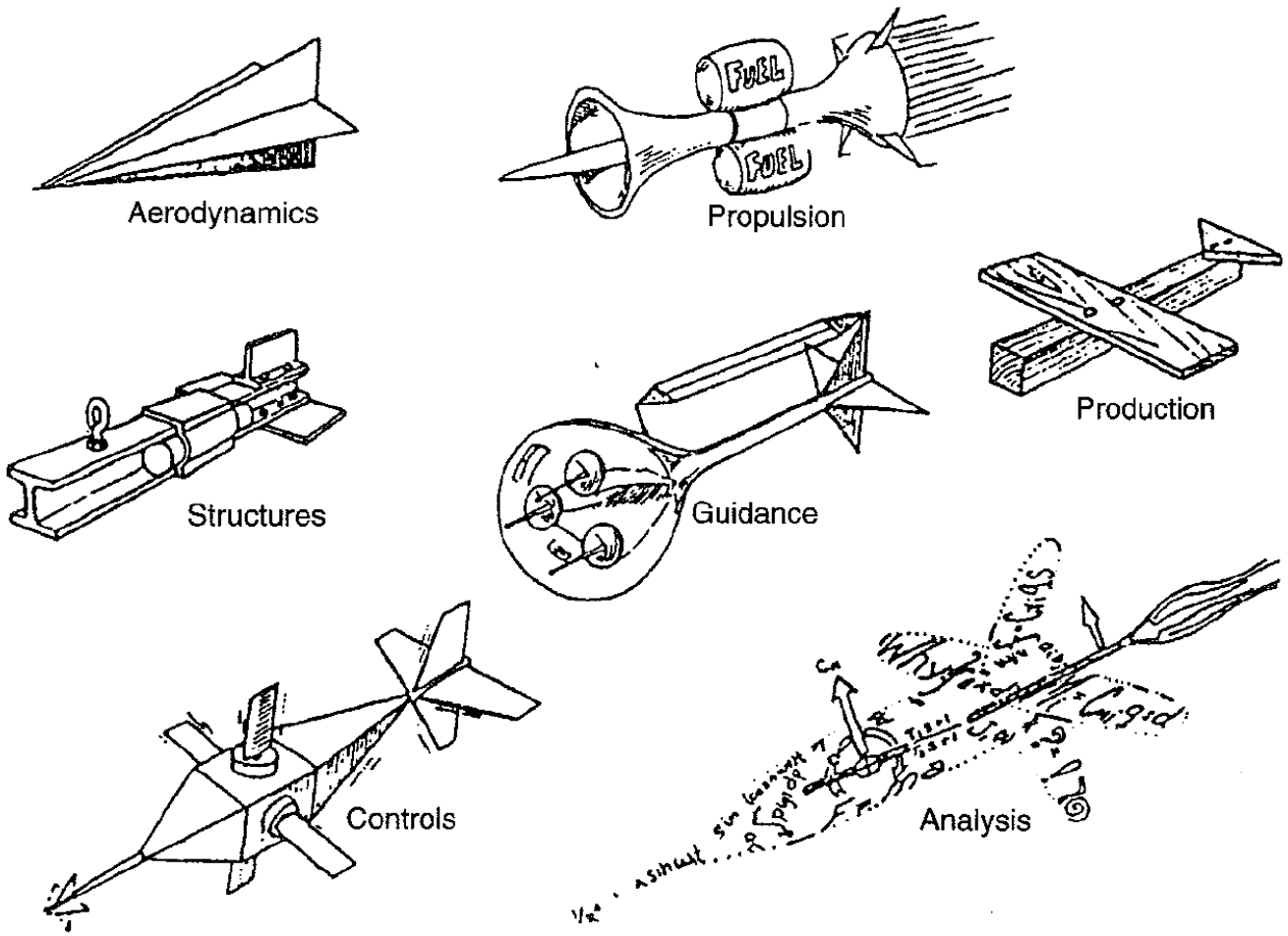
System	Inputs	Process	Outputs
Weather Satellite	Images	• Data Storage • Transmission	Encoded Images
Terminal Air Traffic Control System	Aircraft Beacon Responses	• Identification • Tracking	• Identity • Air Tracks • Communications
Track Location System	Cargo Routing Requests	• Map Tracing • Communication	• Routing Info. • Delivered Cargo
Airline Reservation System	Travel Requests	Data Management	• Reservations • Tickets
Clinical Information System	• Patient ID • Test Records • Diagnosis	Information Management	• Patient Status • History • Treatment

Complex Engineered Systems – More Examples

System	Inputs	Process	Outputs
Passenger Aircraft	• Passengers • Fuel	• Combustion • Thrust • Lift	Transported Passengers
Modern Harvester Combine	• Grain Field • Fuel	• Cutting • Threshing	Harvested Grain
Oil Refinery	• Crude Oil • Catalysts • Energy	• Cracking • Separation • Blending	• Gasoline • Oil Products • Chemicals
Auto Assembly Plant	• Auto Parts • Energy	• Manipulation • Joining • Finishing	Assembled Auto
Electric Power Plant	• Fuel • Air	• Power Generation • Regulation	• Electric AC Power • Waste Products

Systems Engineering Viewpoint

The Ideal Missile Design from Various Viewpoints



Systems Engineering Characteristics

- **Multidisciplinary knowledge** *… required to interact effectively with a wide range of specialists in different technical disciplines and enable disparate groups of specialists to function as a team to achieve a common objective*

- **Approximate Calculations** *… required to make a rough estimate, or an order-of-magnitude "sanity check," on the results of complex calculations to catch gross errors or major omissions*

- **Skeptical (but *POSITIVE)* Thinking …**
  - *Tempers optimism on the probability of success of a given design approach with prudent risk taking based on early validation of that approach*
  - *Conveys a "can-do" attitude that generates and sustains confidence and team spirit across the organization*

- **Systems Thinking – see next slides**

Systems Thinking: Is this Class a System?

If so, what must be considered in its design?

Expectations
- Student
- Instructor
- "Bosses"
- Sponsors

Impact
- Career
- Performance
- Credentials
- Certification

Inter-relationship
- Student-peers
- Student-instructor
- Support-staff
- Hierarchies?

Process
- Methods
- Assessment
- Activities
- Lecture



Facilities
- Materials
- Class room
- Equipment
- Furniture
- Support S/W
- Access

Usefulness
- Relevance
- Applicability
- Can be implemented
- Understandability

Business
- Cost
- Funding
- Perceived ROI

Environment
- Location
- Physical Atmosphere
- Attitudes

Systems thinking is the process of understanding how things influence one another within a whole. (Peter M. Senge)

….and many more….

Systems Thinkers…..

Appreciate the interconnectedness of all things.

One of Leonardo da Vinci's seven elements of genius

See the forest *and* the trees.





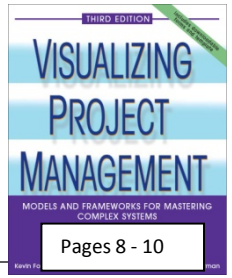
Comprehend the relationships among system components and external factors and how they influence performance.

Systems Thinkers

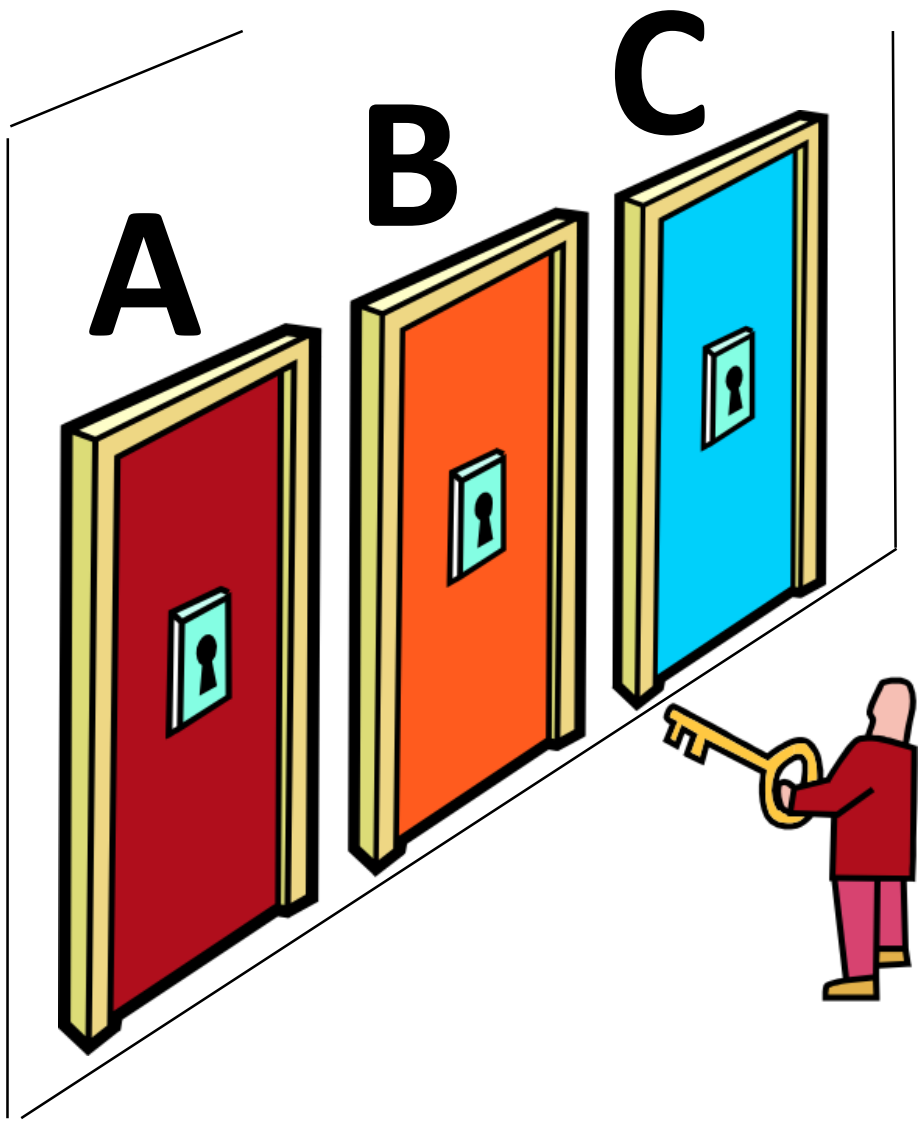
See the whole picture
Seek diverse perspectives
Look for interdependencies
Find opportunities and risks
Focus on structure, not blame
Use tension and paradox to advantage
Consider how mental models create futures
Use peripheral vision to consider cause/effect
Give voice to the long-term



Adapted from *Systems Thinking Playbook*,
Linda Booth Sweeney and Dennis Meadows, 1995



Exercise: Systems Thinking: The Three Doors



There is $100,000 prize behind one of these doors, and nothing behind the other two.

You get to open one of the doors, and take whatever is there
Now, pick your door to open. The probability you have won is _____

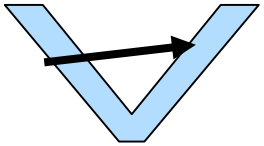
OK, but before I open your selected door, I open another door for you, and you see there is no prize there

Now I give you the opportunity to keep your door, or to change to the last door. What do you do?

Now, what is your probability of winning the prize? _____

Source: Wikipedia

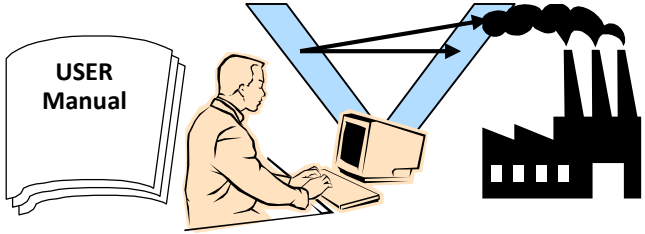
Four Techniques that Promote Systems Thinking



Validation Planning and Solution Requirements

Early determination of the customer validation approach often clarifies requirements. Verification Planning at Concept Development often eliminates flawed concepts that lead to failure.

Session 1



Concurrent Engineering

Concurrent product and process development where all phases of the product life cycle and all stakeholders are considered at the outset and throughout the project cycle.

Session 2



Discovery and Analysis

Ensuring all needs are considered through stakeholder involvement, conceiving of alternate solutions, and rigorous analysis to select the best solution.

Sessions 2 - 4



Modeling and Simulation

Ensuring accurate communication and understanding of all aspects of a problem to be solved including the context of implementation

Throughout

INCOSE (www.incose.org)

The International Council on Systems Engineering



Summary

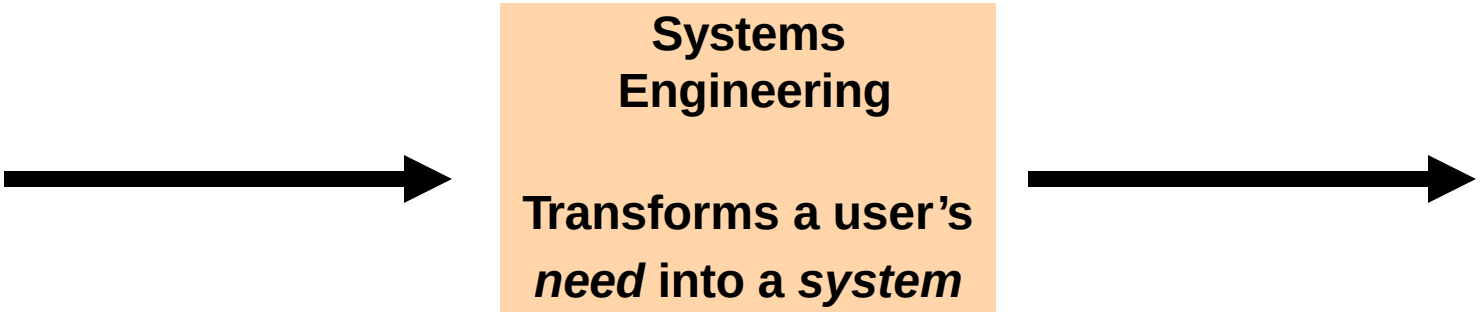
- Systems engineering guides the engineering of complex systems

- A complex engineered system is
  - Composed of many intricate, inter-related diverse components
  - Requires systems engineering to lead it development

- Systems engineering
  - Focuses on the system as a whole
  - Understands customer needs and the operational environment
  - Leads system conceptual design
  - Bridges traditional engineering disciplines

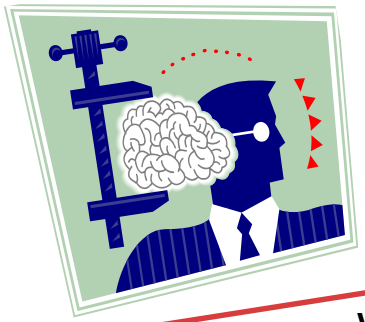
- Systems engineering viewpoint is focused on producing successful systems

Most simply…



Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?

Back-up Slides

Suggested Answers to In-Seat Exercise: What Constitutes System Success?

- ✓Delights customers and stakeholders
- ✓Satisfies specified requirements
- ✓Solves the "problem-to-be-solved" in the target context
- ✓Does what it is supposed to do efficiently and effectively
- ✓Meets budget, schedule, and technical requirements
- ✓Is producible, testable, sustainable, supportable, maintainable, secure, operable, safe, legal, ……..

These are typical results of the earlier brainstorming exercise

Systems Engineering is critical to all of these success criteria

Origins of Systems Engineering

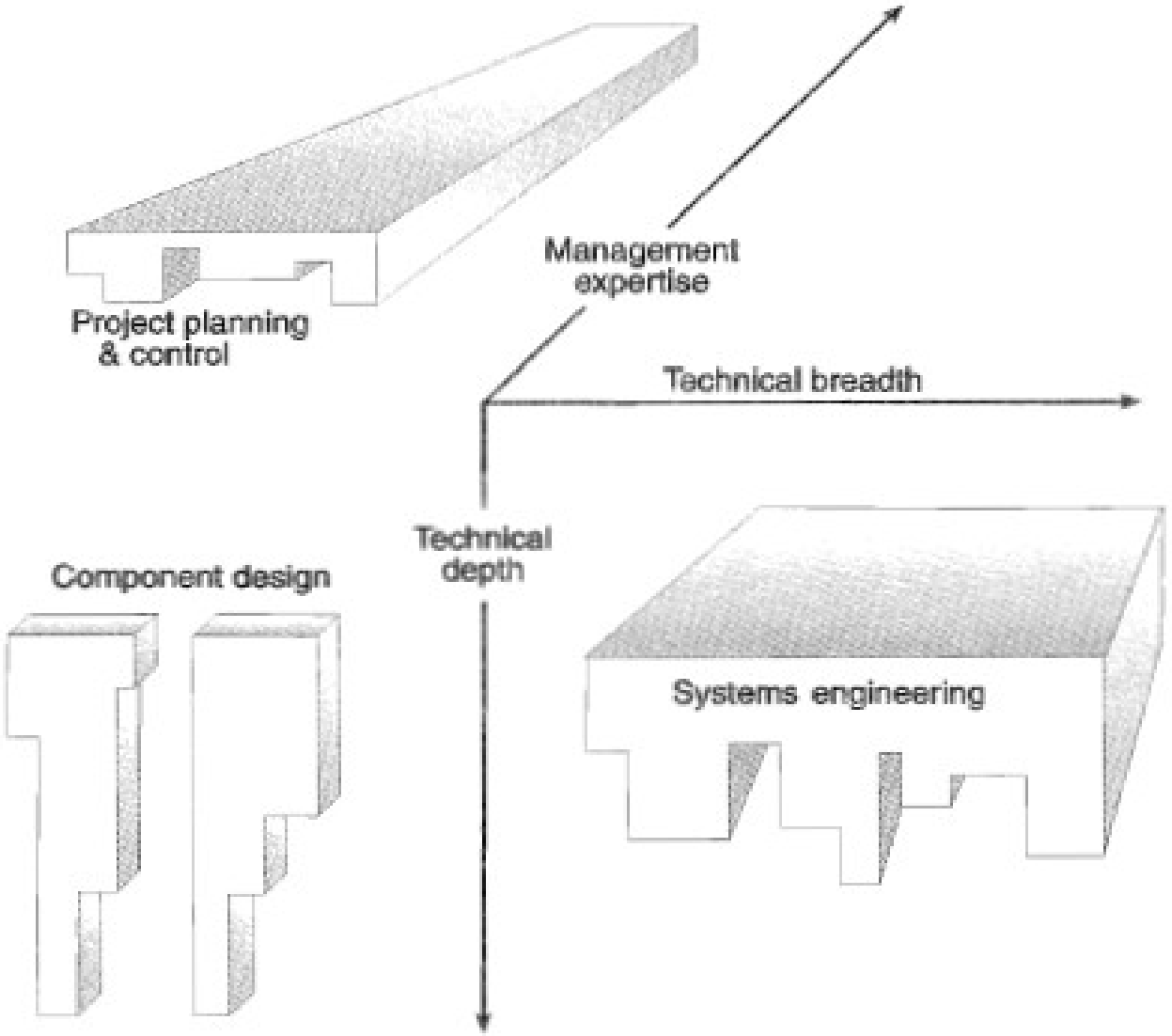
- **Rapid growth of technology …. stimulated by WWII and investments in technology for military advantage**

- **Growth of Automation … stimulated by the advent of the digital computer and software technology**

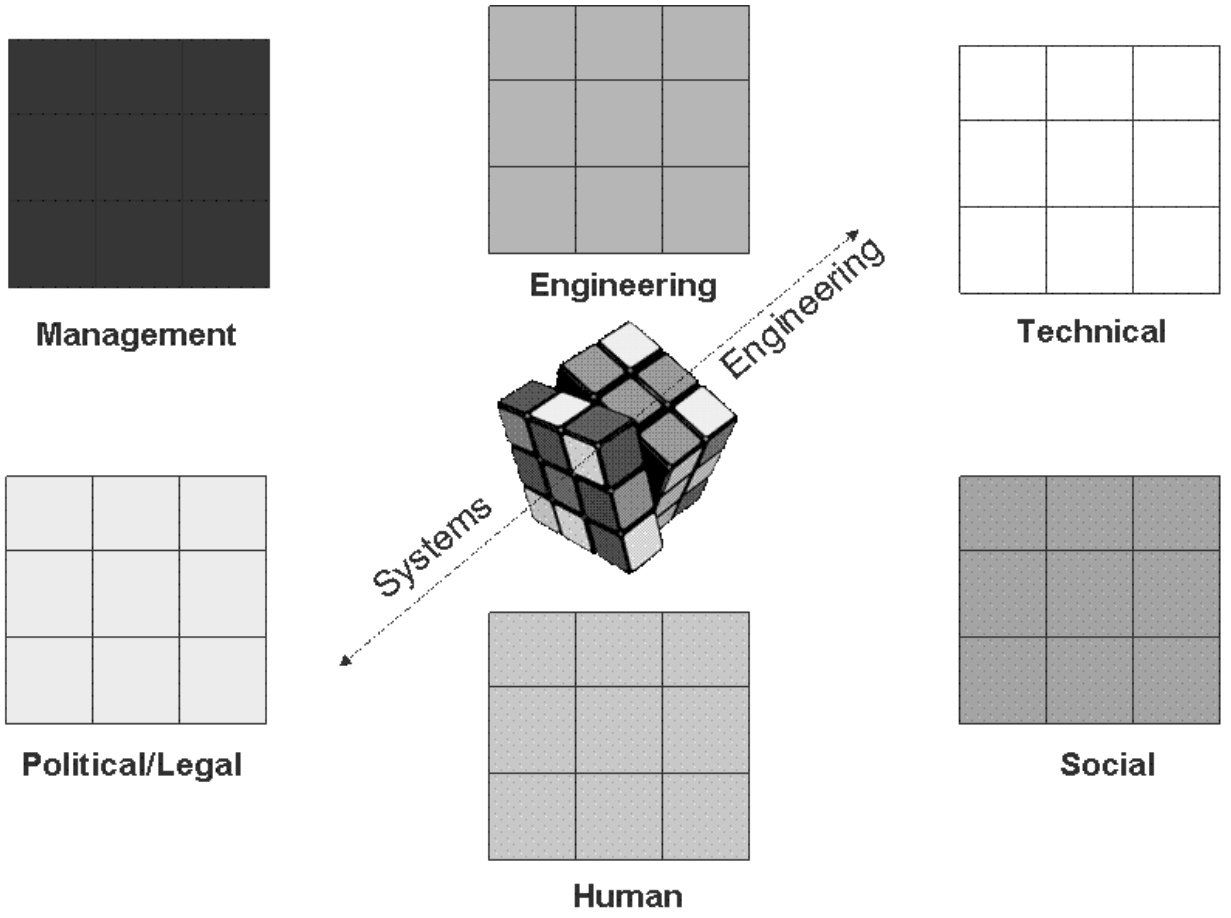
- **Competition … driving innovation and more advanced system solutions …**

- **Specialization** ***… partitioning complex systems into specialized components***

Dimensions of SE, Design and PM

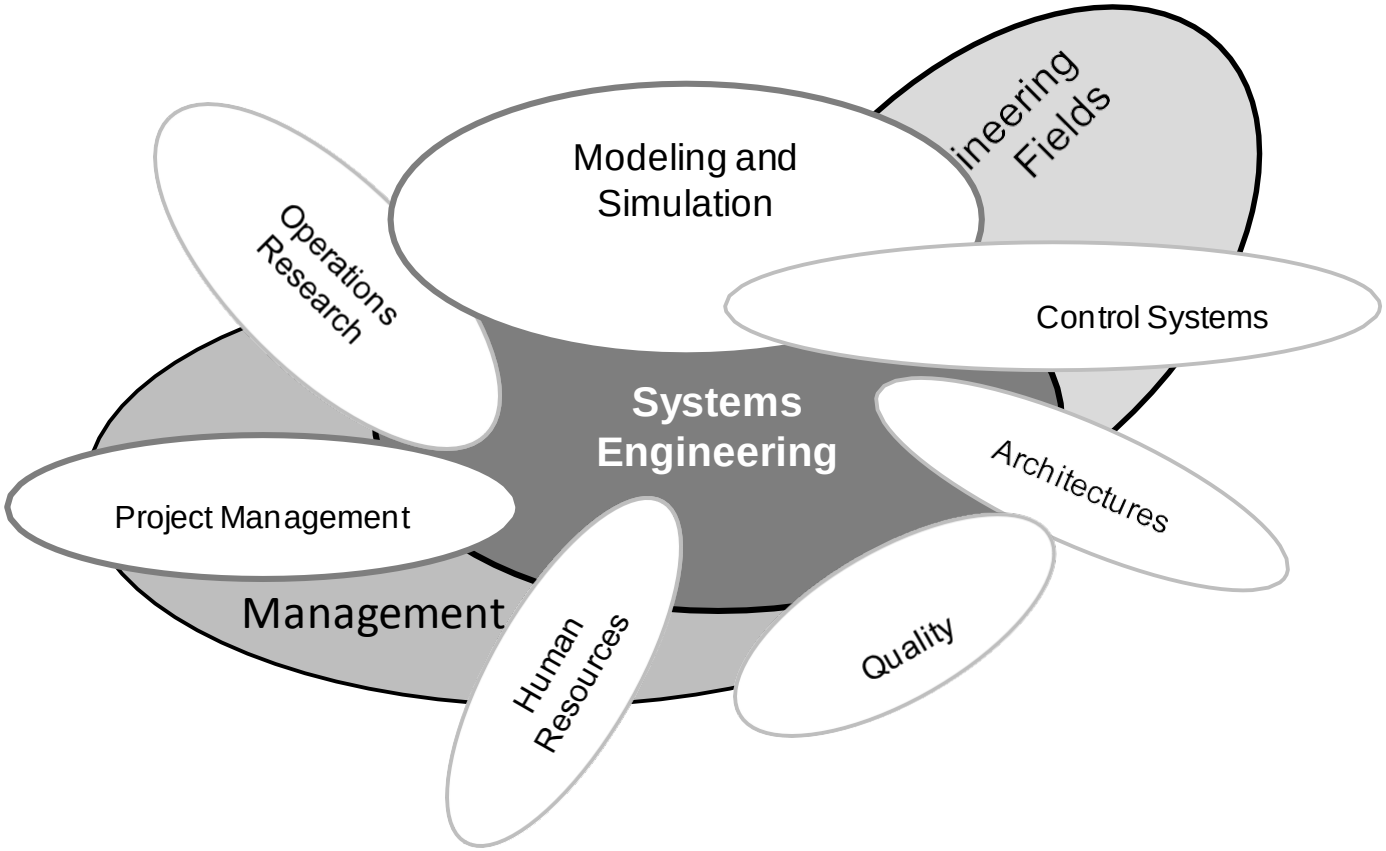


Systems Engineering Domains

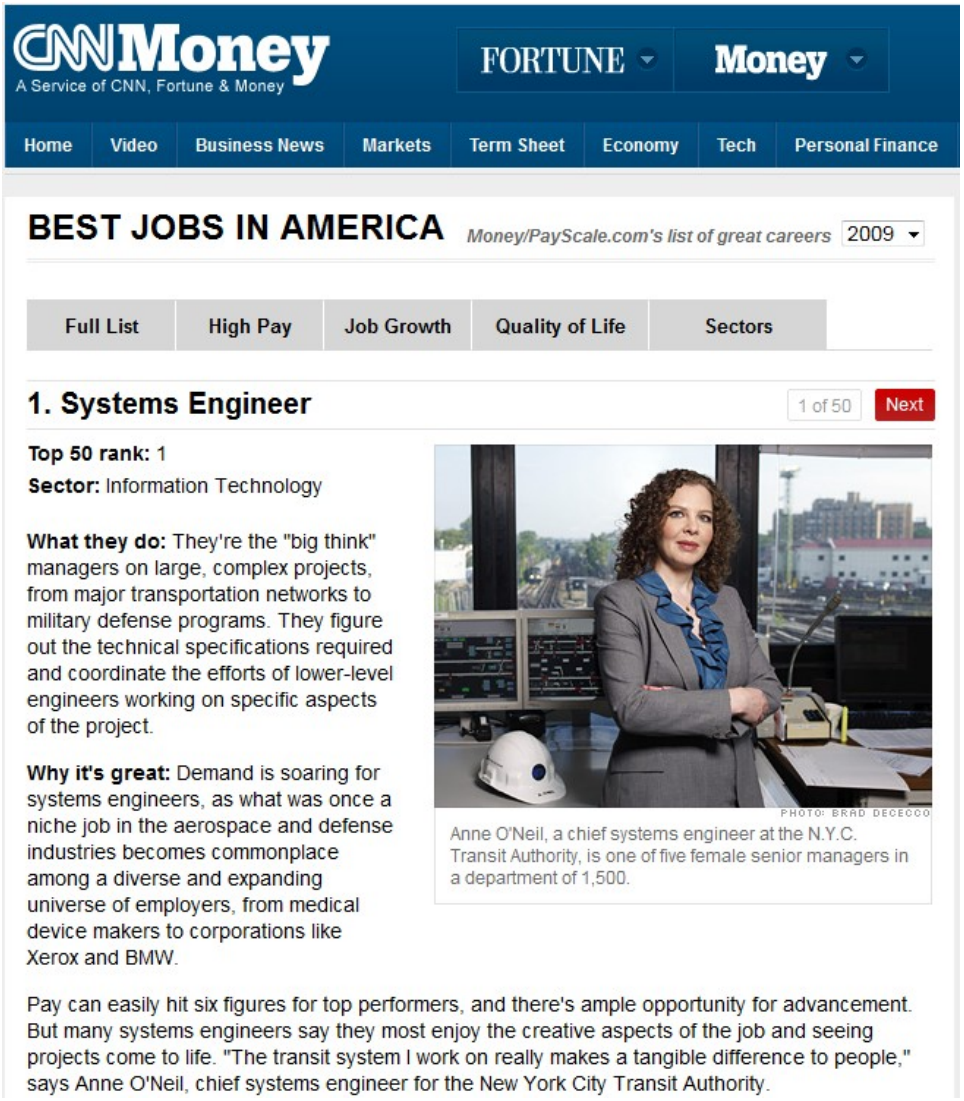


1

Systems Engineering Fields



Systems Engineering as a Profession



CNNMoney
A Service of CNN, Fortune & Money

FORTUNE | Money

Home | Video | Business News | Markets | Term Sheet | Economy | Tech | Personal Finance

BEST JOBS IN AMERICA *Money/PayScale.com's list of great careers* 2009

Full List | High Pay | Job Growth | Quality of Life | Sectors

1. Systems Engineer 1 of 50 Next

Top 50 rank: 1

Sector: Information Technology

What they do: They're the "big think" managers on large, complex projects, from major transportation networks to military defense programs. They figure out the technical specifications required and coordinate the efforts of lower-level engineers working on specific aspects of the project.

Why it's great: Demand is soaring for systems engineers, as what was once a niche job in the aerospace and defense industries becomes commonplace among a diverse and expanding universe of employers, from medical device makers to corporations like Xerox and BMW.

PHOTO: BRAD DECECCO

Anne O'Neil, a chief systems engineer at the N.Y.C. Transit Authority, is one of five female senior managers in a department of 1,500.

Pay can easily hit six figures for top performers, and there's ample opportunity for advancement. But many systems engineers say they most enjoy the creative aspects of the job and seeing projects come to life. "The transit system I work on really makes a tangible difference to people," says Anne O'Neil, chief systems engineer for the New York City Transit Authority.

- **Gaining popularity in the commercial sector**
- **Not just the federal space anymore**
- **Net centric systems and the complexity they contain are primary driver**
- **Telecommunications and Unified Communications are a great example of highly complex commercial systems**



Systems Engineering Technical Short Course

(Session 1)

Appendix A:
Lesson 2 Supplementary Topics

Origins of Systems Engineering

- **Rapid growth of technology** *…. stimulated by WWII and investments in technology for military advantage*
- **Growth of Automation** *… stimulated by the advent of the digital computer and software technology*
- **Competition** *… driving innovation and more advanced system solutions …*
- **Specialization** *… partitioning complex systems into specialized components*

Each of these is discussed in a subsequent slide

Origins of Systems Engineering:

Rapid Growth of Technology

- **Radar technology**
- **Guided missile technology**
- **High performance aircraft technology**
- **Nuclear technology**

Origins of Systems Engineering:

Growth of Automation

- **Digital Computers**
- **Software Technology**
- **Air Traffic Control**
- **Computer-aided Design**
- **Computer-aided Manufacturing**
- **Industrial Automation**

Origins of Systems Engineering:

Competition

- **Innovation increases risks**
  - **Use of new technology may have *unintended consequences***
  - **Characteristics of new materials, processes and components *not well understood***
  - **Unanticipated issues may arise during system development that could**
    - **Increase costs**
    - **Delay program delivery**

- **Conservative approach also risky**
  - **Competitive advantage relies on bringing new products and technologies to market ahead of your competitors**
  - **Inferior products become obsolete quickly**

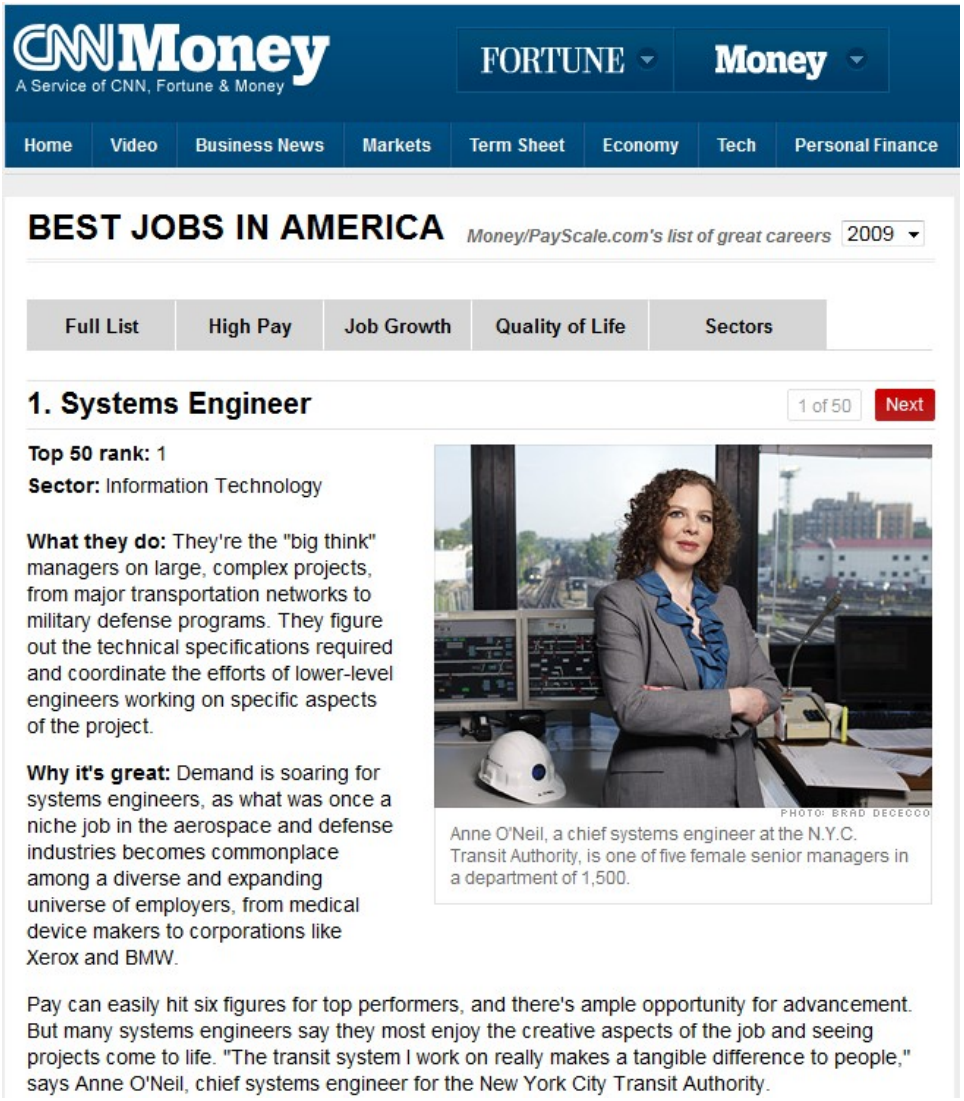
Risk Management ⇔ *the process of dealing with calculated risks through a process of analysis, development, testing and engineering oversight*

Origins of Systems Engineering:

Specialization

- **Partitioning complex systems into modular, disparate building blocks that work together to form an efficient operational system requires:**
  - **Modularity** ***… a measure of the degree of mutual independence of the individual system building blocks***
  - **Decomposition techniques** ***… to form highly modular functional and physical building blocks that can be re-used efficiently.***
  - **Integration processes** ***… which ensure that the disparate building blocks "fit' together perfectly to provided the desired response required for the operational system***
  - **Interfaces** ***… which define boundaries between the modular components***
  - **Interactions** ***… which define the behaviors between and among modular components***

Systems Engineering as a Profession



CNNMoney
A Service of CNN, Fortune & Money

FORTUNE | Money

Home | Video | Business News | Markets | Term Sheet | Economy | Tech | Personal Finance

BEST JOBS IN AMERICA *Money/PayScale.com's list of great careers* 2009

Full List | High Pay | Job Growth | Quality of Life | Sectors

1. Systems Engineer 1 of 50 Next

Top 50 rank: 1

Sector: Information Technology

What they do: They're the "big think" managers on large, complex projects, from major transportation networks to military defense programs. They figure out the technical specifications required and coordinate the efforts of lower-level engineers working on specific aspects of the project.

Why it's great: Demand is soaring for systems engineers, as what was once a niche job in the aerospace and defense industries becomes commonplace among a diverse and expanding universe of employers, from medical device makers to corporations like Xerox and BMW.

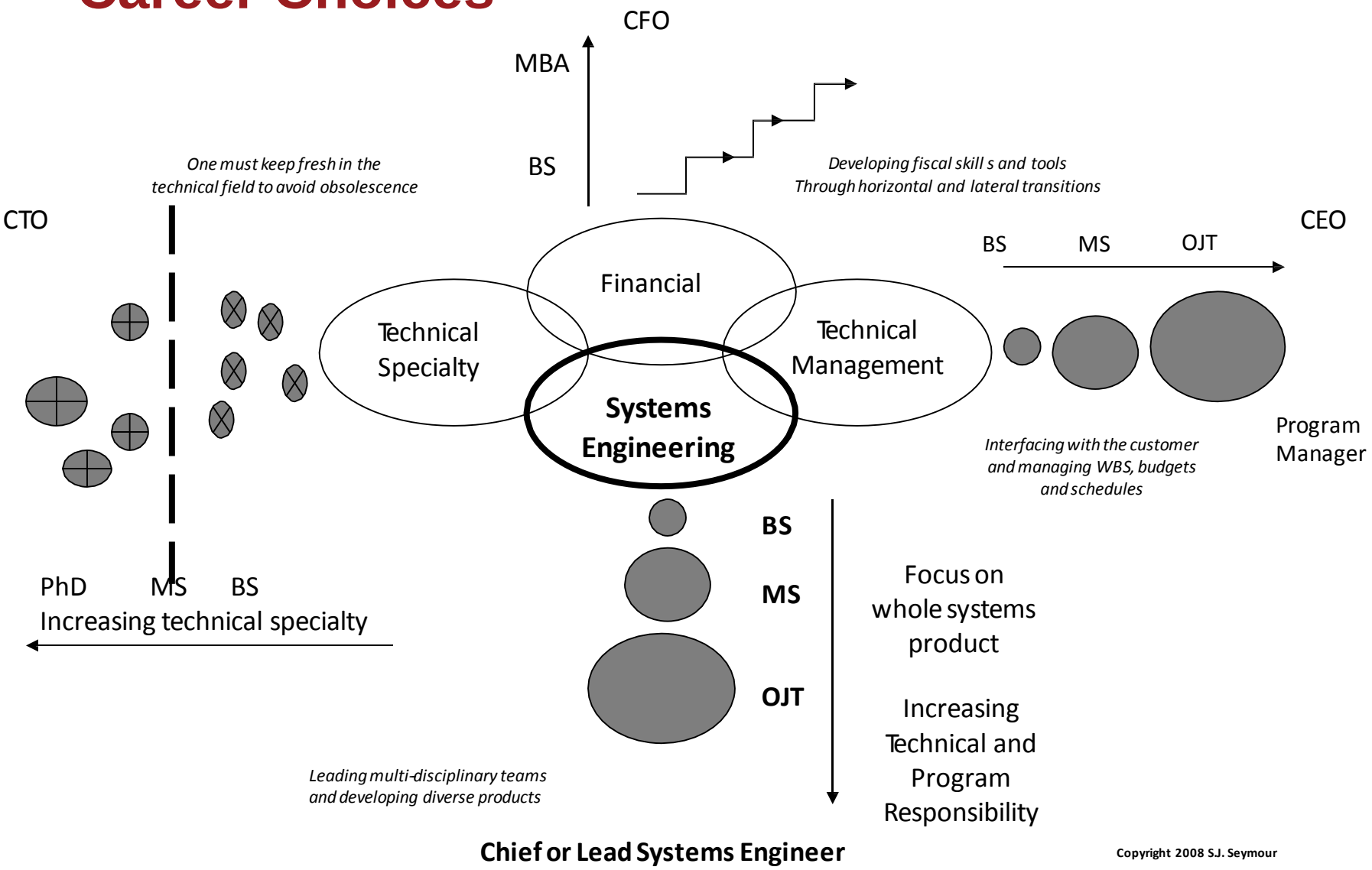
PHOTO: BRAD DECECCO

Anne O'Neil, a chief systems engineer at the N.Y.C. Transit Authority, is one of five female senior managers in a department of 1,500.

Pay can easily hit six figures for top performers, and there's ample opportunity for advancement. But many systems engineers say they most enjoy the creative aspects of the job and seeing projects come to life. "The transit system I work on really makes a tangible difference to people," says Anne O'Neil, chief systems engineer for the New York City Transit Authority.

- **Gaining popularity in the commercial sector**
- **Not just the federal space anymore**
- **Net centric systems and the complexity they contain are primary driver**
- **Telecommunications and Unified Communications are a great example of highly complex commercial systems**

Career Choices

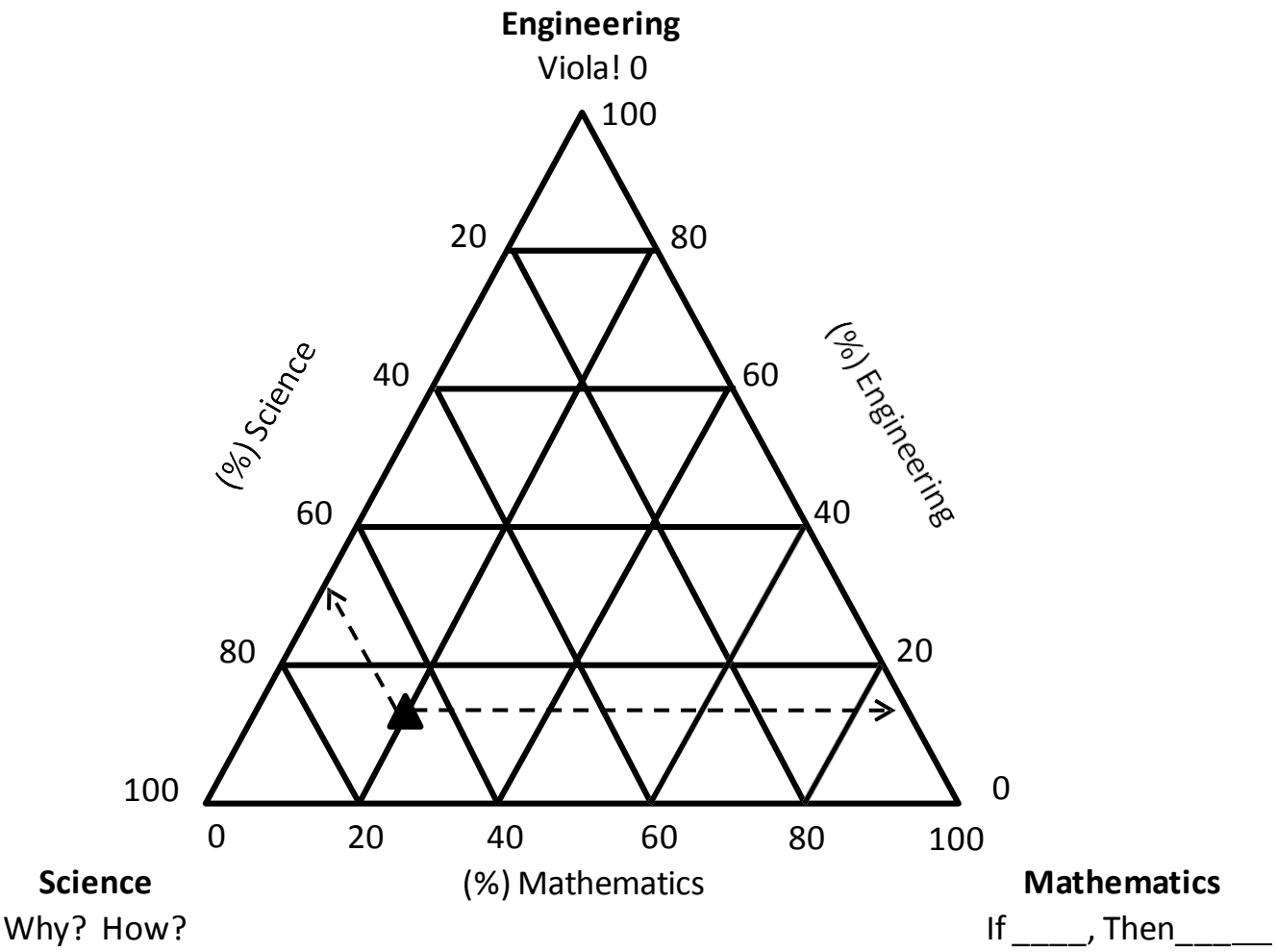


Copyright 2008 S.J. Seymour

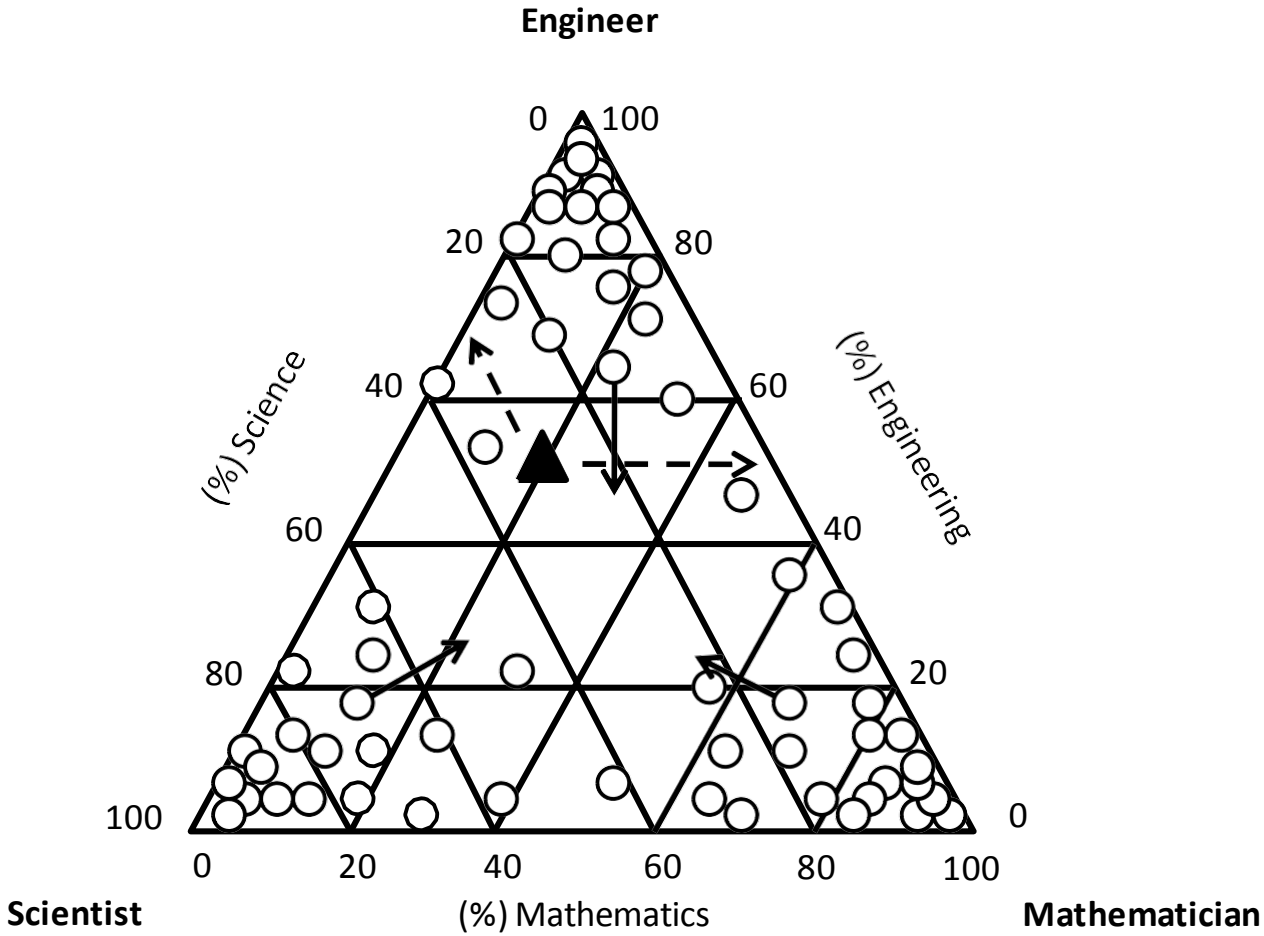
Orientation of Technical Professionals

- The Engineer – Viola!
- The Mathematician – IF A, THEN B
- The Scientist – Why? How?
- Polarization (competitive nature) into distinct disciplines exacerbates the need for systems engineers
- Certain unique individuals resist polarization, cross the boundaries and enjoy taking on the bigger and more complex problem

Technical Orientation Phase Diagram



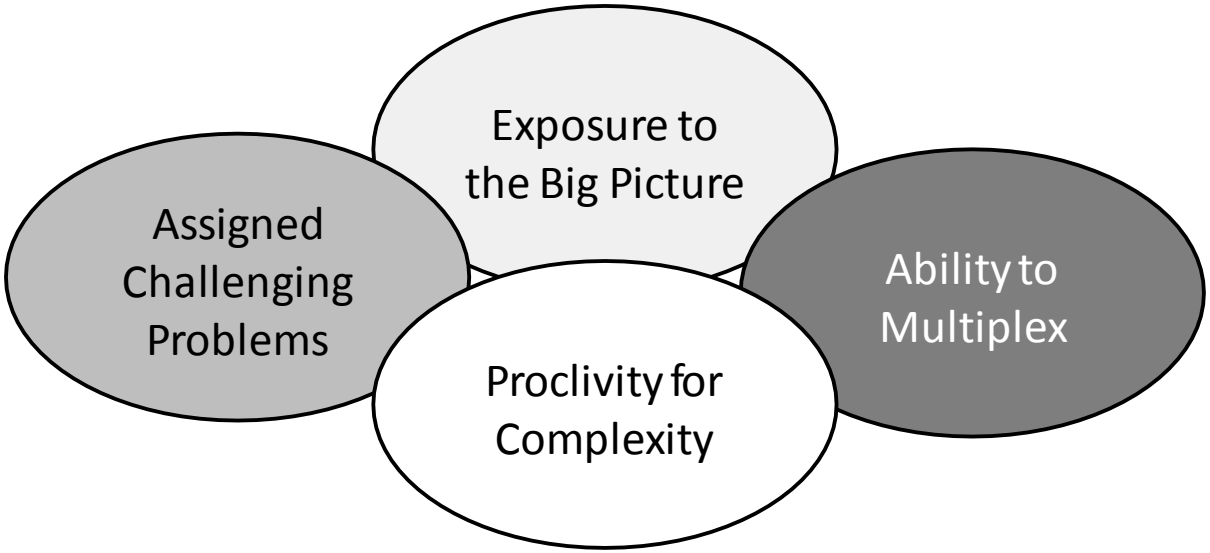
Technical Orientation Population Density Distribution



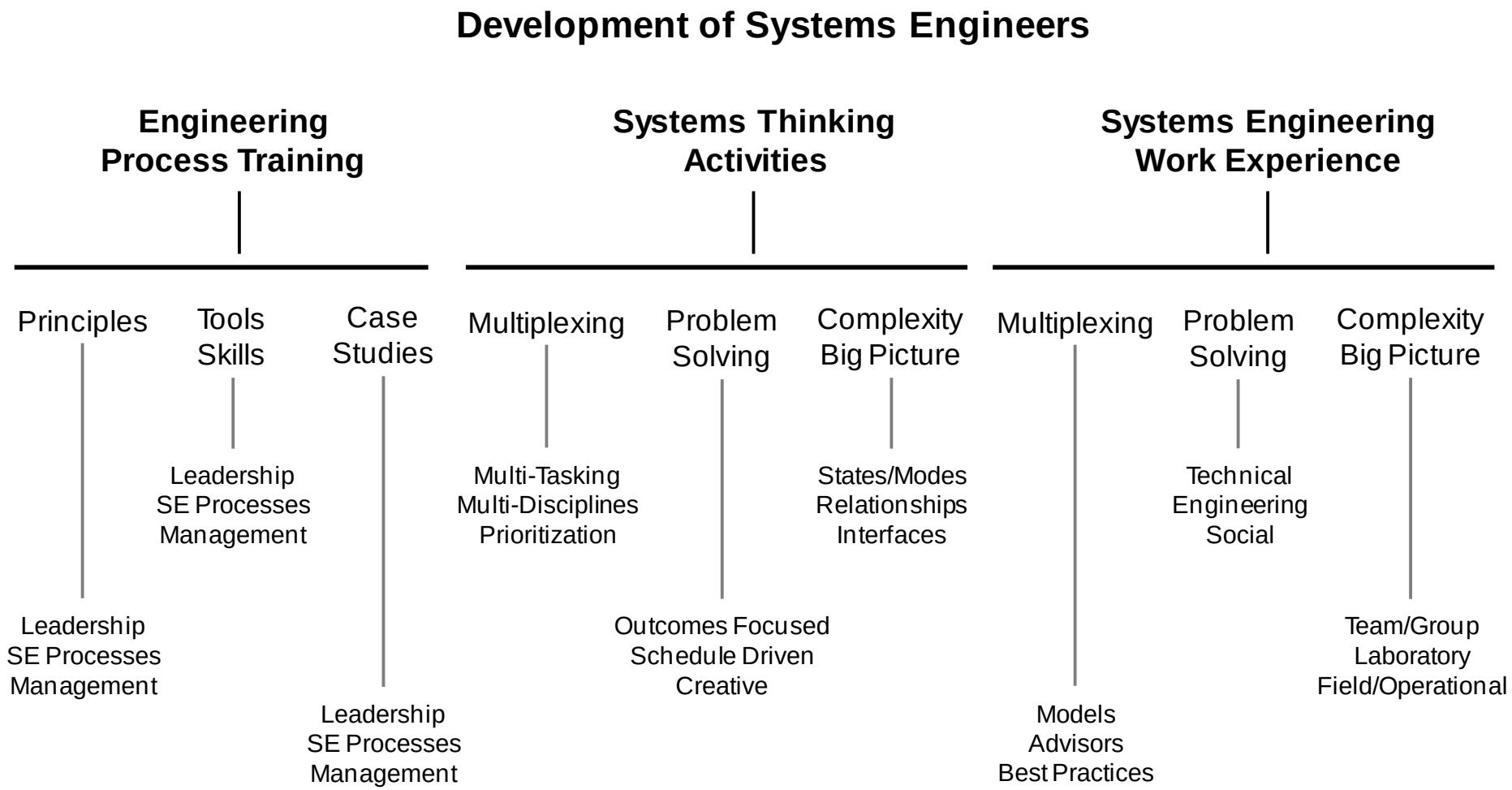
Attributes and Motivations

- Enjoy learning new things and solving problems
- Like challenges
- Skeptical of unproven assertions
- Open minded to new ideas
- A solid background in science and engineering
- Demonstrated technical achievement in a specialty area
- Knowledgeable in several engineering areas
- Pick up new ideas and information quickly
- Good interpersonal and communication skills

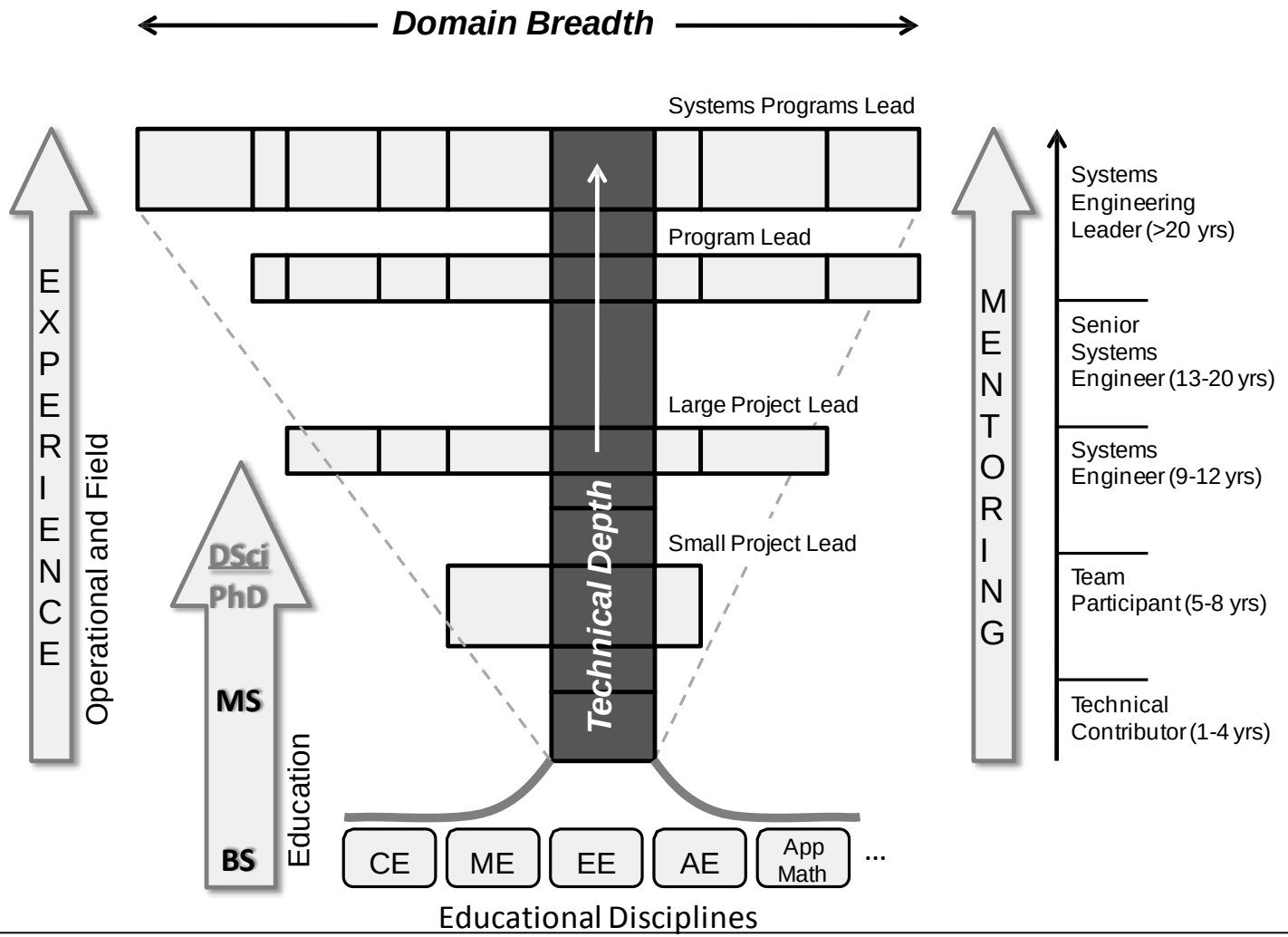
Career Development Model



Career Development Components



T Model for Systems Engineering Career Development



INCOSE (www.incose.org)

The International Council on Systems Engineering



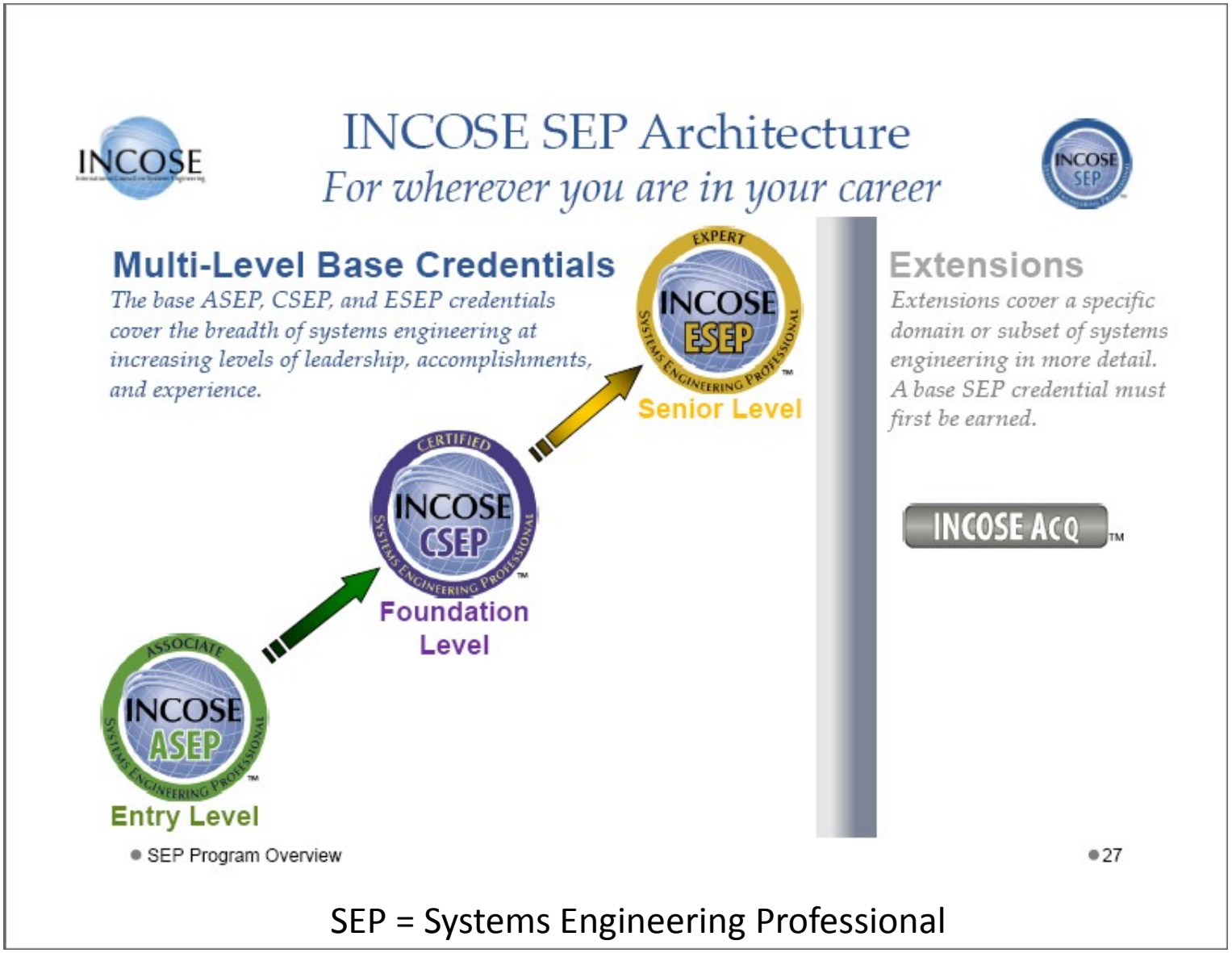
INCOSE Membership

- **Regular Membership dues are $135**
- **Student Membership dues are $35**
  - **Members qualify for the Student category if their course load is at least 3/4 of full time in an engineering or related field**
- **Systems Engineering Handbook**
  - **WPI Student and Members may download a softcopy at *no charge***
  - **As a privilege you have access to the INCOSE INSIGHT Library and Products Area. The link to the site is https://connect.incose.org/products/default.aspx**

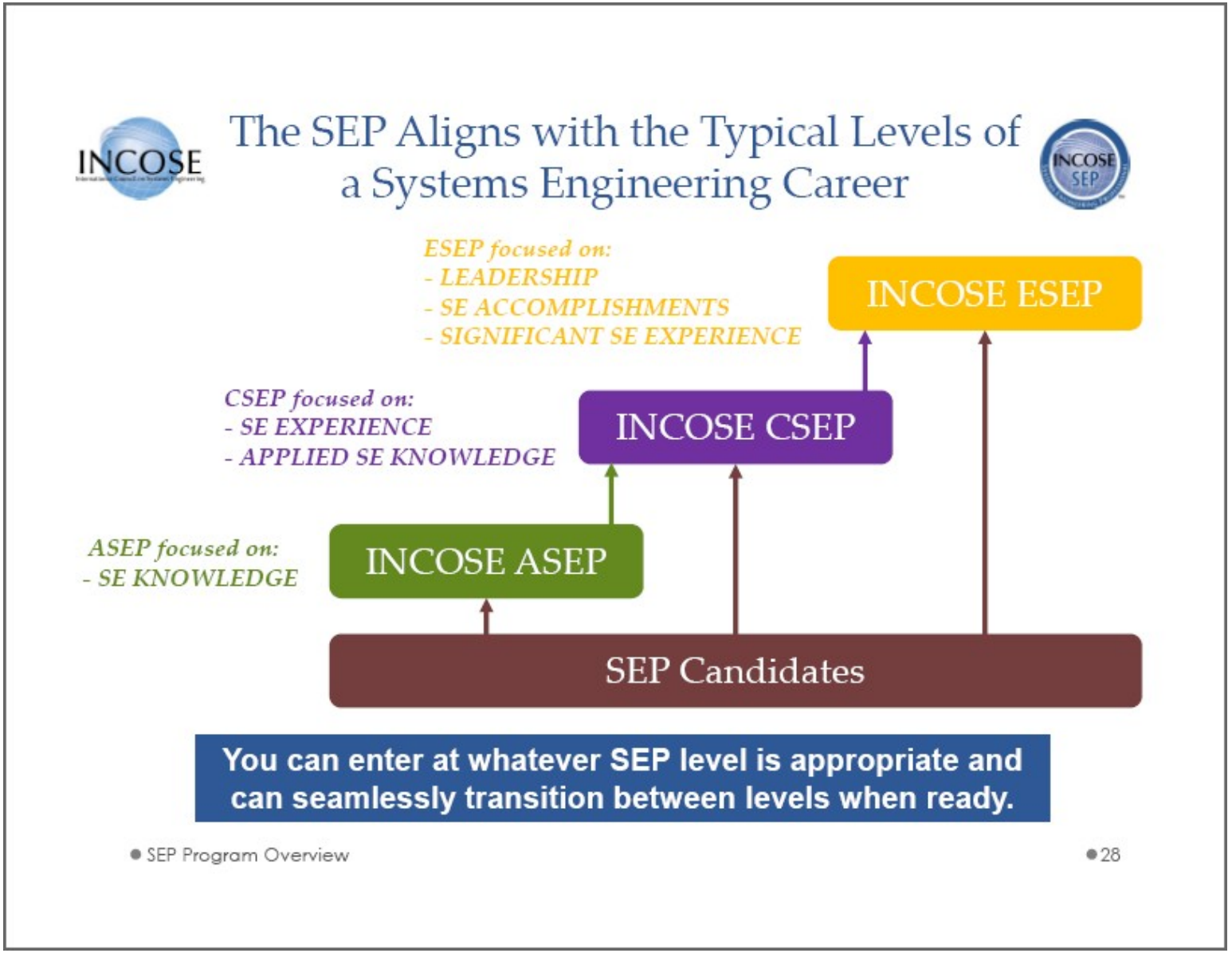
    The username is: worcesterpolytechnic

    The password is: wpicpe

*** All prices are as of June 2013**



SEP = Systems Engineering Professional


The SEP Aligns with the Typical Levels of a Systems Engineering Career
ESEP focused on:
- LEADERSHIP
- SE ACCOMPLISHMENTS
- SIGNIFICANT SE EXPERIENCE
INCOSE ESEP
CSEP focused on:
- SE EXPERIENCE
- APPLIED SE KNOWLEDGE
INCOSE CSEP
ASEP focused on:
- SE KNOWLEDGE
INCOSE ASEP
SEP Candidates
You can enter at whatever SEP level is appropriate and can seamlessly transition between levels when ready.
SEP Program Overview
28

Acq Extension

US DoD Acquisition Extension

- Targeted towards systems engineers who support or work in a US Department of the Defense acquisition environment
- Candidates must first become an ASEP, CSEP, or ESEP
- Acquisition knowledge items tested through both the core ASEP/CSEP exam and an additional Acq exam (Note: ESEPs must also take both exams)
- Extension renewed concurrently with the corresponding base certification
- Available since 2008

INCOSE ACQ™



Systems Engineering Technical Short Course

(Session 1)

Lesson 3: Structure and Environment of Systems

Robert B. Wheeler, Instructor

Reference : "Systems Engineering, Principles and Practice," Kossiakoff et. al – Chapter 3

Session 1 Agenda (Jan 27 – 29): Timing

Day	Overview and Concepts of Systems Engineering	Hours*
1	Waters Management kick-off	1/2
	Introduction and overview of entire program (Lesson 1), including Introduction and overview of the projects (15 minutes each)	2
	Presentation and discussion of Concepts of SE (Lessons 2 – 9)	4
	Project assignments, team formations, and working sessions to apply SE concepts to the projects	1
	Day 1 wrap-up: Takeaways and Q&A	1/2
2	Review of day 1 working sessions	1
	Concepts of Systems Engineering (concluded) (Lesson 10 – 14)	4
	Project team working sessions (continued)	2
	Day 2 wrap up and Q&A	1
3	Team working session feedback to class (informal presentations)	1
	Project team working sessions (concluded)	5
	Check-in with class on application of SE concepts to their projects	1
	Session 1 wrap up: Q&A and homework due next session	1

* Times include breaks, and may be changed based on needs of the class

Concepts of Systems Engineering

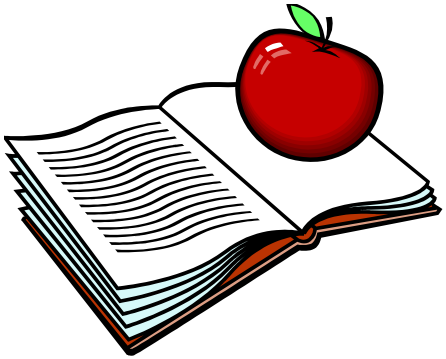
1. Program Overview
2. What is Systems Engineering?
3. Structure and Environment of Systems
4. Systems Development Process
5. Systems Engineering Processes Overview
6. Systems Engineering in Needs Assessment
7. Risk Management
8. Concept Exploration
9. Concept Definition
10. Advanced Development Stage
11. Engineering Design Stage
12. Integration and Evaluation Stage
13. Production and Deployment
14. Operations, Support, and Disposal

Plus Appendices

Lesson Content:

- What is a system?
- System hierarchy
- System building blocks
- Interfaces
- Context diagrams

- Exercise: Context diagram



This slide intentionally left blank

What Is a System?



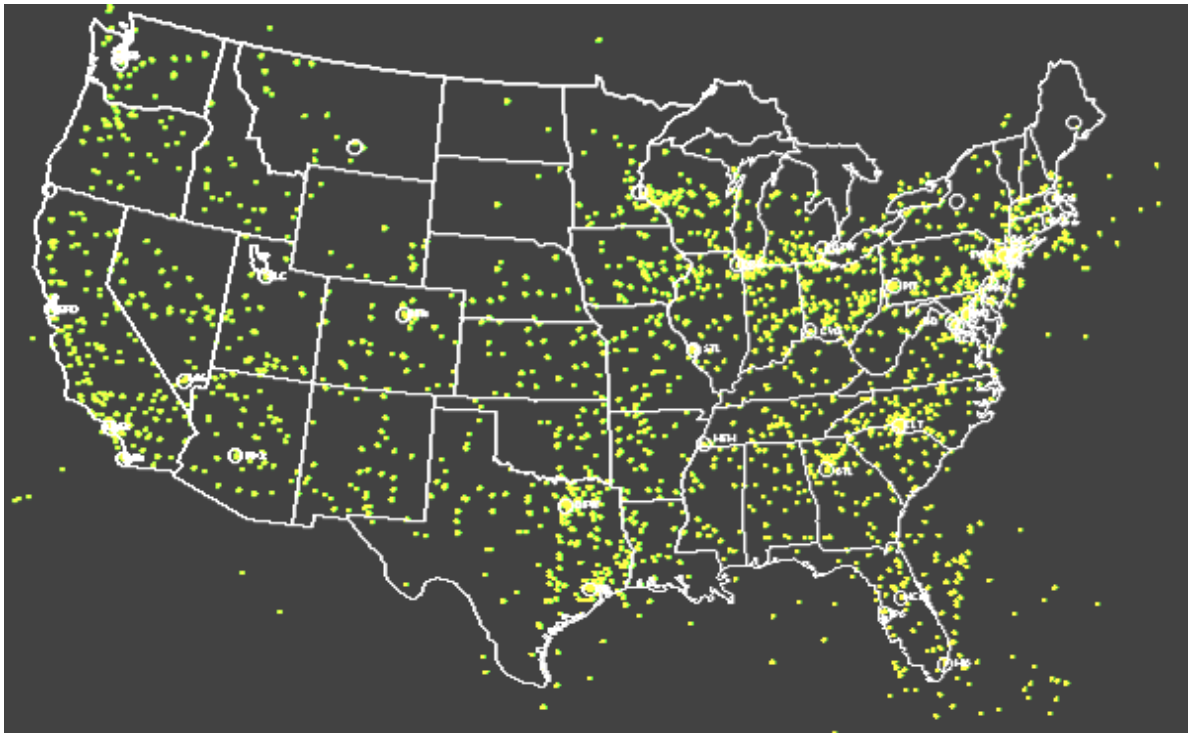




What Is a System?

Air Traffic Control System

National



Local / Regional







What Is a System?

AT&T Network Control Center



What Is a System?

NORAD - Command Post, Missile Warning, Space Control, Air Operations, and Support Centers



Combined Command Center (CCG)



Air Operations Center



Space Control



CCC



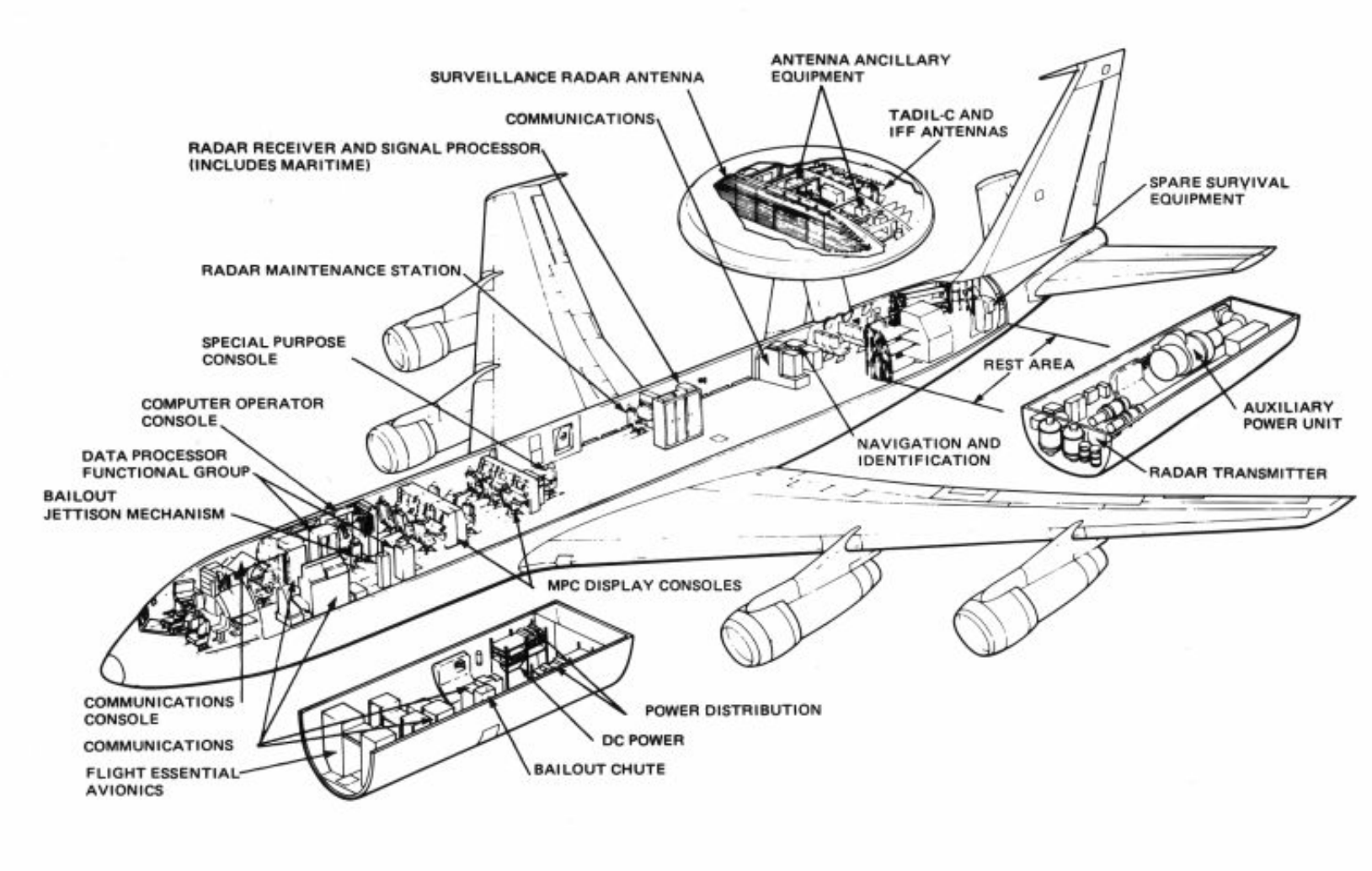
Missile Warning Center



Tech Control Center

What Is a System?

E-3A - AWACS



What Is a System?

Waters Functional Model

Posted slide is blank

System Hierarchy – A Taxonomy

- **System:** An entity which performs a significant useful service
- **Subsystems:** A major interacting elements of a system, each of which perform a subset of closely related system functions
- **Components:** The middle level of system elements which comprise a subsystem (Configuration Items)
- **Subcomponents:** Perform elementary functions and are composed of several parts
- **Parts:** The smallest primitive elements of a subcomponent

In this course, the term “entity” is used to refer to any level in the hierarchy

System Hierarchy Examples

Systems			
Communications Systems	Information Systems	Material Processing Systems	Aerospace Systems

Subsystems			
Signal Networks	Databases	Material Preparation	Engines

Components					
Signal Receivers	Data Displays	Database Programs	Power Transfer	Material Reactors	Thrust Generators

Subcomponents					
Signal Amplifiers	Cathode Ray Tubes	Library Utilities	Gear Trains	Reactive Valves	Rocket Nozzles

Parts					
Transformer	LED	Algorithms	Gears	Couplings	Seals

Knowledge Domains

- **Systems Engineer**
  - Extends across many systems…
  - …through multiple subsystems…
  - … to "systems-of-systems"…
  - … to key characteristics of and specifications for components
    - Interoperability
    - Performance
    - Reliability, etc.

- **Design Specialist**
  - Extends from the lowest level (parts) to the component level
  - Adapts components and sub-components to specific applications
  - Guided by specifications provided by the systems engineer

System Building Blocks:

The Components

- **Functional Building Blocks** – entities which constitute the functional properties of a system and can be classified as one of the following function elements:
  - **Signal** Elements *… which sense and communicate information*
  - **Data** Elements *… which interpret, organize and manipulate information*
  - **Material** Elements *… which provide structure and process material*
  - **Energy** Elements *… which provide energy and power*
- **Physical Building Blocks** - physical instantiations of functional building blocks, composed of hardware and/or software

Functional Building Blocks

Class Function	Element Function	Applications
Signal— generate, transmit, distribute and receive signals used in passive or active sensing and in communications	Input signal Transmit signal Transduce signal Receive signal Process signal Output signal	TV camera FM radio transmitter Radar antenna Radio receiver Image processor
Data— analyze, interpret, organize, query and/or convert data and information into forms desired by the user or other systems	Input data Process data Control data Control processing Store data Output data Display data	Keyboard Computer CPU Operating system Word processor Printer
Material— provide system structural support or enclosure, or transform the shape, composition or location of material substances	Support material Store material React material Form material Join material Control position	Airframe Shipping container Autoclave Milling machine Welding machine Servo actuator
Energy— provide and convert energy or propulsive power to the system	Generate thrust Generate torque Generate electricity Control temperature Control motion	Turbojet engine Reciprocating engine Solar cell array Refrigerator Auto transmission

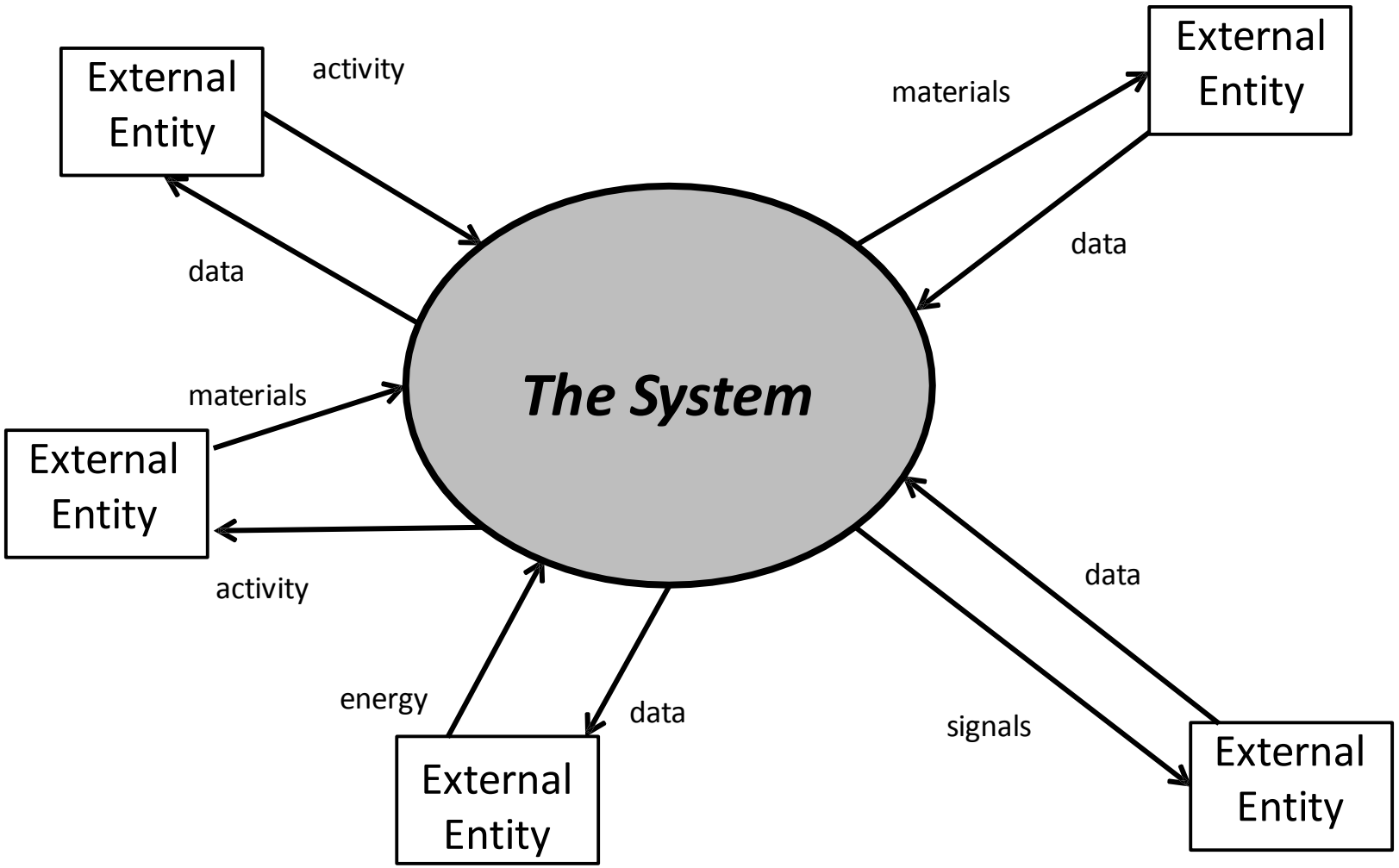
Physical Building Blocks

Category	Component	Functional Element(s)
Electronic	Receiver Transmitter Data processor Signal processor Communications processors Special electronic equipment	Receive signal Transmit signal Process data Process signal Process signal/data Various
Electro-optical	Optical sensing device Optical storage device Display device High energy optics device Optical power generator	Input signal Store data Output signal/data Form material Generate electricity
Electromechanical	Inertial instrument Electric generator Data storage device Transducer Data input/output device	Input data Generate electricity Store data Transduce signal Input/output data
Mechanical	Framework Container Material processing machine Material reactor Power transfer device	Support material Store material Form/join material React material Control motion
Thermomechanical	Rotary engine Jet engine Heating unit Cooling unit Special energy source	Generate torque Generate thrust Control temperature Control temperature Generate electricity
Software	Operating system Application Support software Firmware	Control system Control processing Control processing Control system

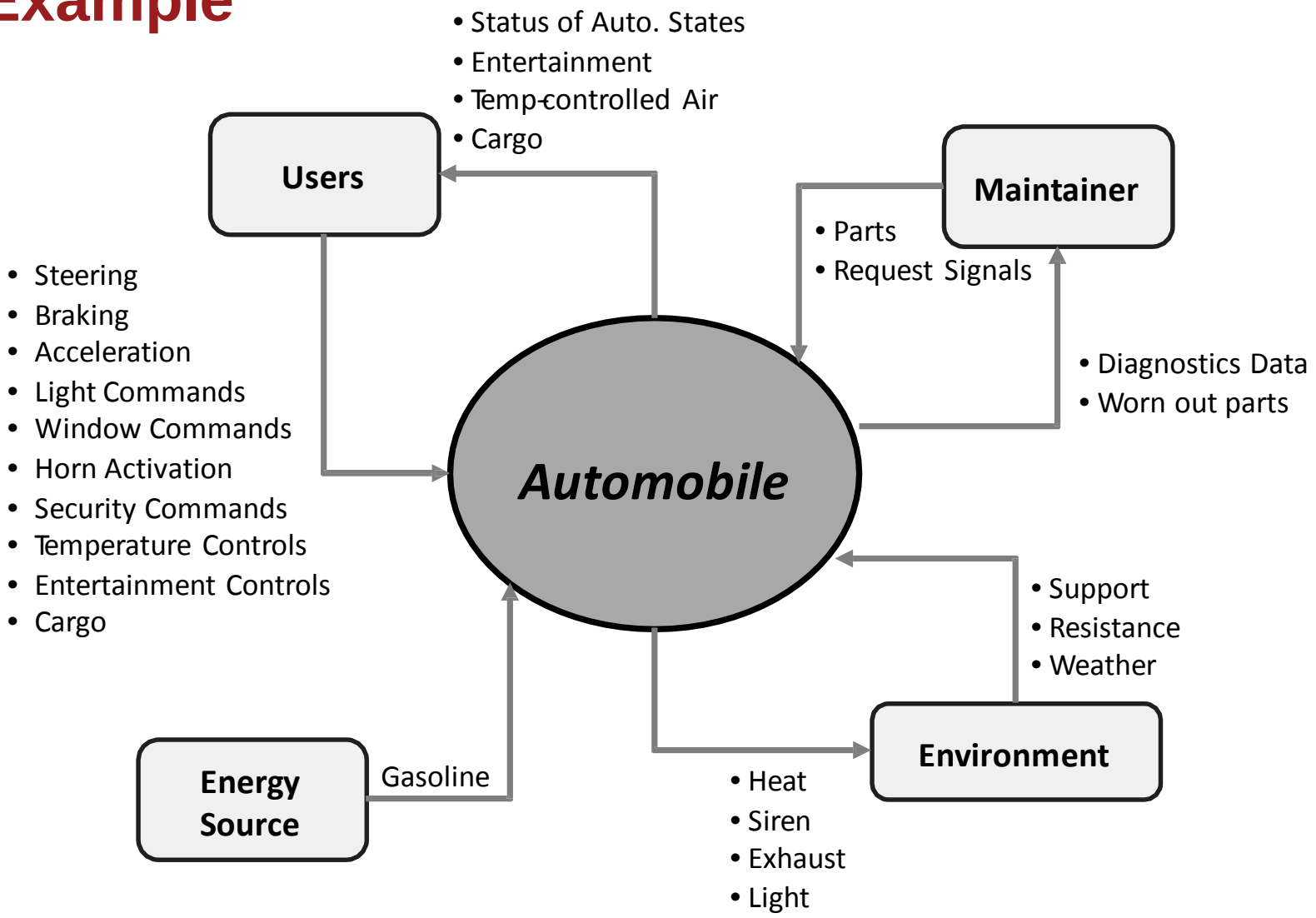
Environment of a System

- **Boundaries** *… defines that which is inside the system and that which is outside the system.*
  - **Inside – the product that is to be developed and delivered**
  - **Outside – the environment in which the system operates, including its operators**
  - **Interfaces – provide the mechanisms by which the system can interact with its environment**
- **(Boundaries of truly complex systems may be difficult to define)**
- **Environmental Interactions** *… define the interfaces and nature of interacts between the system and its surroundings.*
- **(Interactions of truly complex systems are key)**

Context Diagram



Context Diagram – An Automobile Example

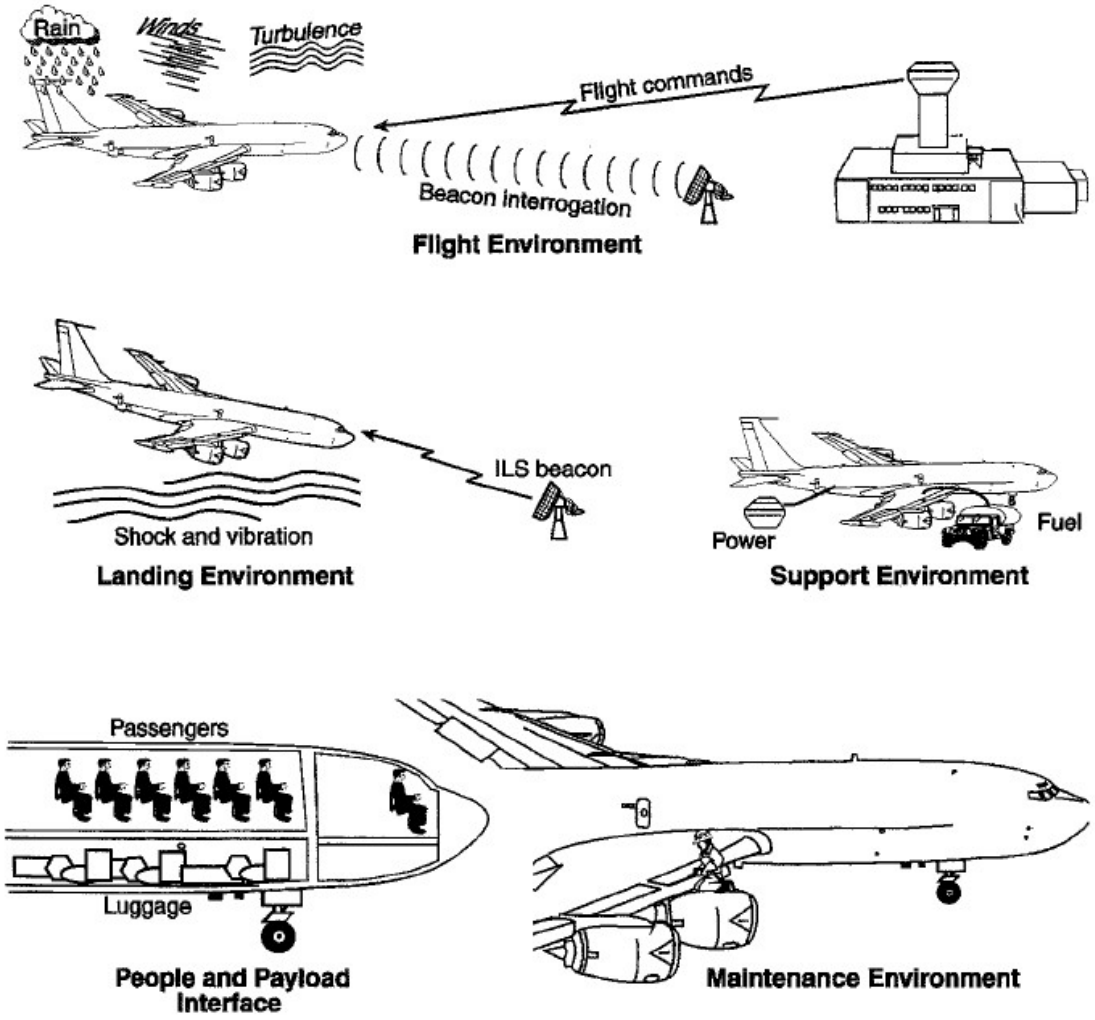


Environment of a System

Examples of Environmental Interactions

- **Inputs and Outputs**
- **System Operators**
- **Operational Maintenance**
- **Support Systems**
- **System Housing**
- **Shipping and Handling**

Environment of a System: An Example



Environments of a passenger airliner.

Interfaces and Interactions

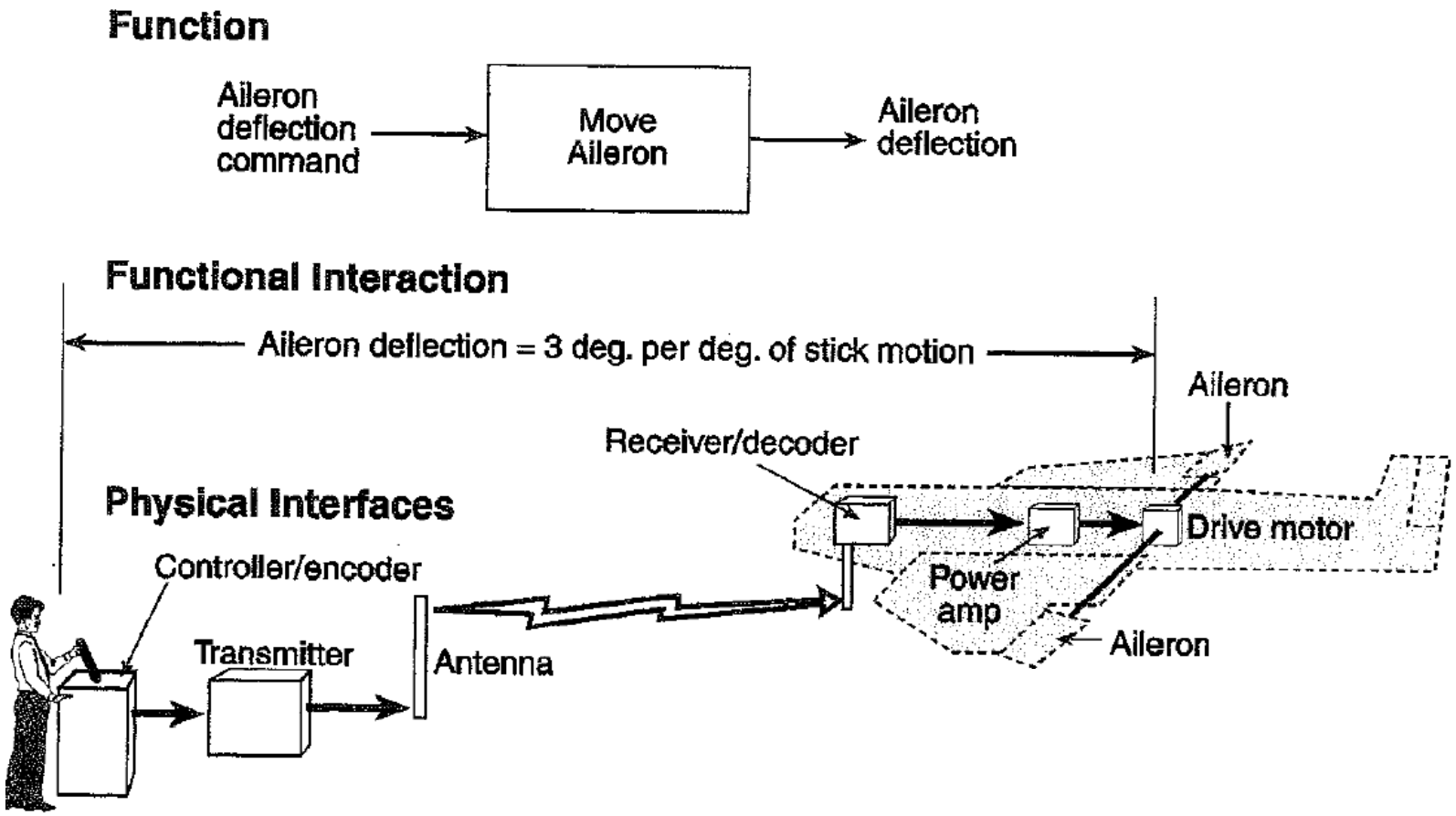
- **Interfaces** *… are defined at system boundaries*
  - **Internal** *… the boundaries between individual system components*
  - **External** *… the boundaries between the system and its environment*
- **Interactions** *… communications between the system, its environment and/or components enabled by its interfaces*
- **Interface Elements** *… categorized as either*
  - **Connectors** *… which facilitate the transmission of data, power, force, etc., between components*
  - **Isolators** *… which inhibit interactions across an interface*
  - **Converters** *… which alter or transform interaction medium*

Interface Elements: An Example

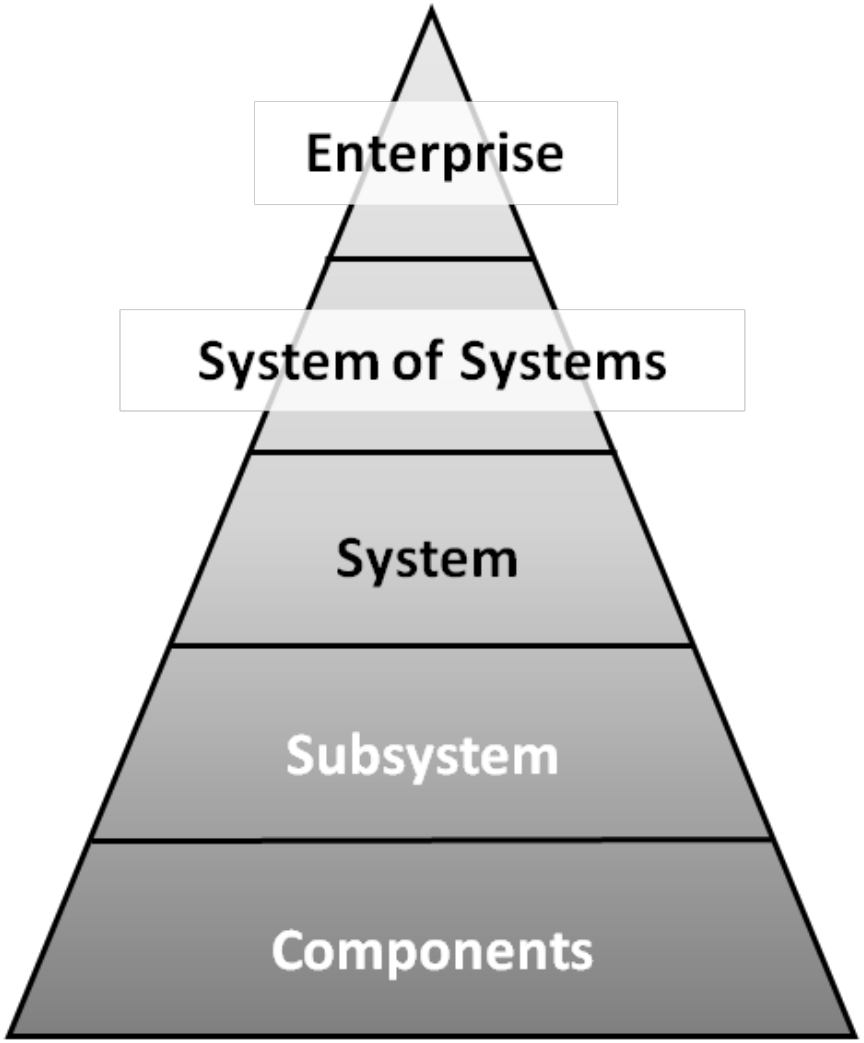
Type	Electrical	Mechanical	Hydraulic	Human-Machine
Interaction medium	Current	Force	Fluid	Information
Connectors	Cable Switch	Joint coupling	Pipe valve	Display Control panel
Isolator	RF shield insulator	Shock mount bearing	Seal	Cover Window
Converter	Antenna A/D converter	Gear train piston	Reducing valve Pump	Keyboard

Physical Interfaces and Functional Interactions

An Example: Radio-Controlled Air Vehicle



Enterprise Systems Engineering



Summary

- Complex systems can be represented as a hierarchical structure composed of things like subsystems, components, subcomponents, and parts
- System building blocks are components which are
  - Characterized by both physical and functional attributes
  - Distinct and perform a significant function
  - Specialized within the scope of a single engineering discipline
  - Common and re-usable across a variety of systems
- Functional elements are the functional equivalent of components and can be categorized into four classes of operating media:
  - Signal Elements, which sense and communicate information
  - Data Elements, which interpret, organize and manipulate information
  - Material Elements, which provide structure and process material
  - Energy Elements, which provide energy and power

Summary (continued)

- **Components are physical realizations of functional elements**
- **Interfaces are a critical systems engineering consideration, which**
  - **Effect interactions between and among components**
  - **Must be identified, specified coordinated and controlled**
  - **Must be tested to ensure smooth integration and ease of maintenance**

Summary (concluded)

- The system environment is everything outside the system which interacts with it, including:
  - System operators
  - Maintenance and support systems
  - Shipping, storage and handling
  - Weather and other physical environments

- Context Diagrams are helpful in identifying and understanding system context including:
  - The environment(s) in which the system will exist through its lifecycle
  - System interfaces
  - System boundaries
  - System stakeholders

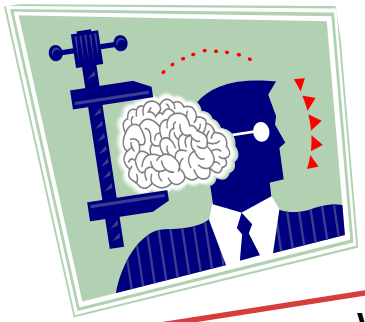
Exercise: Project Context Diagram



Instructor will provide direction for the exercise

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



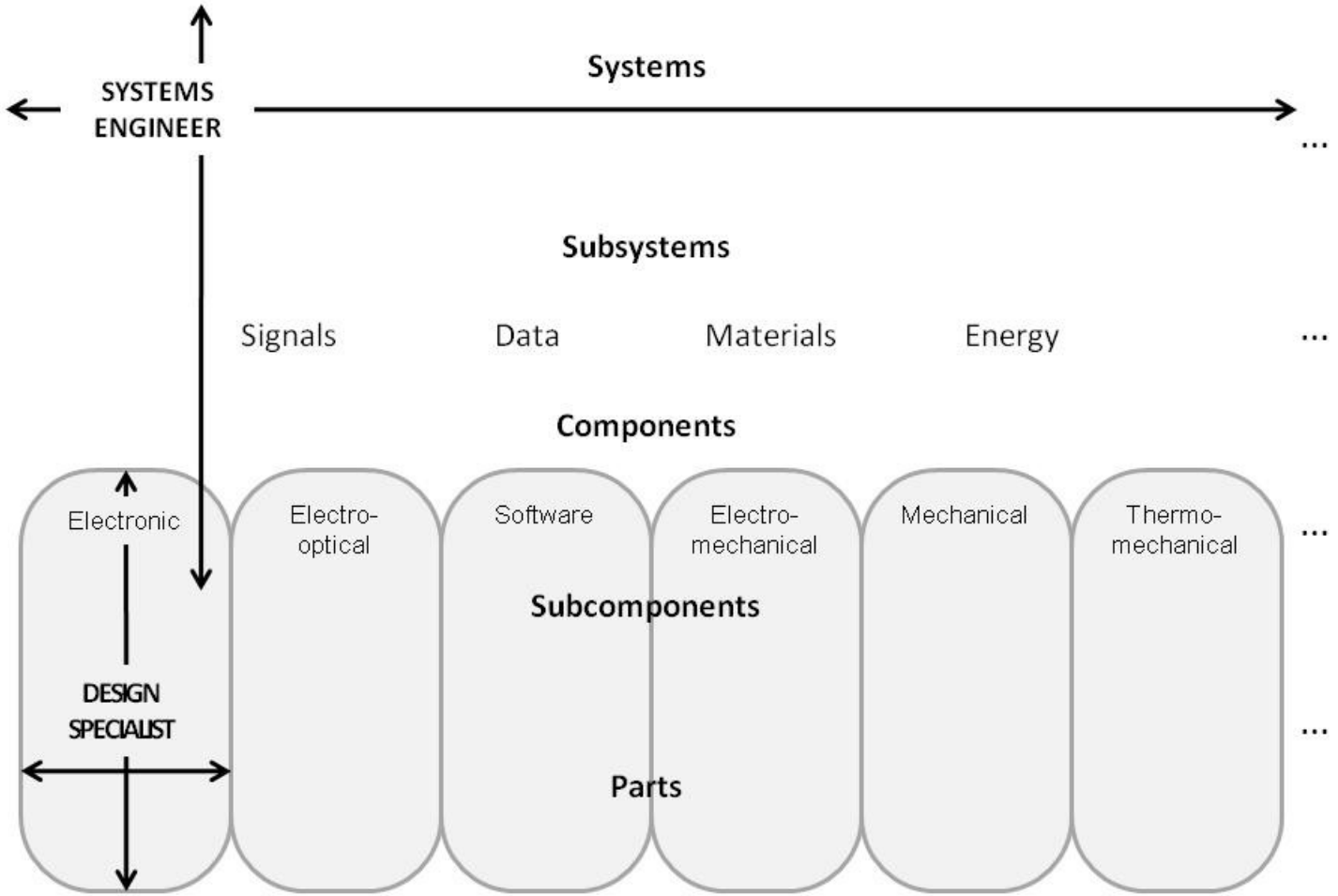
All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?

Back-up Slides

Knowledge Domains: An Example



Systems Engineering Technical Short Course

(Session 1)

Lesson 5:
Systems Engineering Processes Overview

Reference : "Systems Engineering, Principles and Practice," Kossiakoff et. al – Chapter 4

Session 1 Agenda (Jan 27 – 29): Timing

Day	Overview and Concepts of Systems Engineering	Hours*
1	Waters Management kick-off	1/2
	Introduction and overview of entire program (Lesson 1), including Introduction and overview of the projects (15 minutes each)	2
	Presentation and discussion of Concepts of SE (Lessons 2 – 9)	4
	Project assignments, team formations, and working sessions to apply SE concepts to the projects	1
	Day 1 wrap-up: Takeaways and Q&A	1/2
2	Review of day 1 working sessions	1
	Concepts of Systems Engineering (concluded) (Lesson 10 – 14)	4
	Project team working sessions (continued)	2
	Day 2 wrap up and Q&A	1
3	Team working session feedback to class (informal presentations)	1
	Project team working sessions (concluded)	5
	Check-in with class on application of SE concepts to their projects	1
	Session 1 wrap up: Q&A and homework due next session	1

* Times include breaks, and may be changed based on needs of the class

Concepts of Systems Engineering

1. Program Overview
2. What is Systems Engineering?
3. Structure and Environment of Systems
4. Systems Development Process
5. Systems Engineering Processes Overview
6. Systems Engineering in Needs Assessment
7. Risk Management
8. Concept Exploration
9. Concept Definition
10. Advanced Development Stage
11. Engineering Design Stage
12. Integration and Evaluation Stage
13. Production and Deployment
14. Operations, Support, and Disposal

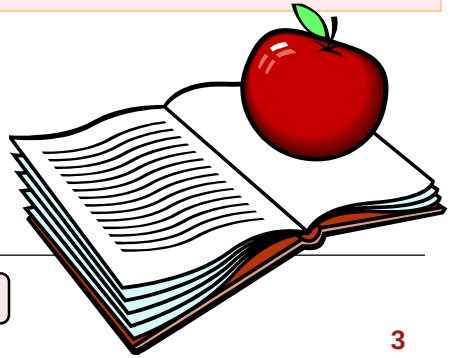
Plus Appendices (See next slide)

Lesson Content:

- An enterprise process model
- SE Process identification
- Requirements process overview
- Pointer to where the processes are covered in this program

Exercise: Waters Processes

Appendix: ~~TBD~~ D



The course will reference, but will not cover, the Appendices

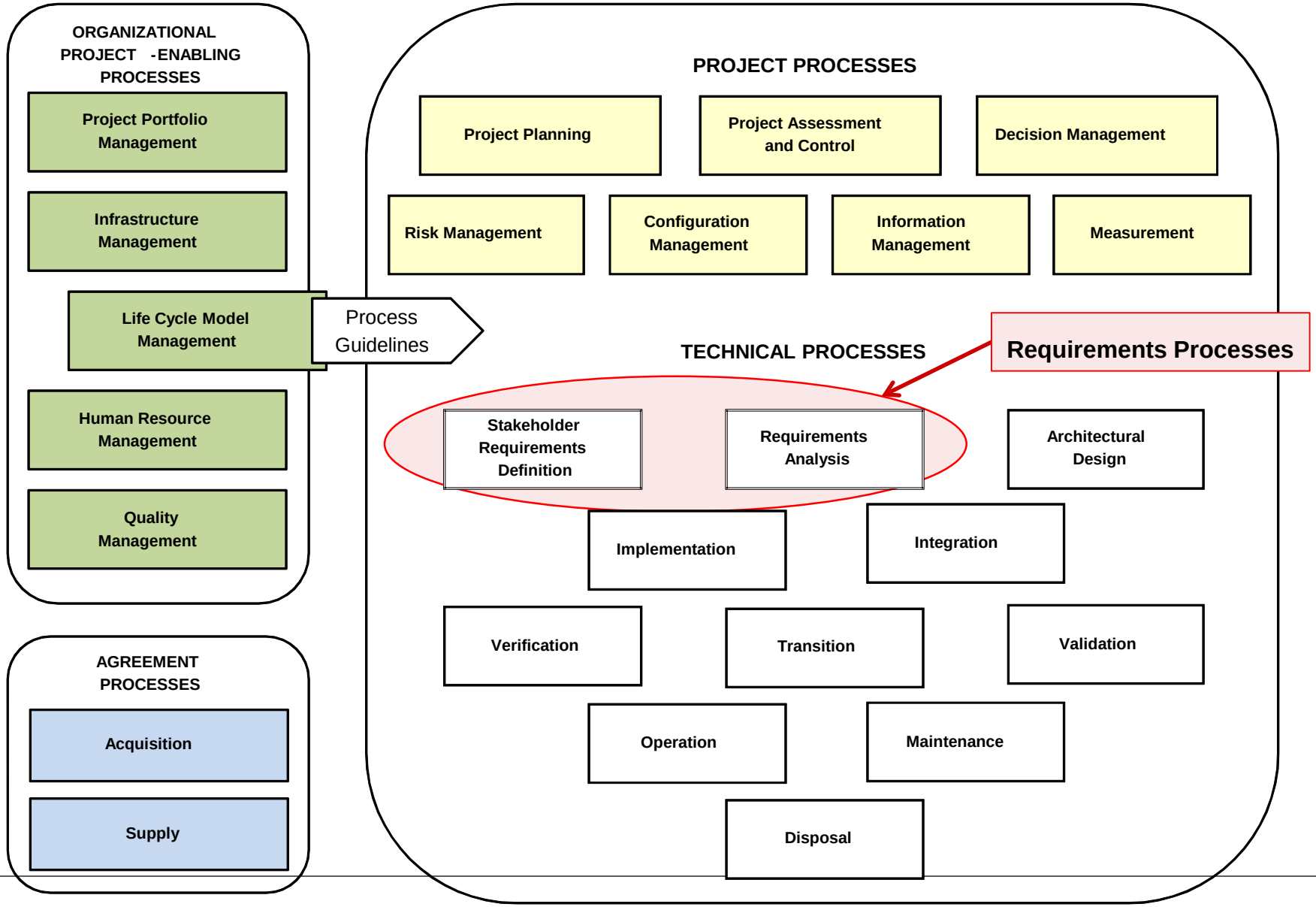
Lesson Overview

- **This lesson**
  - **Introduces the concept of an enterprise process model, and provides one illustrative model (INCOSE/ISO 15288).**
  - **Identifies the processes to be discussed throughout this class**
  - **Provides a very brief overview of those processes that will be covered in depth in sessions 2 through 5 of this course**
    - **Requirements development**
    - **Baseline management (to be provided)**
    - **Communication (to be provided)**
  - **Identifies and points to other processes that are covered elsewhere in this session 1**

An Enterprise Process Model

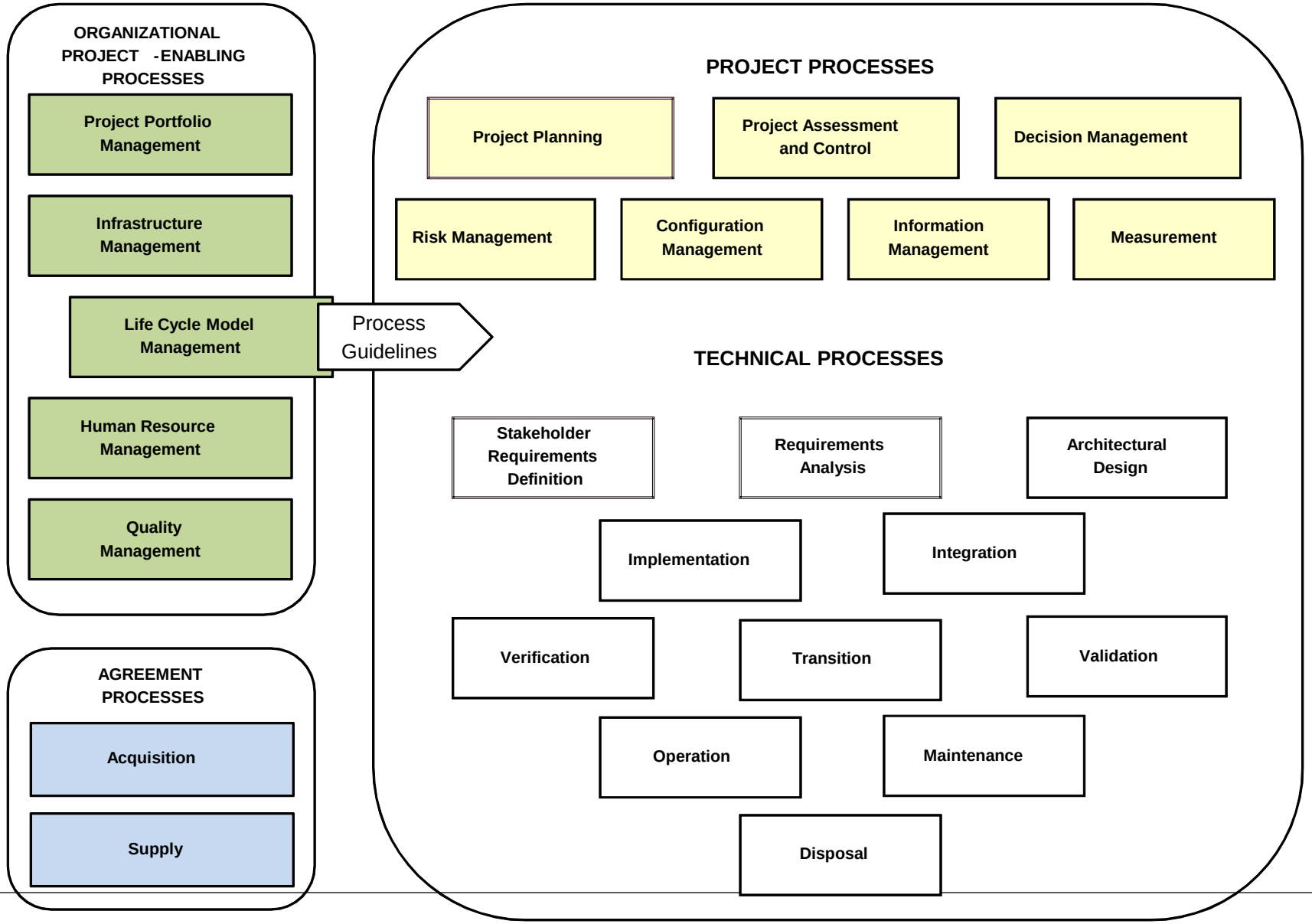
- Every Enterprise or organization has a way of doing business, whether documented or formalized in some manner, or not
- The next slide illustrates an enterprise process model from ISO15288 via the INCOSE Handbook
- The model consists of:
  - Enterprise enabling processes (Upper left), one of which is a overall Lifecycle Management Process
  - The Lifecycle Management Process, which identify and manage The technical and management processes used for system development projects (right side)
  - Agreement processes (lower left) for acquiring from, or supplying to, other organizations
- An enterprise needs to make “conscious decisions” about the degree to which they formalize and manage these processes

INCOSE SE Process Model based on ISO/IEC 15288 – 2008



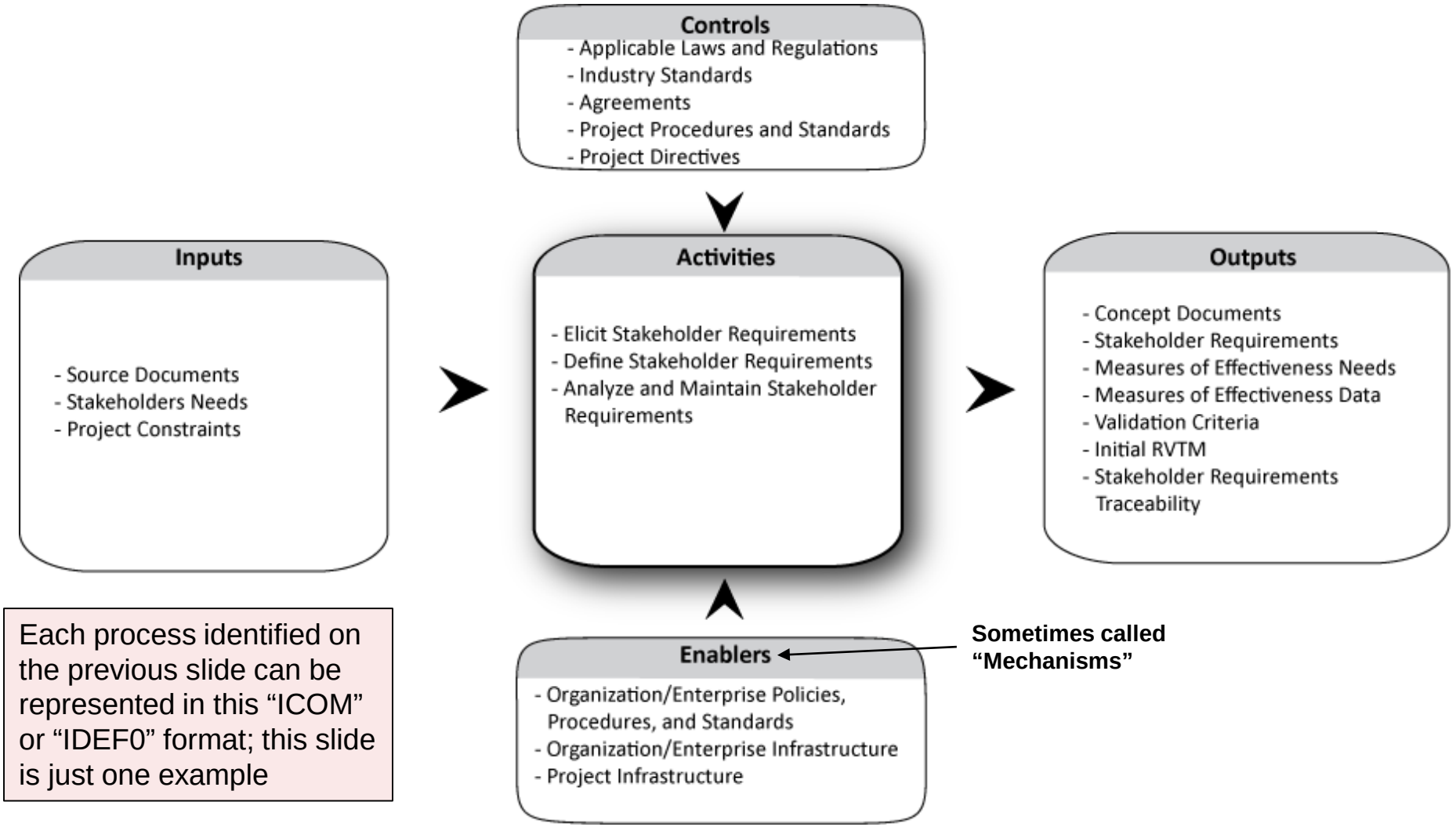
Source: INCOSE Systems Engineering handbook, version 3.2.2, Figure 1-1

INCOSE SE Process Model based on ISO/IEC 15288 – 2008



Source: INCOSE Systems Engineering handbook, version 3.2.2, Figure 1-1

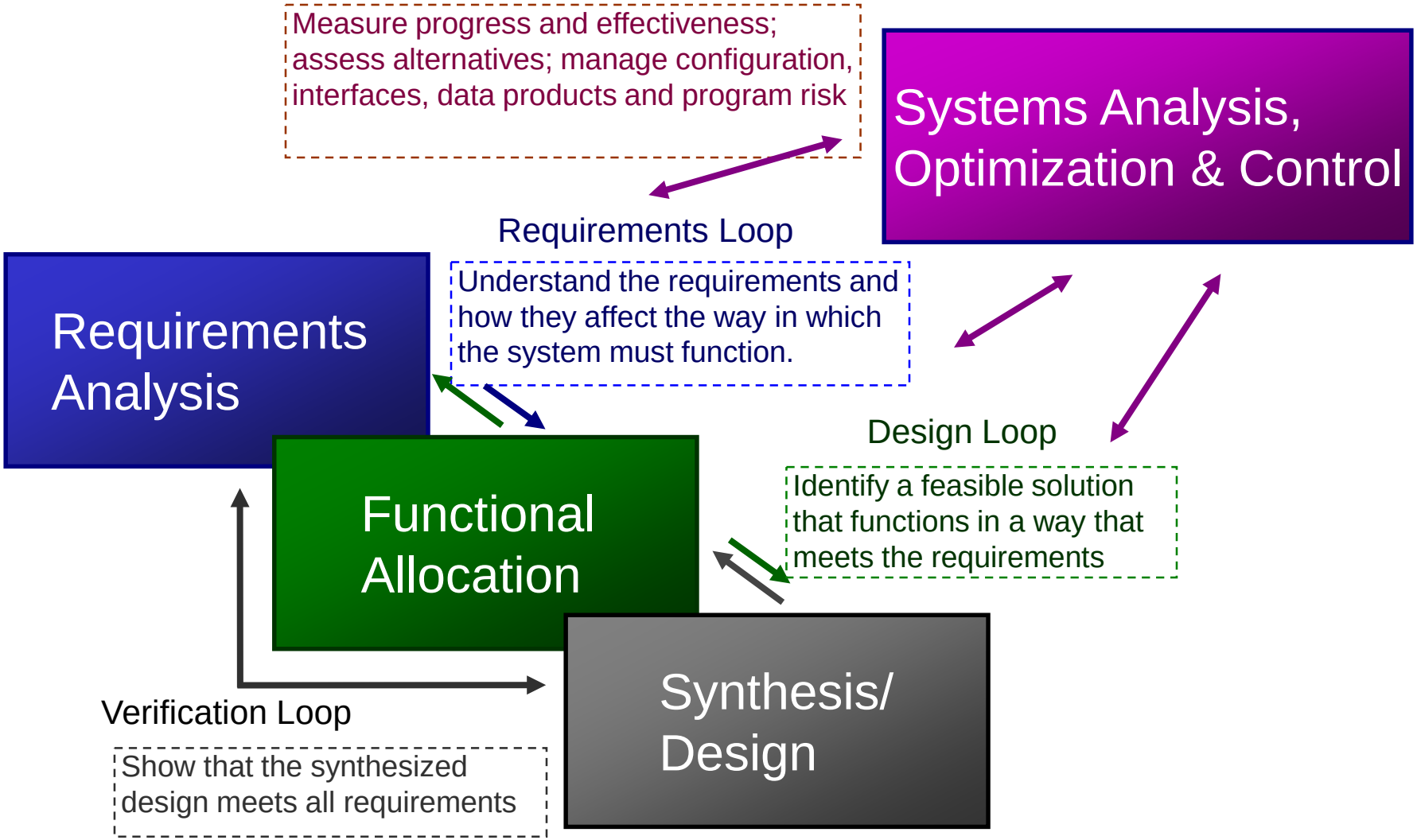
Stakeholder Requirements Definition Process Context Diagram (INCOSE)



Each process identified on the previous slide can be represented in this "ICOM" or "IDEF0" format; this slide is just one example

Source: INCOSE Systems Engineering Handbook, version 3.2.2, Figure 4-2

A Traditional View of the Systems Engineering Process Begins with Requirements Analysis



Slide Adapted from SYS501

What is Involved?

- **How do we define what the user wants**

Needs/Requirements/Capabilities

- **Needs are the user's perspective**
  - **"A lack of something wanted or deemed necessary"**
  - **"A necessity arising from the circumstance of a situation or case"**

- **Requirements are engineering-speak on describing what a system must do, not how the system will do it.**
  - **Intermediate step between the user needs and the design**

- **Defining requirements means to surface, understand, and document every required capability to meet the user needs.**

Requirements

- A characteristic that a system or software item is required to possess in order to be acceptable to the acquirer
- A binding statement . . . requirements are expressed using the word “shall”
- See EIA/IEEE J-STD-016

Capabilities

- **A *capability* is the ability to achieve a desired effect under specified standards and conditions through combinations of means and ways to perform a set of tasks**

- **Capability descriptions should be general enough so as not to prejudice decisions in favor of a particular means of implementation, but specific enough to evaluate alternative approaches to implement the capability**

Characteristics of Good Requirements

- **Necessary and verifiable**
  - **What's essential**
  - **Traceable and measurable/demonstrative**
- **Concise and unambiguous**
  - **Avoid generalities (best, user-friendly, minimize, etc.)**
  - **Little explanation/rationale (…but sometimes useful)**
- **Attainable – within reason, otherwise challenge**
- **Complete and consistent — may need system idea**
- **Implementation-free — what, not how**
- **Programmatic and compliance — levied rules, standards, etc.**
- **Use of the term "shall"**

Section 4.2.2.2 of SE Handbook

Levels of Requirements

▪ User Needs	▪ My house must be comfortably cool, even on the hottest days
▪ Operational Requirement	▪ The system shall keep the house cooled down to 74 degrees F throughout the year
▪ System Requirement	▪ The system shall be capable of cooling a 2600 sq ft house down to 74 degrees F in a climatic environment of up to +120 deg F
▪ Subsystem Requirement	▪ The Chiller subsystem shall dissipate up to 22,000 BTU per hour
▪ Component Requirement	▪ The Compressor motor shall develop at least 2 HP.

Good Requirement Examples

- **The system shall be capable of transporting a minimum of 20 passengers.**
- **The system shall be capable of a maximum speed of at least 200 km/hr.**
- **The system shall determine its location to within an accuracy of 10 meters.**
- **The system shall detect failures of its components and display them to the operator.**
- **The system shall operate in ambient air temperatures between -40 deg C and +51 deg C.**
- **The system shall have a mean time between failures of no less than 200 hours.**

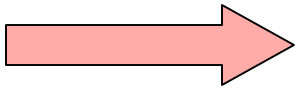
Poor Requirement Examples

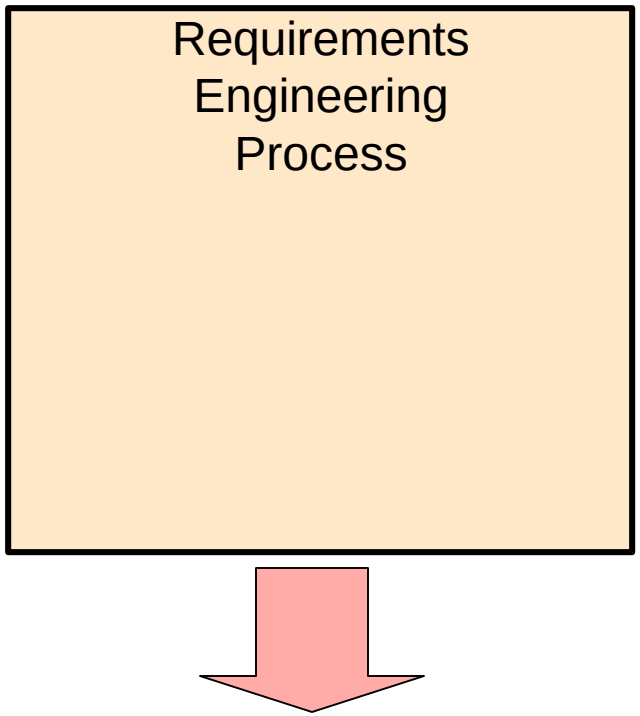
- **The land transport system will be capable of travelling up to 1600 km/hr.**
  - ***"Will" is a statement of intent. "Shall" is a binding requirement per standard practice and in legal contracts.***
  - ***The required velocity performance is beyond current technology and will turn the development into a research program.***
- **The system shall require minimal maintenance.**
  - ***"Minimal" is subjective. How minimal is good enough?***
- **The system shall be compatible with external power sources.**
  - ***Ambiguous statement. Are there specific power sources that the system must be compatible with or does this mean all possible sources?***
- **The system shall not cause harm to operators.**
  - ***The phrase "shall not" is a very poor term to use in a requirement as it is often difficult to prove a negative. What test could you create for this?***
  - ***Also, "cause harm" is a subjective term.***

Requirements Engineering

Inputs and Outputs

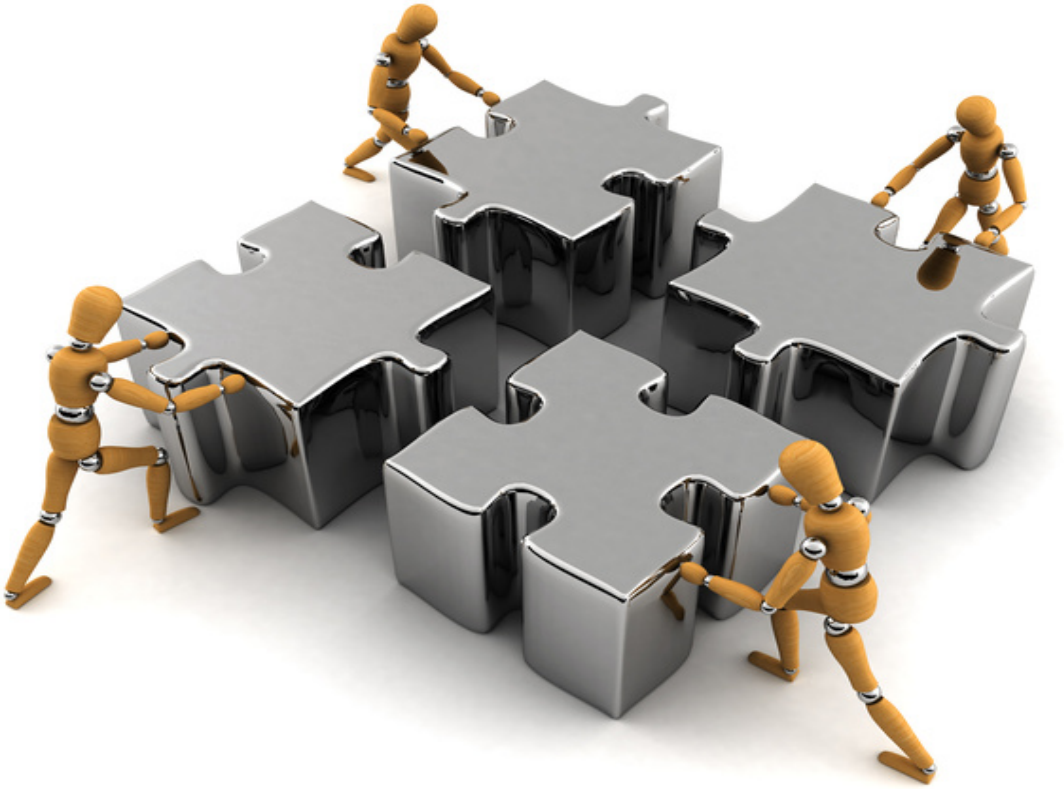
- Existing System Requirements
- Stakeholder Needs
  - Customer(s)
  - Users
  - Maintainers
  - Security
  - Management
- Organizational Standards
- Regulations
- Domain Information
- Type of Software System





- Agreed-Upon Requirements
- System Specifications
- System Models

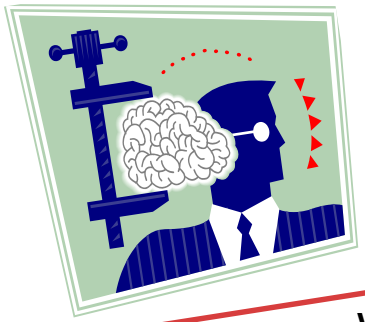
Exercise: Waters Process Models Applicable to the Projects



Instructor will provide direction for the exercise

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?



Systems Engineering Technical Short Course Session 1

Lesson 4: The Systems Development Process

Reference : "Systems Engineering, Principles and Practice," Kossiakoff et. al – Chapter 4

Session 1 Agenda (Jan 27 – 29): Timing

Day	Overview and Concepts of Systems Engineering	Hours*
1	Waters Management kick-off	1/2
	Introduction and overview of entire program (Lesson 1), including Introduction and overview of the projects (15 minutes each)	2
	Presentation and discussion of Concepts of SE (Lessons 2 – 9)	4
	Project assignments, team formations, and working sessions to apply SE concepts to the projects	1
	Day 1 wrap-up: Takeaways and Q&A	1/2
2	Review of day 1 working sessions	1
	Concepts of Systems Engineering (concluded) (Lesson 10 – 14)	4
	Project team working sessions (continued)	2
	Day 2 wrap up and Q&A	1
3	Team working session feedback to class (informal presentations)	1
	Project team working sessions (concluded)	5
	Check-in with class on application of SE concepts to their projects	1
	Session 1 wrap up: Q&A and homework due next session	1

* Times include breaks, and may be changed based on needs of the class

Concepts of Systems Engineering

1. Program Overview
2. What is Systems Engineering?
3. Structure and Environment of Systems
4. Systems Development Process
5. Systems Engineering Processes Overview
6. Systems Engineering in Needs Assessment
7. Risk Management
8. Concept Exploration
9. Concept Definition
10. Advanced Development Stage
11. Engineering Design Stage
12. Integration and Evaluation Stage
13. Production and Depoyment
14. Operations, Support, and Disposal

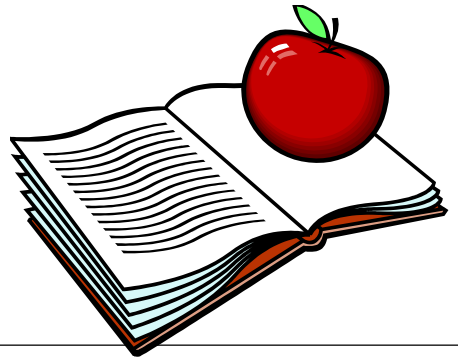
Plus Appendices

Lesson Content:

- SE Lifecycle
- SE Process activities
- Kossiakoff Model
- Vee Model
- SE Standards

Exercise: LC Model

Appendix B: DOD Lifecycle Model



Systems Development Process – Why?

- Humans are inherently fallible and make mistakes
- Systems development is complex
- Needs and technology driven development both involve a significant amount of risk (time & $$$)
- Whether hardware, software, large or small, a step-wise process is required to validate criteria before proceeding
- The multitude of entities and disciplines need to be on the same, top-level sheet of music
- How we determine when we're done

Systems Development Process

The Systems Development process is an integration of:

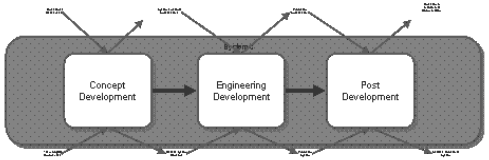
- **A Lifecycle Model**
- **Process descriptions**

The processes are applied iteratively and recursively throughout the system lifecycle

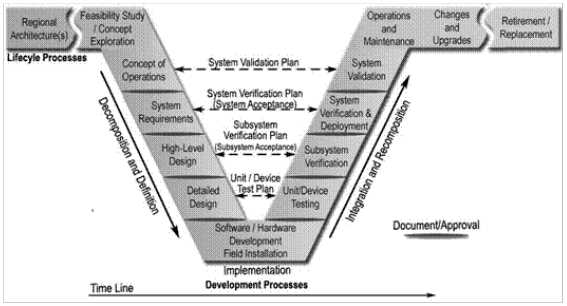
System Life Cycle Models

- **There are a multitude of them!**
- **One size does not fit all**

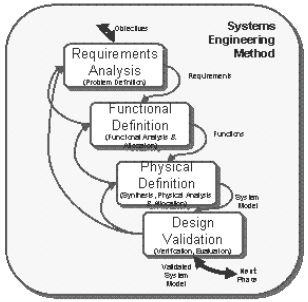
Linear



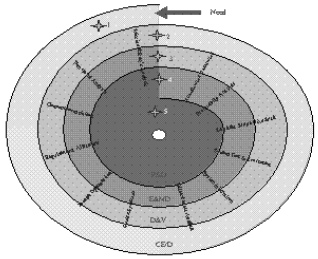
The "V"



Waterfall



Spiral



Generic Life Cycle (ISO 15288:2002)

Concept Stage	Development Stage	Production Stage	Utilization Stage / Support Stage	Retirement Stage

Typical High-Tech Commercial Systems Integrator

Study Period				Implementation Period			Operations Period		
User Requirements Definition Phase	Concept Definition Phase	System Specification Phase	Acq Prep Phase	Source Select. Phase	Development Phase	Verification Phase	Deployment Phase	Operations and Maintenance Phase	Deactivation Phase

Typical High-Tech Commercial Manufacturer

Study Period			Implementation Period			Operations Period		
Product Requirements Phase	Product Definition Phase	Product Development Phase	Engr Model Phase	Internal Test Phase	External Test Phase	Full-Scale Production Phase	Manufacturing, Sales, and Support Phase	Deactivation Phase

US Department of Defense (DoD) 5000.2

User Needs / Tech Opport / Resources	Pre-Systems Acquisition (A)		Systems Acquisition (B / C, IOC)		Sustainment (FOC)
	Materiel Solution Analysis	Technology Development	Engineering and Manufacturing Development	Production and Deployment	Operations and Support (including Disposal)

NASA

Formulation		Approval		Implementation		
Pre-Phase A: Concept Studies	Phase A: Concept & Technology Development	Phase B: Preliminary Design & Technology Completion	Phase C: Final Design & Fabrication	Phase D: System Assembly Integration & Test, Launch	Phase E: Operations & Sustainment	Phase F: Closeout

Feasible Concept → Top-Level Architecture → Functional Baseline → Allocated Baseline → Product Baseline → As Deployed Baseline

US Department of Energy (DoE)

Project Planning Period			Project Execution			Mission	
Pre-Project	Preconceptual Planning	Conceptual Design	Preliminary Design	Final Design	Construction	Acceptance	Operations

Typical Decision Gates: New Initiative Approval; Concept Approval; Development Approval; Production Approval; Operational Approval; Deactivation Approval

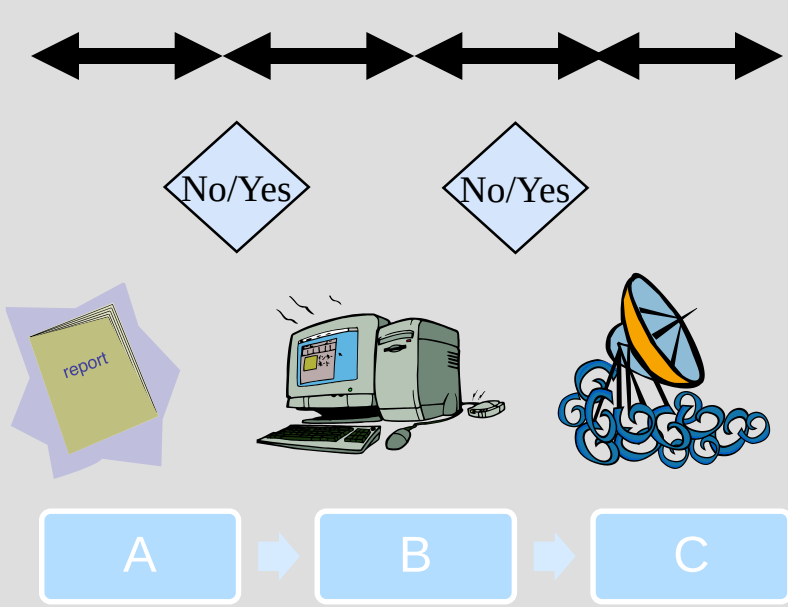
System Lifecycle Model Basics

Periods/Phases/Stages

Control (Decision) Gates

Products/Deliverables

Activities



1. Start with **periods and phases**
2. Then add **decision gates** for each phase
3. Then add **deliverables** required to make the decisions
4. Then add **activities** required to produce the deliverables

This sequence works for **creating new**, or **tailoring old**, project cycles

The Systems Engineering Process

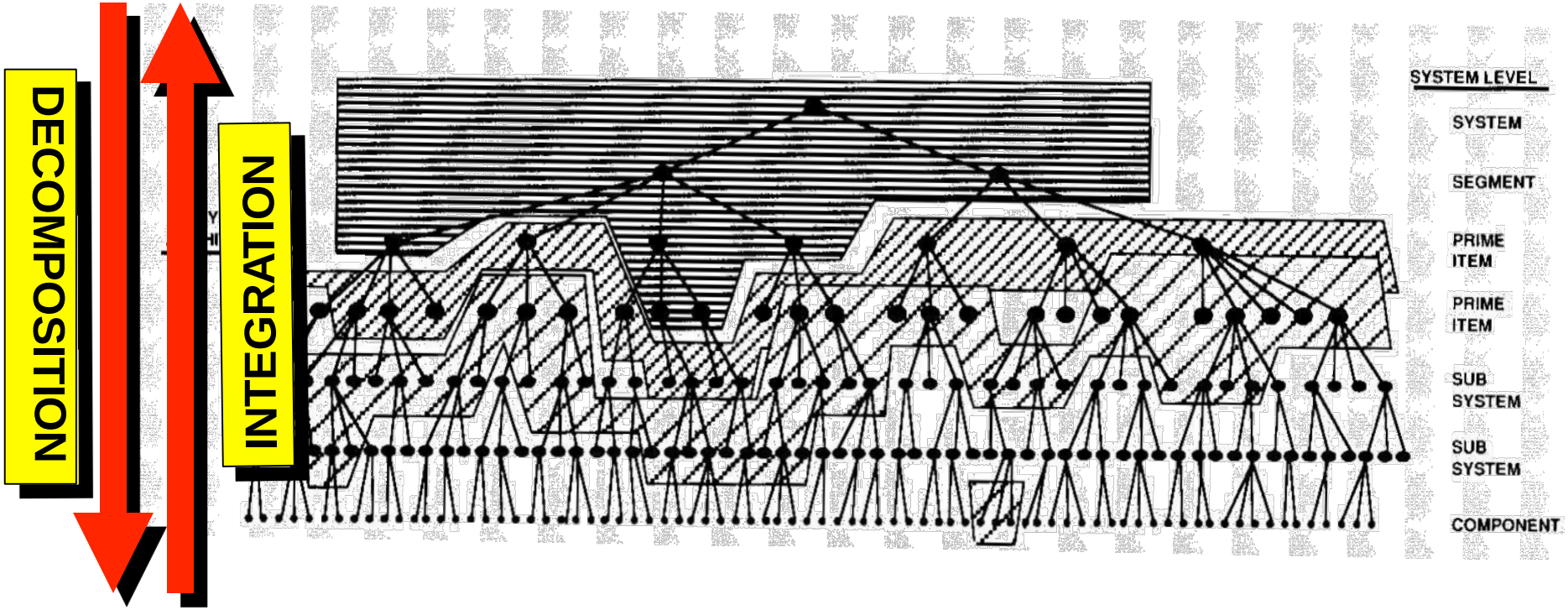
- **The major steps in the completion of a typical systems engineering project are**
  1. **Problem statement**
  2. **Identification of objectives**
  3. **Generation of alternatives**
     - **Analysis of these alternatives**
     - **Selection of one of them**
  4. **Creation of the system**
  5. **Operation and Sustainment**
  6. **Retirement**

Examples of Systems Engineering Process Activities

- Defining needs, operational concept, and requirements
- Functional analysis, decomposition, and allocation
- System modeling, systems analysis, and tradeoff studies
- Requirements allocation, traceability, and control
- Prototyping, integration, and verification
- Systems engineering product and process control
- Configuration and data management
- Risk management approaches
- Engineering technical reviews and their purposes

Managing Complexity

- **Decomposition techniques create “chunks” that can be handled by design teams and eventually individual designers**



Lifecycle Model Examples

- **Described later in this lesson**
  - **The Academic Modell (Kossiakof)**
  - **The Vee Model (Forsberg and Mooz)**

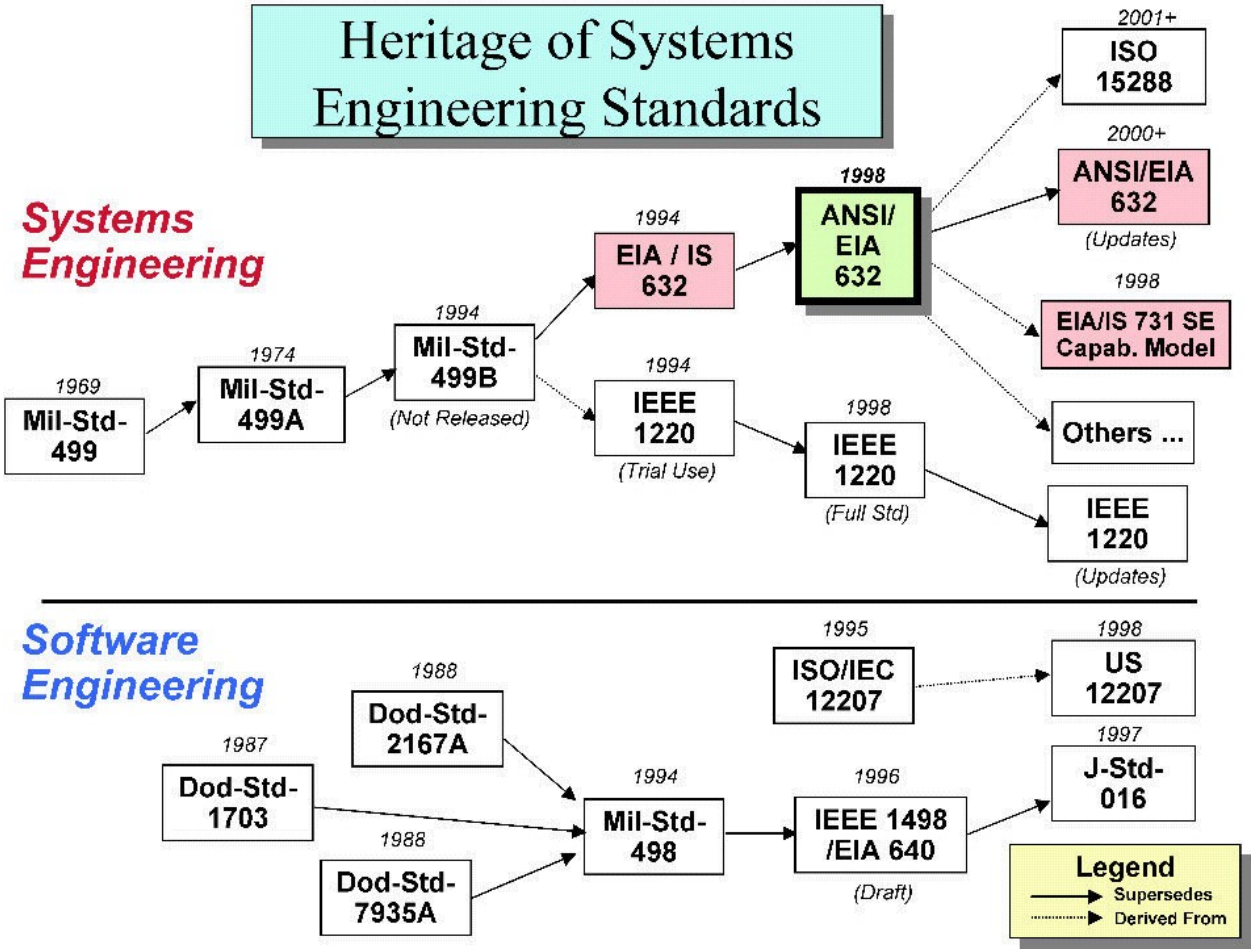
- **Others:**
  - **Spiral (Boehm)**
  - **DOD 5000.02 (Appendix)**
  - **Agile (several approaches)**

Using Systems Engineering Methodologies

- **Some people think of SE tools and methodologies as solution providers - plug in a bunch of "stuff" and get the answer: design, schedule, cost estimate, etc.**
  - **THIS IS WRONG!**
- **Systems Engineering provides a means for discretizing systems problems into chunks that can be solved, managed, and implemented**
  - **The scheduling, costs, and interdisciplinary issues are identified, but continuously change and emerge**

SE Standards

- Depends on your customer (MIL-STD, IEEE STD, etc.)
- Individual private programs can be managed in an *ad hoc* manner
- Government or large corporate contracts may require Mil spec or other spec to ensure process compliance
- INCOSE provides guidance somewhat independent of specific company or government standards





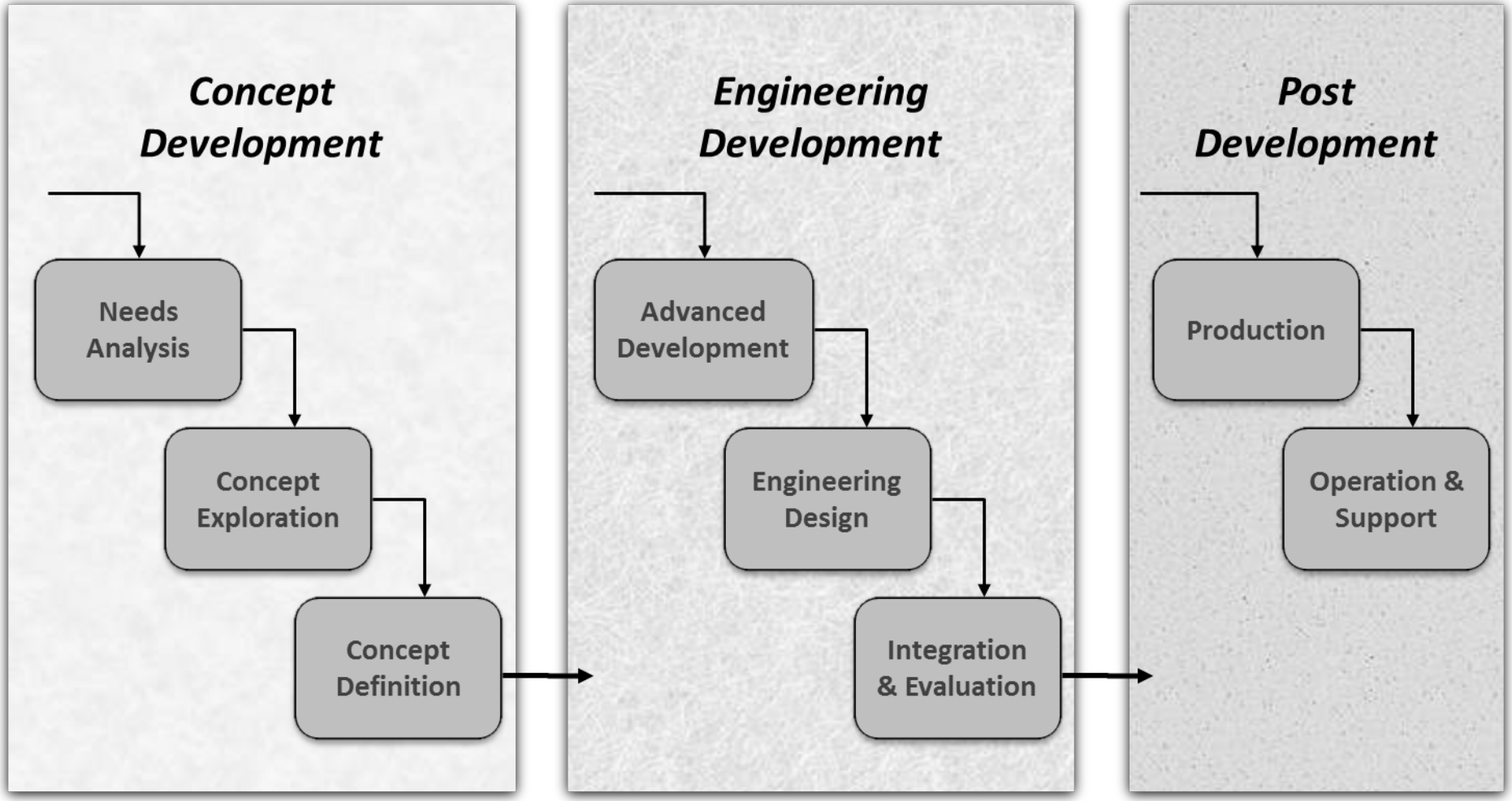
The Life Cycle Model á la Kossiakoff

This model is briefly described in this lesson, and is then elaborated through the course

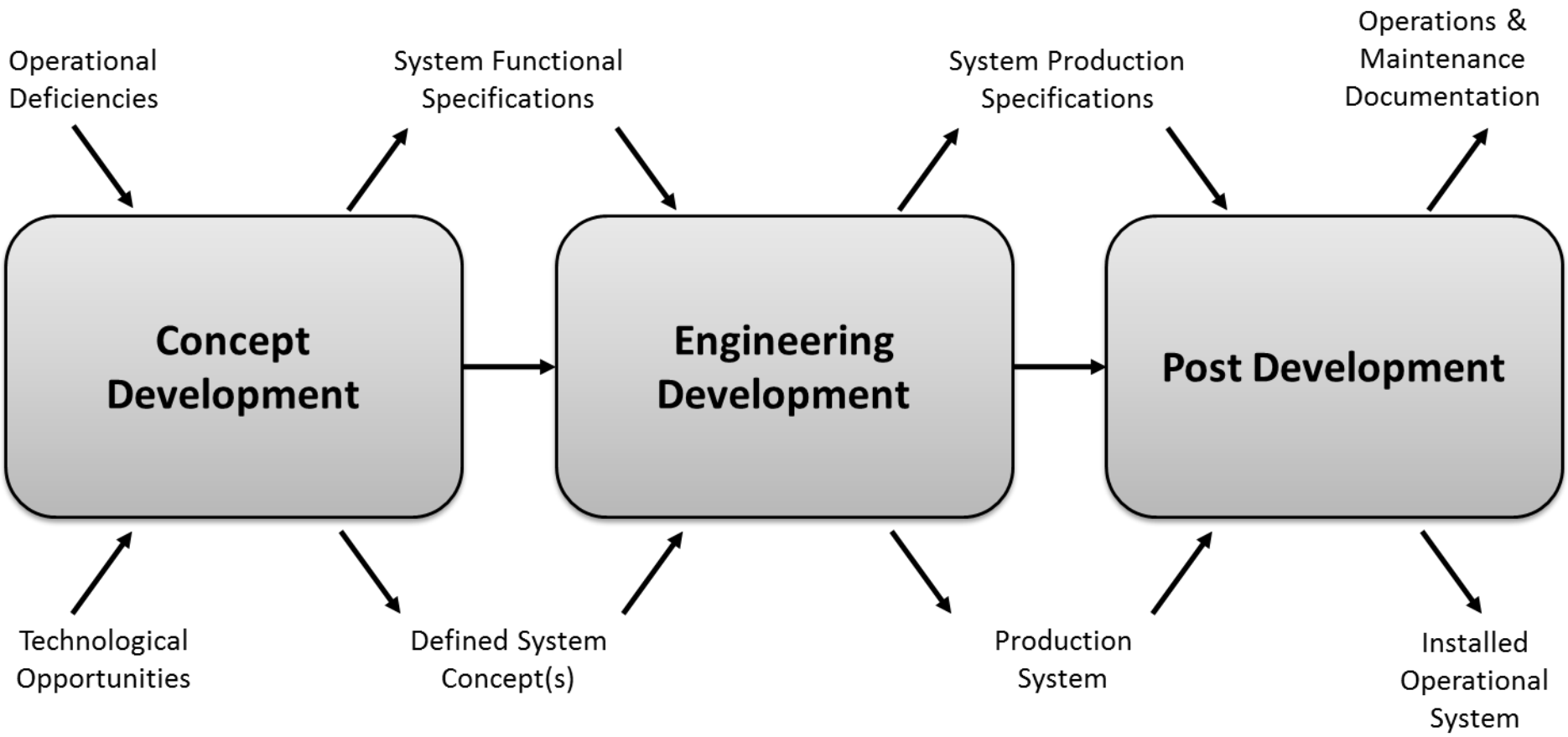
What is a Life Cycle Model?

- "Step-wise" description of the evolution of a system from its first conceptual idea through deactivation and disposal
- It includes…
  - Defining what is needed
  - Defining a concept or top level design to create a solution
  - Developing a solution
  - Building and deploying the solution
  - Using the system
  - Maintaining and fixing the system
  - Disposing of the system

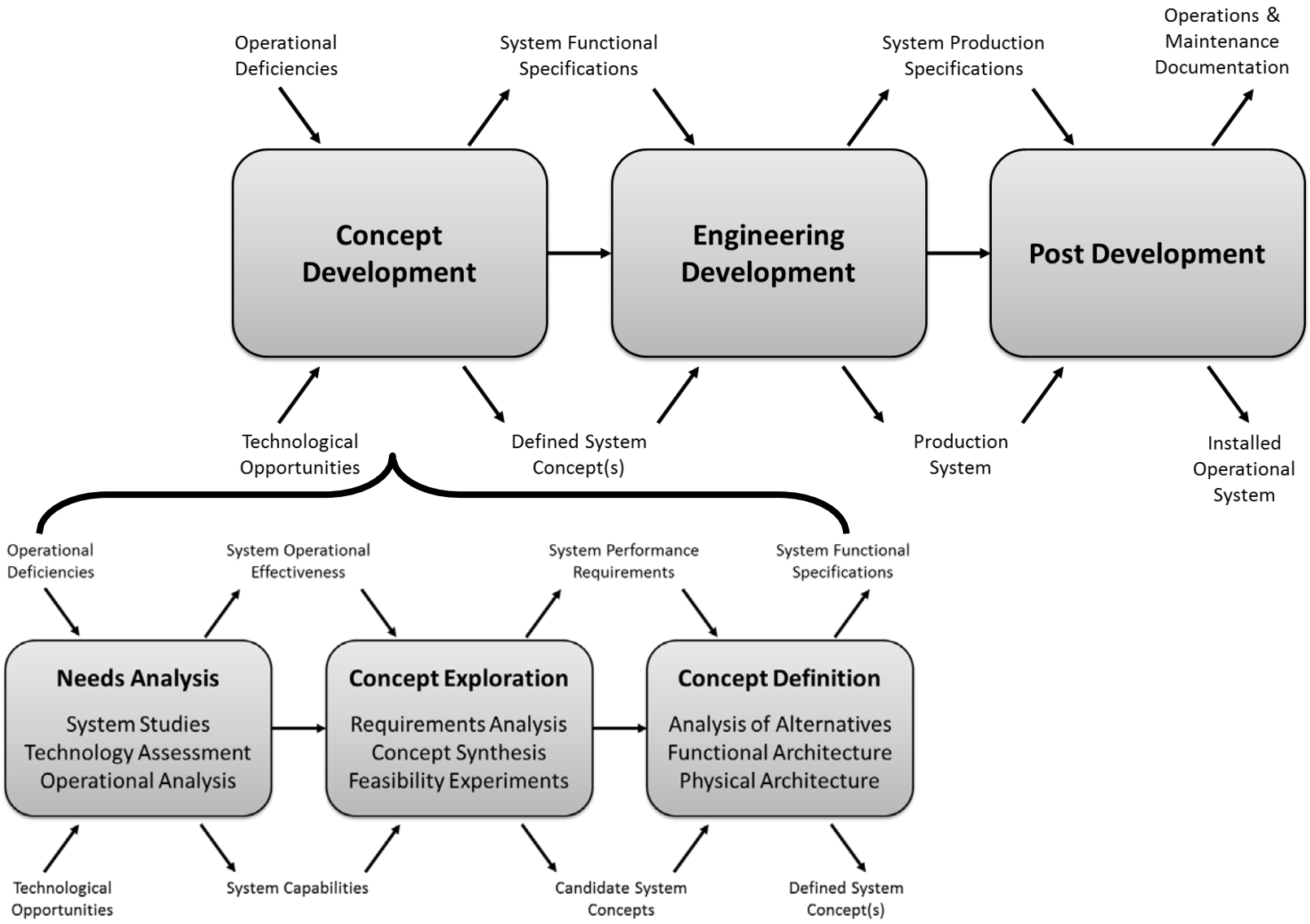
Life Cycle Model – Three Broad Stages



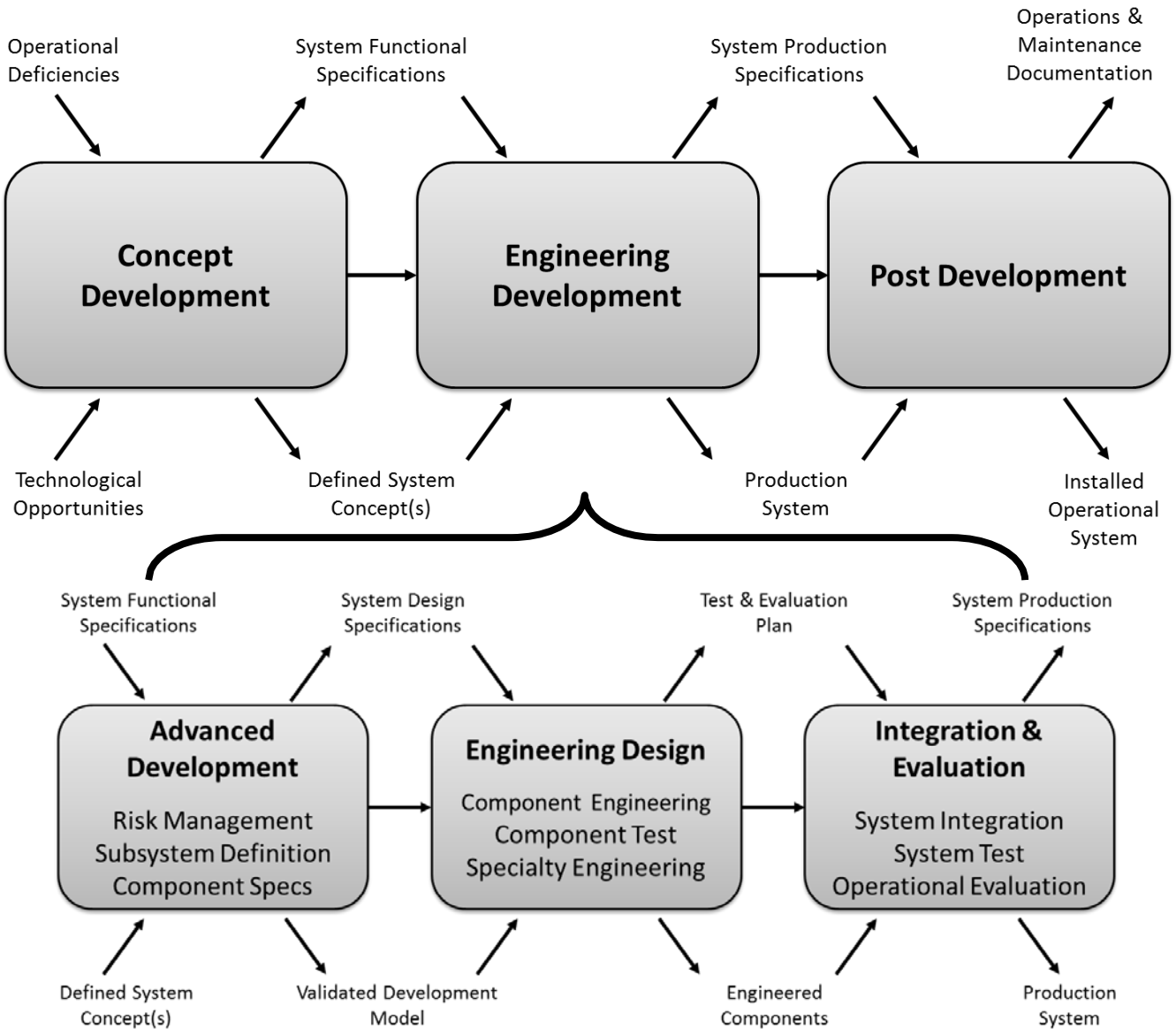
System Life Cycle Model – Inputs/Outputs



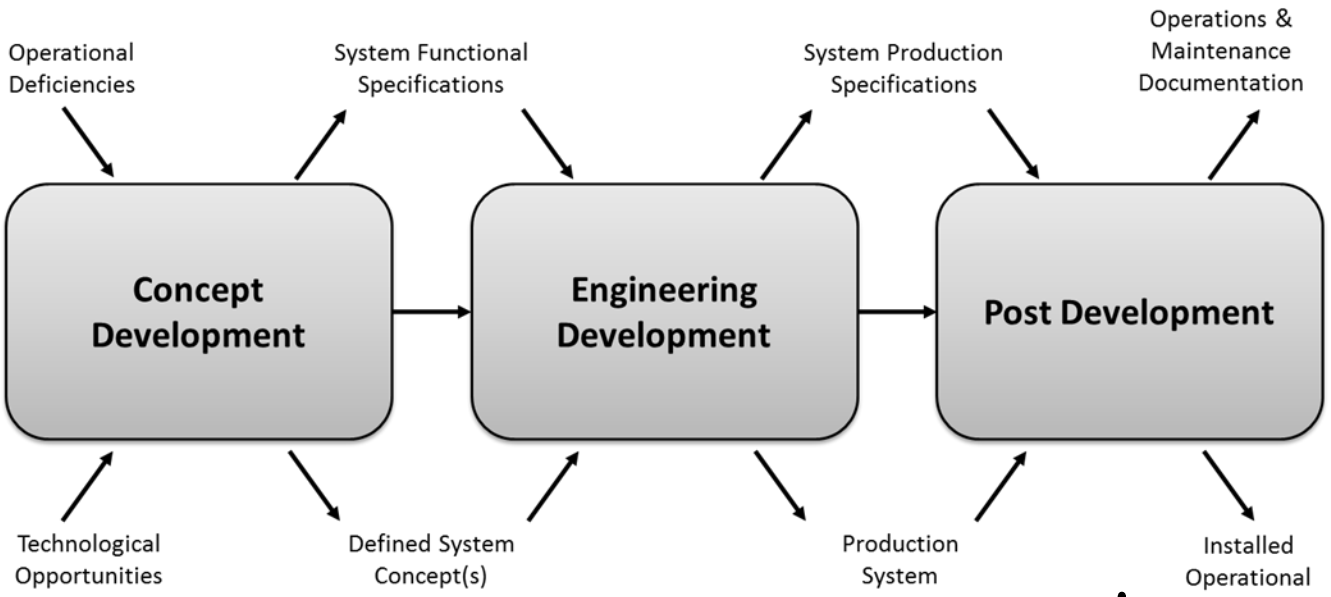
System Life Cycle - Looking at the Details



Developing the Solution



Fielding, Using and Disposing



System Production

Operation

Maintenance

Sustainment

Disposal



The Vee Model á la Forsberg and Mooz

Start with the end in mind

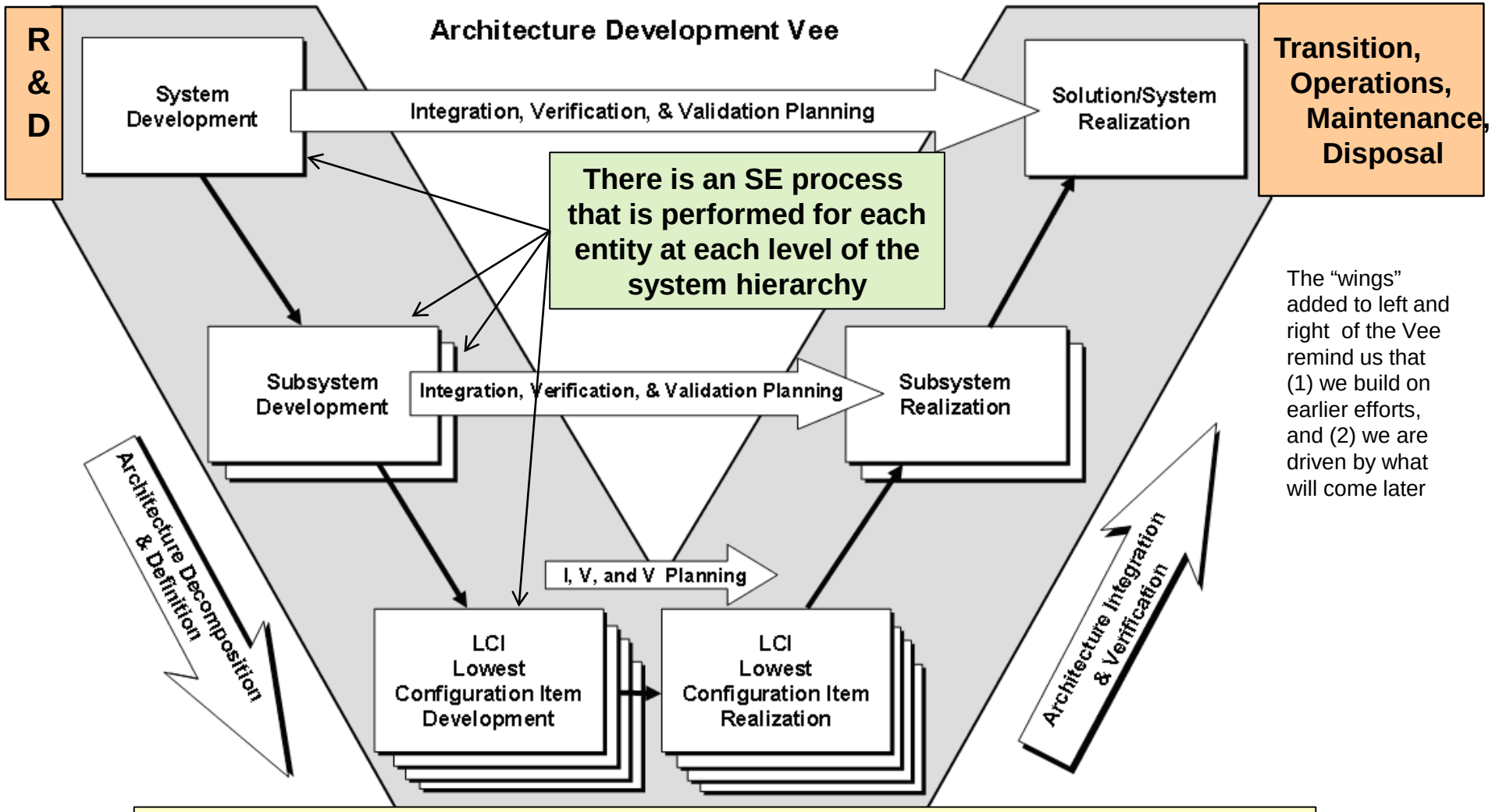
Source: Although the slides illustrating the Vee have been adapted from INCOSE and Buede, much of the defining work on the Vee was done by Forsberg and Mooz

Using the Vee to Model the Systems Engineering Process

- The Vee is particularly valuable in illustrating the systems engineering process
- The Vee model consists of three parts:
  - The Architectural Vee that illustrates the system hierarchy and the relationship between the levels in the hierarchy
  - The Systems Design Process used for every entity at every level of the system hierarchy (left side of the Vee)
  - The Systems Integration Process used for every entity at every level of the system hierarchy (right side of the Vee)

Whatever model you use, make sure your team understands it

Architectural Vee Process Model

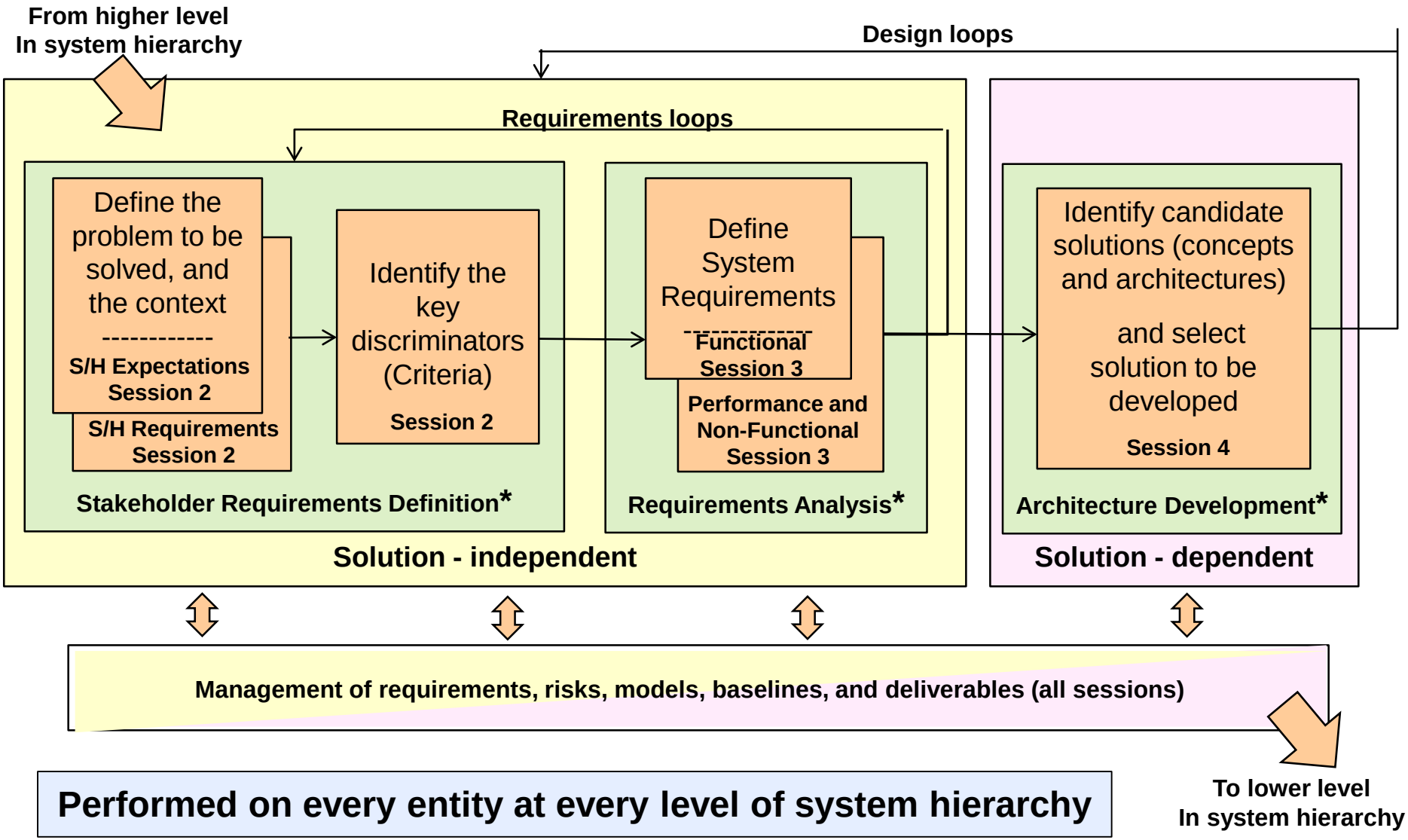


The "wings" added to left and right of the Vee remind us that (1) we build on earlier efforts, and (2) we are driven by what will come later

The next two slides show the two processes used on the left side and on the right side of the Vee respectively

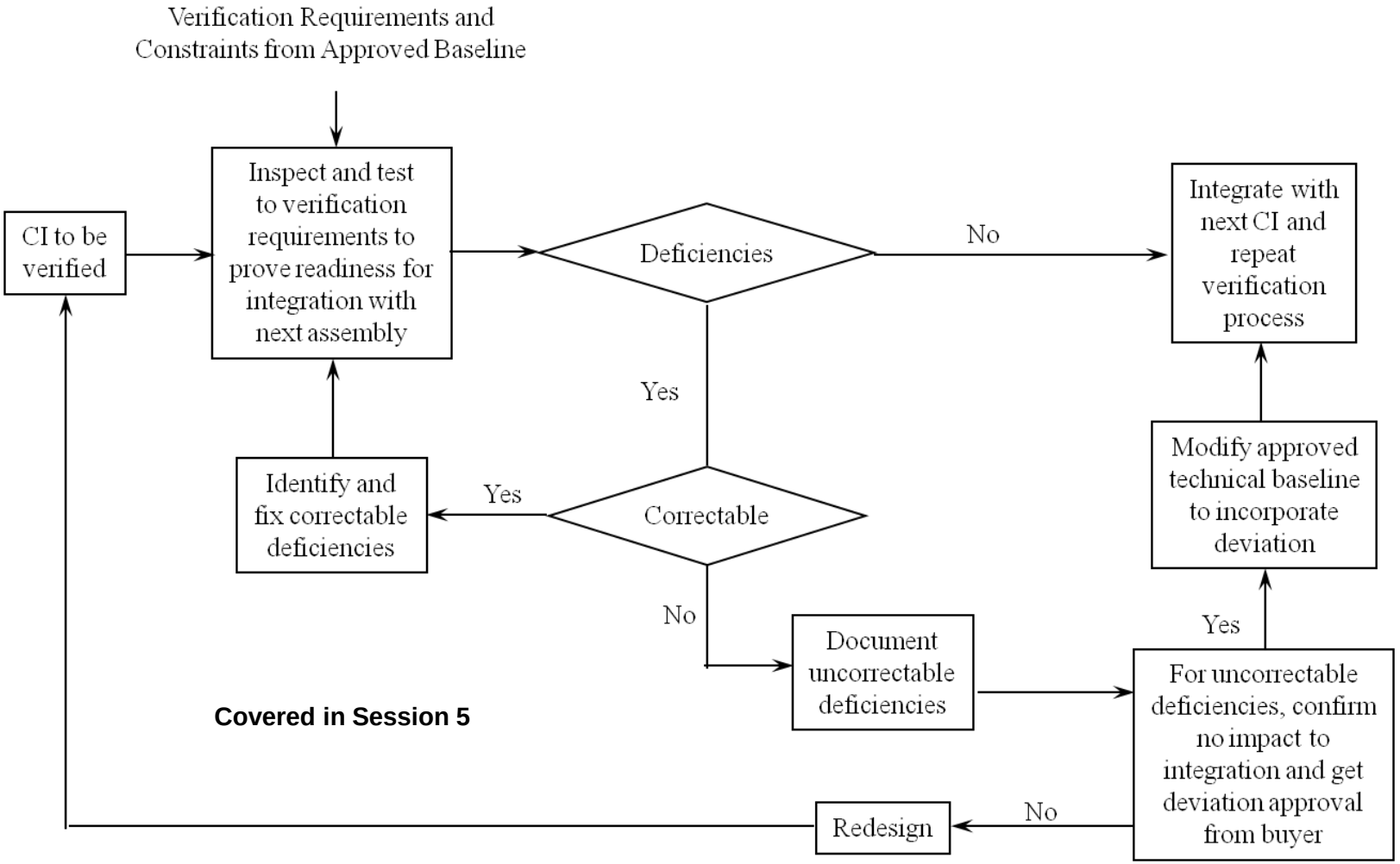
Adapted from INCOSE SE Handbook, Version 3.2.2, Figure 3-4

The Systems Development Process (Left Side of Vee)



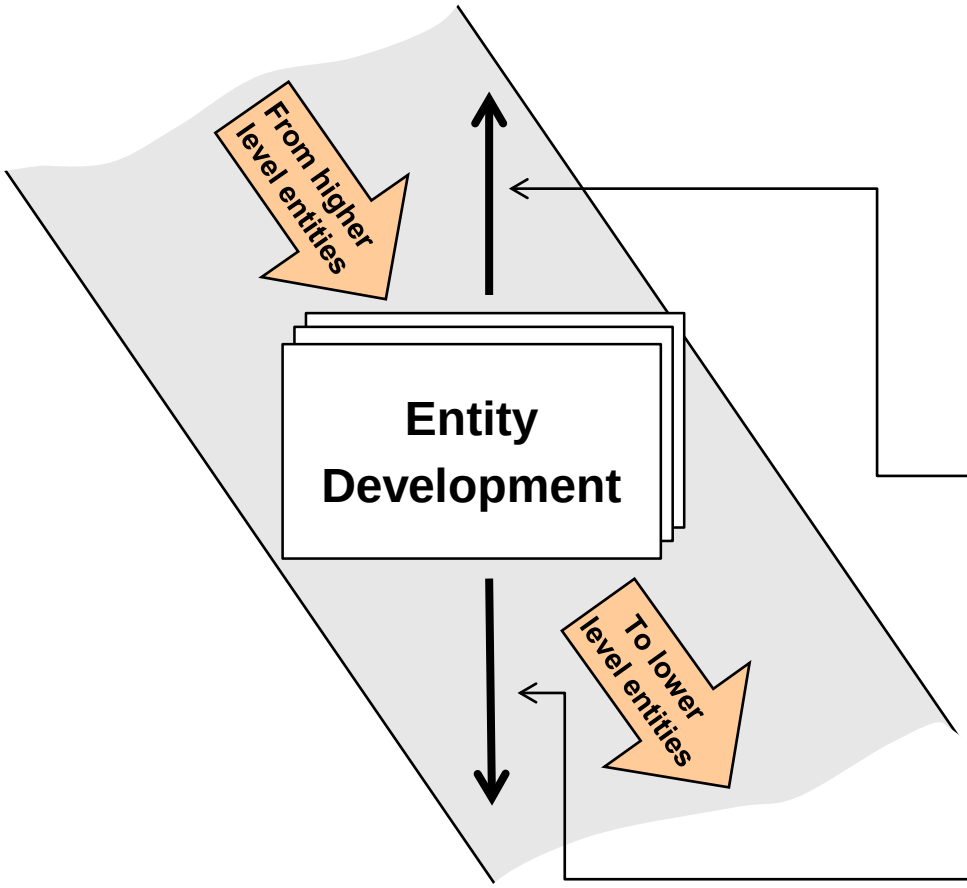
* Indicates INCOSE SE Handbook process name

The Systems Engineering Integration Process (Right Side of Vee)



Source: Adapted from Buede text, Figure 1.21

The Vee Model Incorporates Risk Management into the SE Process



- Entity development progresses down the "core" of the left side of the Vee, from system to subsystem to lower level entities
- Information gained while developing a lower level entity might require revisiting an upper level entity

This is called "off core" in-process validation

- To manage risk, It may be necessary to work at a lower level prior to completing the higher level development

This is called "off core" risk management

Can you think of examples of "off core" activities?

System Materialization

Level \ Phase	*Concept Development*			*Engineering Development*		
	Needs Analysis	Concept Exploration	Concept Definition	Advanced Development	Engineering Design	Integration & Evaluation
System	Define system capabilities & effectiveness	Identify, explore and synthesize concepts	Define selected concept with specifications	Validate Concept		Test & Evaluate
Subsystem		Define requirements & ensure feasibility	Define functional & physical architecture	Validate subsystems		Integrate & test
Component			Allocate functions to components	Define specifications	Design & test	Integrate & test
Sub-Component	*Visualize*			Allocate functions to subcomponents	Design	
Part					Make or buy	

Documents and Models

	Concept Development			*Engineering Development*		
	Needs Analysis	Concept Exploration	Concept Definition	Advanced Development	Engineering Design	Integration & Evaluation
Documents	System capabilities & effectiveness	System performance requirements	System functional requirements	System design specifications	Design documents	Test plans & evaluation reports
System Models	Operational diagrams; mission simulations	System diagrams; high-level system simulations	Architecture products & views; simulations; mock-ups	Architecture products & views; detailed simulations; breadboards	Architecture drawings & schematics; engineered components; CAD products	Test set-ups, simulators, facilities & test articles

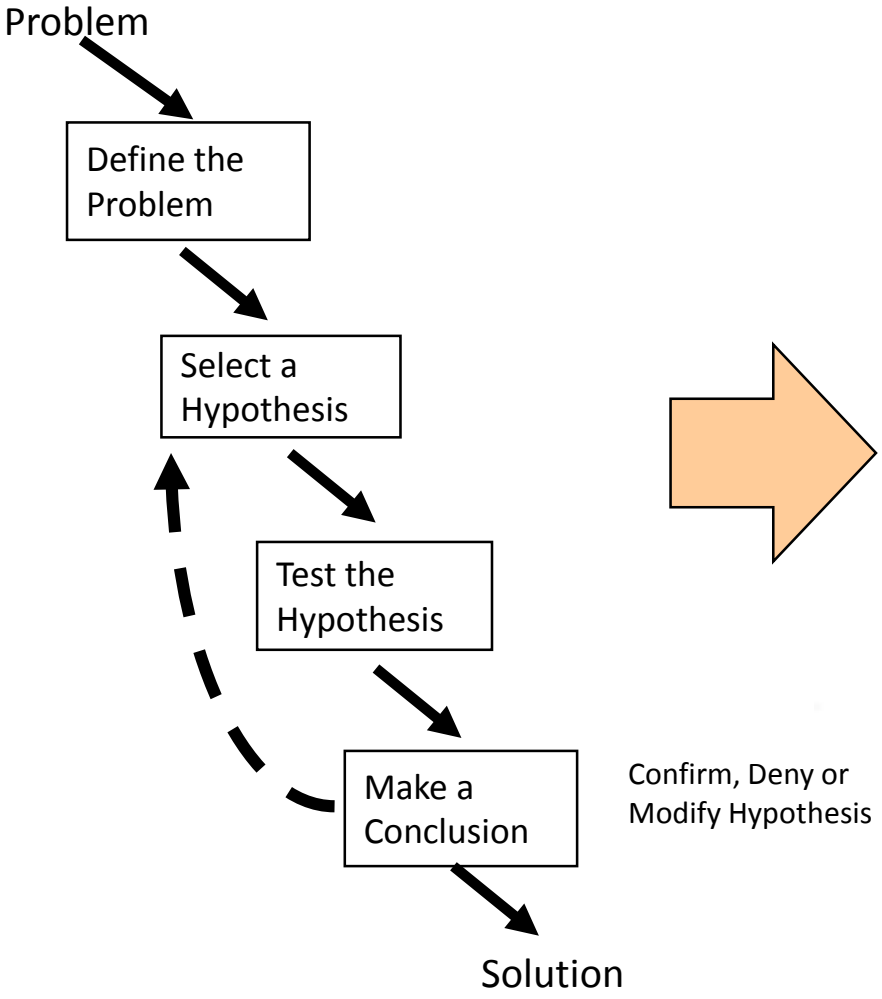


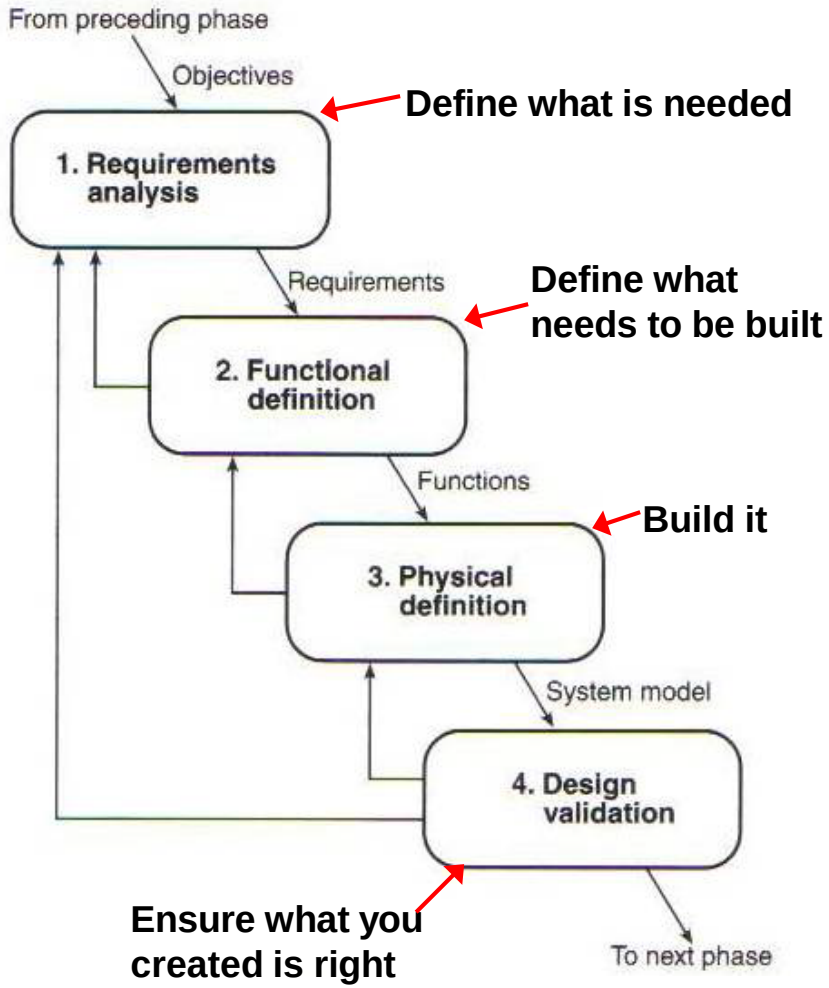
Other Models (See Back-up)

- DOD MIL-STD499B
- IEEE 1220
- EIA-632

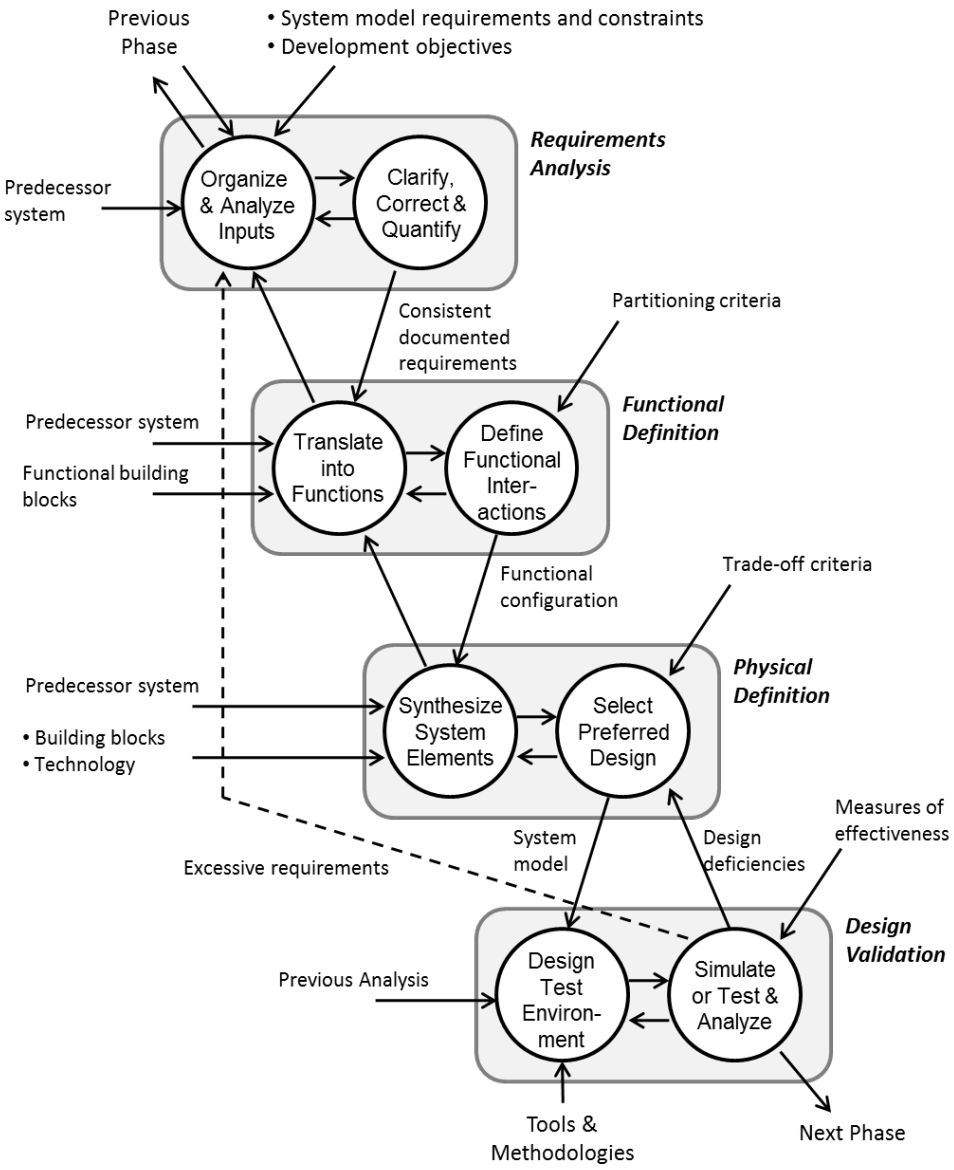
The Systems Engineering Process

The scientific method…. informs the systems engineering process

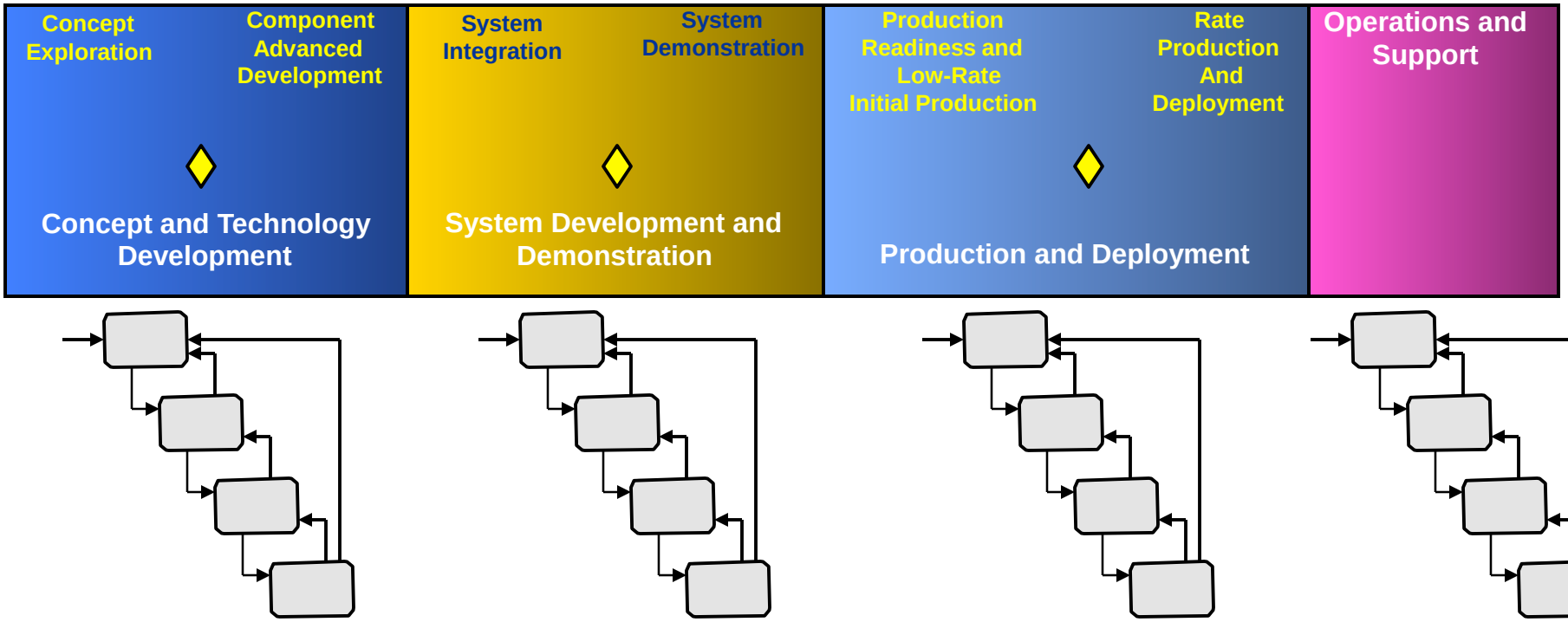




The Engineering Method – Detailed View



The System Engineering Method Repeats Throughout the Life Cycle



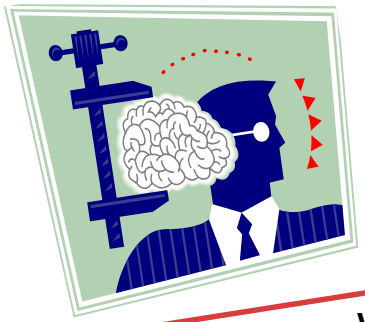
Exercise: Systems Lifecycle Model for your Projects



Instructor will provide direction for the exercise

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



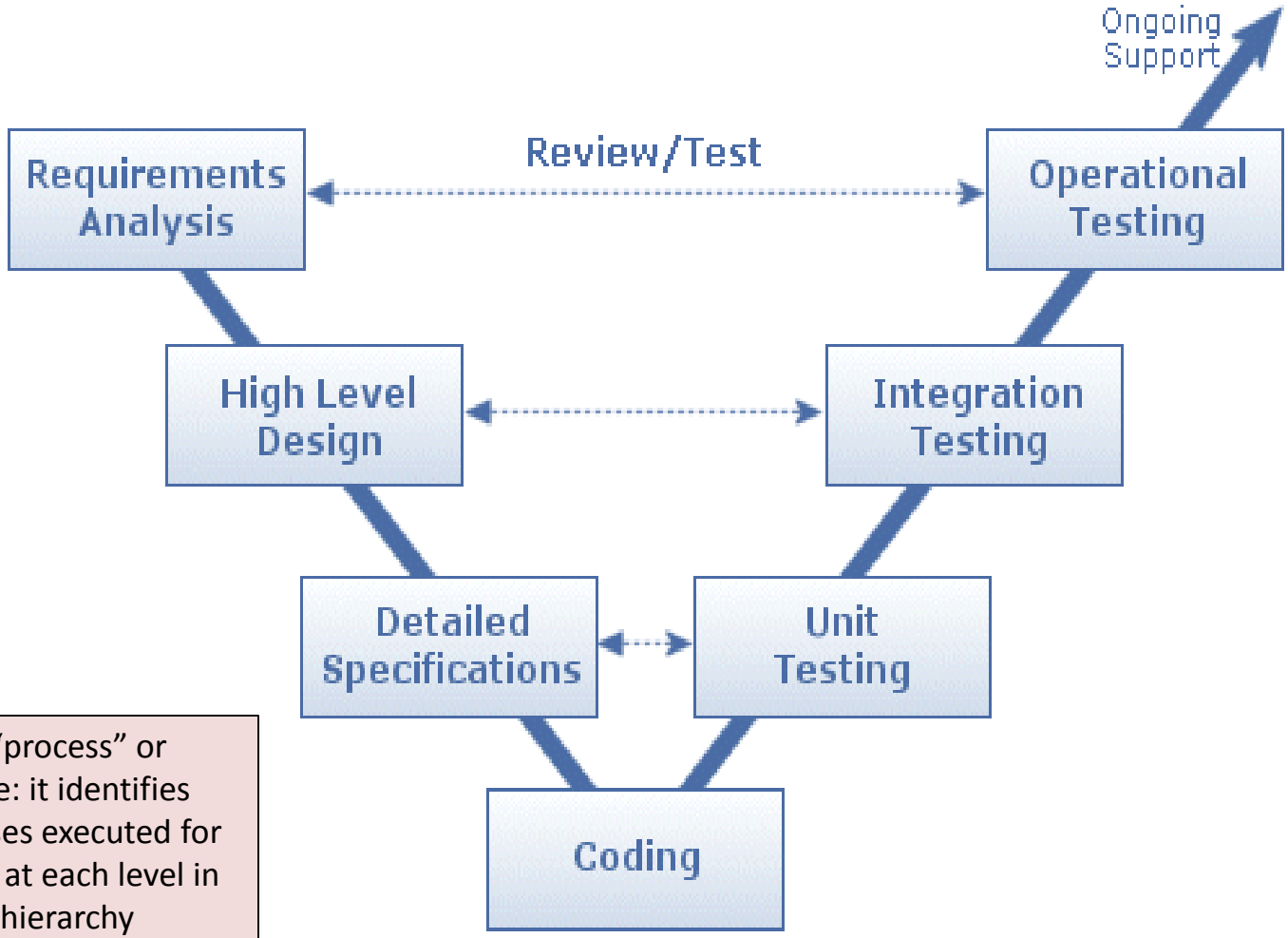
All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?

Back-up Slides

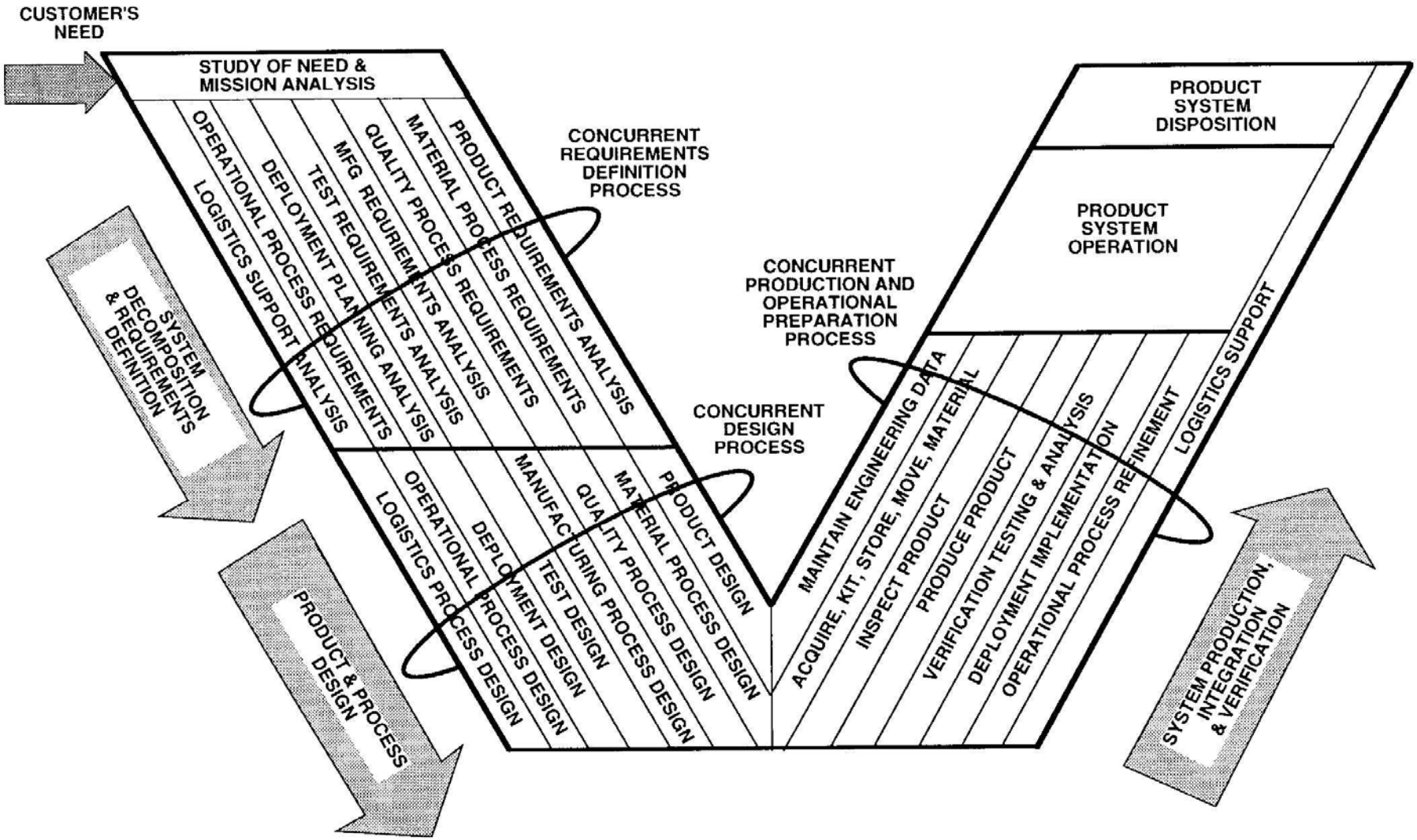
Simple Vee Model (Forsberg and Mooz)



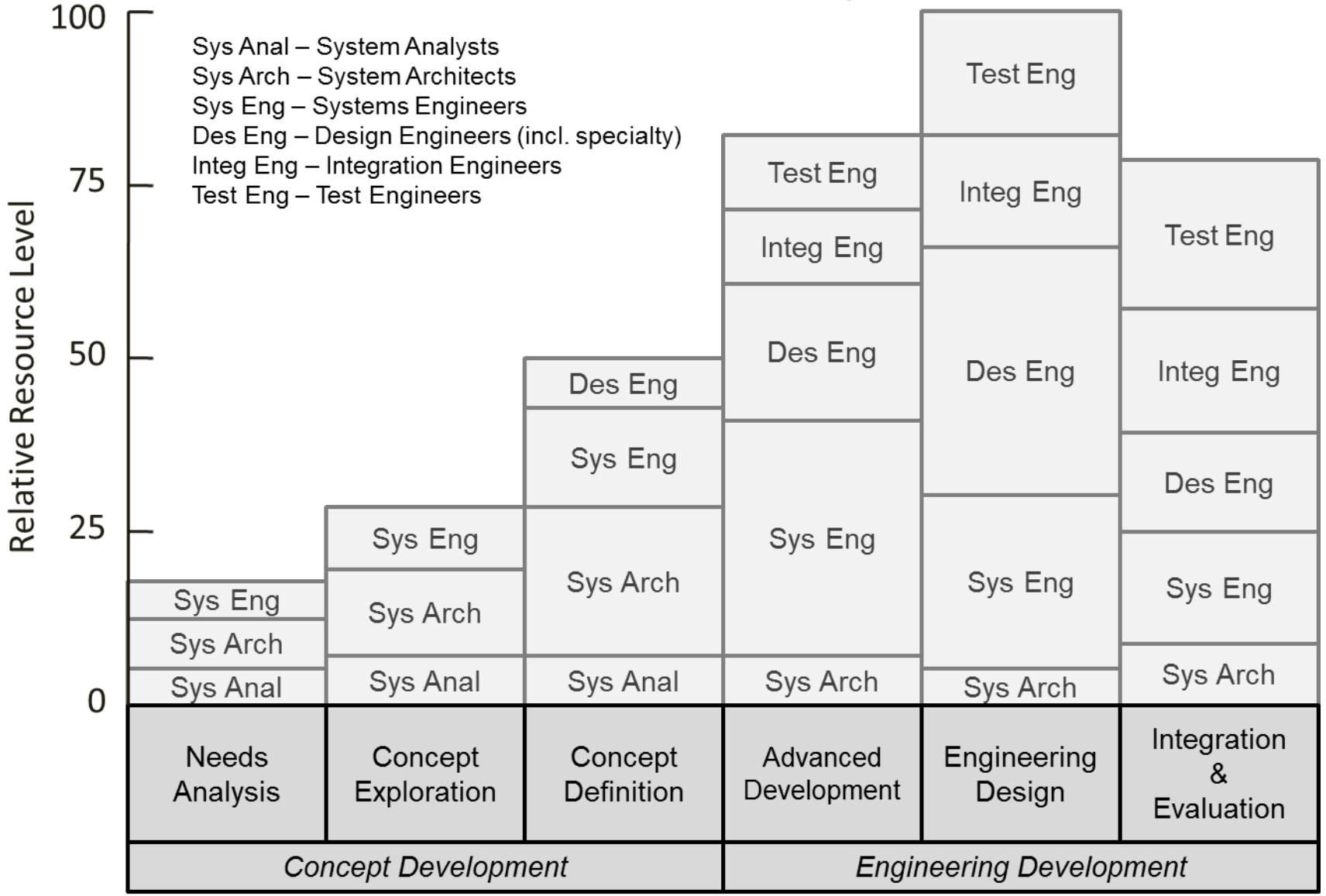
This is the "process" or "entity" Vee: it identifies the processes executed for each entity at each level in the system hierarchy

The Vee model will be discussed more thoroughly later in this course

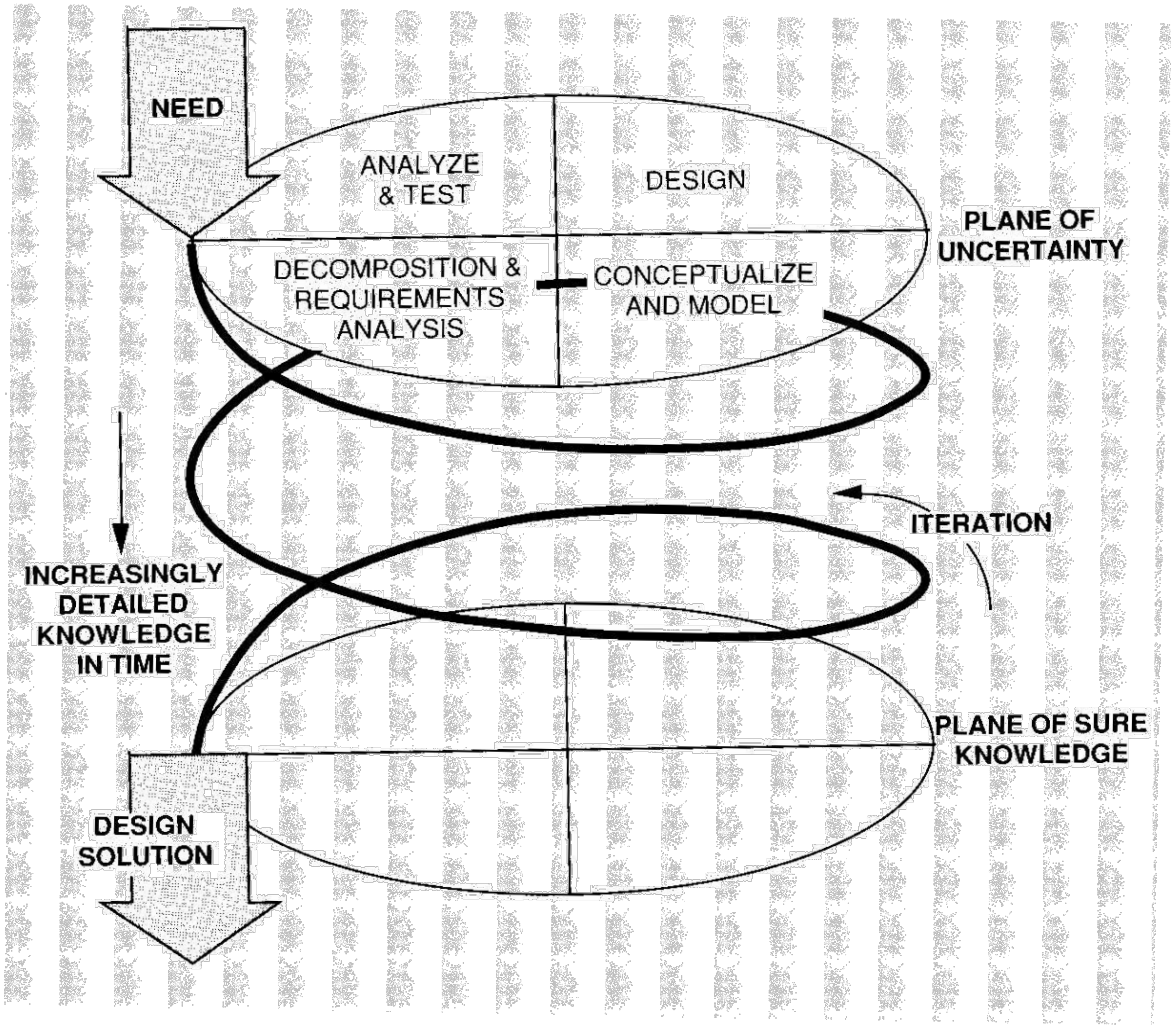
A More Complicated Vee Model



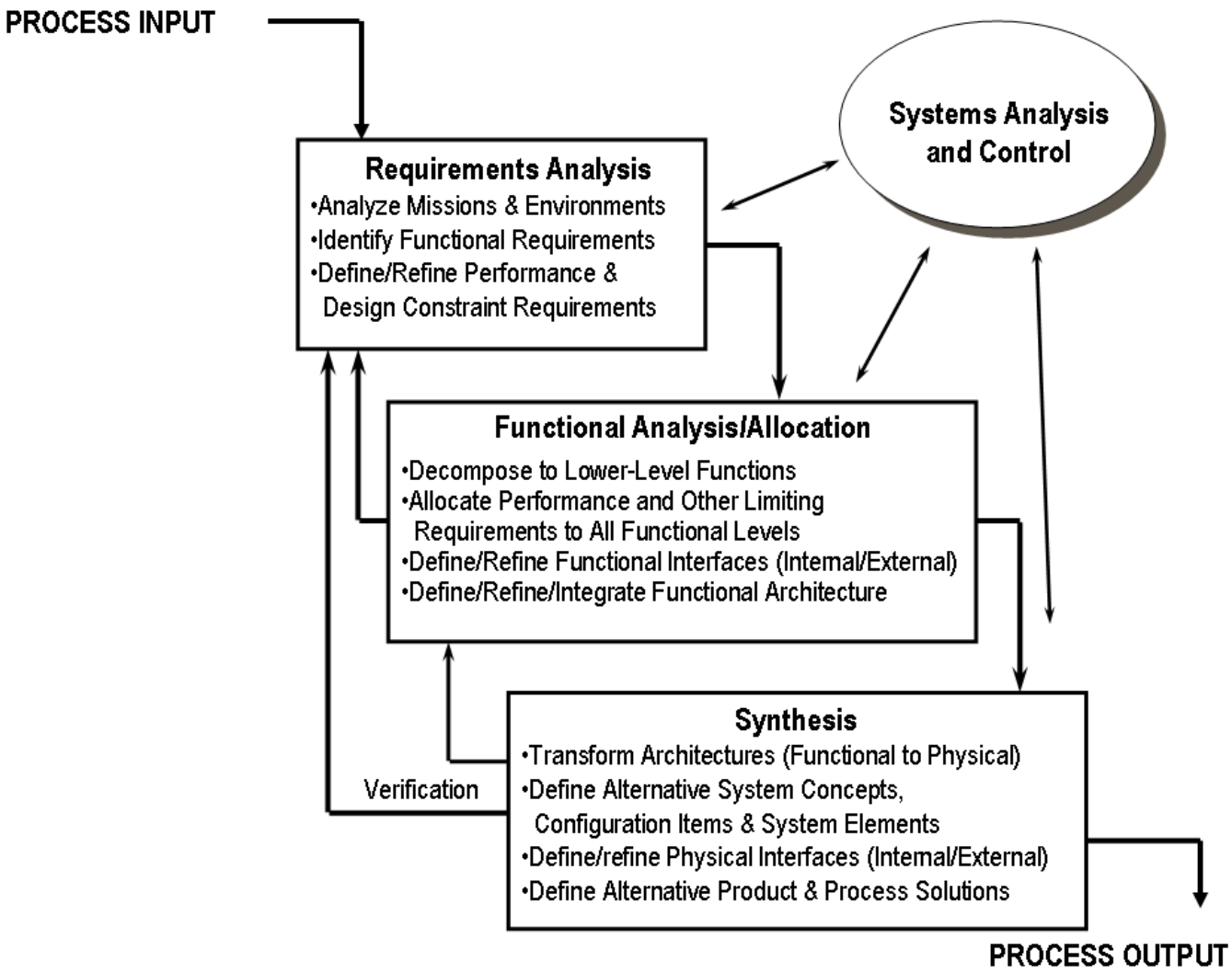
A Typical Aerospace Staffing Profile



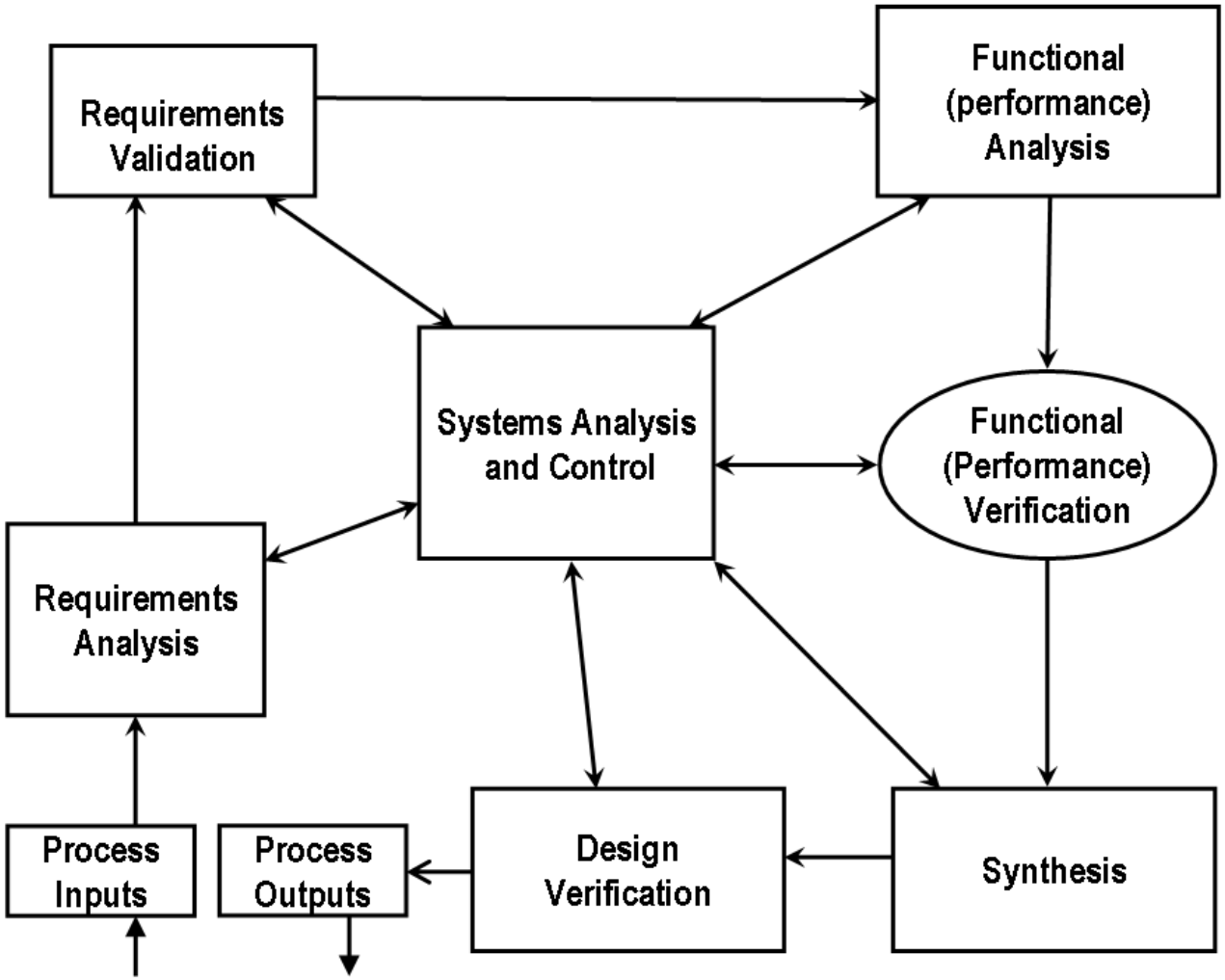
A Spiral Model



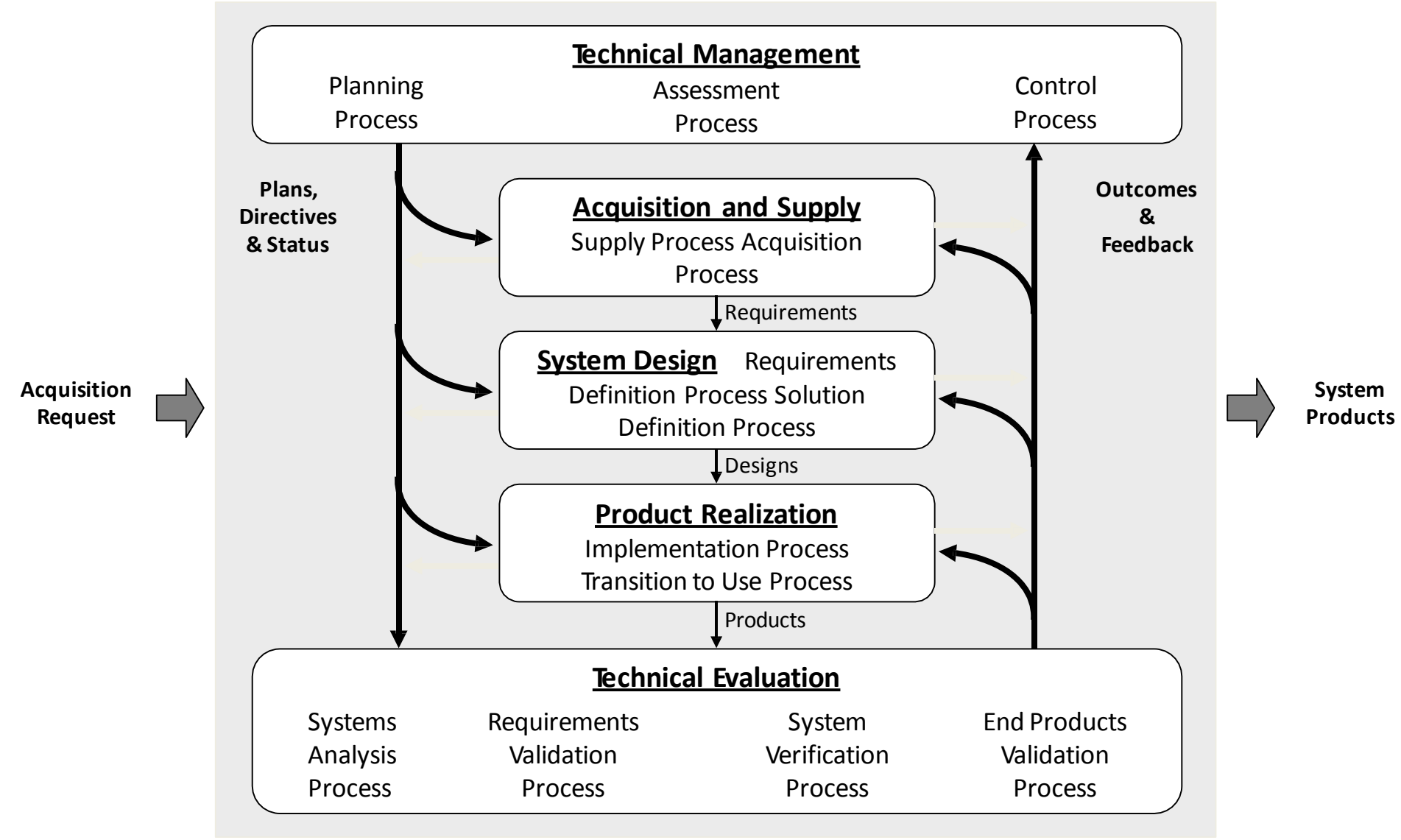
DoD MIL-STD499B



IEEE-1220 Systems Engineering Process



EIA-632 Systems Engineering Process







Systems Engineering Technical Short Course

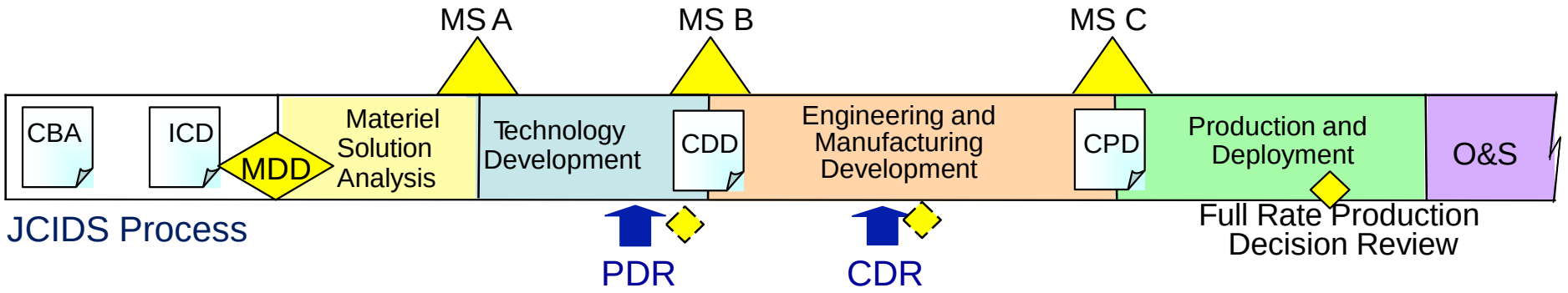
(Session 1)

Appendix B:
Supplement to Lesson 4:
DoD Instruction 5000.02
8 December 2008

DoD Instruction 5000.02
8 December 2008

Operation of the Defense Acquisition System
Statutory and Regulatory Changes

Defense Acquisition Management System

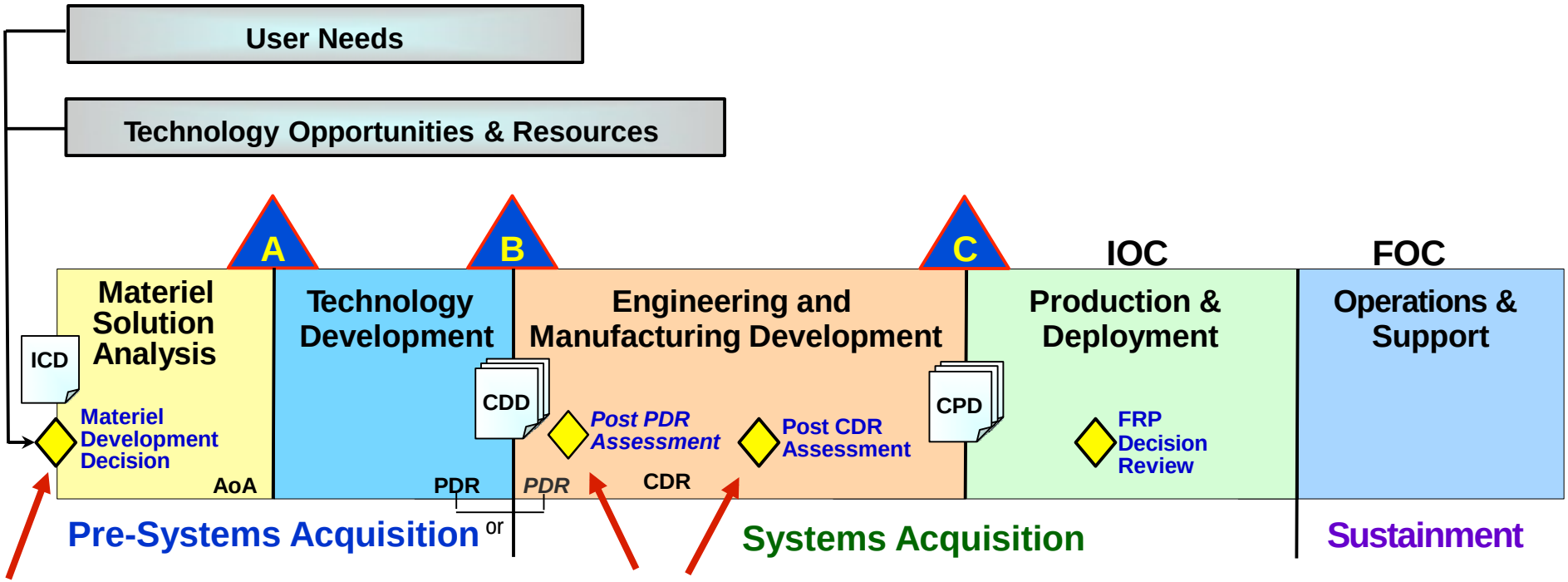


Some Terms

MDAP: Major Defense Acquisition Program
MS: Milestone
JCIDS: Joint Capability Integration Development System
CBA: Capabilities Based Assessment
ICD: Initial Capabilities Document
MDD: Materiel Development Decision
CDD: Capability Development Document
CPD: Capability Production Document
O&S: Operations and Support
PDR: Preliminary Design Review
CDR: Critical Design Review

DISTRIBUTION STATEMENT A -- Cleared for public release by OSR on 11 Jan 2010 -- SR case number 10-S-0779 refers.

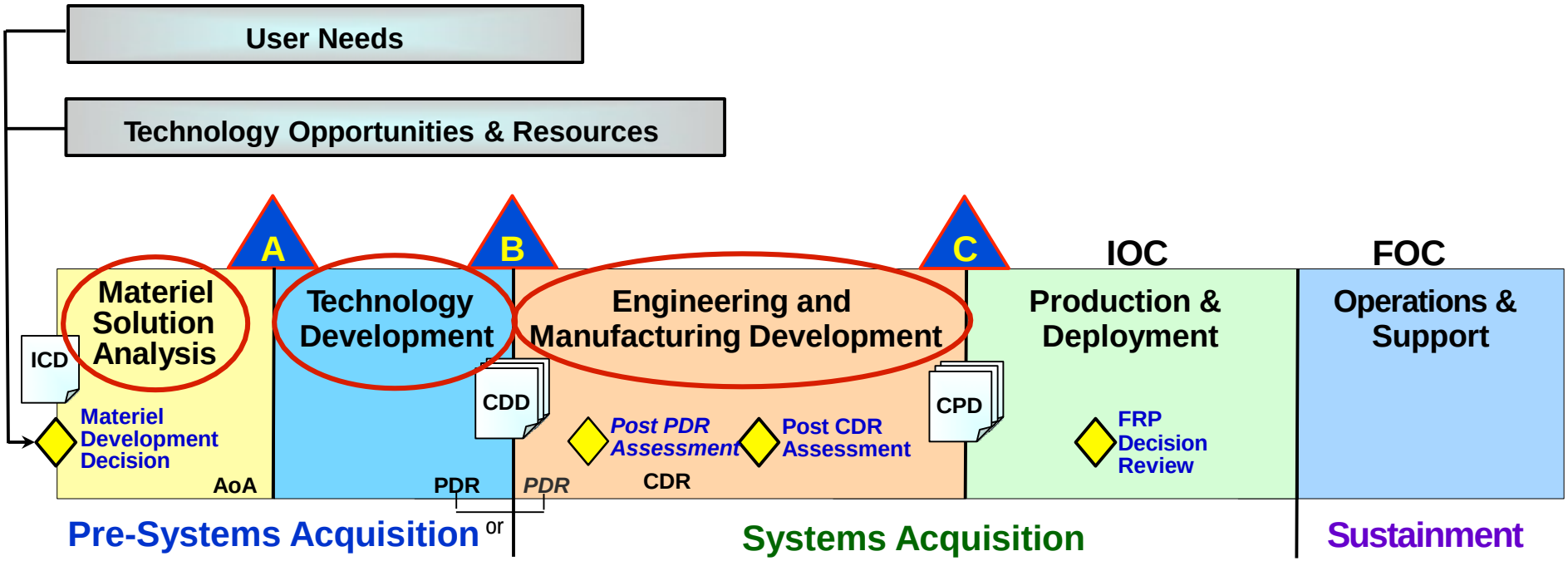
The Defense Acquisition Management System 2008



Changes to Decision Points

Old (2003)	New (2008)	Change from 2003
Concept Decision (CD)	Materiel Development Decision (MDD)	MDD required prior to entering the process at any point
N/A	*Post-PDR Assessment*	*MDA's assessment of PM's PDR Report (if PDR after MS B)*
Design Readiness Review DRR	Post-CDR Assessment	MDA's assessment of PM's CDR Report

The Defense Acquisition Management System 2008

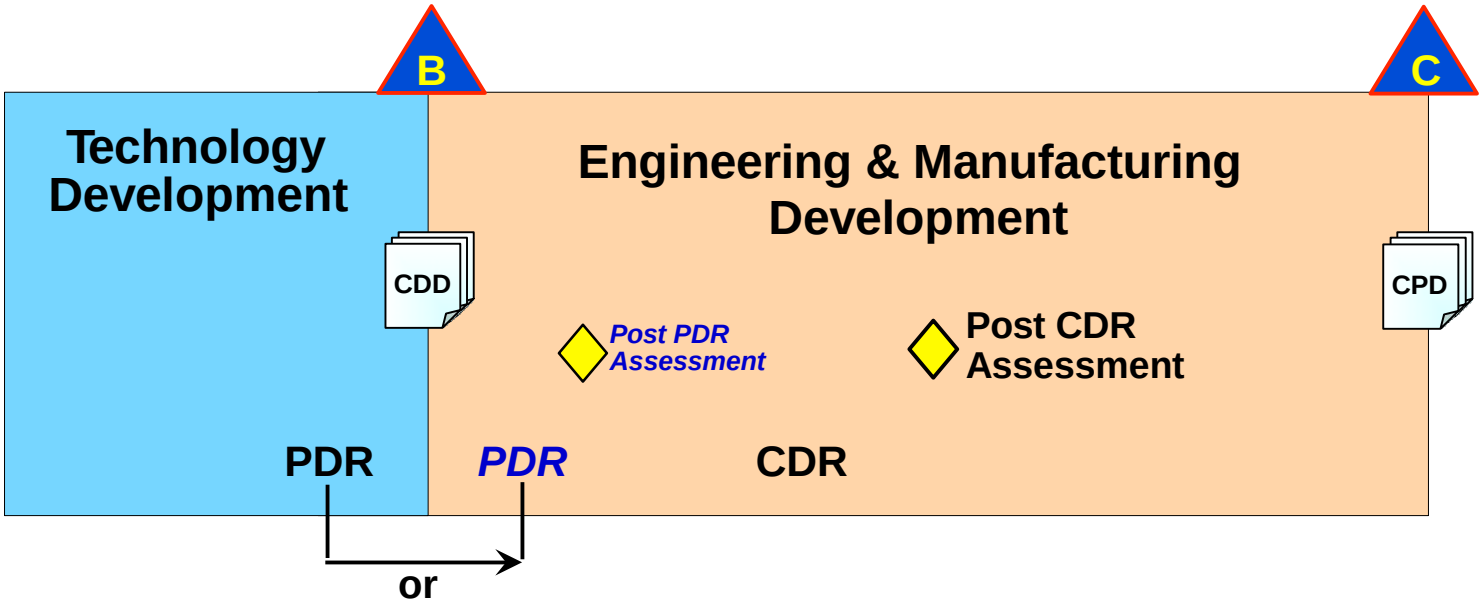


Changes to Phases

Old (2003)	New (2008)	Change from 2003
Concept Refinement (CR)	Materiel Solution Analysis	More robust AoA (result of changes to JCIDS)
Technology Development (TD)		Competitive prototyping
Systems Development & Demonstration (SDD)	Engineering & Manufacturing Development (EMD)	More robust system engineering

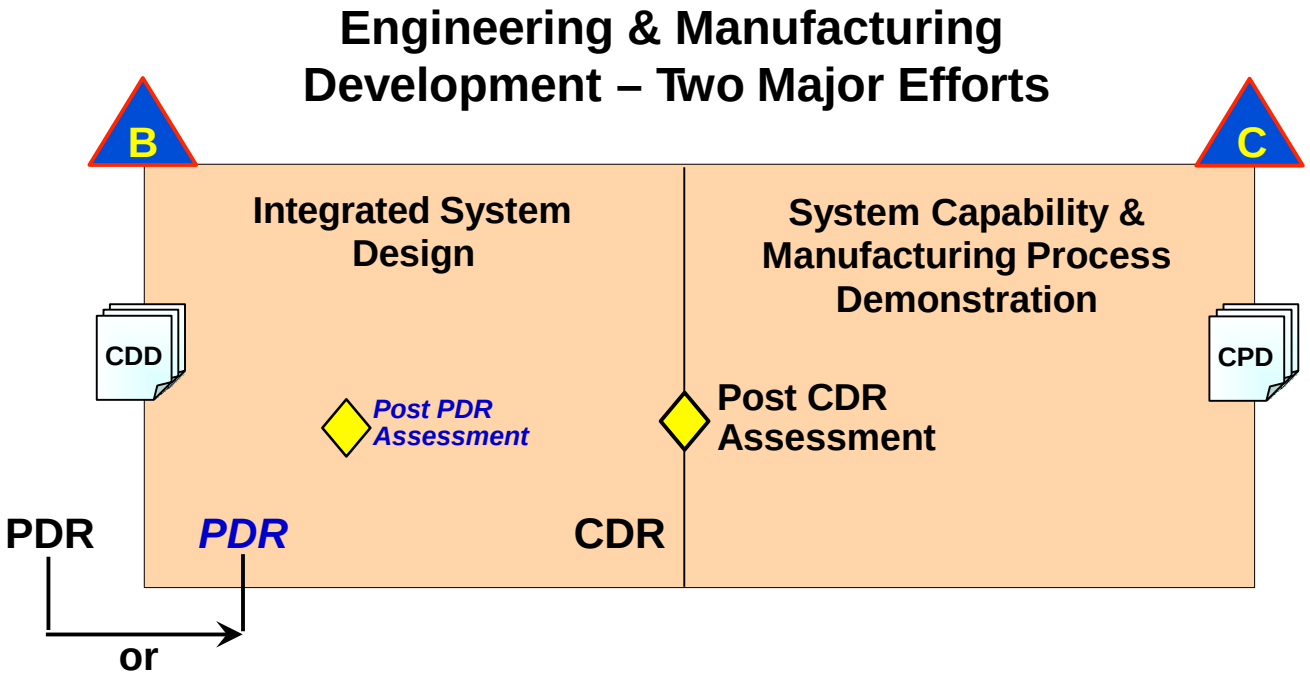
The Defense Acquisition Management System 2008

Preliminary Design Review



PDR Before Milestone B	PDR After Milestone B
• Planned for in Technology Development Strategy • PDR Report provided to MDA at MS B • Includes recommended requirements trades	• Planned for in Acquisition Strategy • PDR Report provided to MDA prior to Post PDR Assessment • Reflects requirements trades • At Post PDR Assessment, MDA considers PDR report; determines action(s) required to achieve APB objectives and issues ADM

The Defense Acquisition Management System 2008

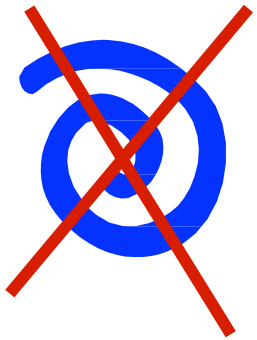


Old (2003)	New (2008)	Change from 2003
System Design	Integrated System Design	Establishment of Product Baseline for all Configuration Items
System Demonstration	System Capability & Manufacturing Process Demonstration	Manufacturing processes effectively demonstrated; production-representative article(s) demonstrated in intended environment; T&E assesses improvements to mission capability and operational support based on user needs.

Evolutionary Acquisition

From two processes… → **To one process…**

- Incremental Development: End-state is known; requirements met over time in several increments
- Spiral Development: End-state is not known; requirements for increments dependent upon technology maturation and user feedback.



- Capability delivered in increments, recognizing up front need for future capability improvements
- Each increment:
  - depends on mature technology
  - is a militarily useful and supportable operational capability
  - Successive Technology Development Phases may be needed to mature technology for multiple increments

Encl 12, Systems Engineering (New)

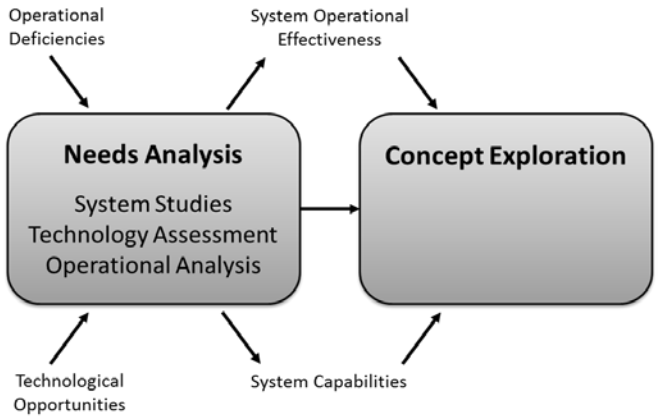
- **Systems Engineering Plan (SEP) required at each milestone**
- **Milestone Decision Authority (MDA) is approval authority for the SEP**
- **For programs where USD(AT&L) is MDA, and programs on the DT-only portion of OSD T&E Oversight List, SEPs must be submitted to Director, Systems and Software Engineering 30 days prior to DAB/ITAB review**
- **Program Executive Offices (PEOs) must have lead systems engineer – oversees SE across PEOs portfolio; reviews SEPs; assesses performance of subordinate systems engineers with PEO and program manager (PM)**
- **Event-driven technical reviews required – with SMEs independent of program, unless waived by MDA**
- **Requires configuration management to establish and control product attributes and the technical baseline**
- **Spectrum Supportability determination required**

Encl 12, Systems Engineering (New)
Continued…

- **Environmental Safety and Occupational Heath (ESOH) risk management required to be integrated with overall SE process; Programmatic ESOH Evaluation (PESHE) required of all programs regardless of ACAT**
- **NEPA and EO 12114 (Environmental Effects Abroad of Major Federal Actions) analysis required of PM, approved by CAE**
- **Addresses PM support of Mishap Accident Investigations**
- **Requires Corrosion Prevention Control Plan for ACAT I programs at MS B and C**
- **Requires PMs to employ modular open systems approach to design**
- **Data Management Strategy (DMS) required to assess long-term technical data needs of the system – included in Acquisition Strategy**

Systems Engineering Technical Short Course

(Session 1)



Lesson 6: Needs Assessment

Reference : "Systems Engineering, Principles and Practice," Kossiakoff et. al – Chapter 6

Session 1 Agenda (Jan 27 – 29): Timing

Day	Overview and Concepts of Systems Engineering	Hours*
1	Waters Management kick-off	1/2
	Introduction and overview of entire program (Lesson 1), including Introduction and overview of the projects (15 minutes each)	2
	Presentation and discussion of Concepts of SE (Lessons 2 – 9)	4
	Project assignments, team formations, and working sessions to apply SE concepts to the projects	1
	Day 1 wrap-up: Takeaways and Q&A	1/2
2	Review of day 1 working sessions	1
	Concepts of Systems Engineering (concluded) (Lesson 10 – 14)	4
	Project team working sessions (continued)	2
	Day 2 wrap up and Q&A	1
3	Team working session feedback to class (informal presentations)	1
	Project team working sessions (concluded)	5
	Check-in with class on application of SE concepts to their projects	1
	Session 1 wrap up: Q&A and homework due next session	1

* Times include breaks, and may be changed based on needs of the class

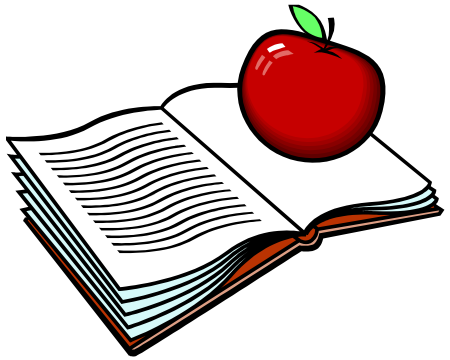
Concepts of Systems Engineering

1. Program Overview
2. What is Systems Engineering?
3. Structure and Environment of Systems
4. Systems Development Process
5. Systems Engineering Processes Overview
6. Systems Engineering in Needs Assessment
7. Risk Management
8. Concept Exploration
9. Concept Definition
10. Advanced Development Stage
11. Engineering Design Stage
12. Integration and Evaluation Stage
13. Production and Deployment
14. Operations, Support, and Disposal

Plus Appendices

Lesson Content:

- **Needs Assessment**
- **Operational deficiency**
- **Technology opportunity**
- **SE in Needs Assessment**



Objectives of the Needs Analysis Phase

- **To identify…**
  - **A valid operational need for a new system**
  - **A feasible approach to meeting the need**

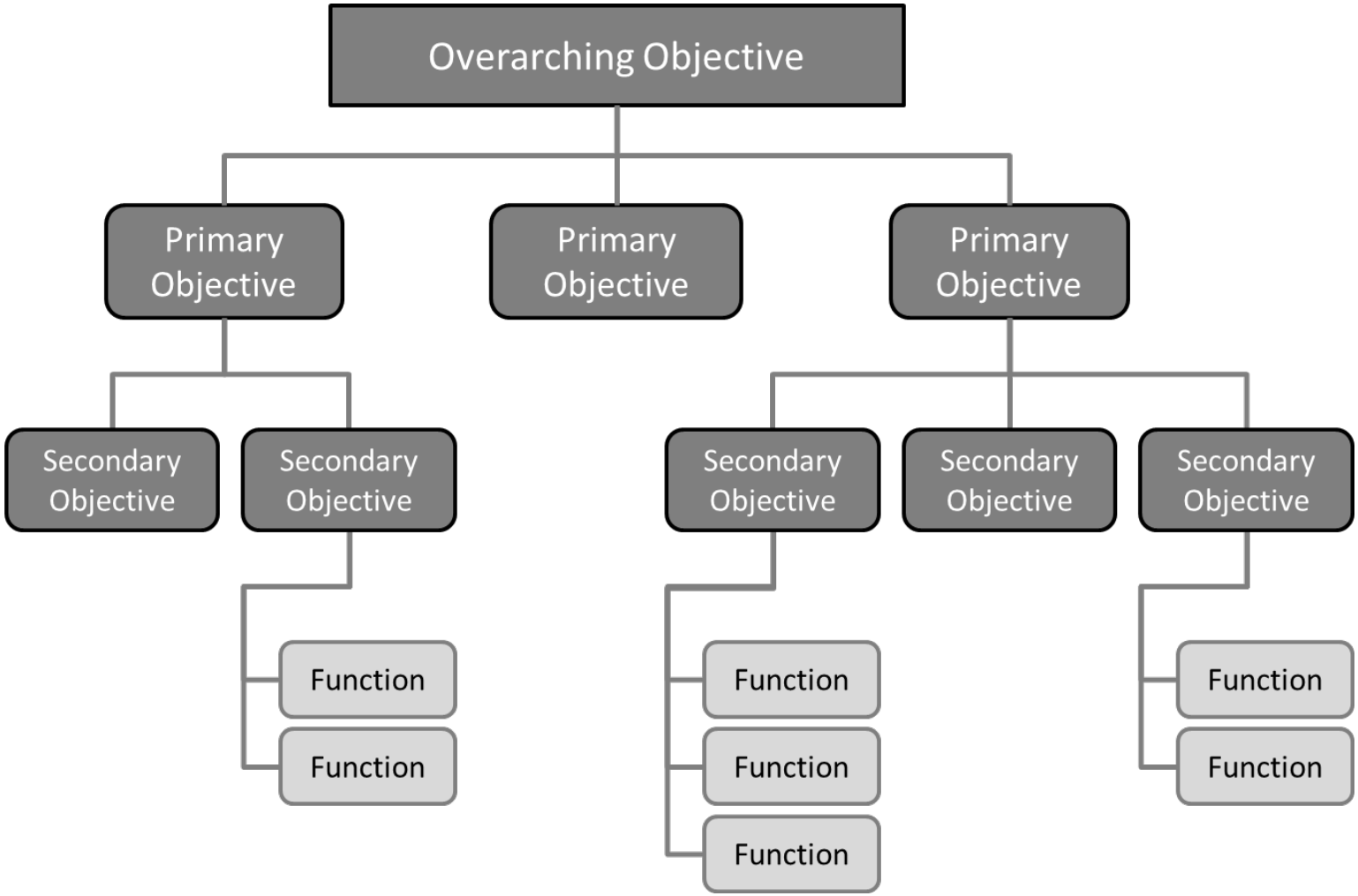
- **A *needs-driven* system development…**
  - **Stems from a deficiency in current system capabilities**
  - **Requires a feasible and affordable technical approach**
  - **Is characteristic of most defense and other government programs**

- **A *technology-driven* system development…**
  - **Stems from a major technological opportunity to better meet a need**
  - **Requires demonstration of practicality and marketability**
  - **Is characteristic of most commercial system developments**

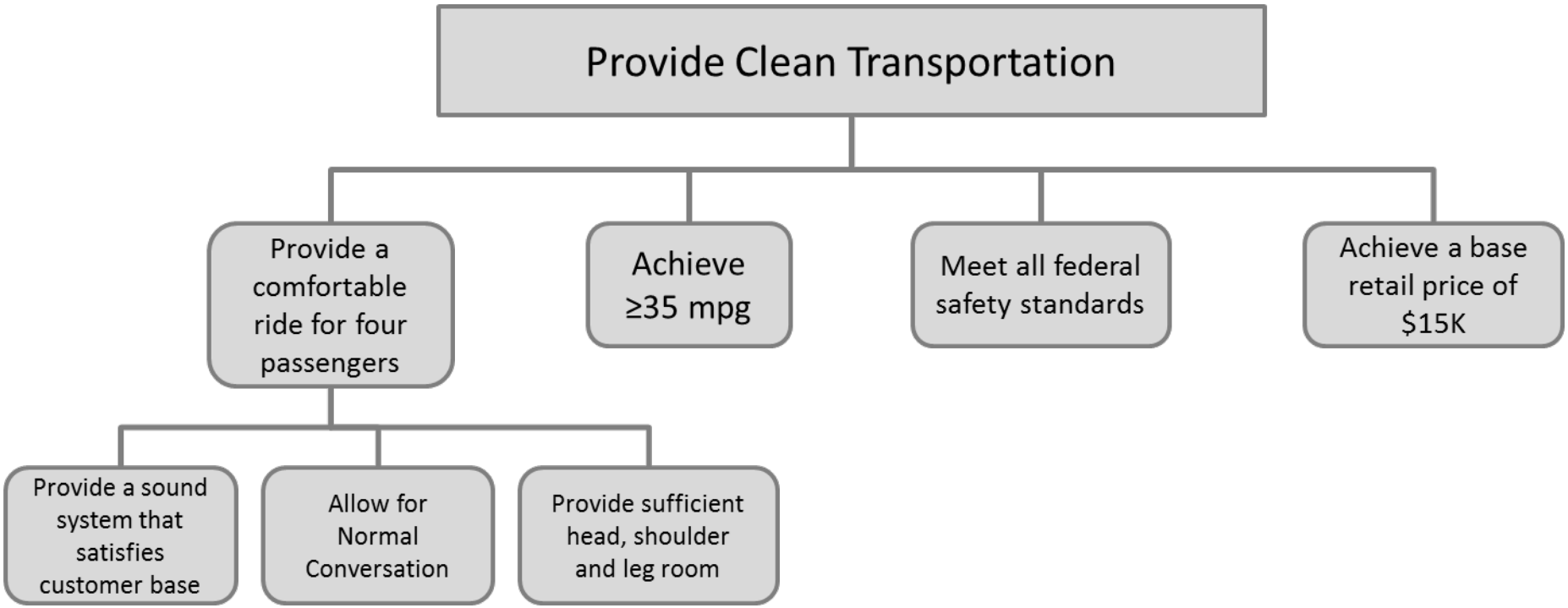
Activities comprising needs analysis

- **Operations analysis—understanding needs for a new system**
- **Functional analysis—deriving functions required to accomplish operations**
- **Feasibility definition—visualizing a feasible implementation approach**
- **Needs validation—demonstrating cost-effectiveness.**

Objectives Tree



Example: An Automobile



The Needs Analysis Phase: Where Do I Begin?

- **Goal:**
  - **Show clearly and convincingly that there is an operational need that requires something to be done to address it**
    - **Define what the need is**
    - **Decide that a new system is required**

What is the problem-to-be-solved?

Needs Analysis Phase

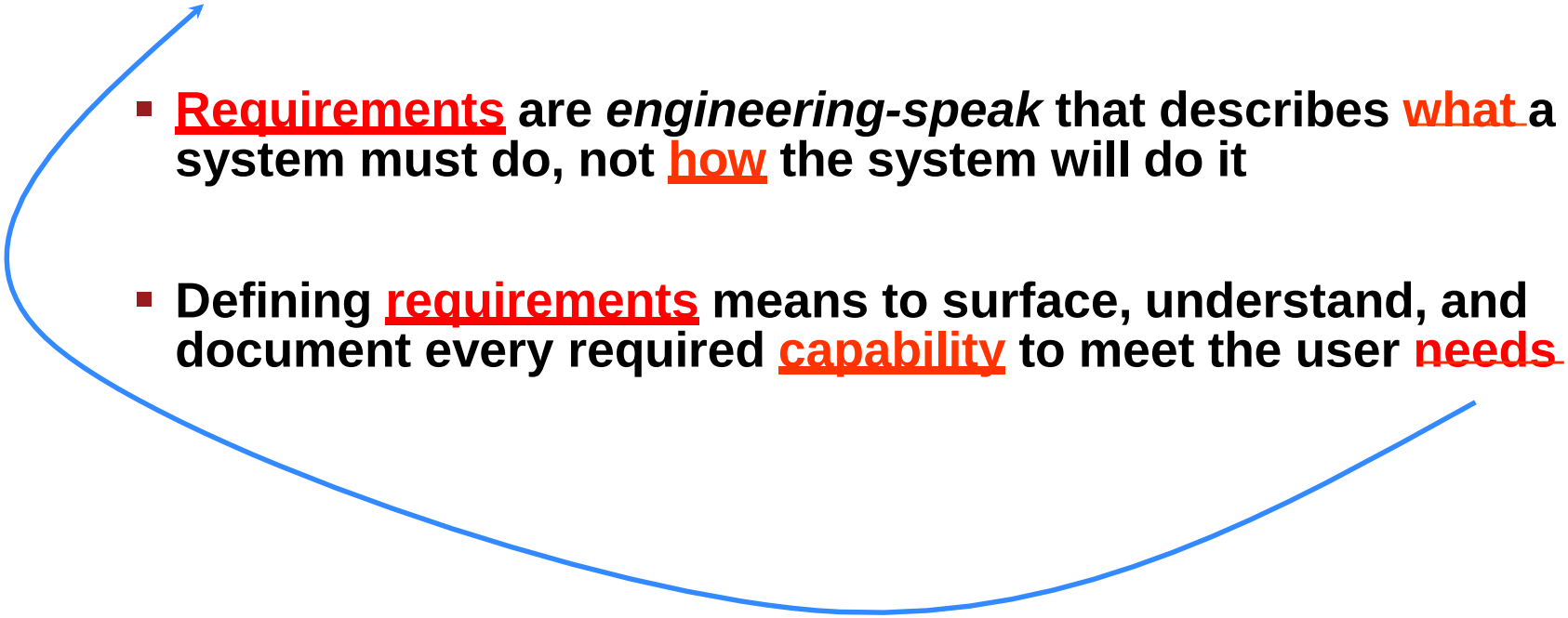
- **Considered the earliest active effort to solve a user problem**
- **At this point, little is known**
  - **Not sure:**
    - **What exactly is needed**
    - **Who is going to solve the problem**
    - **How the problem will be solved**



- **Often starts with a small, informal group**

Needs/Requirements/Capabilities

- **Needs are the user's perspective**
- **Requirements are *engineering-speak* that describes what a system must do, not how the system will do it**
- **Defining requirements means to surface, understand, and document every required capability to meet the user needs**



Concept Development Phase in the System Life Cycle

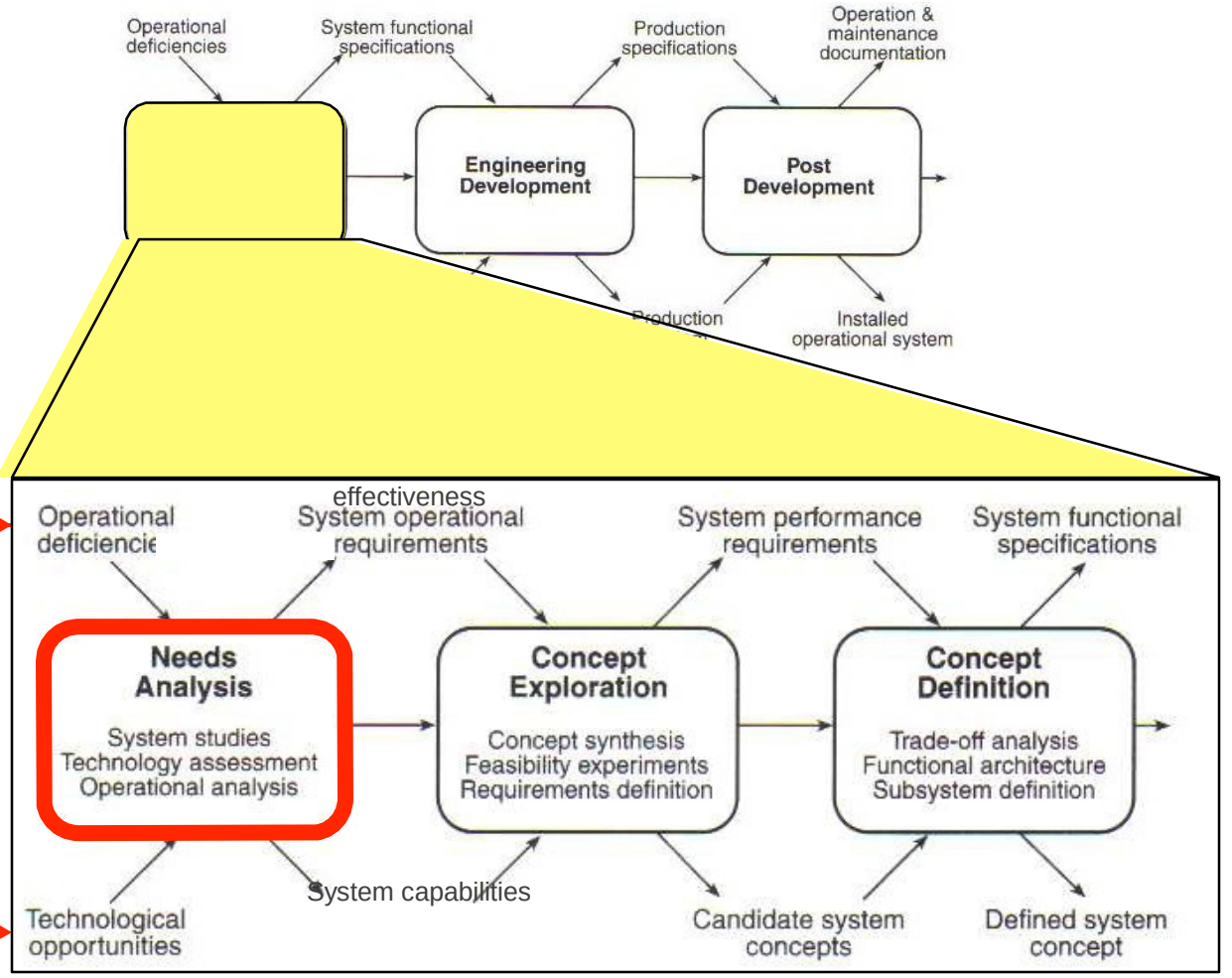


Fig. 3-3 Concept development phases of system life cycle.

Two possible starting points based on an idea that something can not be satisfied by current systems (or there is no current system)

Starting Point: Operational Deficiency

- Users can't do ____with the available system(s)
  - There is no current system that does _____
  - The current systems do _____but not well enough
  - The current system does work the way it is intended to be used
  - "User Pull"

- Causes
  - New capability is desired by users
  - External Events
    - A competitor has fielded a better product
    - New government regulation the current product does not meet
    - Another, related system has changed
  - Internal Events
    - Current system has become unreliable or can no longer be used
    - Current system that does _____has become too costly to operate

Starting Point: Technological Opportunity

- **Technological Opportunity**
  - **New technology allows for new/improved system**
    - **Additional functions**
    - **Better performance**
    - **Higher efficiency**
    - **Lower cost**
  - **"Technology push"**



- **Users may not even know they need it yet**

System Engineering Method in Needs Analysis

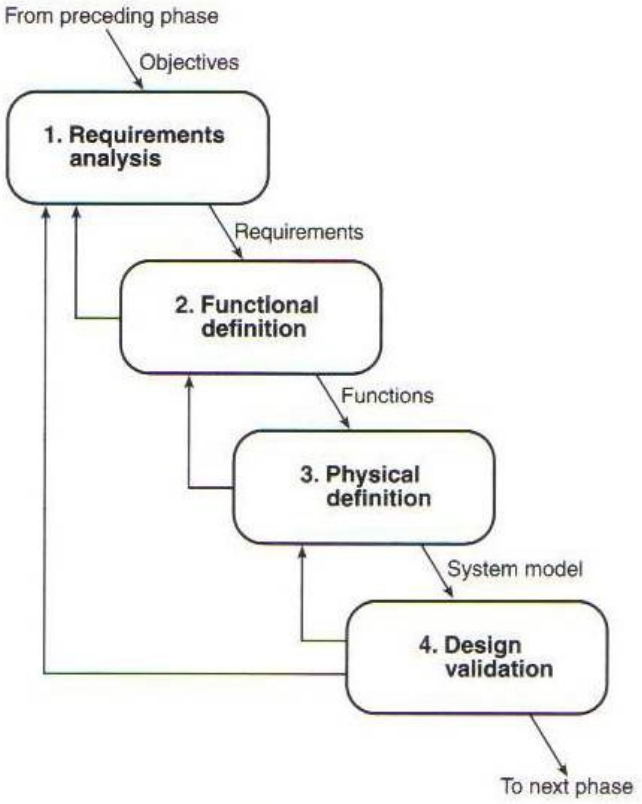


Fig. 3-5 System engineering method top-level flow diagram.

- Every phase applies the system engineering method
- There are many feedback paths
- The process is often iterative

System Engineering Method in Needs Analysis

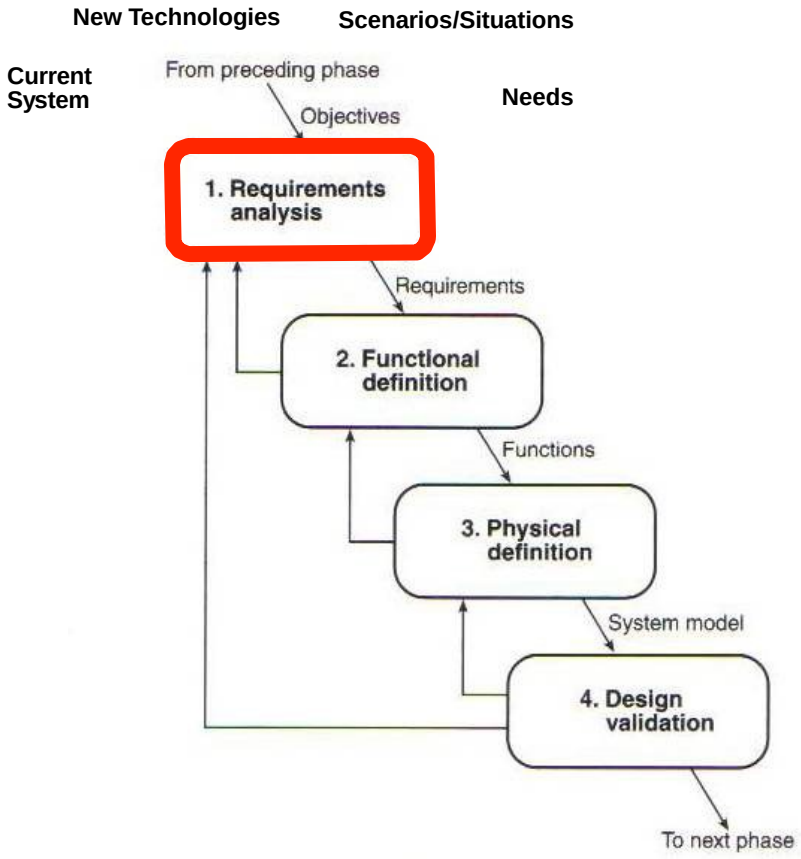


Fig. 3-5 System engineering method top-level flow diagram.

- **Requirements Analysis**
  - **Analyze the needs**
    - **Understand the need**
    - **Understand the value in fulfilling the need**
    - **Understand the need and why existing systems do not solve the problem**
  - **Request clarifications if necessary**

System Engineering Method in Needs Analysis

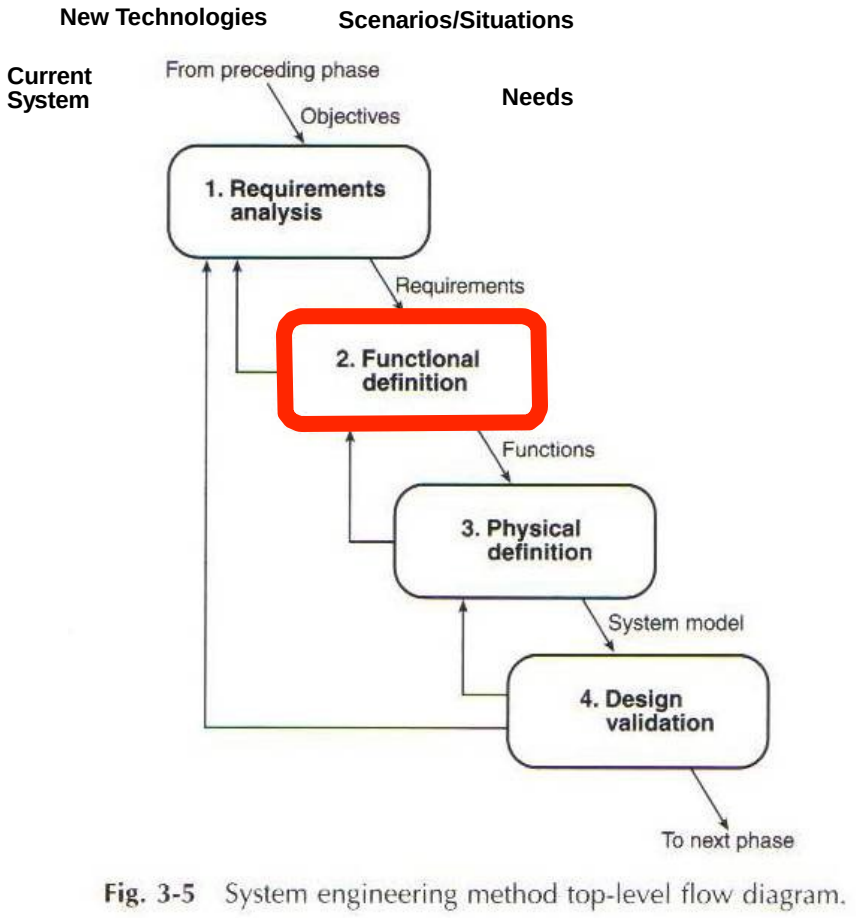


Fig. 3-5 System engineering method top-level flow diagram.

- **Requirements Definition**
  - **Translate operational needs into functions that must be performed**
    - **Organize into related functions**
    - **Define candidate subsystems**
  - **Capture top level <u>operational requirements</u>**
  - **Define top level scope of system**
    - **Where is the system boundary?**

System Engineering Method in Needs Analysis

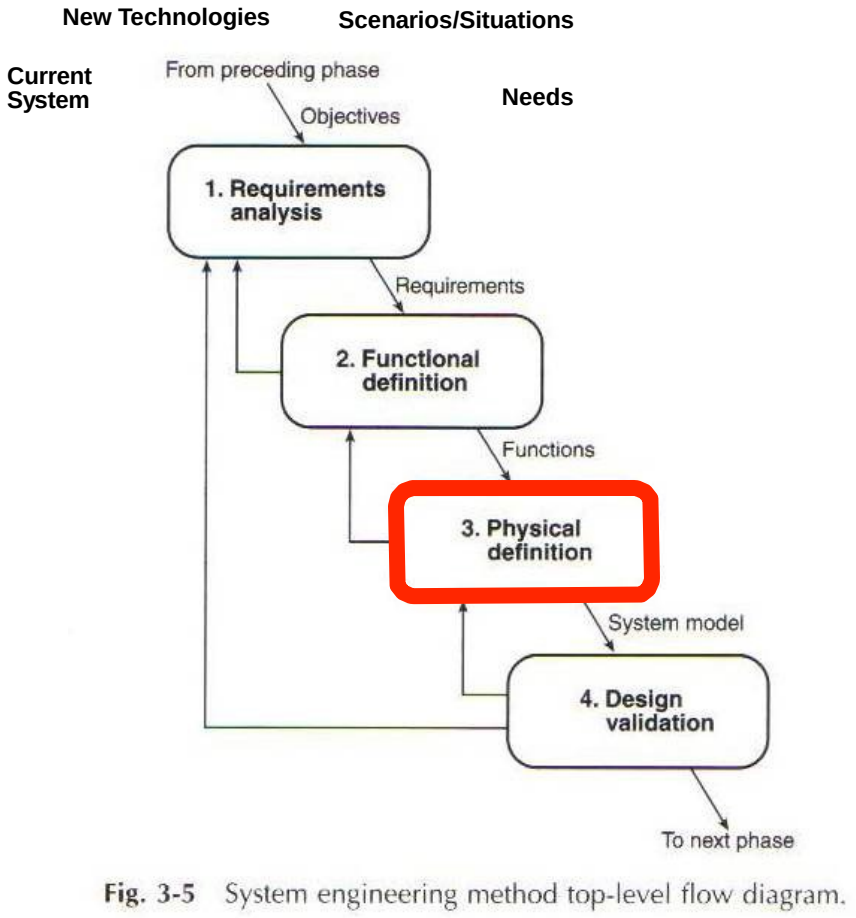


Fig. 3-5 System engineering method top-level flow diagram.

- **Visualize candidate conceptual solutions**
  - **Very top level, often has little detail**
    - **Often look for comparisons with existing systems**
    - **"Like system X except modified as follows..."**
- **Assess feasibility**
  - **Identify new technology developments**
  - **Estimate ability to perform functions, meet performance, cost, schedule**

System Engineering Method in Needs Analysis

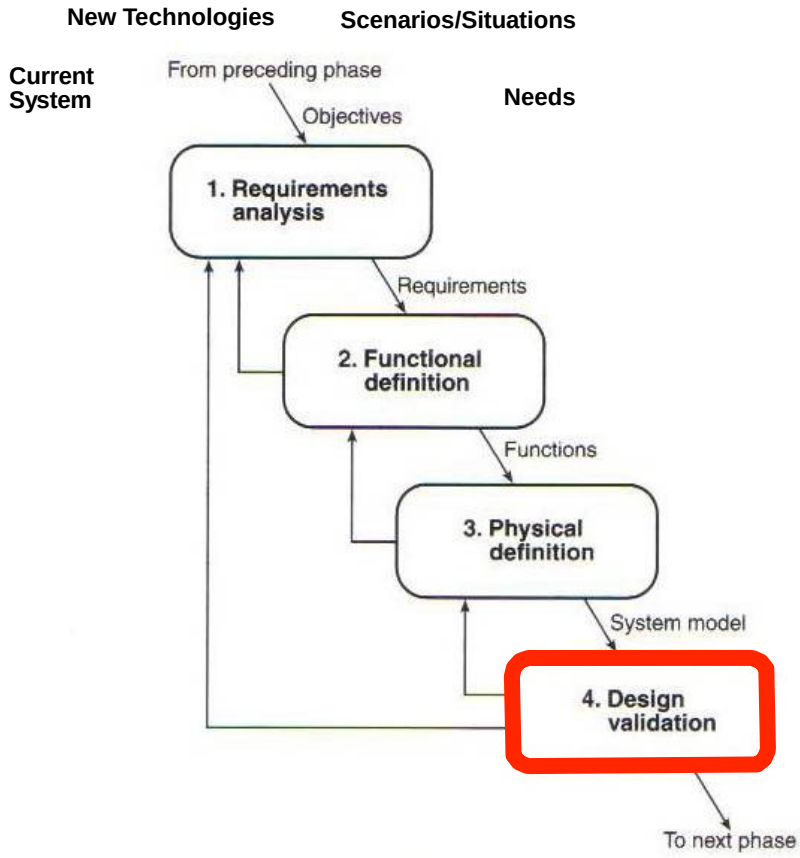
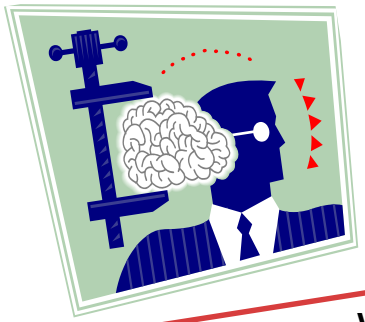


Fig. 3-5 System engineering method top-level flow diagram.

- **Validate Operational Requirements**
  - **Develop a model and show how the new system solves the problem**
    - **Demonstrate effectiveness**
    - **Simulate operational situation**
  - **Create a business case that can justify the new system development**

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?

Back-up Slides

Systems Engineering Technical Short Course

(Session 1)

Lesson 7
Risk Management

Robert B. Wheeler, Instructor

Reference : "Systems Engineering, Principles and Practice," Kossiakoff et. al – Chapter 5.4

Concepts of Systems Engineering

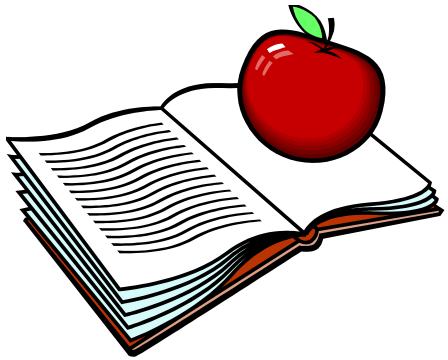
1. Program Overview
2. What is Systems Engineering?
3. Structure and Environment of Systems
4. Systems Development Process
5. Systems Engineering Processes Overview
6. Systems Engineering in Needs Assessment
7. Risk Management
8. Concept Exploration
9. Concept Definition
10. Advanced Development Stage
11. Engineering Design Stage
12. Integration and Evaluation Stage
13. Production and deployment
14. Operations, Support, and Disposal

Plus Appendices

Lesson Content:

- **What is risk**
- **Risk management process**
- **Identifying risk**
- **Mitigating risk**

Exercise: Project Risk



Session 1 Agenda (Jan 27 – 29): Timing

Day	Overview and Concepts of Systems Engineering	Hours*
1	Waters Management kick-off	1/2
	Introduction and overview of entire program (Lesson 1), including Introduction and overview of the projects (15 minutes each)	2
	Presentation and discussion of Concepts of SE (Lessons 2 – 9)	4
	Project assignments, team formations, and working sessions to apply SE concepts to the projects	1
	Day 1 wrap-up: Takeaways and Q&A	1/2
2	Review of day 1 working sessions	1
	Concepts of Systems Engineering (concluded) (Lesson 10 – 14)	4
	Project team working sessions (continued)	2
	Day 2 wrap up and Q&A	1
3	Team working session feedback to class (informal presentations)	1
	Project team working sessions (concluded)	5
	Check-in with class on application of SE concepts to their projects	1
	Session 1 wrap up: Q&A and homework due next session	1

* Times include breaks, and may be changed based on needs of the class

Initial Thoughts

- **How would you describe risk?**







vs.



- **Why would a system engineer care about risk?**

Definitions of Risk

- **Dictionary definition: "possibility of loss or injury : PERIL - at risk : exposed to a usually specified danger or loss <patients *at risk* of infection> "**

- **"Risk: a measure of the potential inability to achieve overall program objectives within defined cost, schedule, and technical constraints"**

Uncertainty and Loss

Risk Management Guide for DoD Acquisition, 5th Edition, Version 2.0, June 2003

Systems Engineering and Risk

- **Risk is a core competency in Systems Engineering**
  - **DoD: Risk Management is mandated**
  - **CMMI**

- **Systems Engineer is the key person to manage risk throughout the lifecycle**
  - **Understands the technical breadth of the project**
    - **Requirements**
    - **Technical approach**
  - **Is close to the program manager**
    - **Cost and schedule considerations**
    - **Access to make decisions**

Three Components of Risk

- An **event** that *might* occur
- A **probability** that the event will occur
- A **consequence** resulting from the occurrence of the event

Expressing A Risk

- **A risk is expressed as an "IF….THEN" statement**

- **Examples:**

  - **"IF the new power amplifier we are designing does not put out enough power, THEN the radar performance will not meet its range requirement."**

  - **"IF the test aircraft is not available when the program is ready for flight testing, THEN the test program will be delayed until a suitable replacement can be acquired."**

  - **"IF the antenna overheats, THEN it will melt, causing a delay of X months to the program and a cost of $Y million (time and effort to get/build another antenna)."**

Types of Risk

- **Technical, examples:**
  - **Critical technology development fails to meet expectations**
  - **Failure to meet a requirement – design error; software bug**
  - **A component failure occurs (e.g., in the prototype)**
  - **Needed assets are not available**
    - **Piece of test equipment fails; test range is unavailable**

- **Programmatic, examples:**
  - **Critical personnel leave the project or are not available when needed**
  - **Other project that your project depends on does not provide a suitable product when it is needed**
  - **Loss of funding/budget**

Probability



- A quantitative value
  - 0 – 100% (0 to 1)
- Ideally, it is a calculated value
- But in reality:
  - Often assessed qualitatively
    - High - Medium - Low

Assessing Probability

- Experience with the part/vendor/technology/developer staff
  - Failure rates
  - New or proven? (new startup part supplier, proven part, etc.)
- Number of dependencies
  - Interfaces
  - Suppliers
  - Climate, time windows for lab/test-range access, etc.
- Margin
  - Easy to meet requirement?
  - Is there excess “X” (time, power, budget, space, etc.) available

Types of Consequences

- **Technical**
  - **Performance shortfall**
  - **Functionality shortfall**
  - **Safety**
- **Cost**
  - **Planned costs will rise**
- **Schedule**
  - **Schedule will slip**

Ranking Consequences

Effects of the consequences depends on the situation

- **Technical**
  - **Priority of requirement**
  - **Pass/Fail requirement vs. level of performance**
  - **Special considerations: safety, catastrophic project failure**
    - **Example: What value is assigned to a consequence of harm/death?**
- **Cost**
  - **Increase in cost vs. total cost of the project**
- **Schedule**
  - **Delay in schedule vs. total time of the project**
  - **Critical milestones (major event or for completing project)**

Some real-world risk assessments translate all consequences into monetary units

Rating Risks

- #1 Challenge – Differences in Views on Risk
  - Perceived vs. actual risk → Subjective assessments
    - My high risk is not your high risk
    - Prime example: Safety & environmental requirements
  - Risk Tolerance
    - Risk averse vs. risk seeking
    - I am willing to risk more than you are



- Don't necessarily have to be perfect…
  - …But be consistent
  - Document your project's approach and ensure the everyone understands how the risks are to be rated and how to rank them
  - As systems engineer – <u>you own the risk process!!!</u>

Managing Risk – An Ongoing Process

Making Decisions to Control the Possible Uncertainty and Loss

1. **Identify the Risks**
2. **Assign probabilities and consequences** } **"Assess Risk"**
3. **Rank the risks in order of importance**
4. **Determine candidate risk mitigations** } **"Decide Response"**
5. **Select and implement risk mitigations**
6. **Monitor progress: does the risk go down over time?**
7. **Start again…**

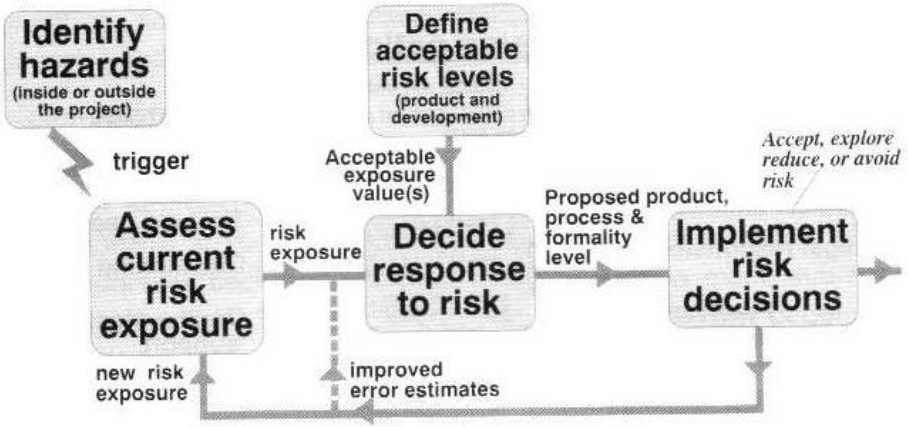
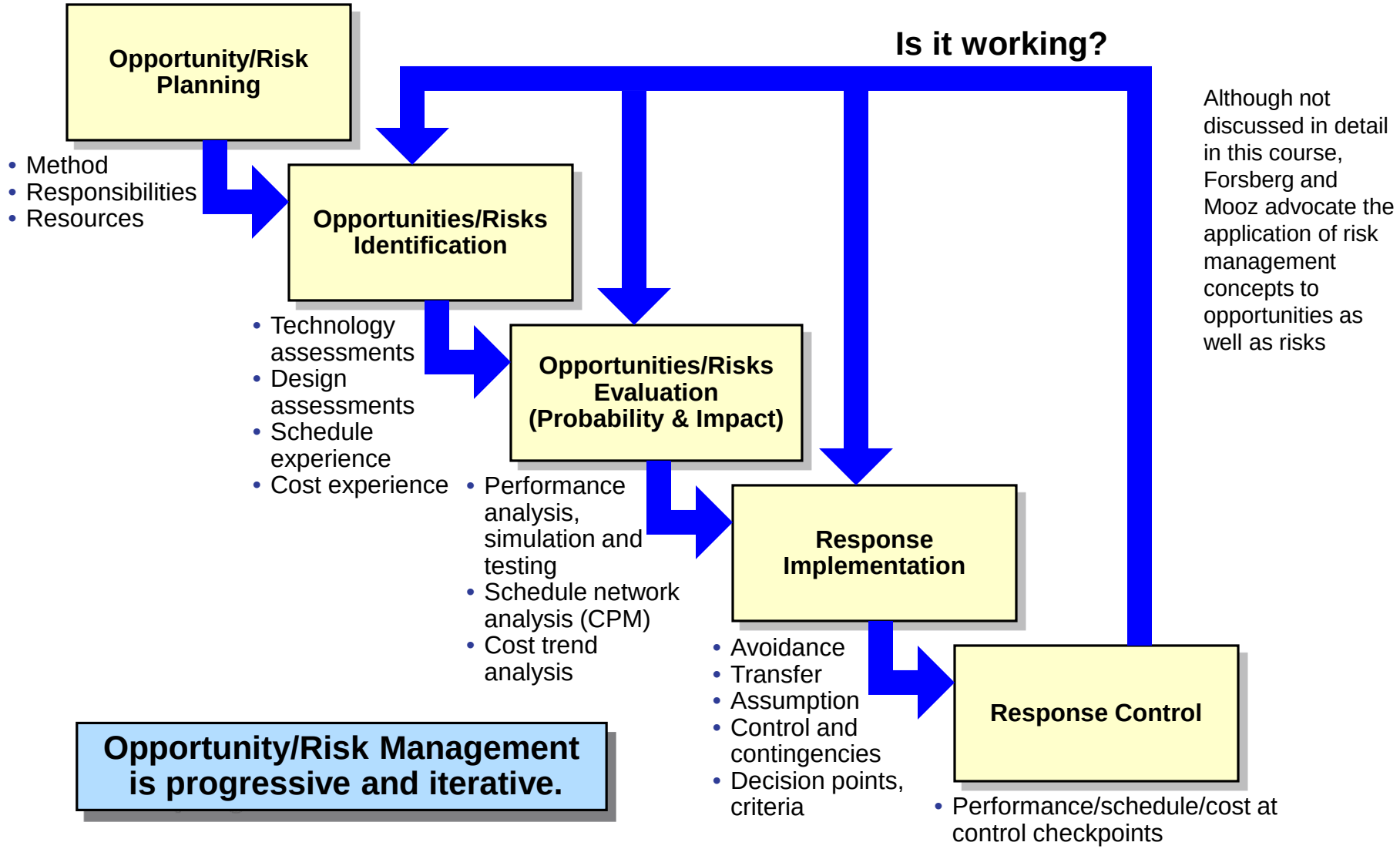


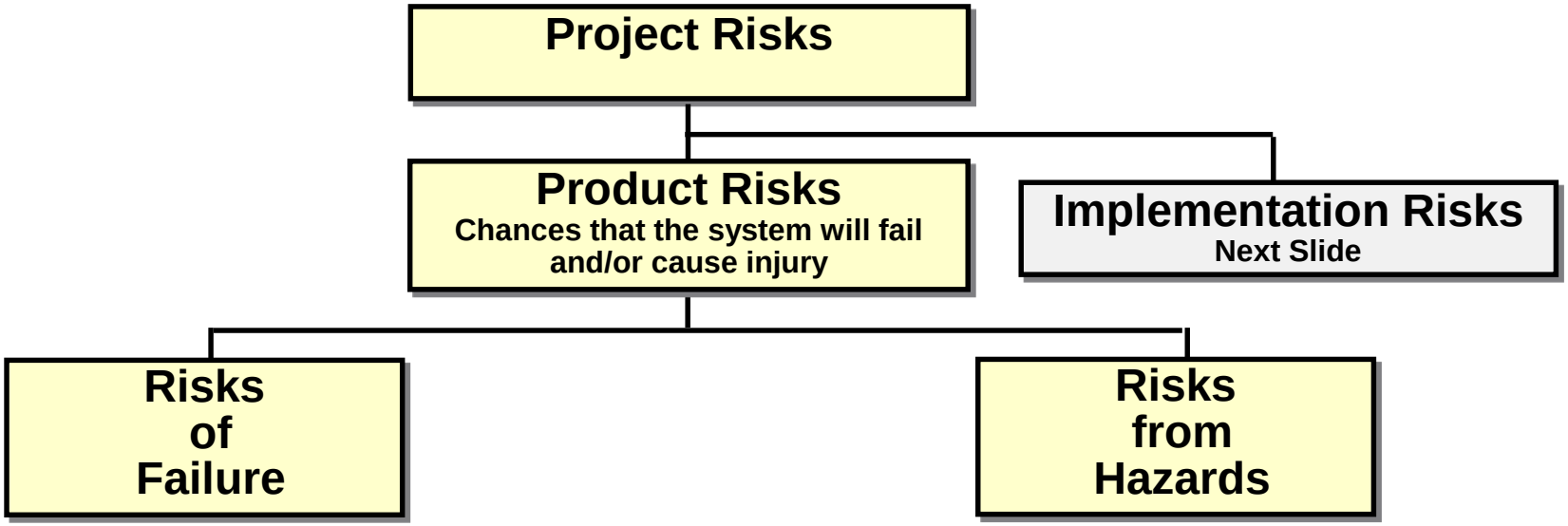
Diagram from R. Stevens, P. Brook, K. Jackson, S. Arnold, "System Engineering: Coping with Complexity" Prentice Hall, 1998 Fig 6-10

Opportunity and Risk Management Process



Reference : “Visualizing Project Management, ” Forsberg, Mooz, et. Al. – Chapter 13

Identifying Project Risks - Product Risks



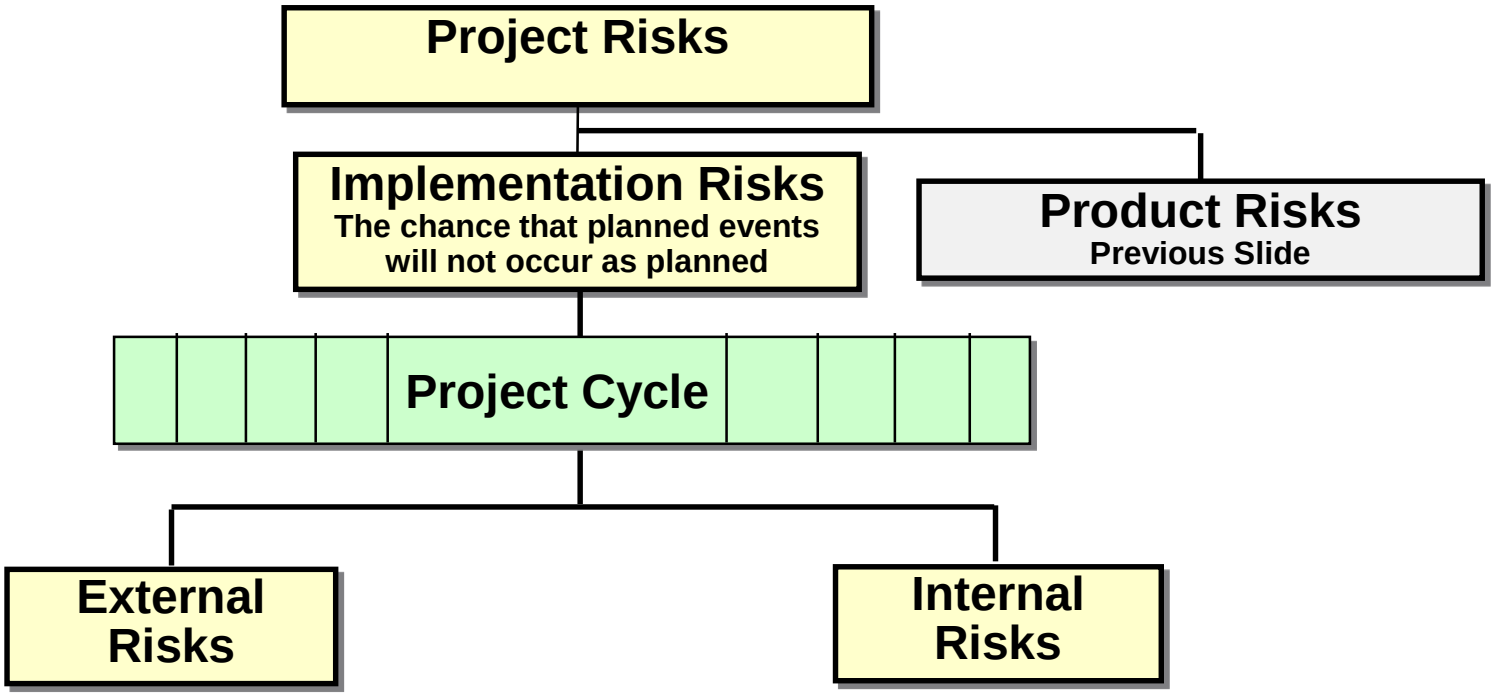
- Single point failure modes
- Limited life items
- Expendables
- Untestables
- Process sensitive

- Radiation
- Explosives
- Potential energy
- High voltage
- Toxins

These can lead to personal injury or project failure.

Reference : “Visualizing Project Management, ” Forsberg, Mooz, et. Al. – Chapter 13

Identifying Project Risks - Implementation Risks



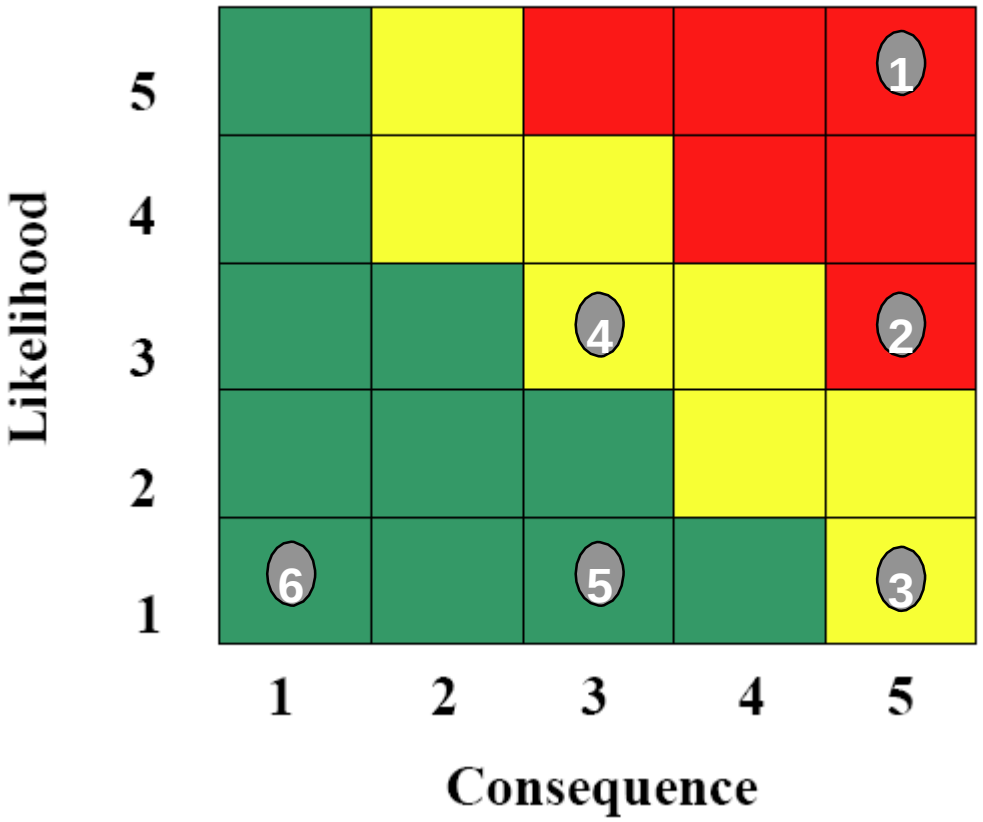
- Funding
- Market Support
- Technology
- Customer Performance
- Supplier Performance

- Planning
- Resources
- Tools
- Skills
- Supplier Performance

These can lead to project cancellation.

Reference : “Visualizing Project Management, ” Forsberg, Mooz, et. Al. – Chapter 13

Ranking Risk



1. Most Important Risk
2. Very Important Risk
3. Not likely, but catastrophic risk
4. Middle of the road risk
5. Lower concern risk
6. Least concern risk

Where should you spend the most effort?

Likelihood (Probability/Frequency)

Likelihood

Level	Likelihood	Probability of Occurrence
1	Not Likely	~10%
2	Low Likelihood	~30%
3	Likely	~50%
4	Highly Likely	~70%
5	Near Certainty	~90%

Consequence (Impact/Severity)

Consequence

Le-vel	Technical Performance	Schedule	Cost
1	Minimal or no consequence to technical performance	Minimal or no impact	Minimal or no impact
2	Minor reduction in technical performance or supportability, can be tolerated with little or no impact on program	Able to meet key dates. **Slip < * month(s)**	Budget increase or unit production cost increases. **< ** (1% of Budget)**
3	Moderate reduction in technical performance or supportability with limited impact on program objectives	Minor schedule slip. Able to meet key milestones with no schedule float. **Slip < * month(s)** **Sub-system slip > * month(s) plus available float.**	Budget increase or unit production cost increase **< ** (5% of Budget)**
4	Significant degradation in technical performance or major shortfall in supportability; may jeopardize program success	Program critical path affected. **Slip < * months**	Budget increase or unit production cost increase **< ** (10% of Budget)**
5	Severe degradation in technical performance; Cannot meet KPP or key technical/supportability threshold; will jeopardize program success	Cannot meet key program milestones. **Slip > * months**	Exceeds APB threshold **> ** (10% of Budget)**

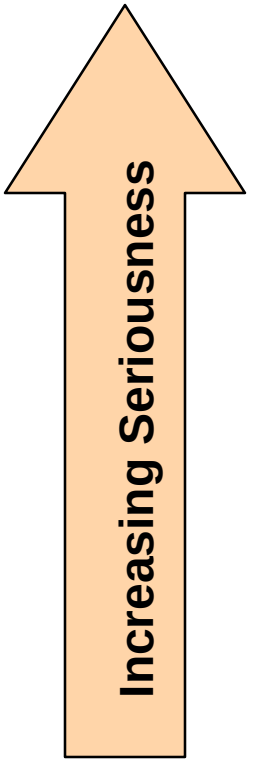
Risk Mitigation

- **Methods**
  - **Reduce probability**
  - **Reduce consequence**

- **End Goal**
  - **Develop a plan that reduces the risks to an acceptable level**

Risk Mitigation – Reducing Probability

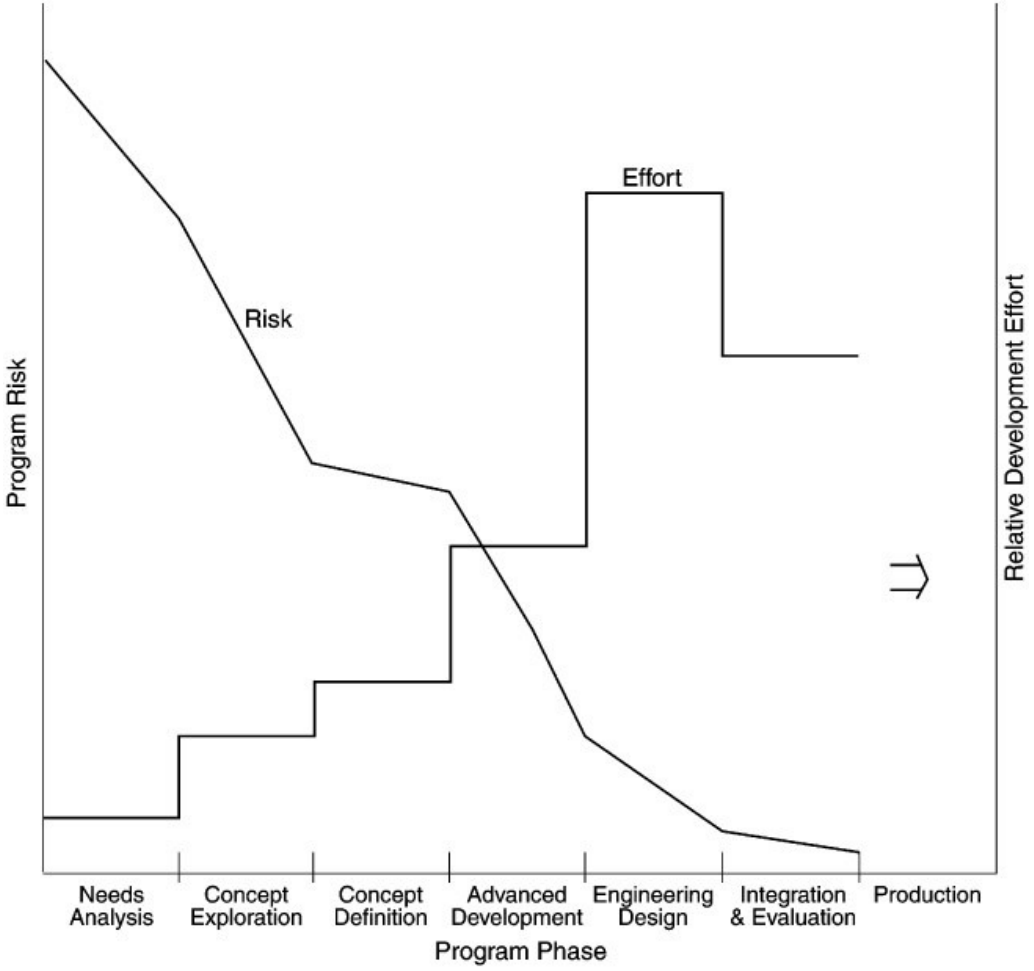
- Develop an alternate backup design
- Consider relief of requirements
  - Scale back on high risk requirements in development
- Prototype – do testing and get feedback early
- Perform analysis and testing of critical design items
- Add special oversight to high risk subsystems/components
- Perform additional technical and management reviews
- Plan ahead – arrange for needed resources early
  - Parts (especially long lead parts), staffing, vendors



Risk Mitigation – Reducing Consequence

- Be prepared to relieve requirements if necessary
  - Close user involvement, prioritized requirements
- Develop project plans that allows for recovery of failures
  - Include time/resources for second iteration of design
  - Incremental design approach
  - Plan to address high risk issues early
- Plan a management reserve of schedule and budget
- Choose flexible design approaches
  - Standards, commodity components

Reducing Risk over the Life Cycle



- **Risk is reduced as we learn more and the number of uncertainties is reduced**
- **Goal: Narrow or eliminate the biggest uncertainties BEFORE we begin the main development part of the effort which carries the largest resource commitment of staff and money**

Risk Analysis in Concept Exploration

- One of the criteria that is evaluated *for each concept* is risk
  - Identify are the sources of risk (technical, programmatic)
    - Perhaps there is something that can be done to manage the risk
  - Helps to assess the likelihood the project will fail
    - Sponsors do not want to end up with a failed project
- Risk is considered as a criteria in comparing alternatives

- What happens if a candidate solution is considered to be "Higher" risk?
  - Likely will have a larger uncertainty factor added to the cost/ schedule estimate to account for the uncertainty
  - Development plan may include additional investigation efforts or experiments to provide confidence the concept will succeed
    - Add decision points in the schedule to continue or stop depending on the experimental results

Risk Analysis in Concept Exploration (concluded)

- A concept may be rejected as a viable concept purely on the basis of risk
  - Depends on the customer's risk tolerance

- Risk tolerance often depends on the urgency of the need and number of available concepts that can provide a solution

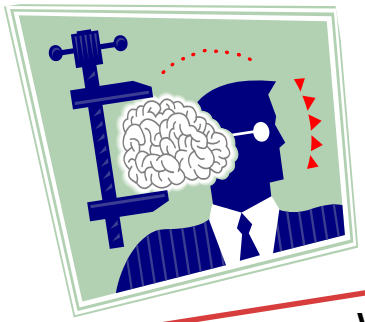
Exercise: Risk Status of Your Projects



Instructor will provide directions for the exercise

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



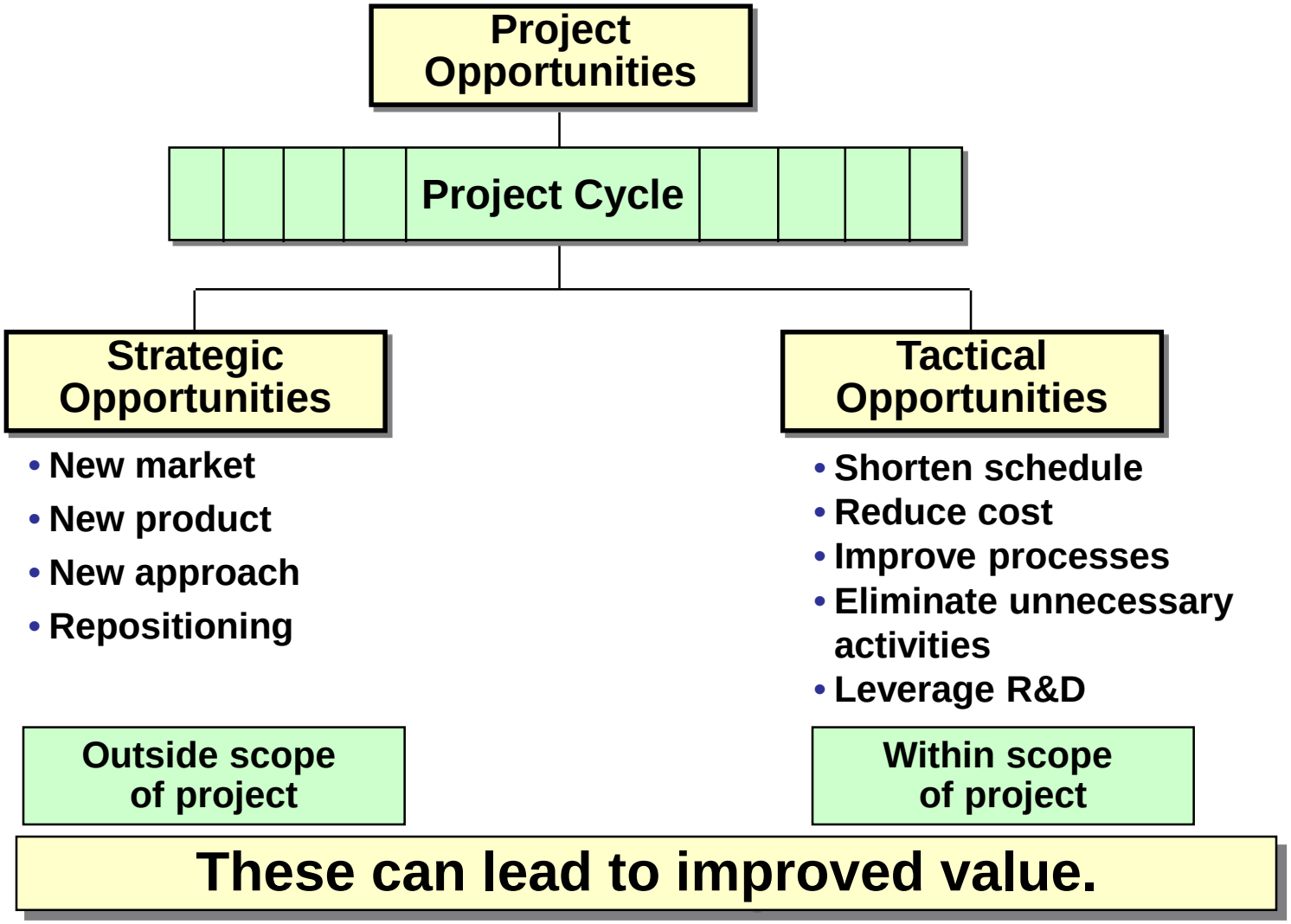
All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?

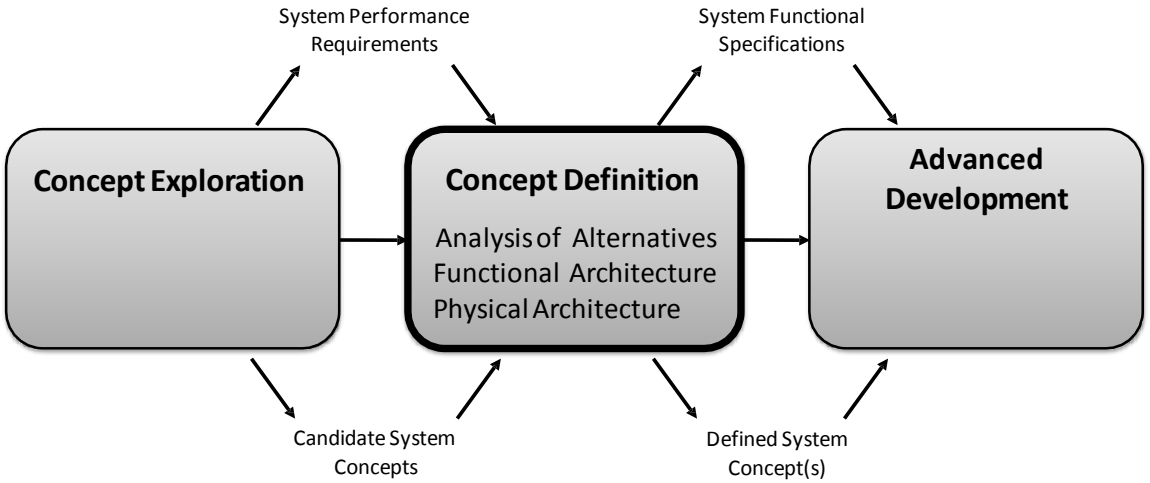
Back-up Slides

Identifying Project Opportunities – Two Categories



Reference : “Visualizing Project Management, ” Forsberg, Mooz, et. Al. – Chapter 13

Systems Engineering Technical Short Course (Session 1)



Lesson 9: Concept Definition

Reference : "Systems Engineering, Principles and Practice," Kossiakoff et. al – Chapter 8

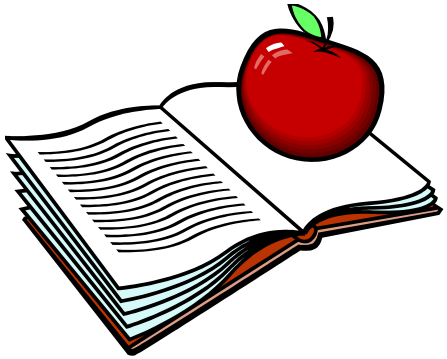
Session 1 Agenda (Jan 27 – 29): Timing

Day	Overview and Concepts of Systems Engineering	Hours*
1	Waters Management kick-off	1/2
	Introduction and overview of entire program (Lesson 1), including Introduction and overview of the projects (15 minutes each)	2
	Presentation and discussion of Concepts of SE (Lessons 2 – 9)	4
	Project assignments, team formations, and working sessions to apply SE concepts to the projects	1
	Day 1 wrap-up: Takeaways and Q&A	1/2
2	Review of day 1 working sessions	1
	Concepts of Systems Engineering (concluded) (Lesson 10 – 14)	4
	Project team working sessions (continued)	2
	Day 2 wrap up and Q&A	1
3	Team working session feedback to class (informal presentations)	1
	Project team working sessions (concluded)	5
	Check-in with class on application of SE concepts to their projects	1
	Session 1 wrap up: Q&A and homework due next session	1

* Times include breaks, and may be changed based on needs of the class

Session 1 Agenda (Jan 27 – 29): Timing

1. Program Overview
2. What is Systems Engineering?
3. Structure and Environment of Systems
4. Systems Development Process
5. Systems Engineering Processes Overview
6. Systems Engineering in Needs Assessment
7. Risk Management
8. Concept Exploration
9. **Concept Definition**
10. Advanced Development Stage
11. Engineering Design Stage
12. Integration and Evaluation Stage
13. Production and Deployment
14. Operations, Support, and Disposal



Lesson Content:
- **Concept Definition**
- **SE in Concept Definition**
- **SE activities**

Plus Appendices

* Times include breaks, and may be changed based on needs of the class

Concept Development Phase in the System Life Cycle

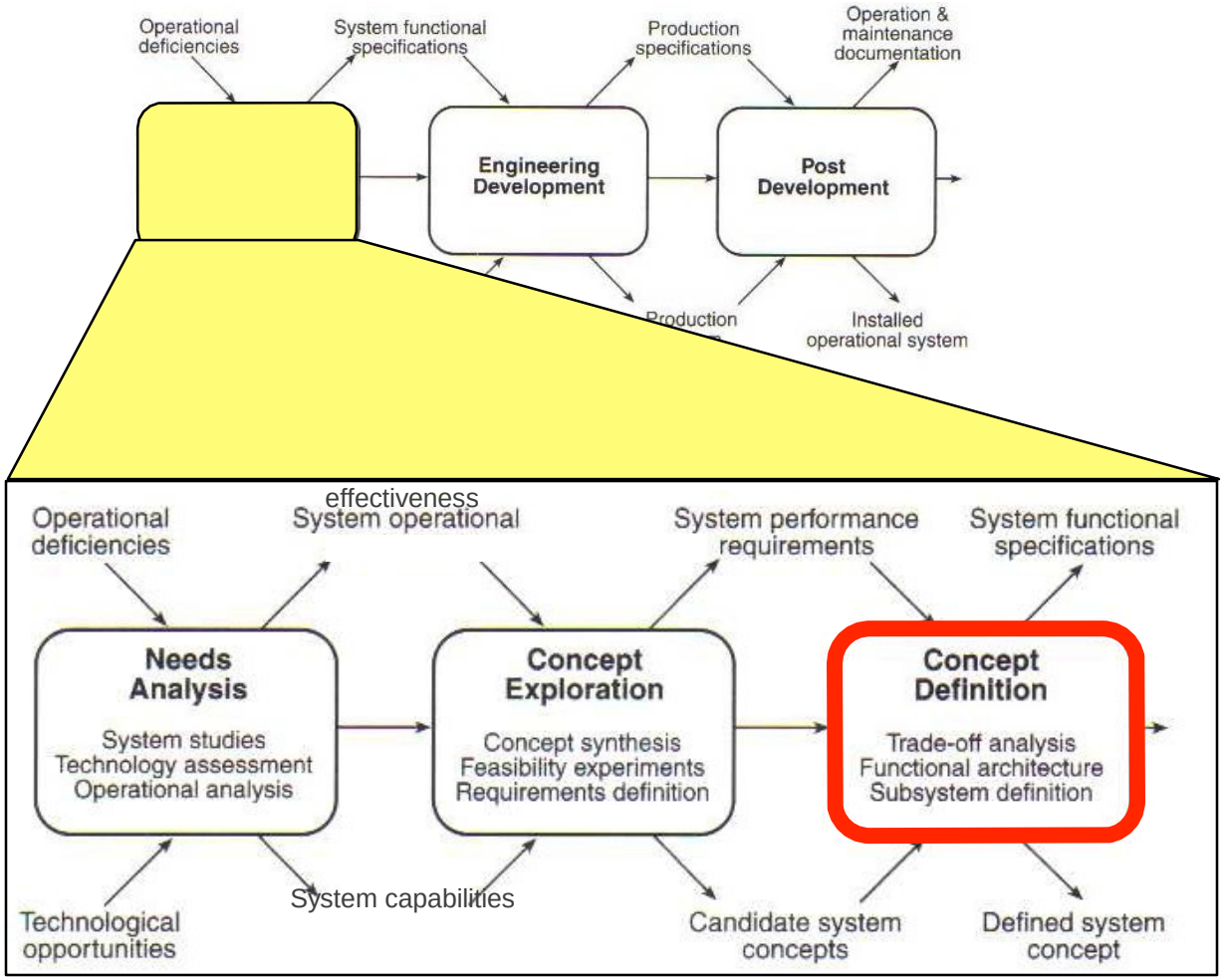
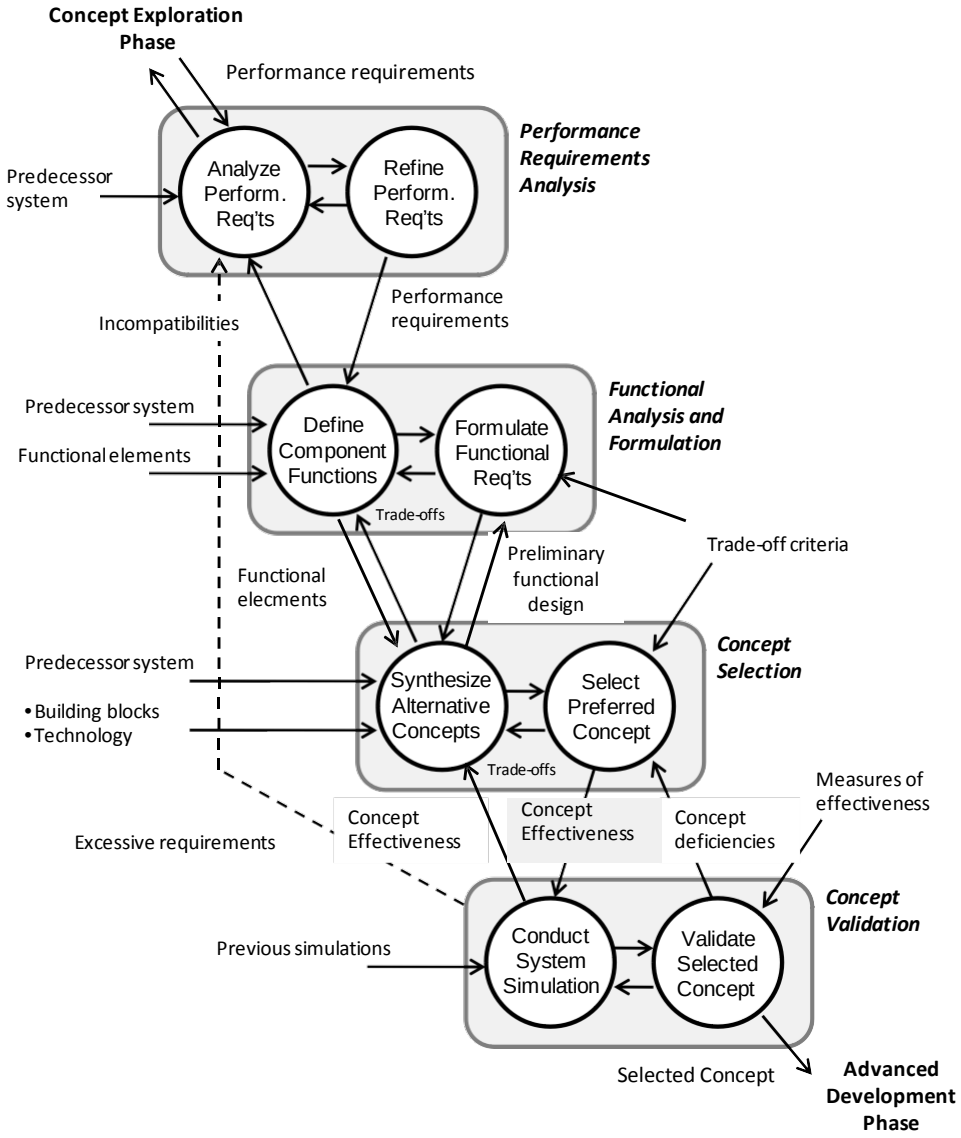


Fig. 3-3 Concept development phases of system life cycle.

Concept Definition Phase Flow Diagram



Status of System Materialization

Level \ Phase	*Concept Development*			*Engineering Development*		
	Needs Analysis	Concept Exploration	Concept Definition	Advanced Development	Engineering Design	Integration & Evaluation
System	Define system capabilities & effectiveness	Identify, explore and synthesize concepts	**Define selected concept with specifications**	Validate Concept		Test & Evaluate
Subsystem		Define requirements & ensure feasibility	**Define functional & physical architecture**	Validate subsystems		Integrate & test
Component			**Allocate functions to components**	Define specifications	Design & test	Integrate & test
Sub-Component				Allocate functions to subcomponents	Design	
Part					Make or buy	

Visualize

Concept Definition: Making a Major Choice

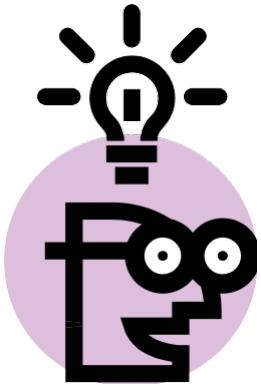
- **Goals:**
  - **Evaluate the alternative concepts and make a decision as to which one to begin implementing**
- **Inputs:**
  - **Candidate system concepts**
  - **System performance specifications**
- **End Result:**
  - **Defined system concept**
  - **System functional specifications**
  - **Cost & schedule estimates for building the solution**

Also Needed for Concept Definition

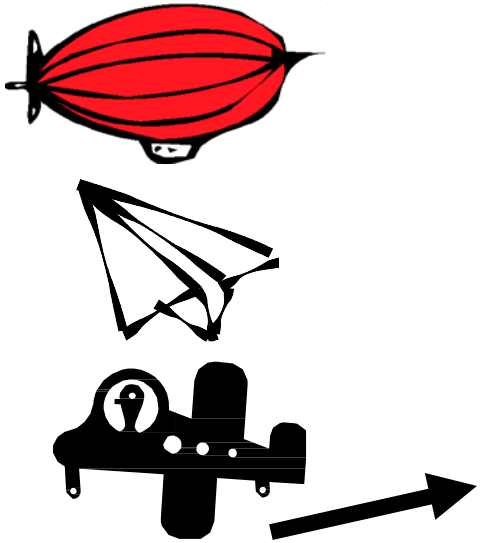
- **Value the decision makers place on the solution**
  - **Resources that will be made available**
  - **Constraints identified (especially cost & schedule)**

Getting Started…

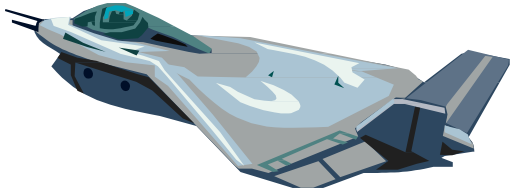
- Take what you have learned in Concept Exploration and go to the next level of detail
- Focus is to flesh out the alternatives to enough detail so one can make a choice between them



Brainstorm Concepts



Concept Exploration



Concept Definition

System Engineering Method in Concept Definition

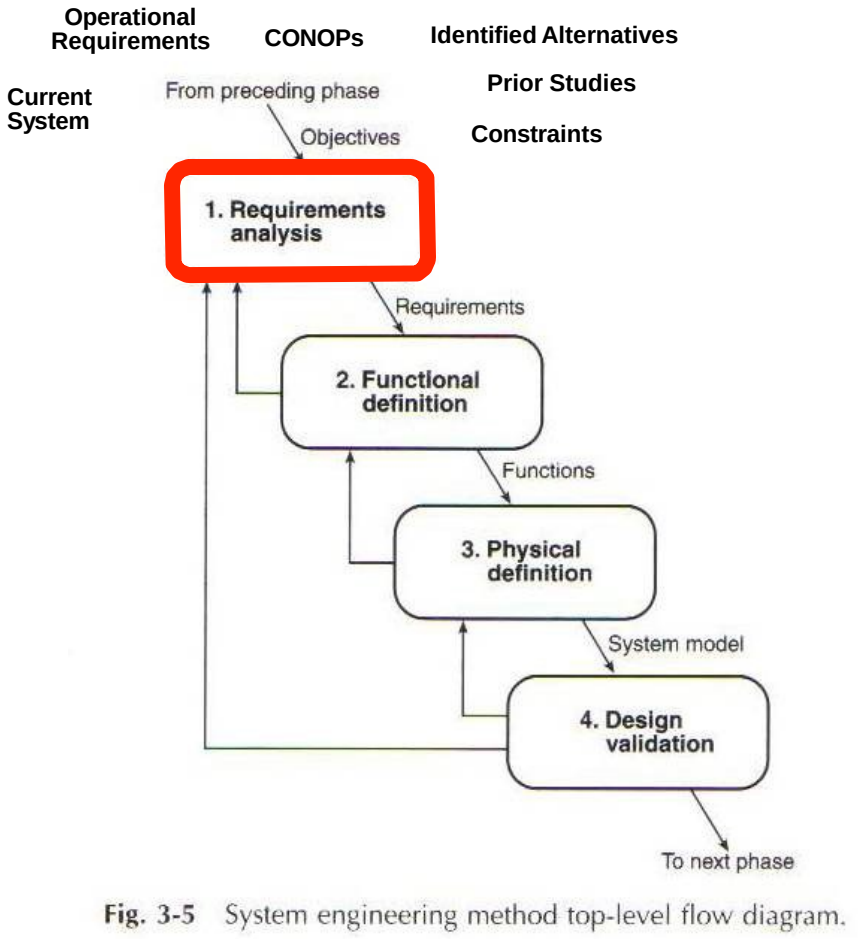


Fig. 3-5 System engineering method top-level flow diagram.

- **Requirements Analysis**
  - **Analyze and ensure understanding of *performance* requirements**
    - **Relate to operational objectives and the entire lifecycle scenario**
      - **Field usage**
      - **Maintenance**
      - **How it will be repaired/supported**
      - **How it will be upgraded/disposed**
  - **Refine requirements as needed**
    - **Capture previously unstated constraints**
    - **Quantify qualitative requirements**

System Engineering Method in Concept Definition

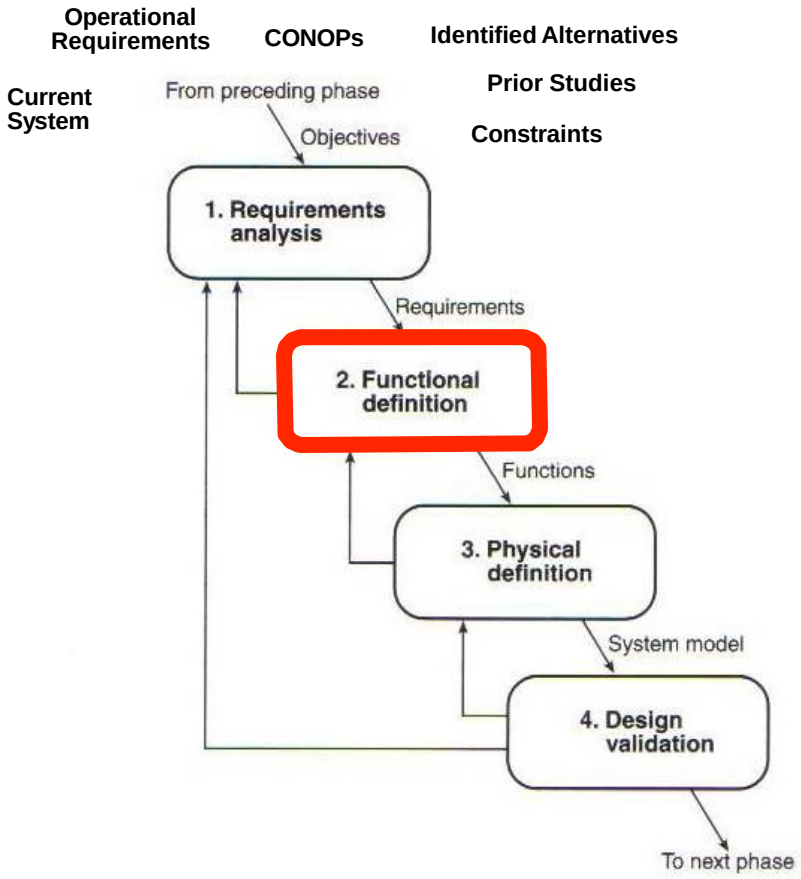


Fig. 3-5 System engineering method top-level flow diagram.

- **Functional Definition**
  - **For the concepts: allocate subsystem functions to major components in terms of functional elements**
    - **Identify major components**
    - **Formulate key component performance requirements**

System Engineering Method in Concept Definition

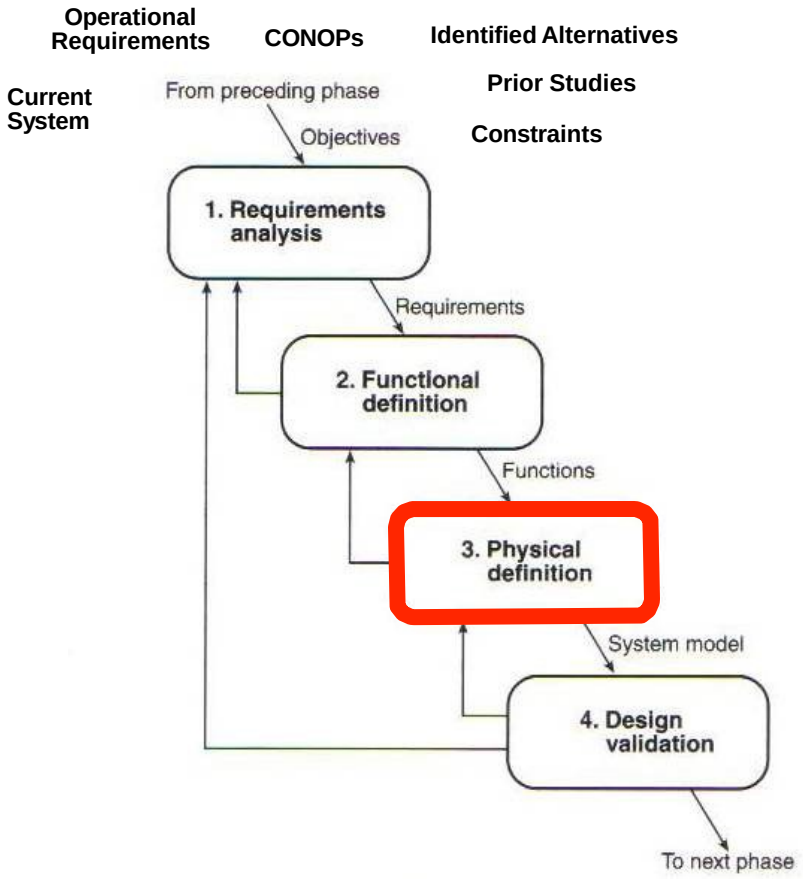


Fig. 3-5 System engineering method top-level flow diagram.

- **Physical Definition**
  - Develop a conceptual design for each alternative so it can be evaluated
    - Candidate hardware/software components
    - Develop enough detail to estimate costs and to estimate the alternative's performance against the requirements
  - Perform a trade study on the alternatives
  - Recommend a selection

Amount of detail will depend on the size, complexity, and cost

The Real Work

System Engineering Method in Concept Definition

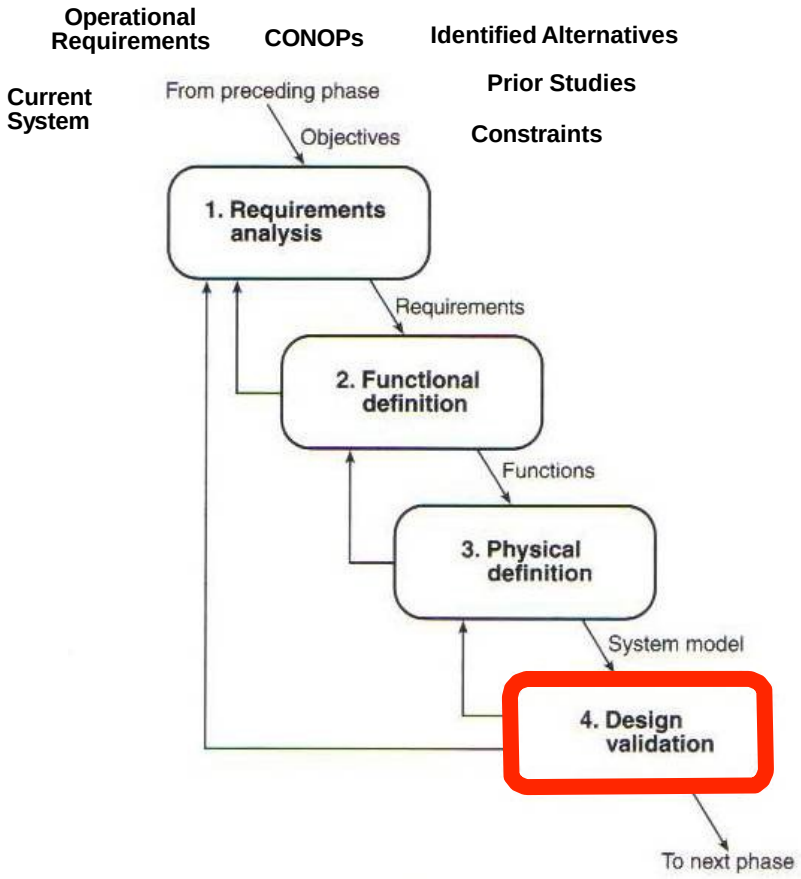


Fig. 3-5 System engineering method top-level flow diagram.

- **Validate Results**
  - **Conduct system analyses and/ or develop simulations to validate the chosen concept against the requirements**
  - **Refine the concept if needed**

Choosing an Alternative-
Factors (Criteria) often part of the Decision:



- **Performance**
  - **Against the user's Key Performance Parameters (most important requirements) from the user requirements?**
  - **Against important system requirements we have derived in our analysis**



- **Cost – Life cycle cost estimate**
  - **Cost to develop, produce, field, operate, maintain/sustain, and dispose**
    - **DoD/Government: Many of the lifecycle phases are funded by designated monies in the Government budgeting process (so called "colors of money")**
      - **Development money can not be spent to produce and field the system, etc.**



- **Schedule**
  - **Decided by the user & sponsor**
  - **DoD/Government consideration: DoD normally budgets 5-6 years in advance. You need to get the right color money at the right time**



- **Risk**
- **Other differentiators unique to the situation**
  - **One-upping competition, safety, public perception, perceived value**

Other Tasks for the Systems Engineer in Concept Definition

- Estimate risk
- Provide inputs to estimate the needed schedule
- Provide inputs to estimate the cost

> At this time, these may be very rough estimates

- Recommend who is needed to support the project
  - Specialists
- Develop technical documentation for follow-on efforts
  - Plan for system engineering efforts & processes
  - Scope of overall engineering efforts
- Recommend next course of action
  - Perform an Advanced Development phase?
    - …and do what?
  - Go straight to System Development?

Other Outputs of Concept Definition

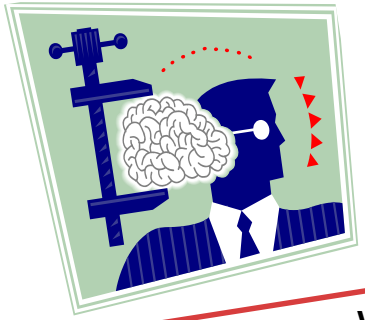
- Selected concept
- System functional specifications
  - Function List
- Information for the program manager
  - Candidate actions
  - Plans and information needed for follow-on program planning
    - System Engineering Plan (SEP) or equivalent
  - Risk Assessment
  - Schedule
  - Cost



- These are Improved Estimates
  - Finer grain of detail
    - e.g., uses work breakdown structure
  - Use of more equivalent references to substantiate estimates

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?

What's Next?

Day	Overview and Concepts of Systems Engineering	Hours*
1	Waters Management kick-off	1/2
	Introduction and overview of entire program (Lesson 1), including Introduction and overview of the projects (15 minutes each)	2
	Presentation and discussion of Concepts of SE (Lessons 2 – 9)	4
	Project assignments, team formations, and working sessions to apply SE concepts to the projects	1
	Day 1 wrap-up: Takeaways and Q&A	1/2
2	Review of day 1 working sessions	1
	Concepts of Systems Engineering (concluded) (Lesson 10 – 14)	4
	Project team working sessions (continued)	2
	Day 2 wrap up and Q&A	1
3	Team working session feedback to class (informal presentations)	1
	Project team working sessions (concluded)	5
	Check-in with class on application of SE concepts to their projects	1
	Session 1 wrap up: Q&A and homework due next session	1

* Times include breaks, and may be changed based on needs of the class

<= New slide since printing

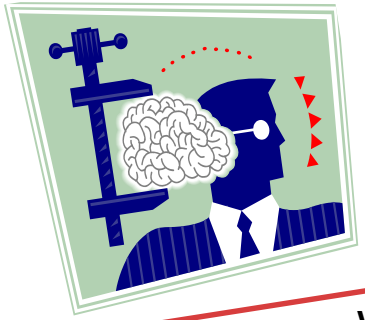
The First Project Group Session

- **Objective: Form your groups and begin organizing the work on the project**
- **Suggested agenda:**
  - **Meet as a project group**
  - **Get to know each other**
  - **Review the project assignment**
  - **Review the homework assignments with emphasis on assignment number 1 due in February**
  - **What were the takeaways from the material covered today that you need to implement on your projects?**
  - **What challenges do you see so far?**

- **Be ready to briefly report on the results of this working session to the rest of the class tomorrow morning at beginning of class**

Day 1 Wrap-up Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



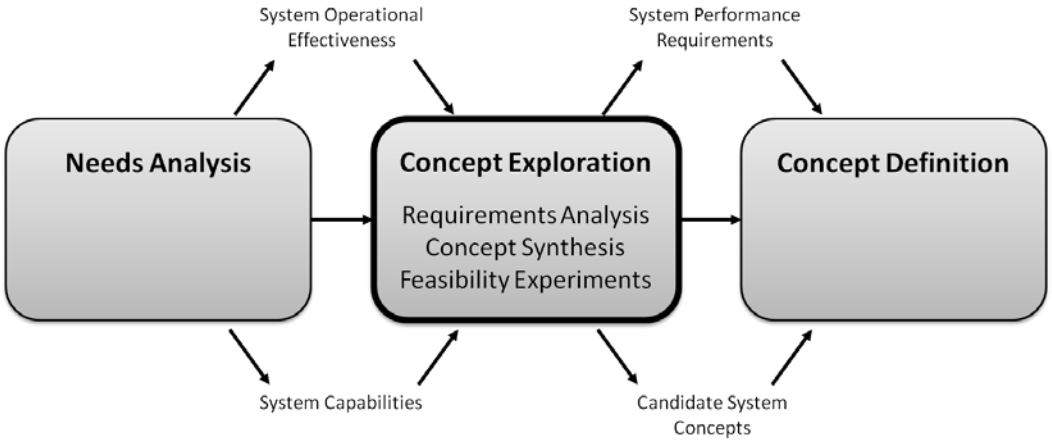
All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?

Back-up Slides

Systems Engineering Technical Short Course (Session 1)



Lesson 8: Concept Exploration

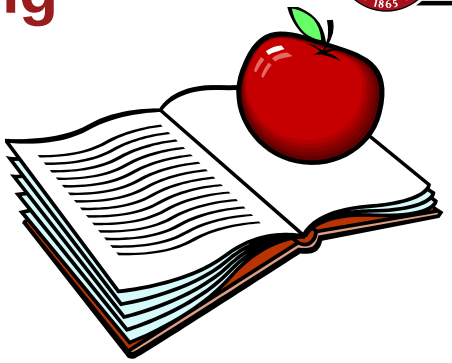
Reference : "Systems Engineering, Principles and Practice," Kossiakoff et. al – Chapter 7

Session 1 Agenda (Jan 27 – 29): Timing

Day	Overview and Concepts of Systems Engineering	Hours*
1	Waters Management kick-off	1/2
	Introduction and overview of entire program (Lesson 1), including Introduction and overview of the projects (15 minutes each)	2
	Presentation and discussion of Concepts of SE (Lessons 2 – 9)	4
	Project assignments, team formations, and working sessions to apply SE concepts to the projects	1
	Day 1 wrap-up: Takeaways and Q&A	1/2
2	Review of day 1 working sessions	1
	Concepts of Systems Engineering (concluded) (Lesson 10 – 14)	4
	Project team working sessions (continued)	2
	Day 2 wrap up and Q&A	1
3	Team working session feedback to class (informal presentations)	1
	Project team working sessions (concluded)	5
	Check-in with class on application of SE concepts to their projects	1
	Session 1 wrap up: Q&A and homework due next session	1

* Times include breaks, and may be changed based on needs of the class

Session 1 Agenda (Jan 27 – 29): Timing



1. Program Overview
2. What is Systems Engineering?
3. Structure and Environment of Systems
4. Systems Development Process
5. Systems Engineering Processes Overview
6. Systems Engineering in Needs Assessment
7. Risk Management
8. Concept Exploration
9. Concept Definition
10. Advanced Development Stage
11. Engineering Design Stage
12. Integration and Evaluation Stage
13. Production and Deployment
14. Operations, Support, and Disposal

Lesson Content:
- Concept Exploration
- SE in Concept Exploration
- SE Tasks

Plus Appendices

* Times include breaks, and may be changed based on needs of the class

Concept Exploration Stage

- *"I have a user with a need….what now?"*

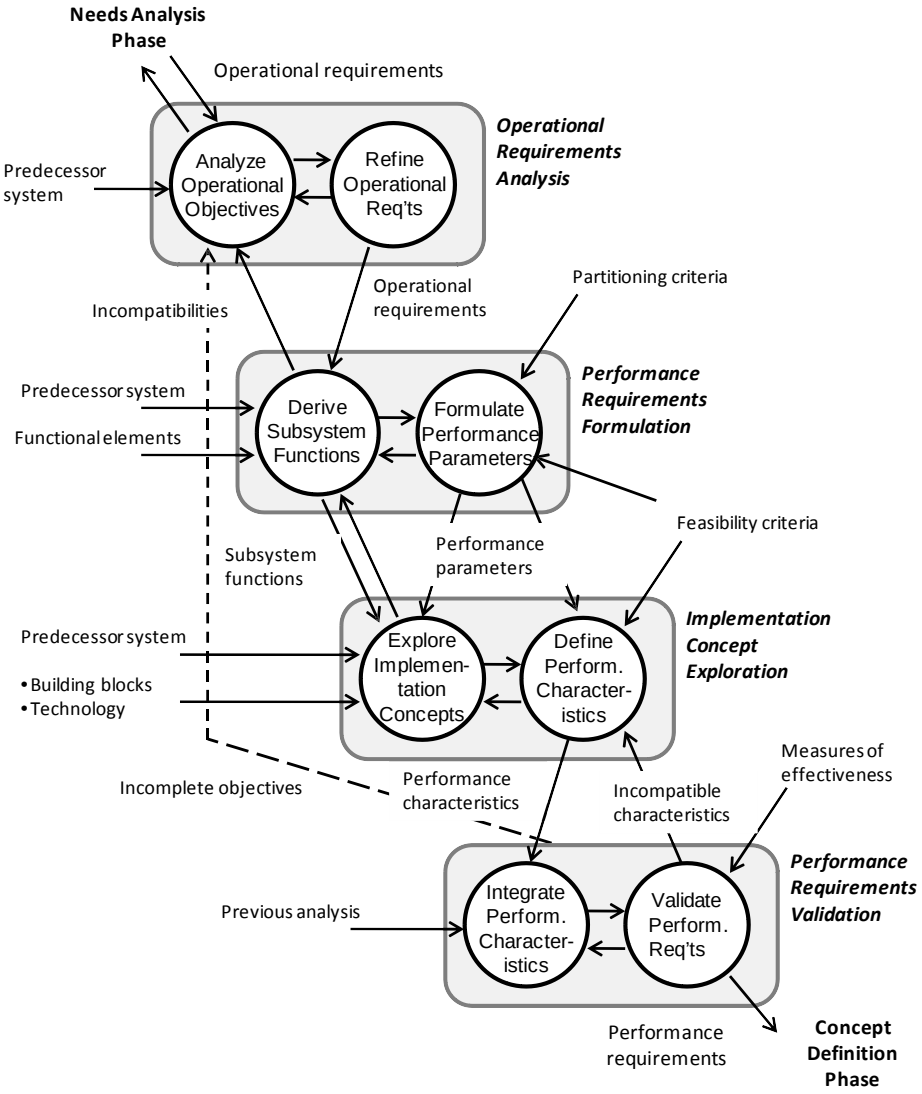
- **Goals:**

  - **To *convert* the operationally-oriented system view into an engineering-oriented view**
    - **Translate "user-speak" into "engineering-speak"**
    - **Needs/Capabilities ⇒ Requirements**

  - **To *consider* alternative concepts**
    - **Brainstorm solutions**
    - **Derive common characteristics**

  - **To *assess* whether there is a feasible solution**

Concept Exploration Phase Flow Diagram



Status of System Materialization

Level \ Phase	*Concept Development*			*Engineering Development*		
	Needs Analysis	Concept Exploration	Concept Definition	Advanced Development	Engineering Design	Integration & Evaluation
System	Define system capabilities & effectiveness	**Identify, explore and synthesize concepts**	Define selected concept with specifications	Validate Concept		Test & Evaluate
Subsystem		**Define requirements & ensure feasibility**	Define functional & physical architecture	Validate subsystems		Integrate & test
Component			Allocate functions to components	Define specifications	Design & test	Integrate & test
Sub-Component	*Visualize*			Allocate functions to subcomponents	Design	
Part					Make or buy	

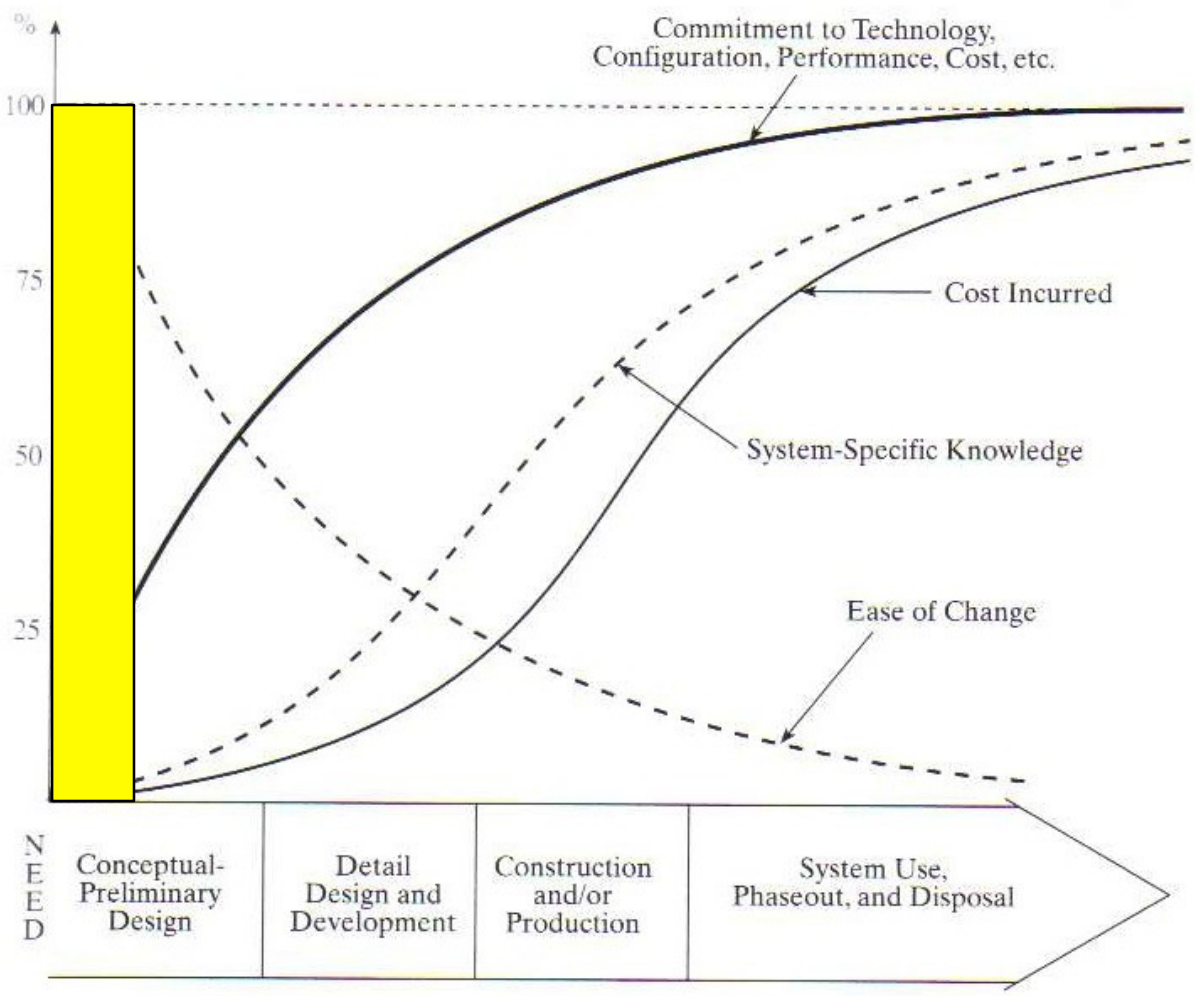
Why Do Concept Exploration?

- Start at the top level of the problem
  - Too many details in a large and/or complex systems
  - Don't want to focus on the details . . . yet

- Don't want to limit solutions this early in the process
  - It's time to brainstorm!
  - People have a natural tendency to go directly to the quick answer
    - This can artificially eliminate solutions

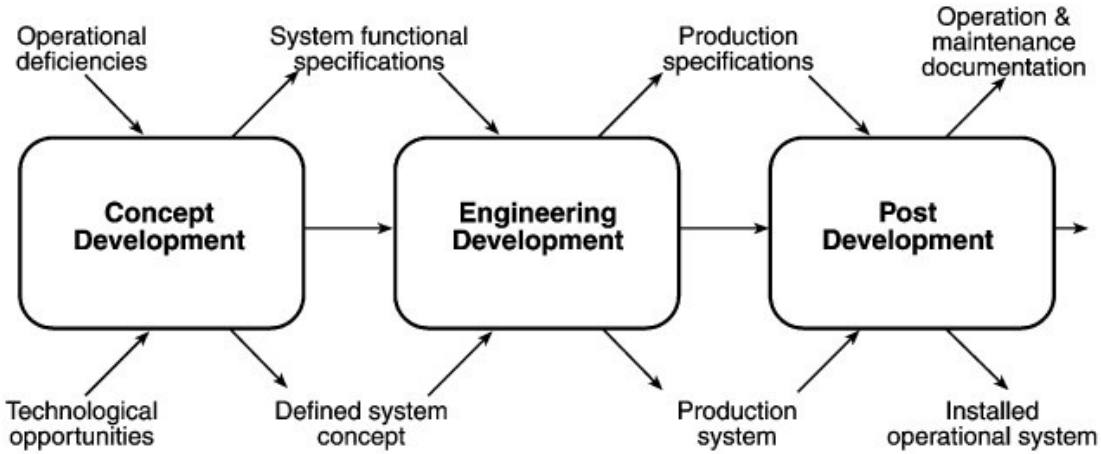


Project Commitment, Knowledge & Cost

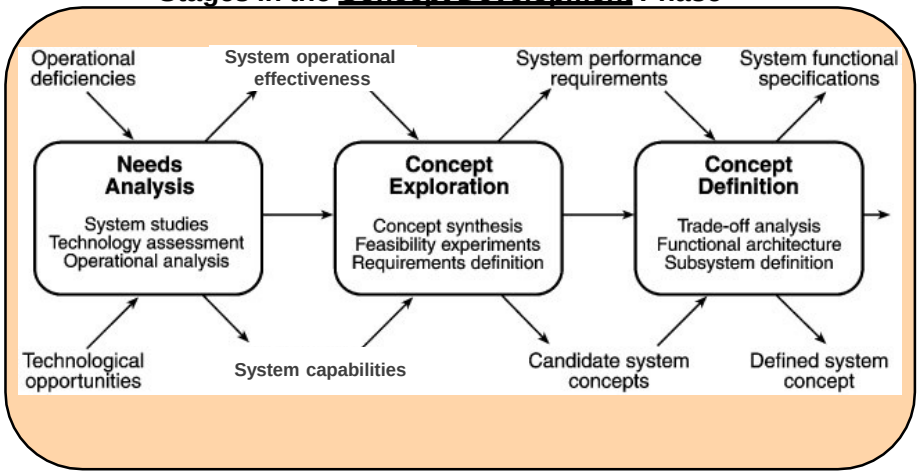


From B. Blanchard & W. Fabrycky, "System Engineering and Analysis" Prentice Hall, 1998 Fig 2.11

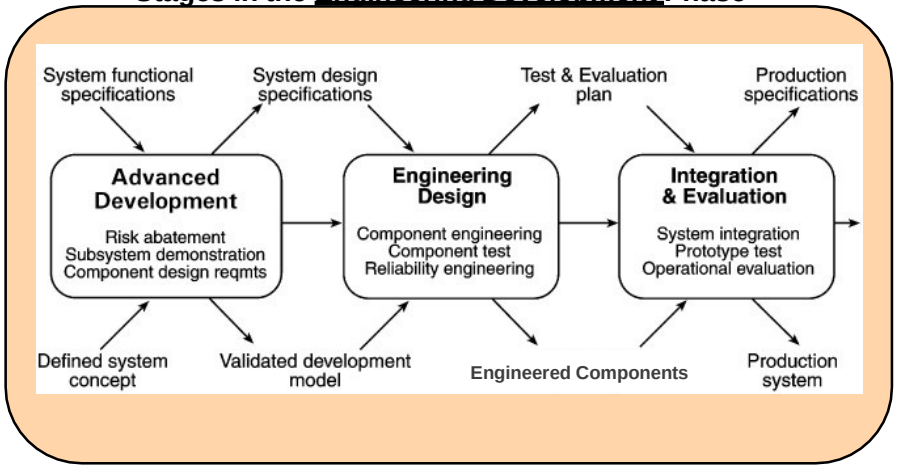
Stages/Phases Reminder (per Kossy)



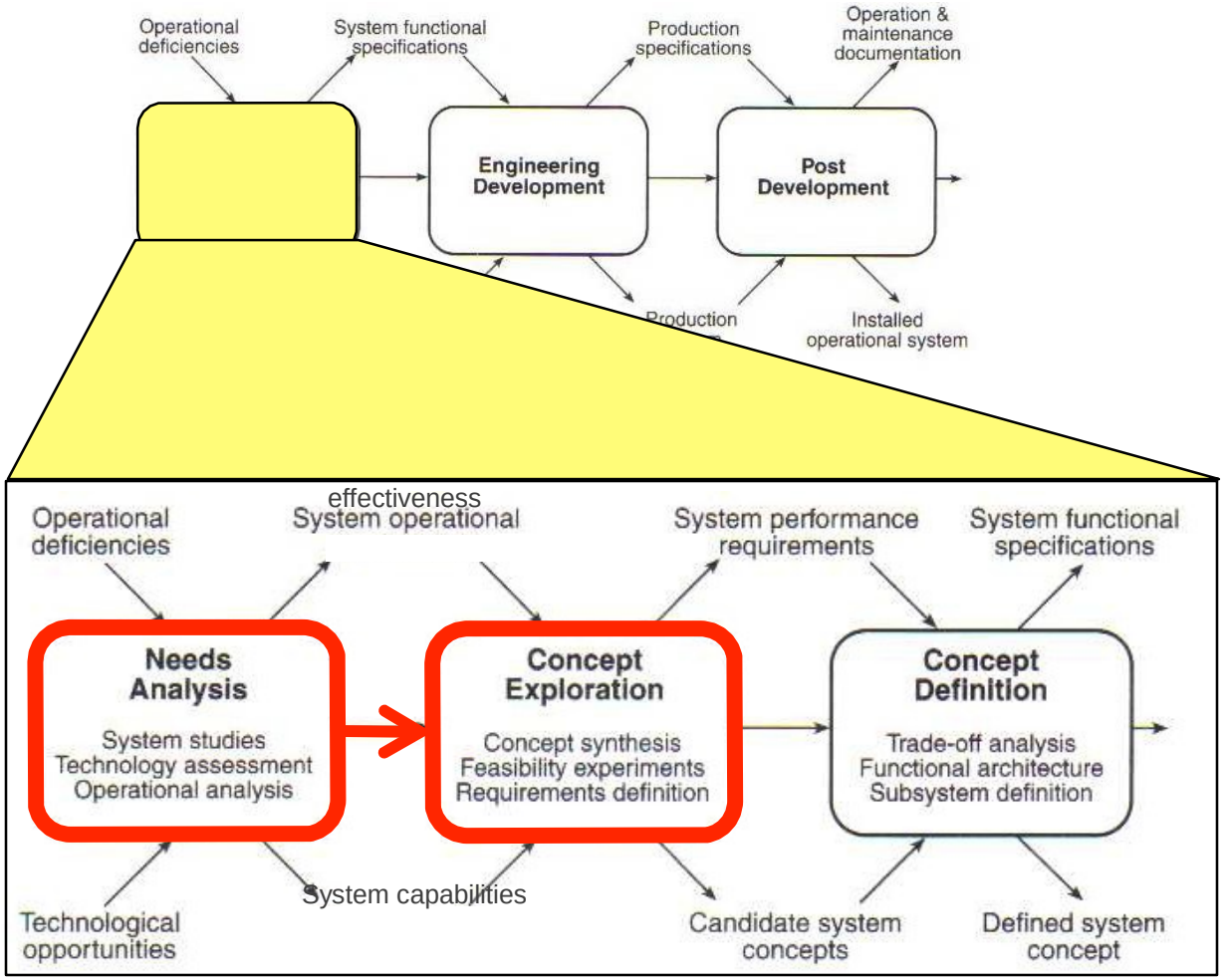
Stages in the Concept Development Phase



Stages in the Engineering Development Phase



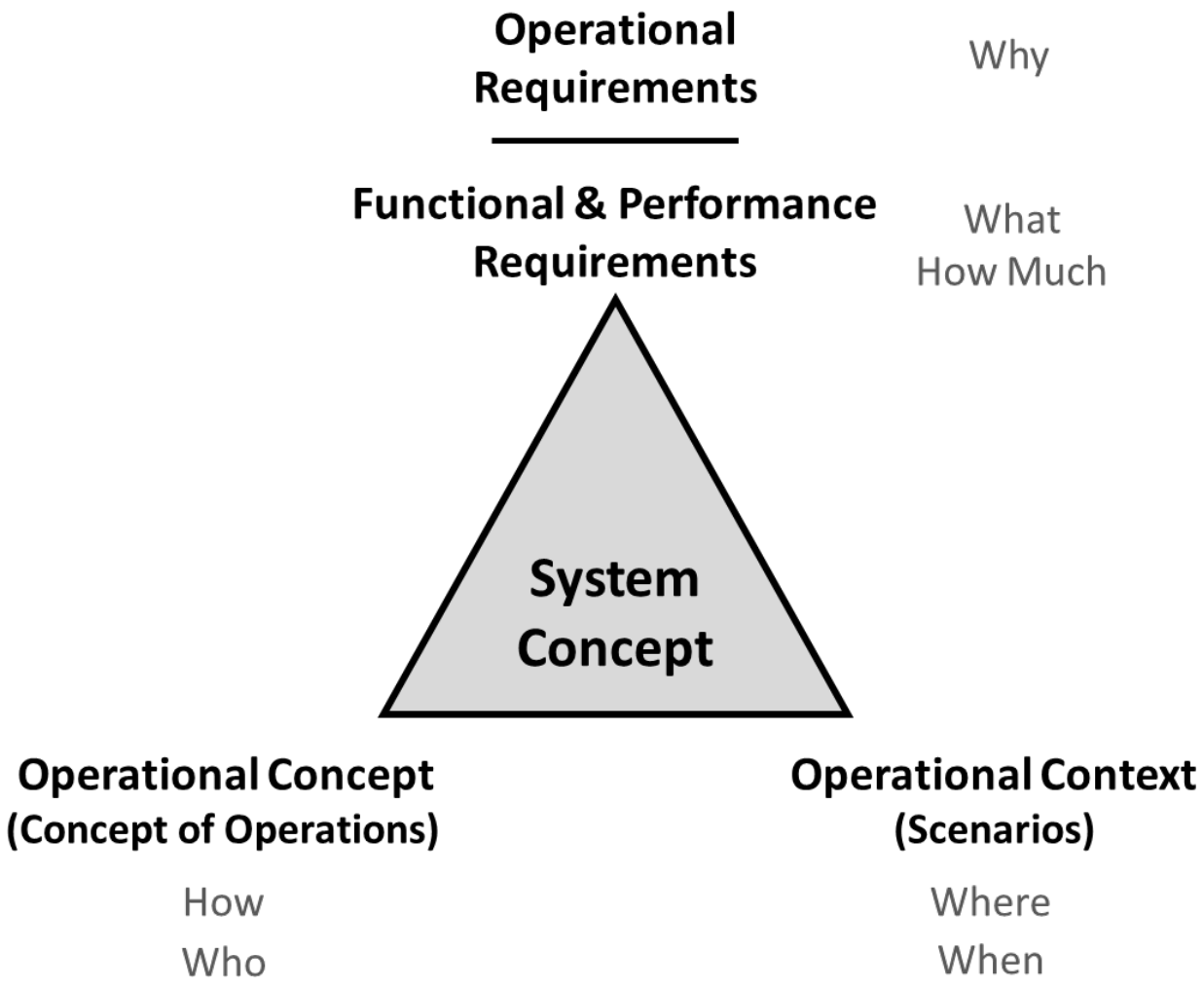
Needs to "Concept Exploration"



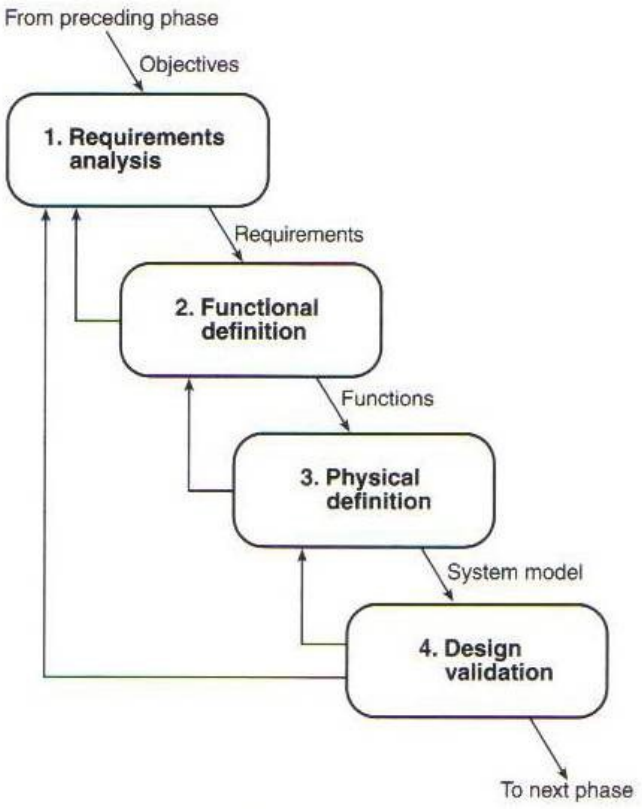
Starting Points for Concept Exploration

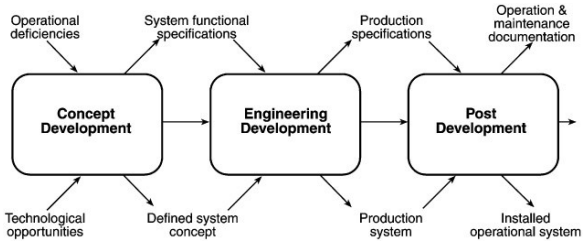
- **User needs**
  - **Operational requirements**
  - **Business product goals**
  - **Request For Information (RFI)**
- **Concept of Operations (CONOPS)**
  - **May be available for military programs**
  - **Other information on how a system would be expected to be used**
- **Current Solutions**
  - **Existing system(s) that do the job today**
- **Technology Opportunities**
  - **New products, algorithms, technology advancement**

Triumvirate of Conceptual Design



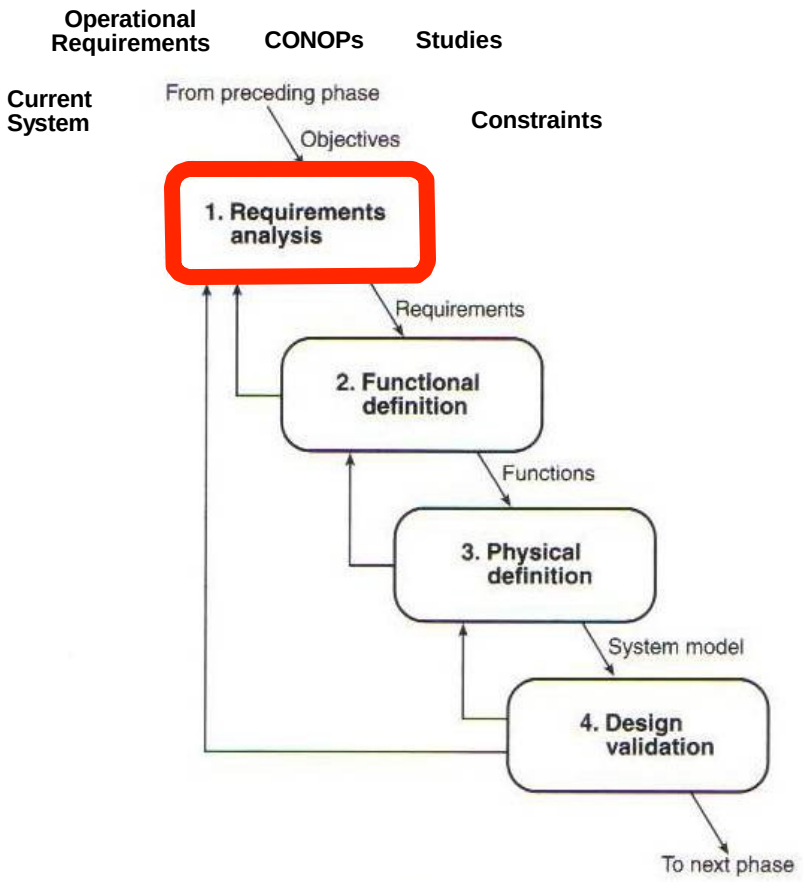
"Systems Engineering Method" (per Kossy)





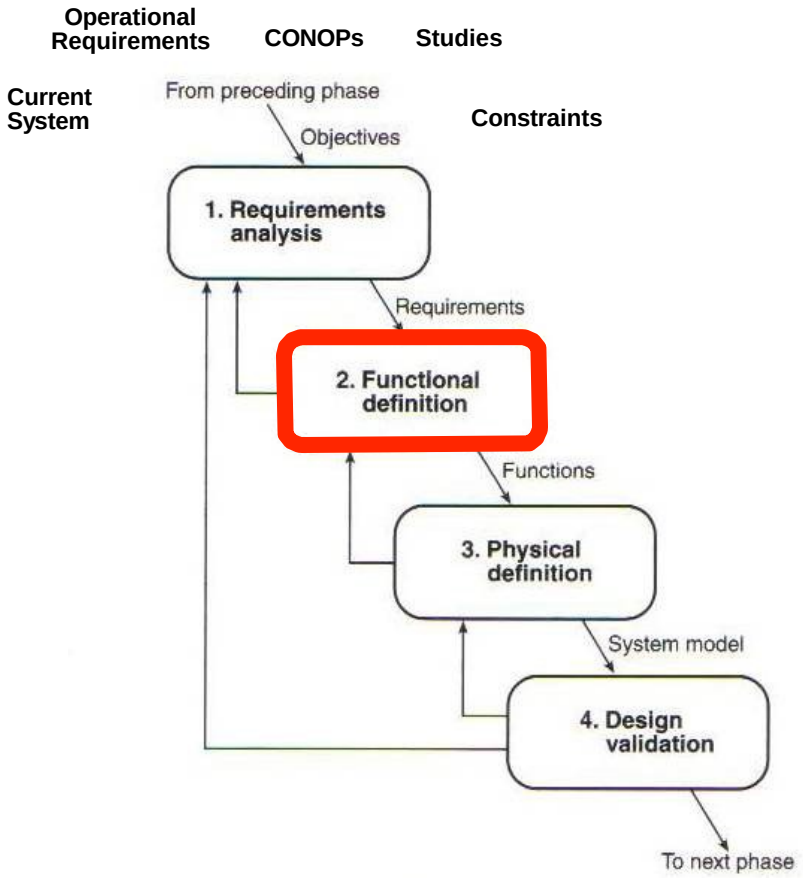
- **Every *phase* applies the system engineering method**
- **There are many feedback paths**
- **The process is often iterative**

Systems Engineering Method in Concept Exploration



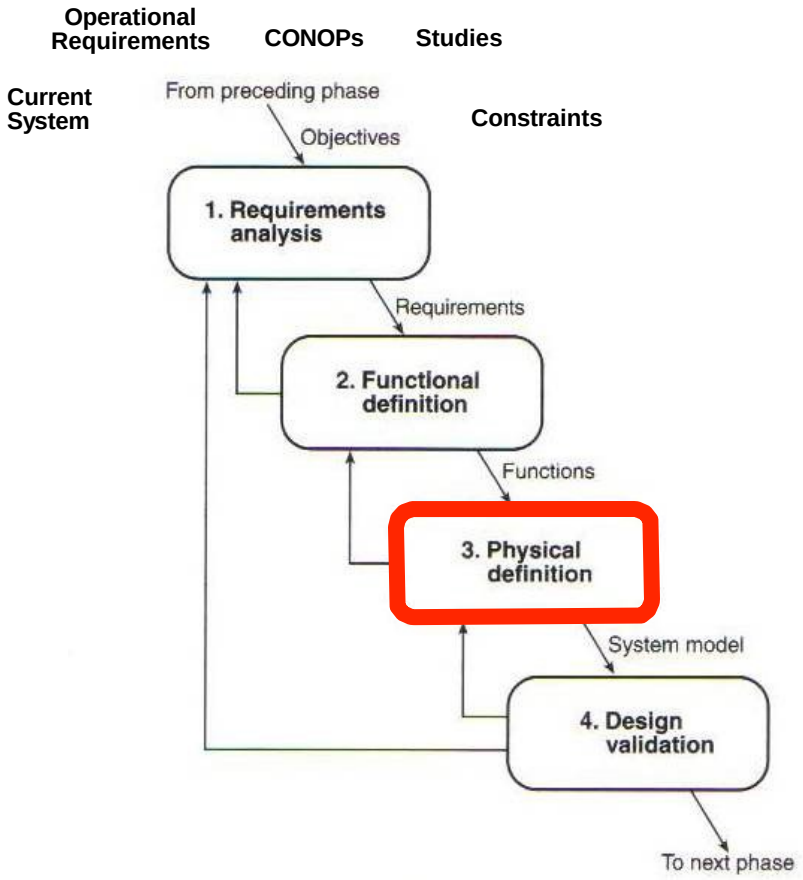
- **Requirements Analysis**
  - **<u>Operational</u> Requirements Analysis**
    - **Understand the operational requirements**
    - **Identify and resolve problems with the operational requirements**
      - **Conflicts**
      - **Incomplete requirements**
      - **Obviously expensive/impossible**
  - ***Provide feedback* to the users**
    - **Request clarifications**
    - **Recommend changes**

Systems Engineering Method in Concept Exploration



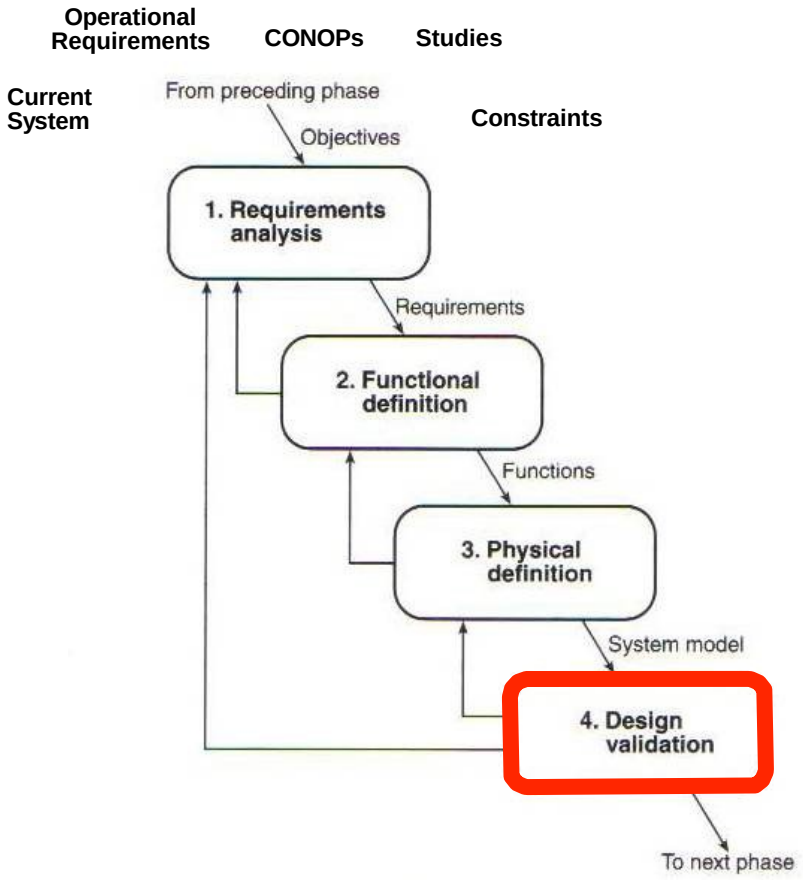
- **Requirements Definition**
  - **Determine system requirements**
  - **Determine required performance**
  - **Define top level scope of system**

Systems Engineering Method in Concept Exploration



- **Define candidate implementation concepts (solutions)**
  - **Define functions**
  - **Assign to candidate subsystems**
  - **Define necessary subsystem requirements and performance characteristics**
- ***Provide feedback* to Functional Definition**
  - **Do the system requirements need to change?**

Systems Engineering Method in Concept Exploration



- **Validate Requirements**
  - **Ensure requirements are consistent and complete**
  - **Ensure concepts are feasible**
- ***Provide feedback* to Design Implementation**
  - **Correct deficiencies in candidate concepts**
  - **Change system requirements**
  - **Recommend changes to operational requirements**

Other Tasks for the Systems Engineer in Concept Exploration

- Estimate risk
- Provide inputs to estimate the needed schedule
- Provide inputs to estimate the cost

At this time, these may be very rough estimates

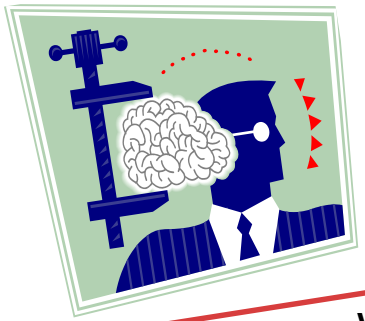
- Recommend who is needed to support the project
  - Specialists
- Establish good system engineering processes
  - I.e., configuration management of requirements, etc.
- Recommend next course of action
  - Prototyping
  - Trade Study
  - Competitive evaluation

Outputs of Concept Exploration

- **Recommendations to change operational requirements**
- **Function List**
- **Preliminary System Specifications**
- **Feasible concept(s) to meet the requirements**
- **Information for the program manager**
  - **Candidate actions/plans**
  - **Schedule**
  - **Risk**
  - **Cost**

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		

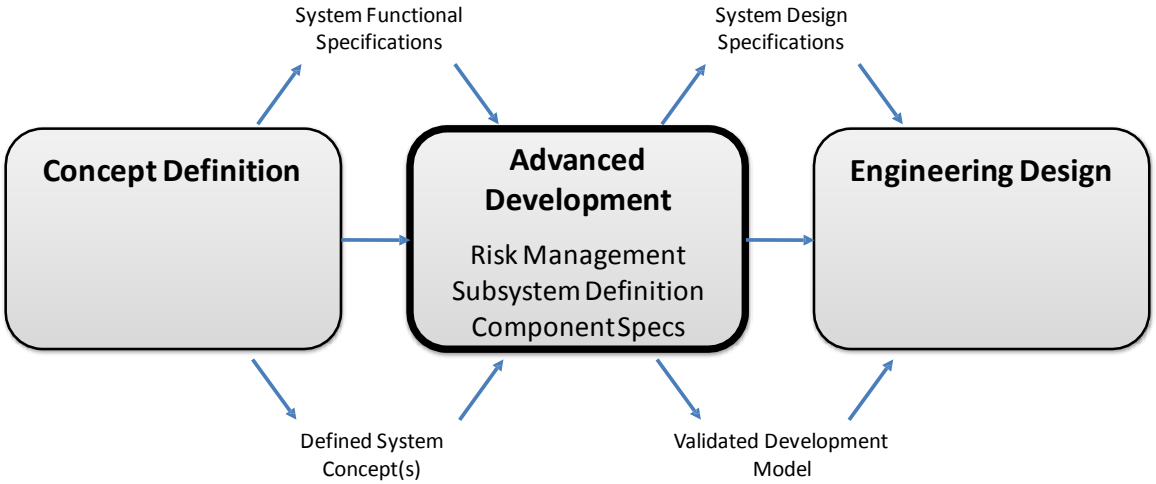


All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?

Systems Engineering Technical Short Course (Session 1)



Lesson 10: Advanced Development

Reference : "Systems Engineering, Principles and Practice," Kossiakoff et. al – Chapter 10

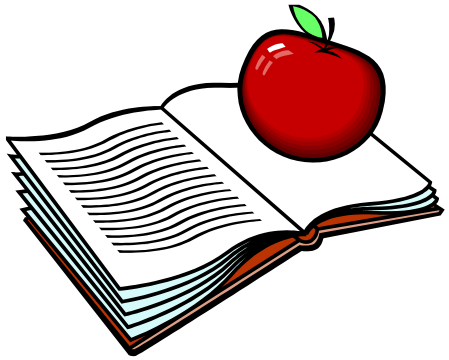
Session 1 Agenda (Jan 27 – 29): Timing

Day	Overview and Concepts of Systems Engineering	Hours*
1	Waters Management kick-off	1/2
	Introduction and overview of entire program (Lesson 1), including Introduction and overview of the projects (15 minutes each)	2
	Presentation and discussion of Concepts of SE (Lessons 2 – 9)	4
	Project assignments, team formations, and working sessions to apply SE concepts to the projects	1
	Day 1 wrap-up: Takeaways and Q&A	1/2
2	Review of day 1 working sessions	1
	Concepts of Systems Engineering (concluded) (Lesson 10 – 14)	4
	Project team working sessions (continued)	2
	Day 2 wrap up and Q&A	1
3	Team working session feedback to class (informal presentations)	1
	Project team working sessions (concluded)	5
	Check-in with class on application of SE concepts to their projects	1
	Session 1 wrap up: Q&A and homework due next session	1

* Times include breaks, and may be changed based on needs of the class

Session 1 Agenda (Jan 27 – 29): Timing

1. Program Overview
2. What is Systems Engineering?
3. Structure and Environment of Systems
4. Systems Development Process
5. Systems Engineering Processes Overview
6. Systems Engineering in Needs Assessment
7. Risk Management
8. Concept Exploration
9. Concept Definition
10. **Advanced Development Stage**
11. Engineering Design Stage
12. Integration and Evaluation Stage
13. Production and deployment
14. Operations, Support, and Disposal



Lesson Content:

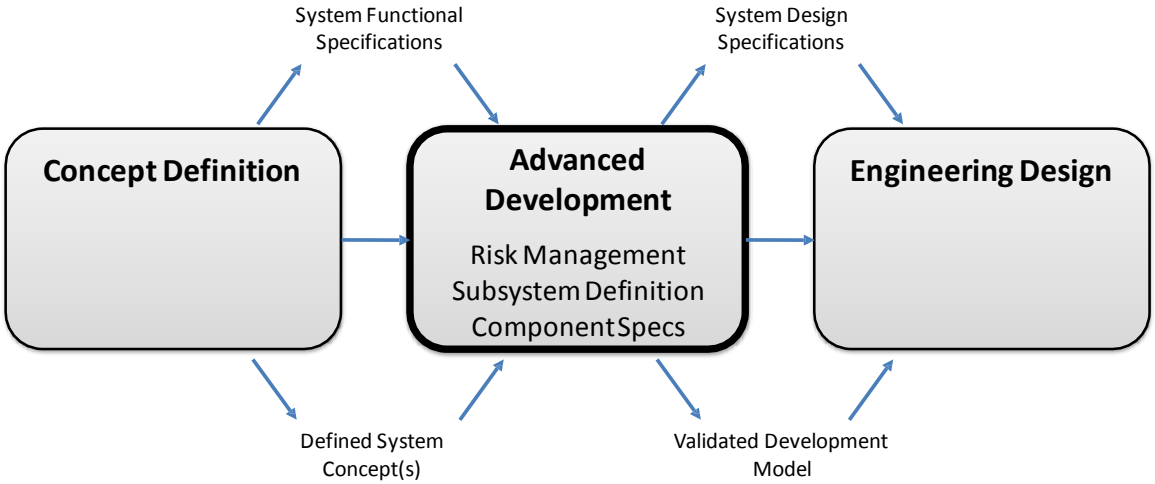
- Advanced Development
- Technology readiness
- SE in Advanced Development
- SE tasks

Exercise: Technology readiness for your projects

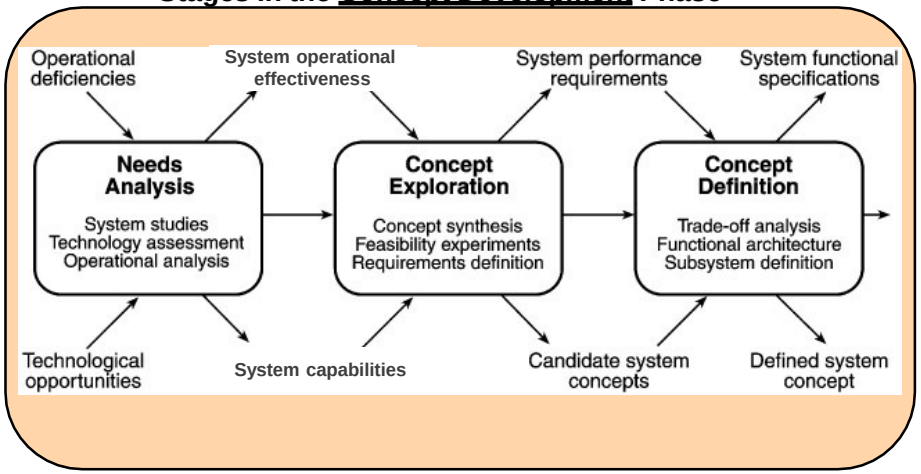
Plus Appendices

* Times include breaks, and may be changed based on needs of the class

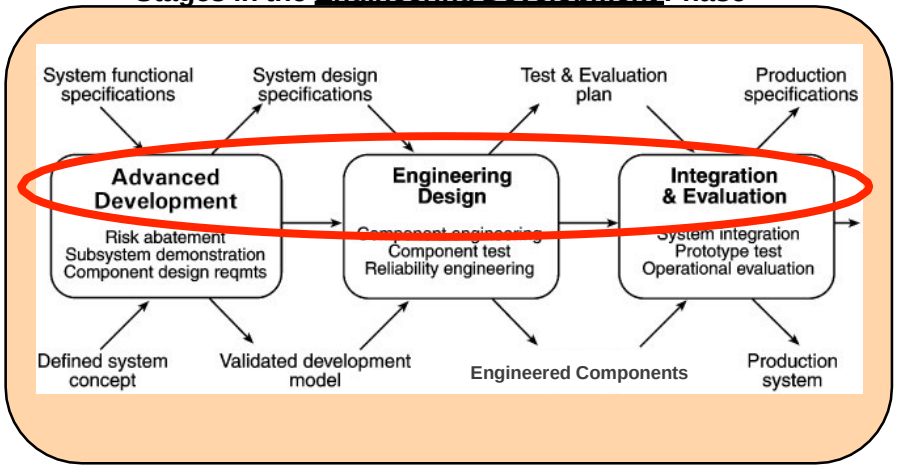
Stages/Phases Reminder (per Kossy)



Stages in the Concept Development Phase



Stages in the Engineering Development Phase



Advanced Development: Let's Start to Get Real

- **Goals:**
  - **To validate the technical approach**
  - **To resolve the major uncertainties in the selected system concept**

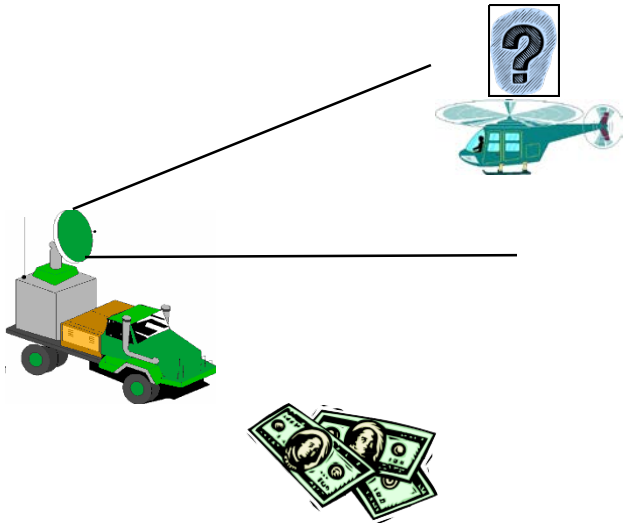
- **End Result:**
  - **Evidence that the chosen concept can be implemented**
  - **System specifications**
  - **Detailed cost & schedule estimates for building the solution**

Advanced Development – Where are we?

- **We have a <u>concept</u> for a solution**
  - **Selected at the end of the Concept Definition Phase**

- **We also have a lot of unknowns (Risk)**
  - **Is the technology mature?**
  - **Will solution really meet the need?**
  - **Have we estimated the size of the job right?**



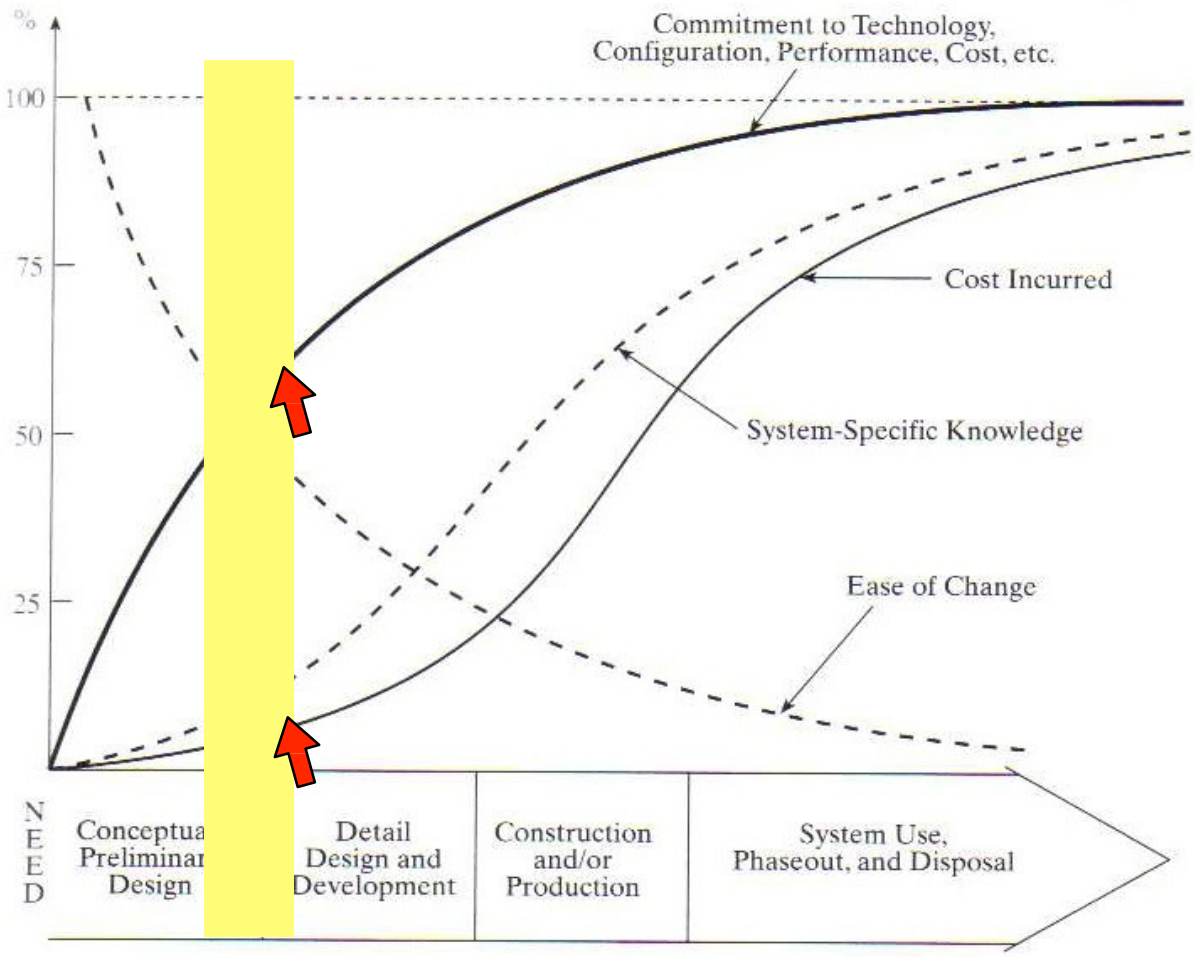


Status of System Materialization

Level \ Phase	*Concept Development*			*Engineering Development*		
	Needs Analysis	Concept Exploration	Concept Definition	Advanced Development	Engineering Design	Integration & Evaluation
System	Define system capabilities & effectiveness	Identify, explore and synthesize concepts	Define selected concept with specifications	**Validate Concept**		Test & Evaluate
Subsystem		Define requirements & ensure feasibility	Define functional & physical architecture	**Validate subsystems**		Integrate & test
Component			Allocate functions to components	**Define specifications**	Design & test	Integrate & test
Sub-Component				**Allocate functions to subcomponents**	Design	
Part					Make or buy	

Visualize

Project Commitment, Knowledge and Cost

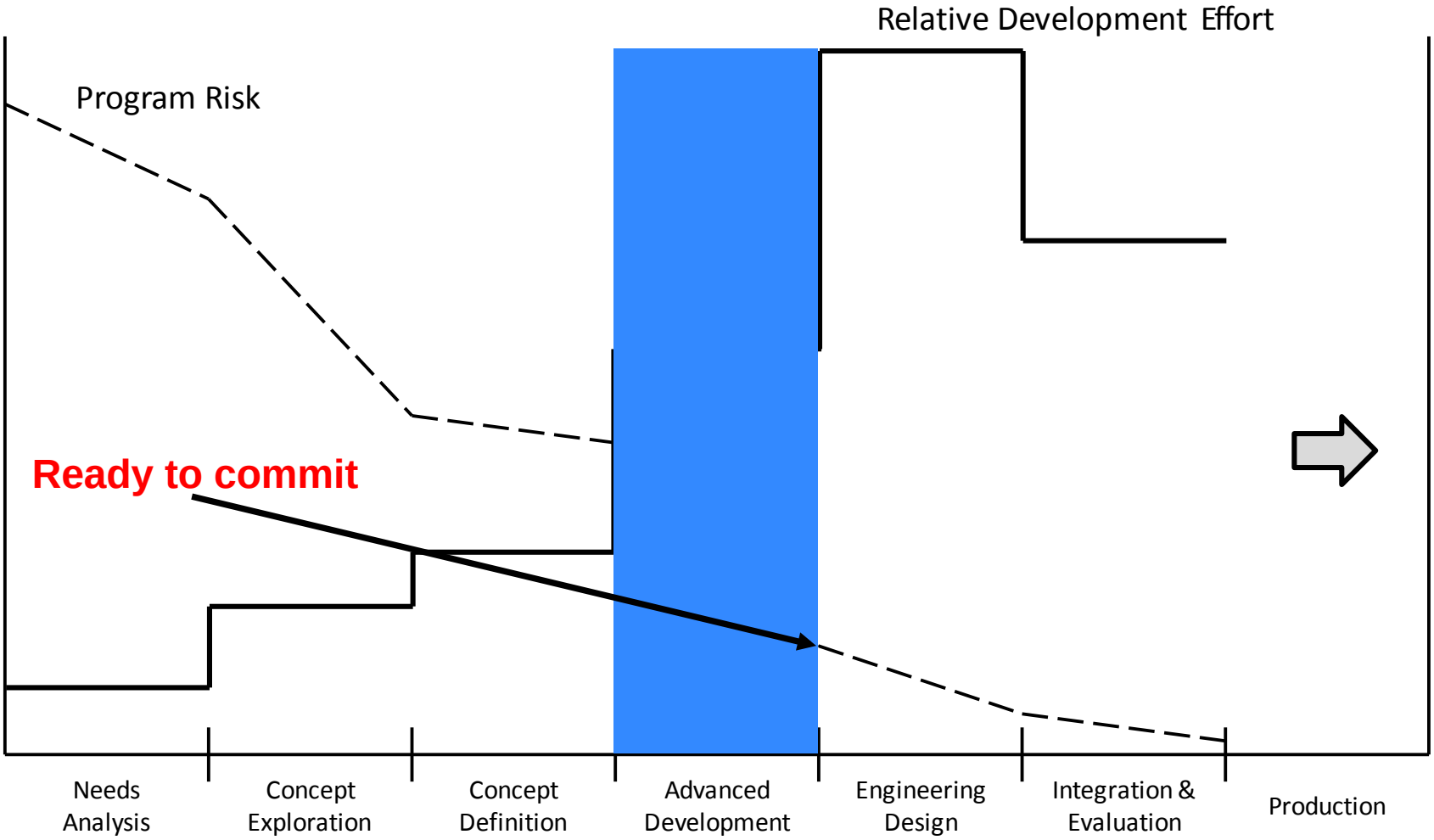


We are about to make large commitments of time and money. Will the concept really work and are we convinced enough to spend a lot of money on a complete development effort?

Technology Readiness Levels (NASA)

Technology Readiness Level		Description
9	Actual system 'flight proven' through successful mission operations	Actual application of the technology in its final form and under mission conditions, such as those encountered in operational test and evaluation. In almost all cases, this is the end of the last "bug fixing" aspects of true system development. Examples include using the system under operational mission conditions.
8	Actual system completed and 'flight qualified' through test and demonstration	Technology has been proven to work in its final form and under expected conditions. In almost all cases, this TRL represents the end of true system development. Examples include developmental test and evaluation of the system in its intended weapon system to determine if it meets design specifications.
7	System prototype demonstration in a operational environment	Prototype near or at planned operational system. Represents a major step up from TRL 6, requiring the demonstration of an actual system prototype in an operational environment, such as in an aircraft, vehicle or space. Examples include testing the prototype in a test bed aircraft.
6	System/subsystem model or prototype demonstration in a relevant environment	Representative model or prototype system, which is well beyond the breadboard tested for TRL 5, is tested in a relevant environment. Represents a major step up in a technology's demonstrated readiness. Examples include testing a prototype in a high fidelity laboratory environment or in simulated operational environment.
5	Component and/or breadboard validation in relevant environment	Fidelity of breadboard technology increases significantly. The basic technological components are integrated with reasonably realistic supporting elements so that the technology can be tested in a simulated environment. Examples include 'high fidelity' laboratory integration of components.
4	Component and/or breadboard validation in laboratory environment	Basic technological components are integrated to establish that the pieces will work together. This is relatively "low fidelity" compared to the eventual system. Examples include integration of 'ad hoc' hardware in a laboratory.
3	Analytical and experimental critical function and/or characteristic	Active research and development is initiated. This includes analytical studies and laboratory studies to physically validate analytical predictions of separate elements of the technology. Examples include components that are not yet integrated or representative.
2	Technology concept and/or application formulated	Invention begins. Once basic principles are observed, practical applications can be invented. The application is speculative and there is no proof or detailed analysis to support the assumption. Examples are still limited to paper studies.
1	Basic principles observed and reported	Lowest level of technology readiness. Scientific research begins with to be translated into applied research and development. Example might include paper studies of a technology's basic properties.

It's All About Reducing Risk



Getting Started

- Perform a risk assessment for the overall system development
  - End goal of advanced development is to reduce risk developing and delivering the end product/capability
  - Requirements Analysis is the starting point
- Determine key areas to apply effort now to reduce the risk
  - E.g., identify risk mitigations for the overall project
- Plan the advanced development stage
  - Identify specific goals
  - Develop program/project to execute the advanced development
    - Work Breakdown Structure (identify the work areas)
    - Systems Engineering Plan (how will we technically manage it)
    - Resources needed (cost, personnel, facilities, equipment)
  - Perform a risk assessment on the effort associated with the advanced development stage
    - This is different than that risk assessment for the overall system development

System Engineering Method in the Advanced Development Stage

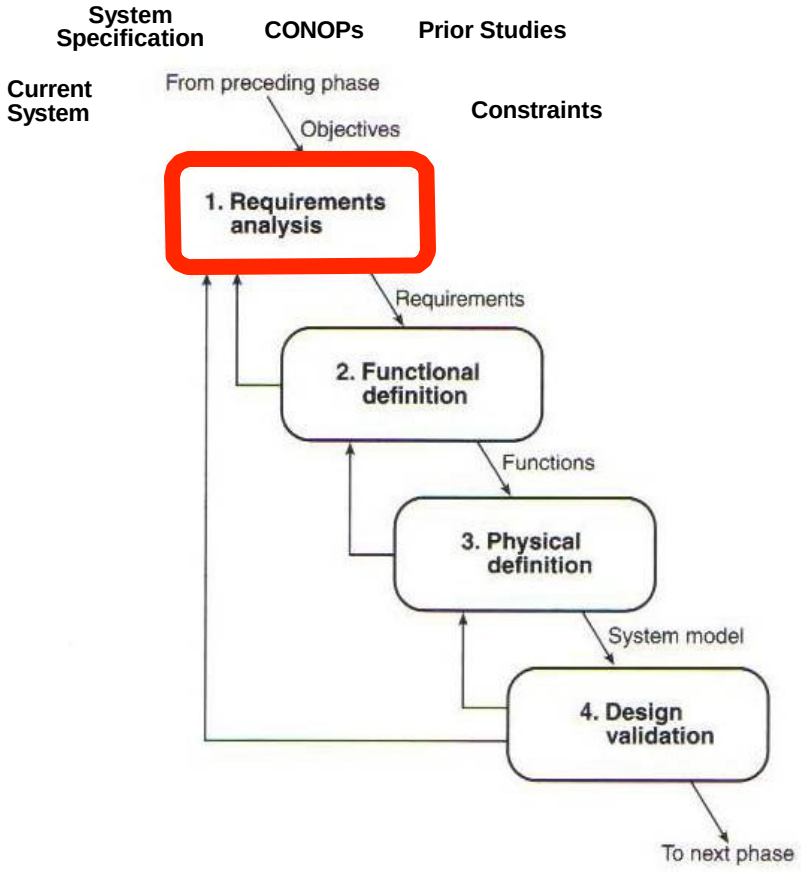


Fig. 3-5 System engineering method top-level flow diagram.

- **Requirements Analysis**
  - **Analyze the system functional specifications**
    - **Understand derivation of functional and performance requirements**
    - **Does proposed derivation of subsystems make sense?**
  - **What requires development?**
    - **Are there components/subsystems that are high risk to develop?**
      - **New technology**
      - **Complexity**
    - **Are there components/subsystems that are uncertain to meet the need/ requirements?**
      - **Never built an item X before**
      - **Uncertain performance**

System Engineering Method in Advanced Development

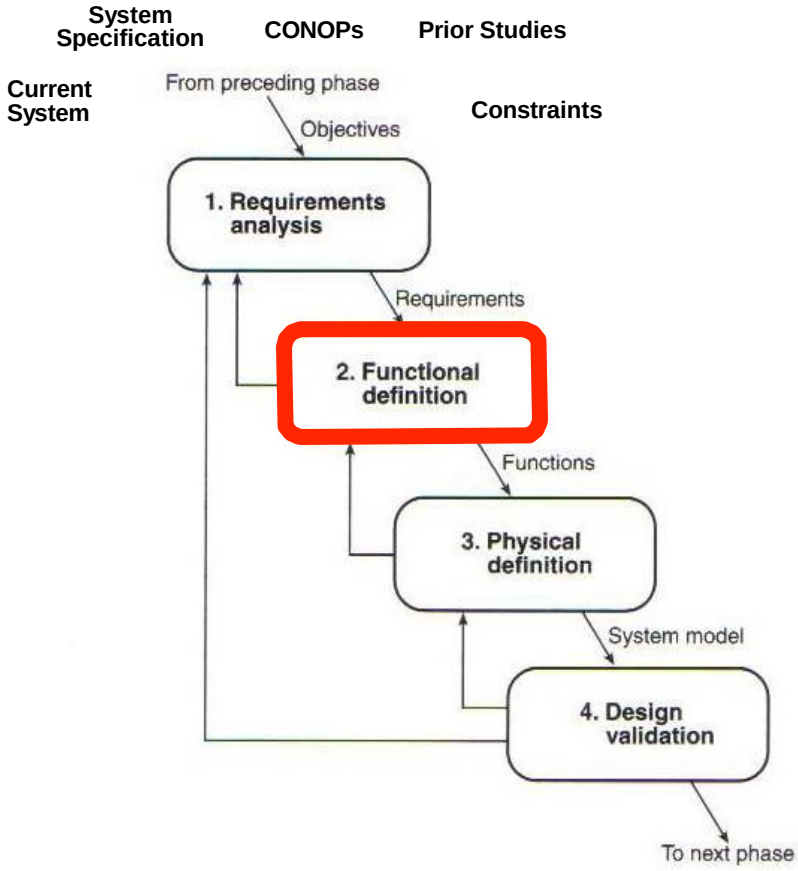


Fig. 3-5 System engineering method top-level flow diagram.

- **Functional Definition**
  - **Analyze allocation of functions to components**
    - **Define your prototype/ simulation functionality and performance**
  - **Identify analogous elements in other systems**
  - **Perform analysis and simulations to resolve outstanding performance issues**

System Engineering Method in Advanced Development

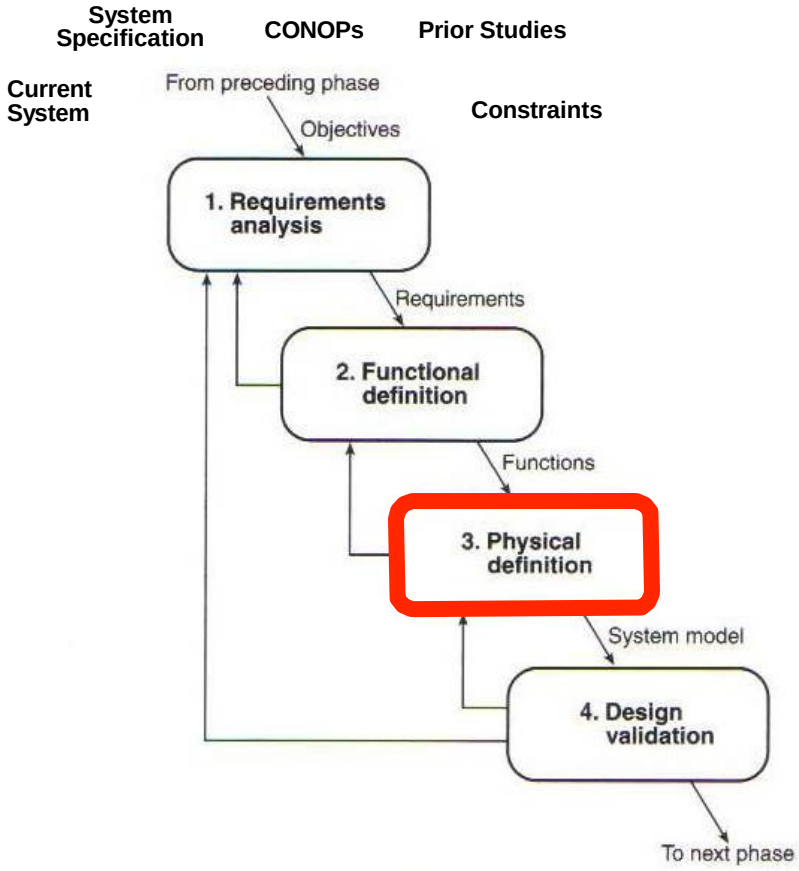


Fig. 3-5 System engineering method top-level flow diagram.

- Create the prototype
  - Hardware
  - Software
- Bench or unit test
  - Ensure that the pieces work before sending to integration and test (in Design Evaluation)
- Correct deficiencies fed back from Design Validation

System Engineering Method in Advanced Development

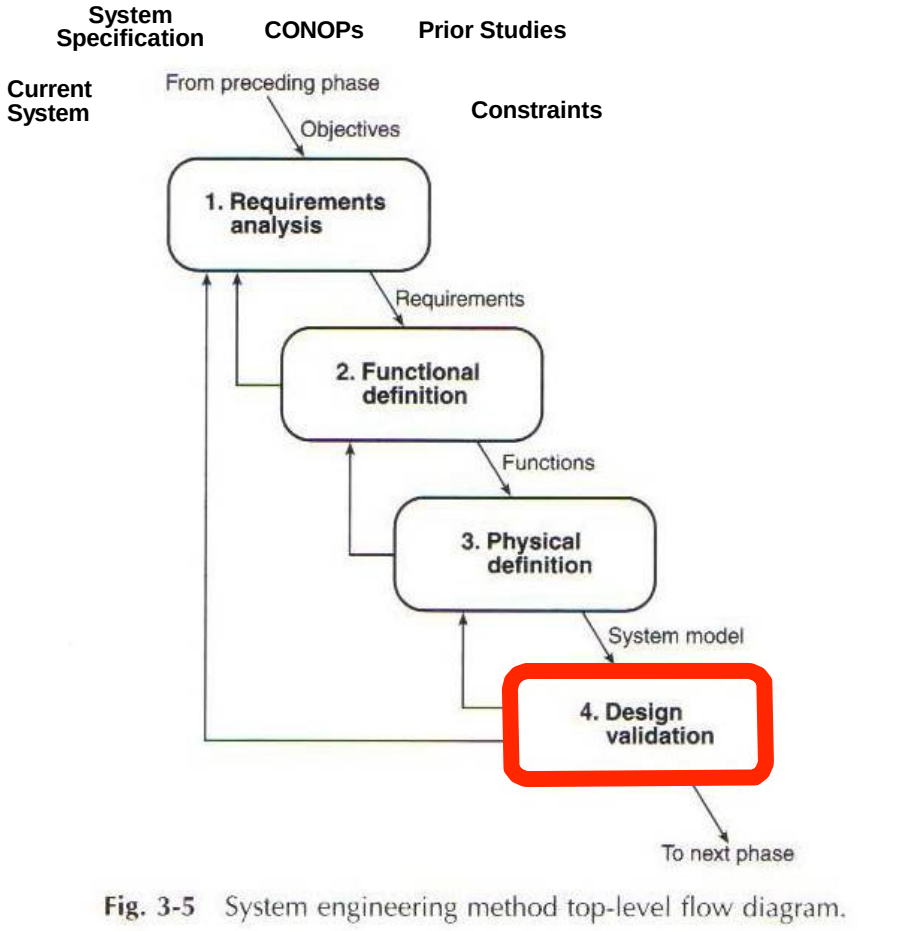


Fig. 3-5 System engineering method top-level flow diagram.

- Integrate the components
- Insure that the Advance Development goals are being met
  - Develop test plan
  - Develop procedures to verify the prototype works as intended
- Provide feedback to Physical Definition to correct deficiencies in the prototype
- Recommend changes to subsystem/component requirements

Testing in Advanced Development

- Prove that the new technology works
  - Functional tests
  - Performance tests
  - Test it in a representative environment
    - Environmental (i.e., temperature, pressure, electromagnetic interference, jamming, etc.)
    - Loading (i.e., 50 simultaneous users, 500 targets, etc.)
- Simulate other components not built yet
- Show a demonstration
  - Let the users interact with it
  - Example: screens of a new user interface
- Create a high fidelity model or simulation
  - More depth based on time spent gathering data, conducting analysis
  - Use to validate requirements and solution conceptual approach
  - Especially applicable to very large projects where even subsystems or components may require large amounts of resources to develop

Other Risk-Reducing Efforts in Advanced Development

- **Develop a formal test plan for Advanced Development and for the remainder of the development**
  - **DoD: Test and Evaluation Master Plan (TEMP)**
- **Develop special test equipment**
  - **Some test equipment items are as complex or hard to build as the subsystem/component that they are built to test**
- **Begin preliminary work for the Engineering Design Phase**
  - **Develop detail specifications for components**
    - **Especially applicable to items that were prototyped in Advanced Development**

Tasks for the Systems Engineer in Advanced Development

- Estimate risk
- Provide inputs to estimate the needed schedule
- Provide inputs to estimate the cost

Responsible for Systems Engineering and Specialty Engineering inputs
May also be responsible to review other inputs

- Recommend who is needed to support the project
  - Specialists
- Establish good system engineering processes
  - i.e., Configuration management of requirements, etc.
  - Develop a system engineering management plan

More Tasks for the Systems Engineer in Advanced Development

- **Execute systems engineering processes**
  - Conduct ongoing risk assessments
  - Perform functional analysis and allocation requirements to lower level component specifications
  - Establish and conduct design reviews
  - Conduct test planning
  - Ensure good traceability occurs between high level requirements through lower level development specifications through test

- Recommend next course of action
  - Has the risk been reduced?
  - Are further advanced development phases needed?
    - If so, to do what?

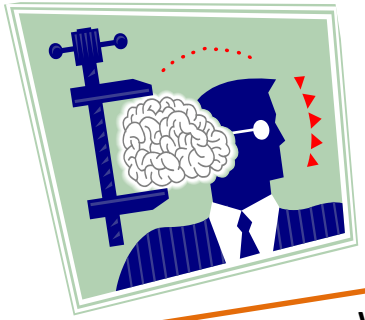
Exercise: Technology Readiness for Your Projects



Instructor will provide directions for the exercise

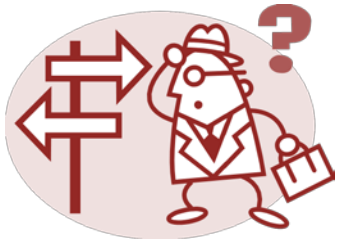
Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future

What are your Questions before We Continue?

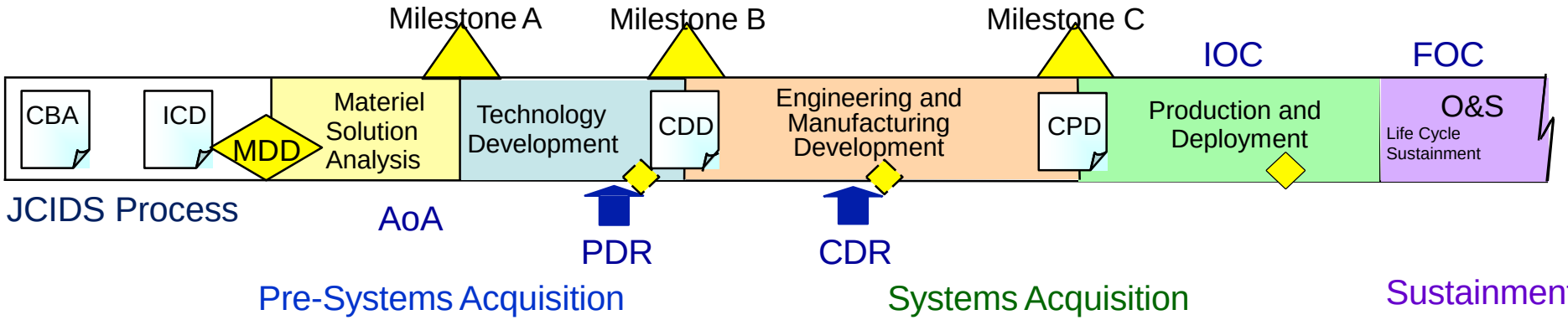


Back-up Slides

Advanced Development Phase in the System Life Cycle – DoD View (2008)

- **Successful "Milestone A" decision to Start**
  - **Requires significant backup data justifying approach and plan for technology development**
    - **Validated user requirements**
    - **System engineering plan, cost, schedule for development**
    - **How the proposed system will fit into the DoD enterprise architecture**

- **Purpose: "to reduce technology risk and to determine the appropriate set of technologies to be integrated into a full system…. It is an iterative process designed to assess the viability of technologies while simultaneously refining user requirements.**

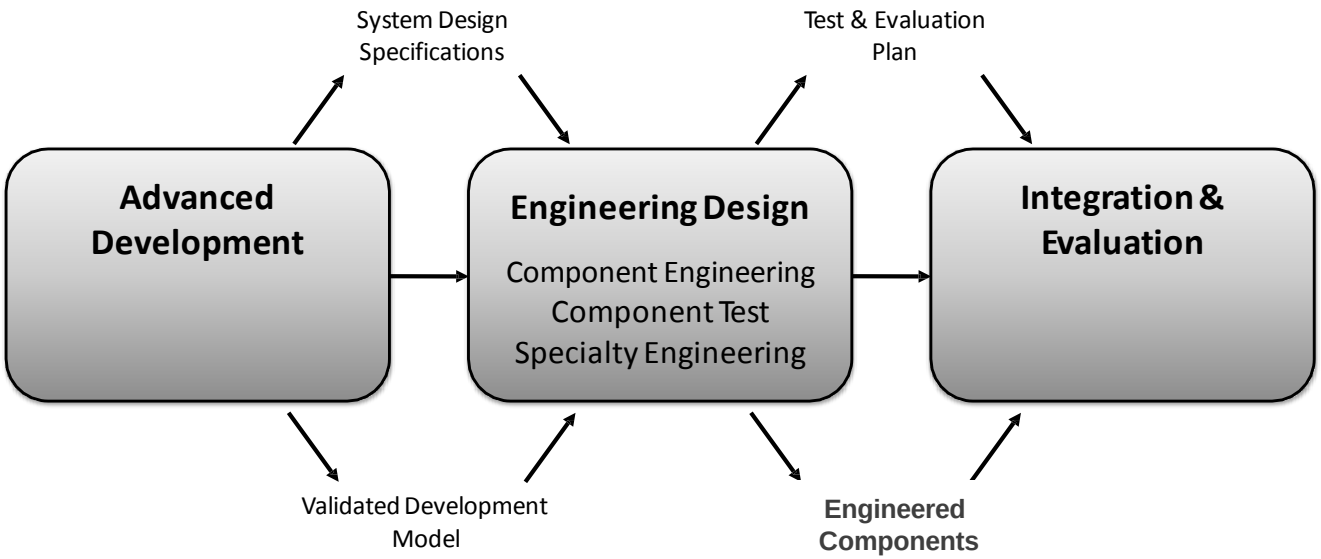


DoD View of Advanced Development

- **"A favorable Milestone A decision into an advanced development phase DOES NOT mean that a new acquisition program has been initiated."**
  - **Feasibility is being investigated, they are not buying yet**
- **"Multiple technology development demonstrations may be necessary before the user and developer agree that a proposed technology solution is affordable, militarily useful, and based on mature technology."**
  - **This phase may be repeated several times before committing to full system development**
- **"The project shall exit Technology Development when an affordable increment of … useful capability has been identified, the technology for that increment has been demonstrated in a relevant environment, and a system can be developed for production within a short timeframe (normally less than five years); or when the … Decision Authority decides to terminate the effort."**

Excerpts from the Defense Acquisition Guidebook Section 3.6 (2005), https://dag.dau.mil/

Systems Engineering technical Short Course (Session 1)



Lesson 11: Engineering Design Stage

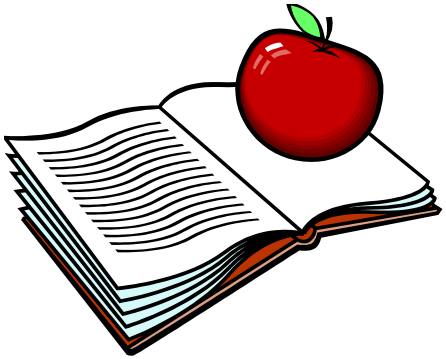
Reference : "Systems Engineering, Principles and Practice," Kossiakoff et. al – Chapter 12

Session 1 Agenda (Jan 27 – 29): Timing

Day	Overview and Concepts of Systems Engineering	Hours*
1	Waters Management kick-off	1/2
	Introduction and overview of entire program (Lesson 1), including Introduction and overview of the projects (15 minutes each)	2
	Presentation and discussion of Concepts of SE (Lessons 2 – 9)	4
	Project assignments, team formations, and working sessions to apply SE concepts to the projects	1
	Day 1 wrap-up: Takeaways and Q&A	1/2
2	Review of day 1 working sessions	1
	Concepts of Systems Engineering (concluded) (Lesson 10 – 14)	4
	Project team working sessions (continued)	2
	Day 2 wrap up and Q&A	1
3	Team working session feedback to class (informal presentations)	1
	Project team working sessions (concluded)	5
	Check-in with class on application of SE concepts to their projects	1
	Session 1 wrap up: Q&A and homework due next session	1

* Times include breaks, and may be changed based on needs of the class

Concepts of Systems Engineering



1. Program Overview
2. What is Systems Engineering?
3. Structure and Environment of Systems
4. Systems Development Process
5. Systems Engineering Processes Overview
6. Systems Engineering in Needs Assessment
7. Risk Management
8. Concept Exploration
9. Concept Definition
10. Advanced Development Stage
11. **Engineering Design Stage**
12. Integration and Evaluation Stage
13. Production and Deployment
14. Operations, Support, and Disposal

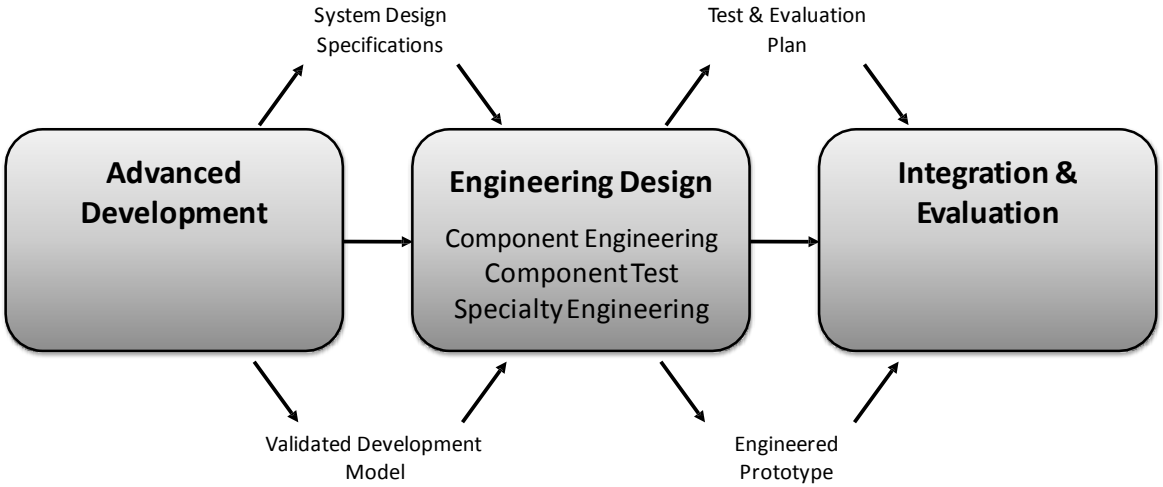
Plus Appendices

Lesson Content:

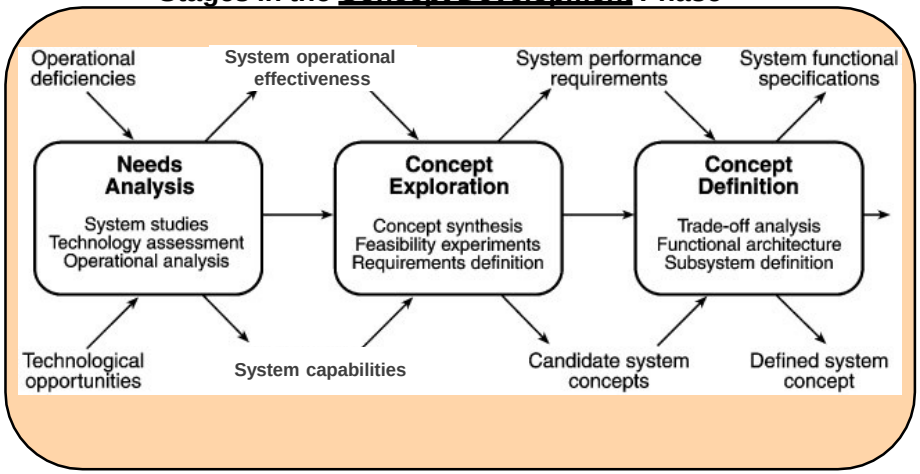
- Engineering Design
- Project planning
- Risk management
- SE in Engineering Design
- Tools and techniques
- Configuration control
- Design review

Exercise: Planning for the projects

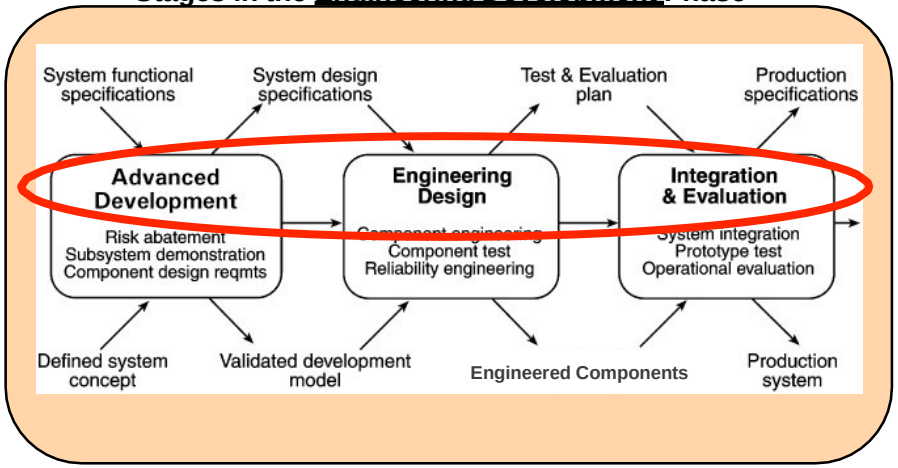
Stages/Phases Reminder (per Kossy)



Stages in the Concept Development Phase



Stages in the Engineering Development Phase



Engineering Development: Now We Need to Really Produce Something!

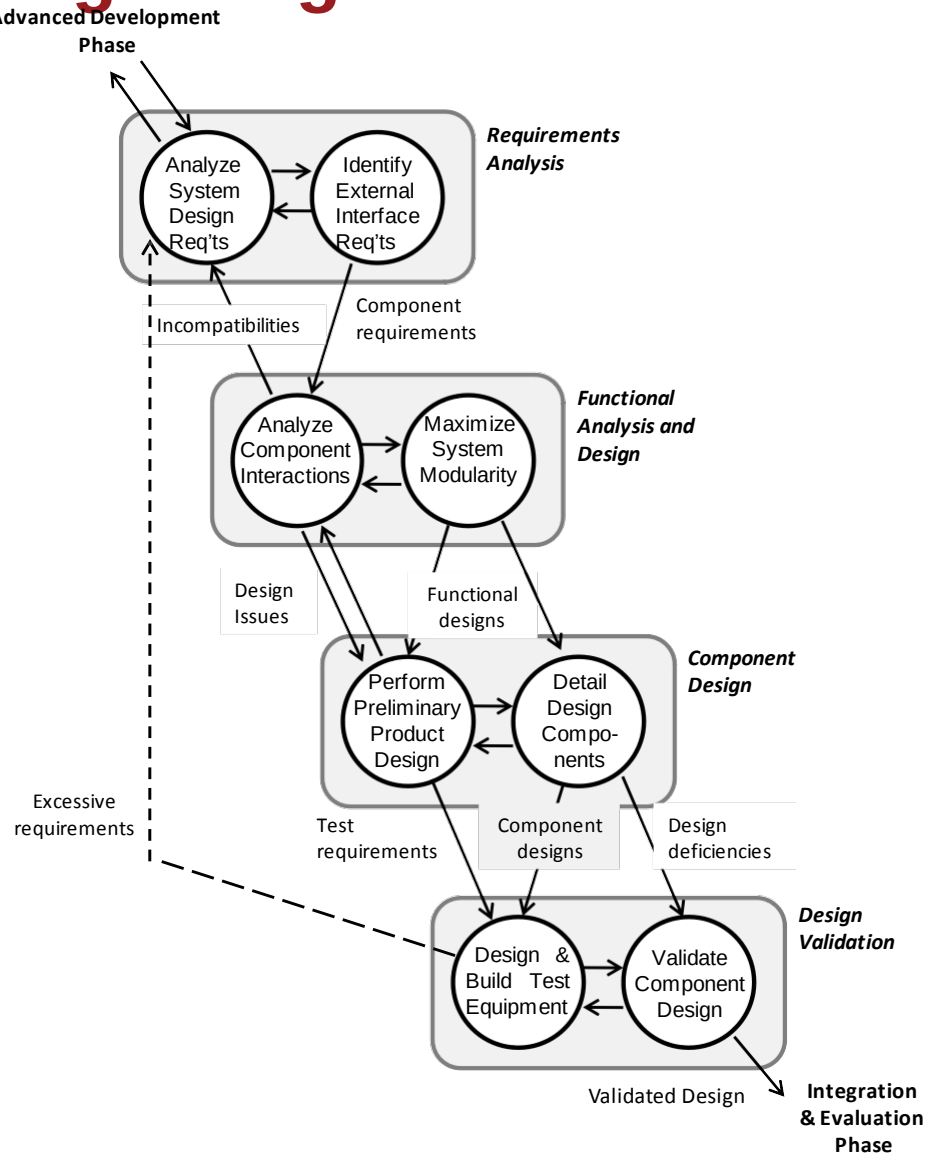
- **Goals:**
  - To develop system components
  - To establish consistent internal and external interfaces
  - To meet technical, cost, and schedule requirements

- **End Result:**
  - Detailed design specifications
  - Interface documentation
  - Drawings
  - At least one finished, representative product
  - Some or all factory tooling and test equipment needed for producing more products
  - DoD: Information to support Milestone C (entry into Production)

Status of System Materialization at the Engineering Design Phase

Level \ Phase	*Concept Development*			*Engineering Development*		
	Needs Analysis	Concept Exploration	Concept Definition	Advanced Development	Engineering Design	Integration & Evaluation
System	Define system capabilities & effectiveness	Identify, explore and synthesize concepts	Define selected concept with specifications	Validate Concept		Test & Evaluate
Subsystem		Define requirements & ensure feasibility	Define functional & physical architecture	Validate subsystems		Integrate & test
Component			Allocate functions to components	Define specifications	**Design & test**	Integrate & test
Sub-Component		*Visualize*		Allocate functions to subcomponents	**Design**	
Part					**Make or buy**	

Engineering Design Phase Flow Diagram



Engineering Development – Where are we?

- We are pretty sure we have a solution
  - We selected it at the end of the Concept Definition Phase
  - We proved major element(s) of it in the Advanced Development Phase (if we had one) to reduce the technical unknowns

- We have also planned enough to convince our sponsor to commit the resources to develop the real product
  - Project planning has been done
    - Cost & schedule defined
  - Risk has been assessed and determined to be reasonable





General Thoughts

- This is the phase that "regular" engineers know best
  - Designing and building the new product

This is also the phase where past mistakes will usually show up to haunt the project

  - Solving the right problem?
  - System requirement definition errors
  - Subsystem definition errors
  - Test approach (test requirements) errors
  - Unmitigated technical risks
- Why Now?
  - Need to complete the development with a defined cost on an defined schedule
  - There is no next development phase to wait and do it later

The SE is Involved Throughout the Engineering Development

- Assist with Project Planning
- Perform Risk Management
- Develop Specifications and Allocate Requirements
  - Functional Analysis – define subsystems/major components
  - Define interfaces (external, between subsystems)
- Perform Configuration Control
- Apply Specialty Engineering
  - Coordinate and Apply Specialty Engineering Disciplines
- Establish Technical Performance Measures
- Plan Qualification activities
- Plan and Run Design Reviews
- Arbitrate cross team technical issues
- Perform Functional Configuration Audit
- Perform Physical Configuration Audit

Project Planning

- **Plan the engineering development effort**
  - **Develop Work Breakdown Structure (identify the work areas)**
  - **Develop the System Engineering Management Plan**
    - **How will we technically manage the project**
  - **Help to identify resources needed**
    - **Cost**
    - **Personnel**
    - **Facilities**
    - **Equipment**

The Systems Engineering Management Plan (SEMP) contains the processes and plans for the technical work on the project

Systems Engineering Management Plan

SYSTEMS ENGINEERING MANAGEMENT PLAN
(Outline, EIA Engineering Standard 632, "Systems Engineering")

1. Title Page, Table of Contents, Applicable Documents
2. Systems Engineering Process
   - 2.1 Systems Engineering Process Planning—major products and results from process, process inputs, technical objectives, work breakdown structure (WBS), training, standards and procedures, resource allocation, constraints, work authorization, verification planning, subcontractor/supplier technical effort.
   - 2.2 Requirements Analysis—reliability and maintainability, survivability, electromagnetic compatibility, human engineering, safety, security, producibility, product support (supportability), test and evaluation, integrated diagnostics, transportability, infrastructure support, other areas of functionality.
   - 2.3 Functional Analysis and Allocation—approach, methods, procedures tools.
   - 2.4 Synthesis — factor-dependent approaches and methods, use of leverage options (COTS/NDI, open systems architecture, reuse).
   - 2.5 Systems Analysis And Control—approach, methods, procedures, and tools (trade studies, system cost-effectiveness analysis, risk management, configuration management, interface management, data management, SE master schedule, technical performance measurement, subcontractor/supplier control, requirements traceability).
3. Transitioning Critical Technologies—activities, risks, and criteria for selecting technologies and for the transitioning of these technologies.
4. Integration of Systems Engineering Effort—organization and integration of design disciplines and related activities (concurrent engineering).
5. Implementation Tasks—technology verifications, process proofing, manufacturing engineering test articles, development test and evaluation, generation and reuse of software, sustaining engineering and problem solution support, other systems engineering implementation tasks.
6. Additional Systems Engineering Activities—long-lead items, engineering tools, design-to-cost, value engineering, system integration, other methods and controls.
7. Notes and Appendicies

Specific content can be tailored to program needs.

From B. Blanchard & W. Fabrycky, "System Engineering and Analysis" Prentice Hall, 1998 Fig 18.3

Systems Engineering Plan (SEP)

Table of Contents

1. Introduction – Purpose and Update Plan
2. Program Technical Requirements
   - 2.1. Architectures and Interface Control
   - 2.2. Technical Certifications
3. Engineering Resources and Management
   - 3.1. Technical Schedule and Schedule Risk Assessment
   - 3.2. Engineering Resources and Cost/Schedule Reporting
   - 3.3. Engineering and Integration Risk Management
   - 3.4. Technical Organization
   - 3.5. Relationships with External Technical Organizations
   - 3.6. Technical Performance Measures and Metrics
4. Technical Activities and Products
   - 4.1. Results of Previous Phase SE Activities
   - 4.2. Planned SE Activities for the Next Phase
   - 4.3. Requirements Development and Change Process
   - 4.4. Technical Reviews
   - 4.5. Configuration and Change Management Process
   - 4.6. Design Considerations
   - 4.7. Engineering Tools

Annex A – Acronyms

- **This is the suggested format from the DoD SEP Preparation Guide**
- **Useable outline, not perfect**
  - **Can accommodate all significant systems engineering topics**
  - **Some redundancy**

Work Breakdown Structure (WBS)

- An itemized list of tasks that will need to be performed to complete the project
  - Often used in contracts
  - Also used in project planning and cost accounting
- Typically shown as a hierarchical structure in outline form
  - Product oriented
  - Also captures categories of activity where is no end deliverable product
    - Management
    - Quality assurance inspections

Work Breakdown Structure Example

0.0	Planetary Lander Probe Project
1.0	Lander Probe
1.1	Sensors (Scientific)
1.2	Locomotion and Terrain Sensing subsyste m
1.3	Control subsystem
1.4	Communication subsystem
1.5	Power subsystem
1.6	Lander Body
1.7	Lander Integration and Test
2.0	Delivery Vehicle
3.0	Orbiter Vehicle
4.0	Ground Mission Control Application
5.0	System Integration and Test
6.0	System Engineering
7.0	Program Management
8.0	Quality Assurance
9.0	Documentation
10.0	Ground Test Equipment

Deliverable product

Components as sub-tasks

Work Breakdown Structure Example(2)

Every sub-element…

May have tasks associated with it…

…or it may have sub-elements…

…that require further breakout to capture the needed tasks.

1.0	Planetary Lander Probe
1.1	Sensors (Scientific)
1.2	Locomotion and Terrain Sensing subsystem
1.3	Control subsystem
1.3.1	Conduct Control Subsystem AOAs
1.3.2	Develop Control Subsystem Requirements
1.3.3	Lander Control and Management Software
1.3.4	Navigation Software
1.3.4.1	Define Navigation Software Requirements Spec.
1.3.4.2	Design Navigation Software
1.3.4.3	Code and Unit Test Navigation software
1.3.4.4	Integrate and Test Navigation Software
1.3.4.5	Prototype Navigation Software
1.3.4.6	Software Engr Support to Integration & Test
1.3.5	Navigation Sensors
1.3.6	Control Processor (Purchased Item)
1.3.7	Control Wiring
1.3.8	Control Subsystem Integration
1.4	Communication subsystem
1.5	Power subsystem
1.6	Lander Body
1.7	Lander Integration and Test

Sample Breakout of a WBS item to its deepest level

Risk Management

- The Systems Engineer owns the risk management process
  - Defines the process
  - Conducts periodic risk reviews
- First Thing: Perform a risk assessment for the Engineering Development Phase
  - What are the risks to completing the development?
- Determine key areas to apply effort toward to reduce risk
  - Address in the project plan

Risk Management is discussed in Lesson 7

System Engineering Method in Engineering Development

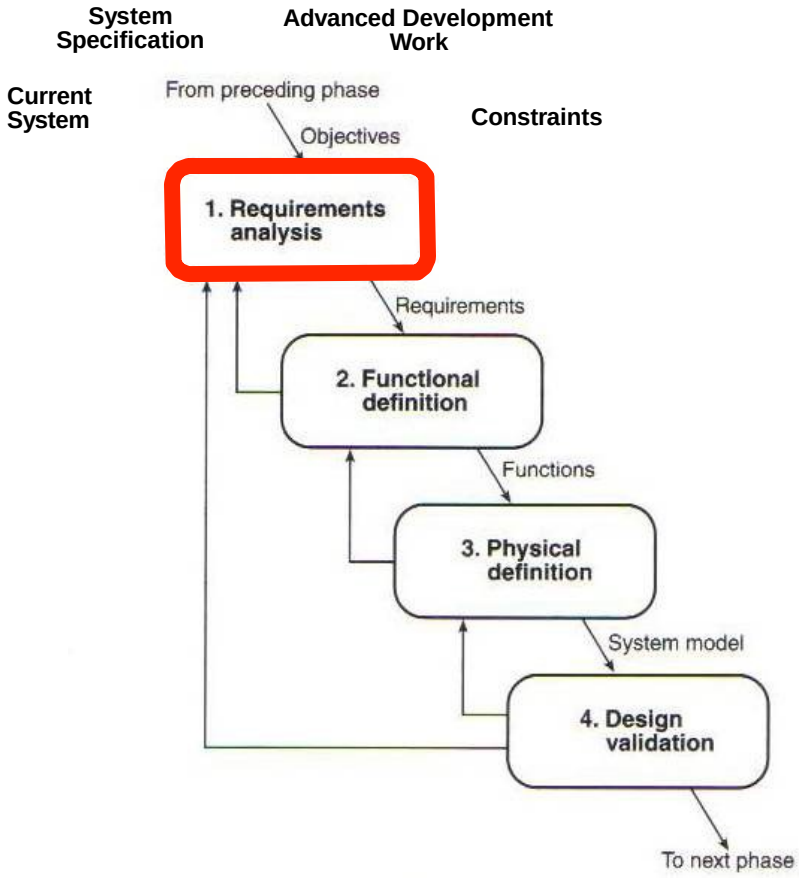


Fig. 3-5 System engineering method top-level flow diagram.

- **Requirements Analysis**
  - **Analyze the system functional specifications**
    - **Understand derivation of functional and performance requirements**
    - **Does proposed derivation of subsystems make sense?**
  - **Are there lower level components/ parts needed?**
    - **Identify interfaces and interactions**

System Engineering Method in Engineering Development

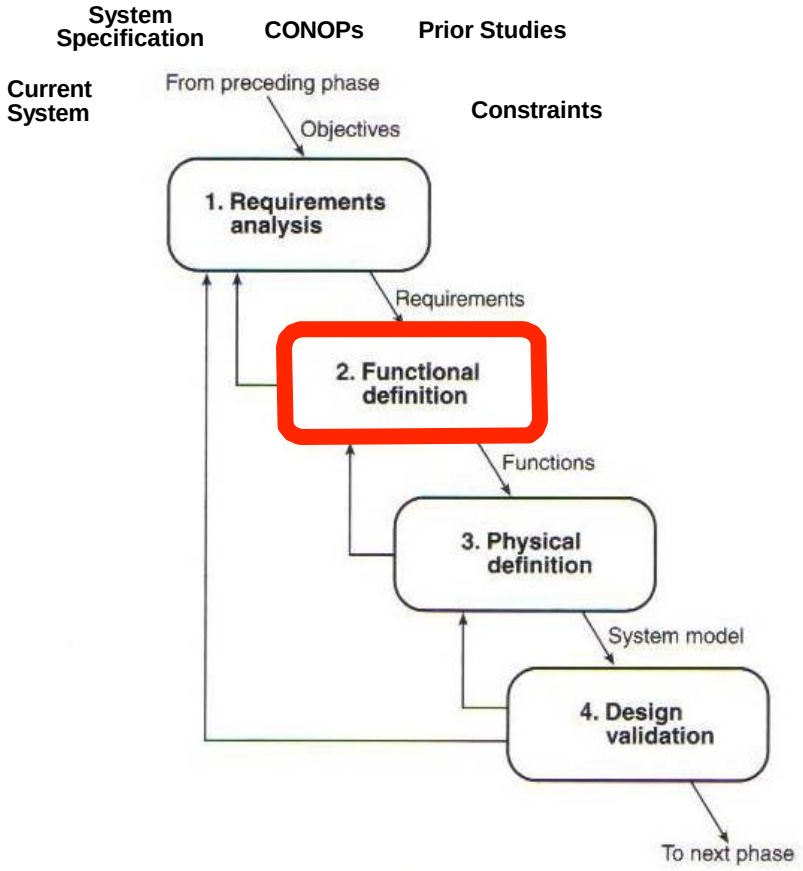


Fig. 3-5 System engineering method top-level flow diagram.

- **Functional Definition**
  - **Allocate functions to next lower level of components/ parts**
  - **Design and prototype user interfaces**
  - **Perform analysis and simulations to identify and/or resolve outstanding performance issues**
  - **Go back and reassess the requirements**
    - **Are the requirements at a low enough level for a single component/part to satisfy it?**
    - **Break down again if needed**

System Engineering Method in Engineering Development

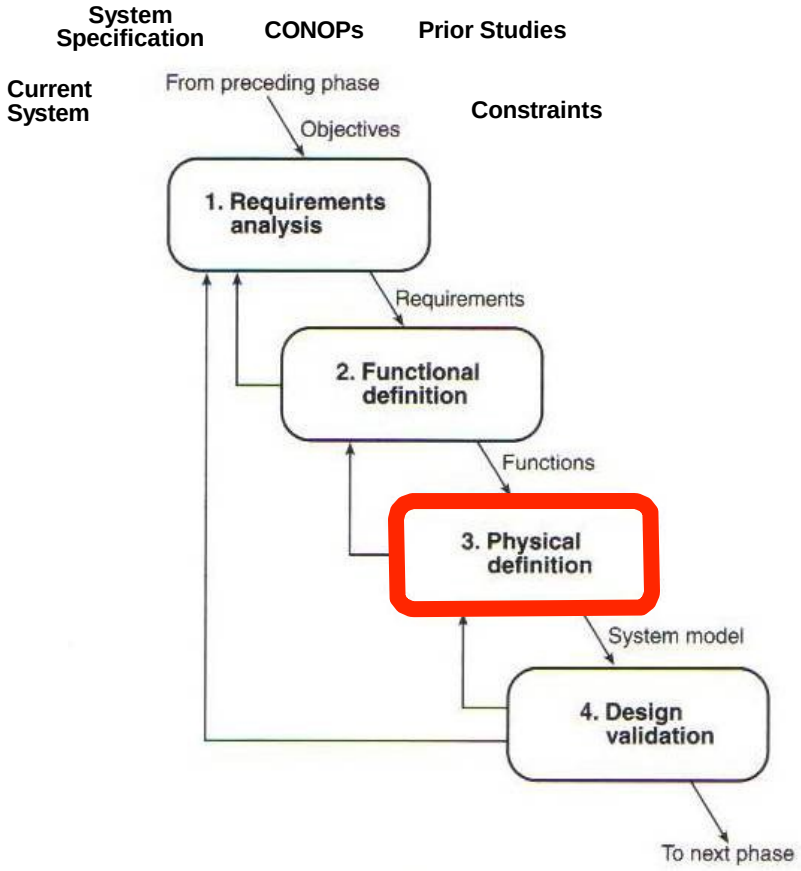


Fig. 3-5 System engineering method top-level flow diagram.

- Build or purchase parts/ components
  - Hardware
  - Software
- Bench or unit test
  - Ensure that the parts and components work before sending to integration and test (in Design Evaluation)
- Correct deficiencies fed back from Design Validation

System Engineering Method in Engineering Development

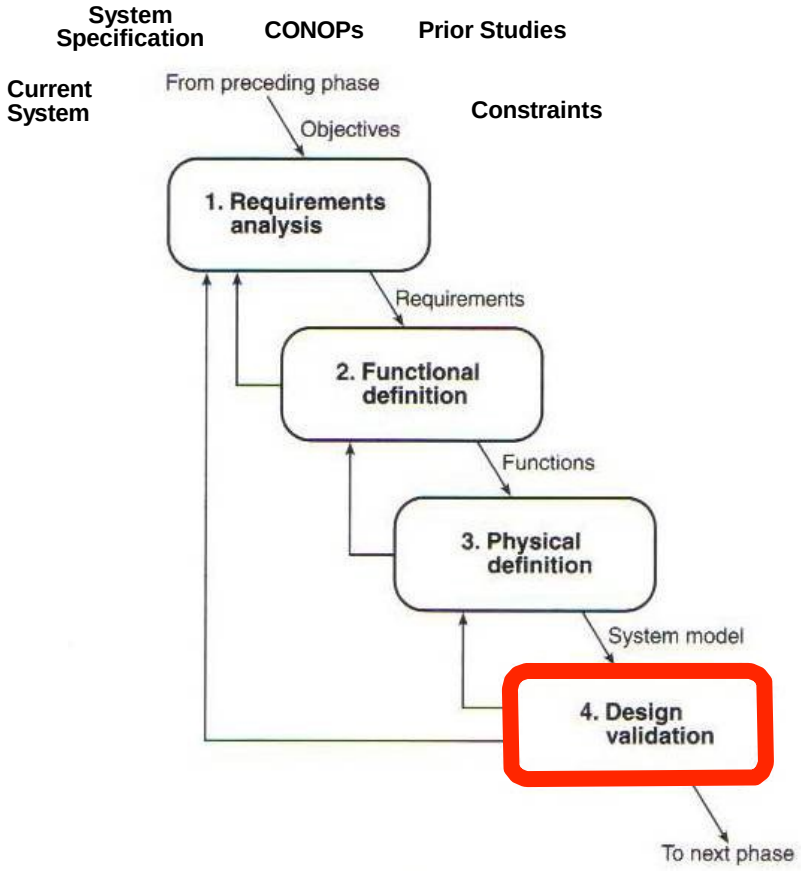


Fig. 3-5 System engineering method top-level flow diagram.

- Integrate the parts & components
- Ensure that the Engineering Development requirements are being met
  - Develop procedures to verify the product meets its specification
- Provide feedback to Physical Definition to correct deficiencies
- Recommend changes to design requirements

Design: Develop Specifications & Allocate Requirements

- The System Engineer owns the System Specifications (and possibly subsystem specifications)
- Performs functional analysis to determine how system requirements should be allocated to subsystems/major components
  - Defines the system building blocks
  - Defines the interfaces between them
- Ensures traceability of requirements
  - From contractual requirement to where requirement is implemented
- May oversee lower level component or part requirement definition being performed by design teams
- This is an iterative process until all parts have been identified and specified by the design teams

Example Tool: Functional Block Diagram

Top Level Blocks

Each Top Level Block can be Broken down into More detail

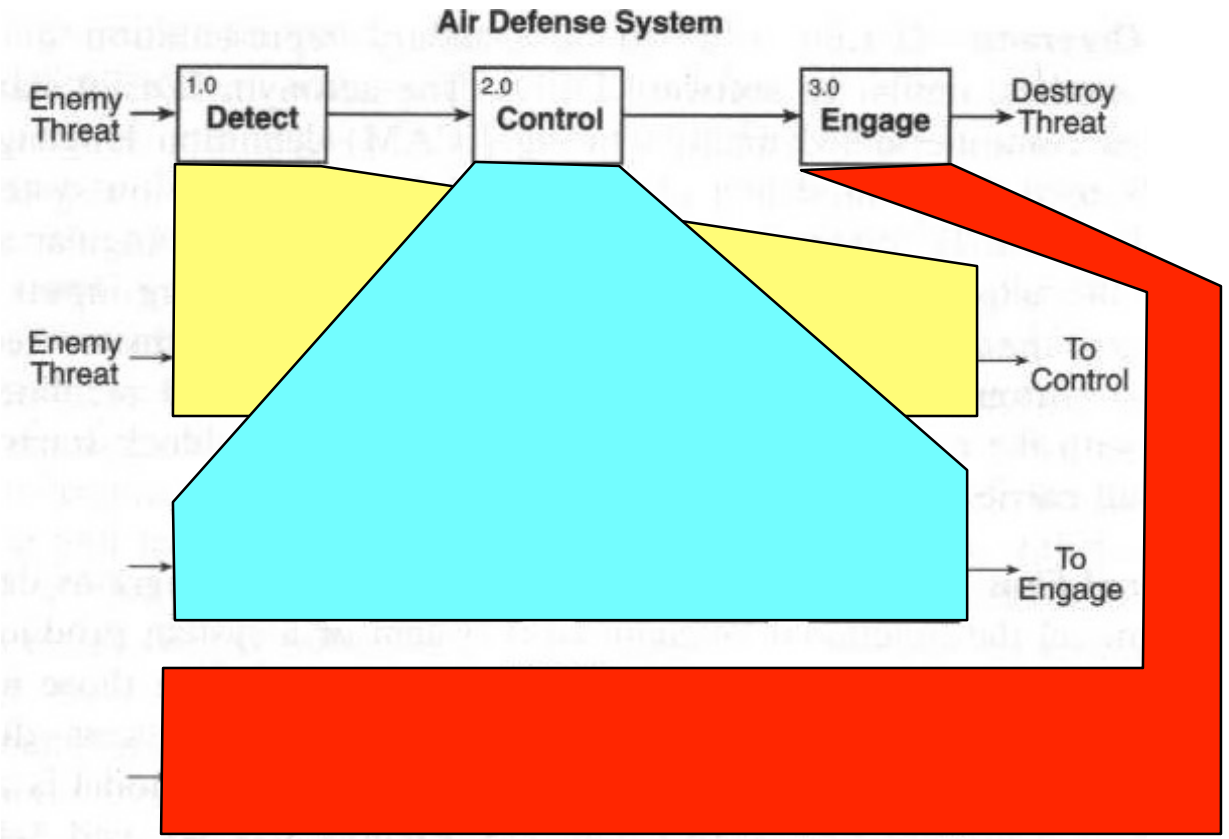
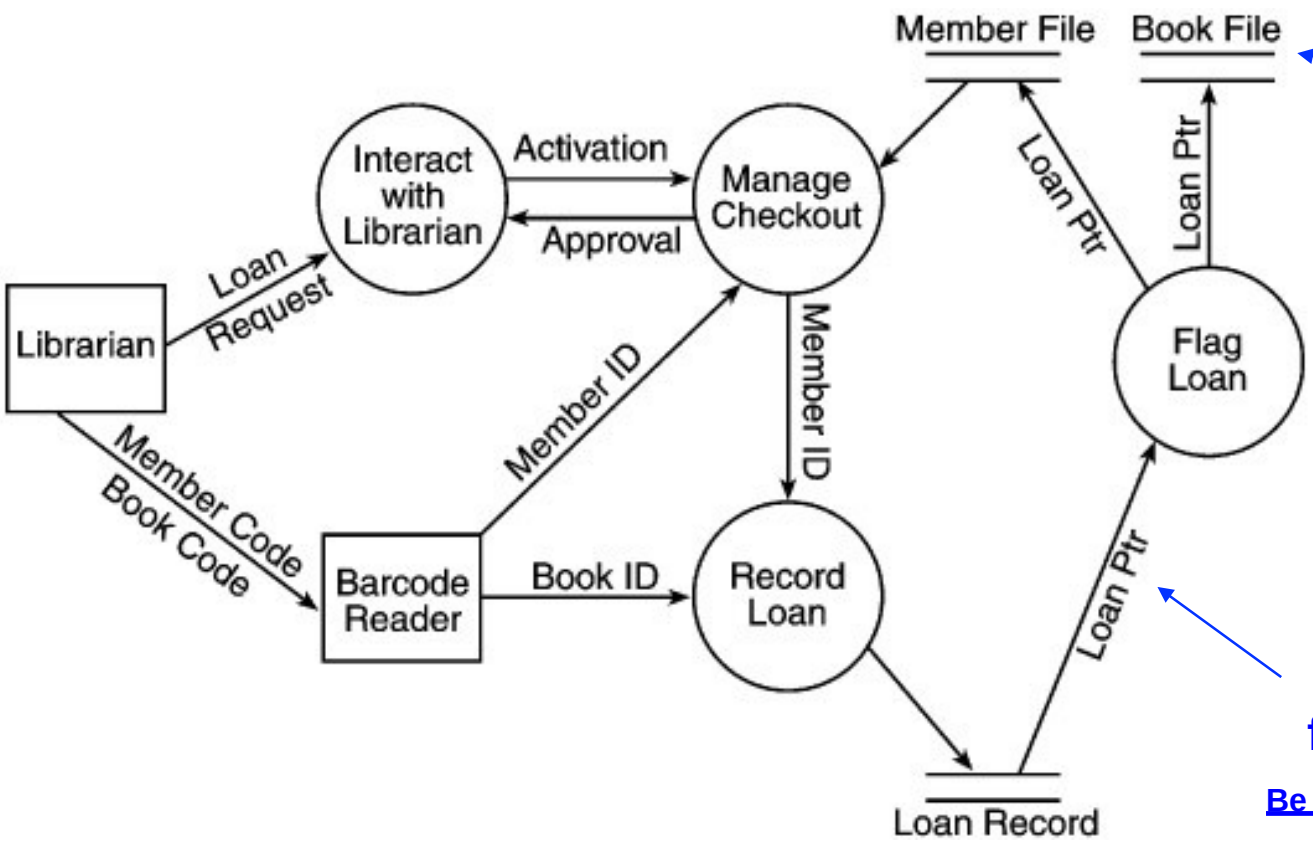


Fig. 14-3 Air defense function flow block diagrams.

Example Tool: Data Flow Diagrams



Data Stores

Functions

Labels of content for data exchanged

Be Specific - Never just use "data"

Depicts system functions and data stores,

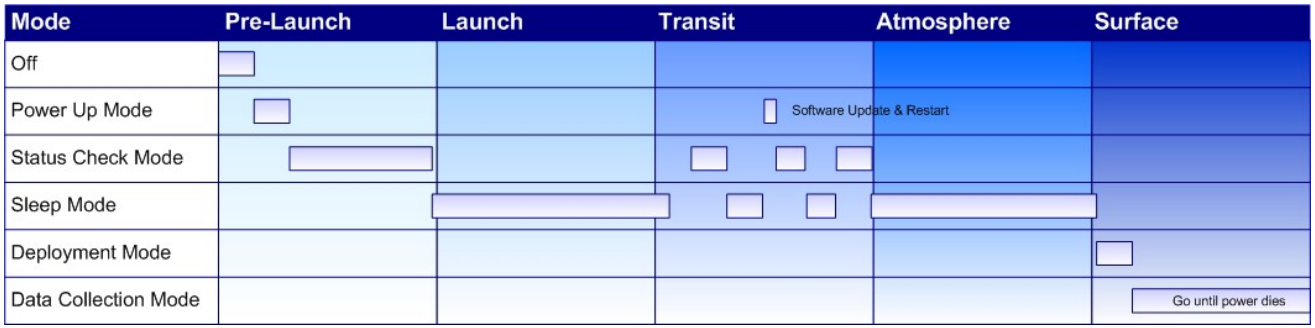
along with interactions between them

Example Tool: Capture States & Modes

- A system (or component) may not always be required to meet all its requirements at all times
- Mode: An application or usage where the system will be required to perform specific functions or have specific performance characteristics
- State: Sometimes used interchangeably with Mode
  - Often used to describe how a system responds in response to a stimulus
  - e.g., state transition diagram
- Address as requirements in the specification

Mode Example

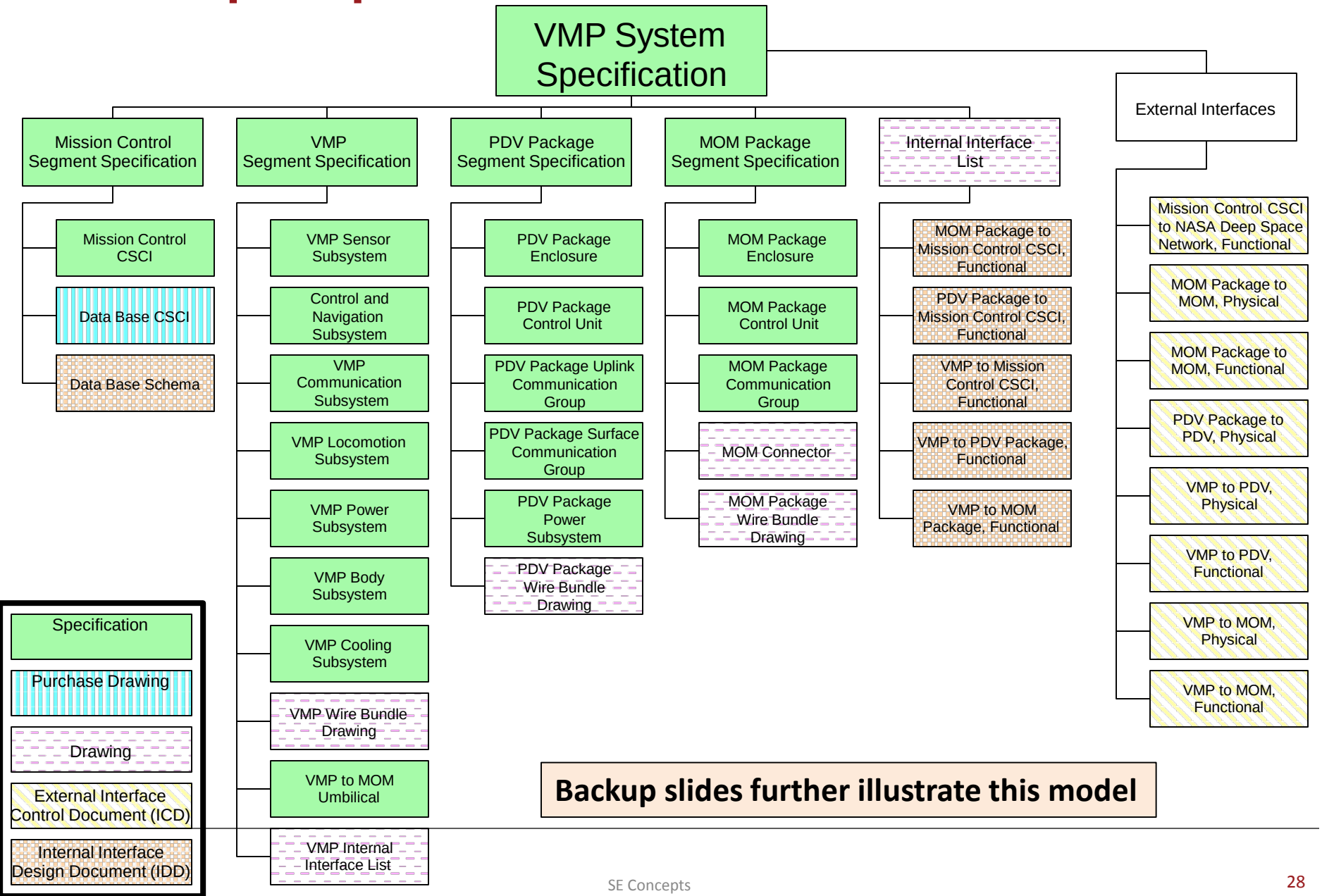
- **Space Probe carried by a launch and landing vehicle**
  - **“Off” – shut down, no functionality, no power usage**
  - **“Power Up Mode” – Perform internal checkout**
  - **“Status Check Mode” - connect to mission control for monitoring**
  - **“Sleep” – Shut down non-essential components to conserve power**
  - **“Deployment Mode” – Unpackage probe and separate from landing vehicle**
  - **“Data Collection Mode” – Turn on sensors, move around and transmit data back to mission control**



Completing the Process

- As more detail is defined, the specifications form a tree of related products
- Lower level specifications are based on allocated requirements from the higher level specification
  - Also guided by enterprise level concerns
    - Preference for specific solutions (commonality)
    - Flexibility for future system or enterprise level evolution
- The systems engineer is responsible for ensuring that the "ilities" are addressed in the process
  - Ensure analyses or evaluations are performed to make sure the end design will meet the overall system requirements
  - Update analyses as required

Sample Specification Tree



Backup slides further illustrate this model

Configuration Control

- **The Systems Engineer owns Configuration Management of specifications and major interfaces (external and cross-subsystem)**

- **Ensures that the current copy of the specifications and interface documentation is consistent and available**
- **Defines a disciplined process for making changes**
  - **Conducts the Configuration Control Board (CCB)**
  - **Records change requests and approved changes**
  - **Ensures specifications and interface documents are updated with the approved changes**
  - **May also decide on a tool to use for managing documentation**

Plan and Run Design Reviews

- A coordinated evaluation of the evolving design
- Purpose:
  - Formal check of the evolving design against the proposed system/subsystem specification requirements
  - Provide a common reference point for all project personnel
    - Cross team communication
    - Opportunity for design engineers to explain design approach
    - Opportunity to identify major problem areas and determine a course of action to correct
  - Provide a formalized record of design decisions and reasons for making them

Preliminary Design Review

- Demonstrate that the chosen system design will be able to meet the system requirements
- Review:
  - Design and interface specifications
    - System, subsystem, major components
  - Supporting design and effectiveness trade studies
  - Mockups, models and breadboards
  - Interface design
  - Software top level design
  - Development, integration and verification test plans
  - Engineering specialty studies
    - Reliability, maintainability, safety, logistics, EMI/EMC, etc.

Detailed (Critical) Design Review

- Present the entire system design
- Demonstrate the detail design will meet allocated requirements
- Review:
  - Design and interface specifications
    - All components and key parts that have a specification
  - Subsystem detailed engineering drawings
  - Mockups, models, breadboards and prototype hardware
  - Interface control drawings
  - Software detail design
  - Development, integration and verification test plans (and possibly procedures)
  - Engineering specialty results based on the design
    - Reliability, maintainability, safety, EMI/EMC, etc.
  - Logistics support plans
  - Quality Assurance Plan
  - Final resolutions to issues raised at the preliminary design review

Arbitrate Cross-Team Technical Issues

- **The System Engineer owns the internal subsystem to subsystem and the external interfaces**

- **If problems arise, the systems engineer adjudicates how to resolve them**
  - **Assign responsibility to subsystem X fix**
  - **Change the interface**
  - **Change the system/subsystem requirement**
  - **Go back to the customer/user to investigate whether the user requirement can be changed**

Plan Qualification Activities

- **The System Engineer is responsible for proving that the specification has been met by the system – "Qualification"**

- **Ensure that the qualification activities will meet the verification requirements in the system specification**
  - **Participate in development of the Qualification Plan**
    - **DoD: Test, and Evaluation Master Plan**
  - **Ensure test procedures will provide the evidence needed to ensure the system has met the requirement**

Perform Functional Configuration Audit (FCA)

- **Purpose: Prove that the system met its requirements and that each of its parts meets their detail design specifications**

- **Step through each specification requirement**
  - **Typically in section 3**
- **See how the requirement is to be verified**
  - **Typically in section 4**
- **Review the associated verification data**
  - **Does the verification information indicate that the right verification was done and does it also provide the evidence that the requirement has been met?**
- **Capture actions that require correction**
  - **Fix the document**
  - **Find the missing evidence**
  - **Perform additional verification if needed**

Perform Physical Configuration Audit (PCA)

- Purpose: Ensure that the as-built system matches its design documentation (e.g., drawings)
  - Allows for formal delivery of design documentation
  - Also allows for contracting of future copies of the product to "built to print"

- Examine each document/drawing (typically sampled)
- Ensure the document/drawing adequately captures the necessary design information
- Inspect the associated part/component/assembly and ensure that the as-built unit matches the document/drawing
- Capture actions that require correction
  - Fix the document
  - Fix the part
    - Re-verify if needed

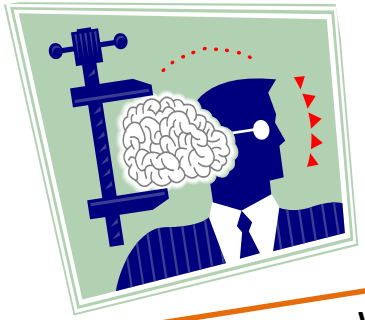
Exercise: Planning for the Projects



Instructor will provide directions for the exercise

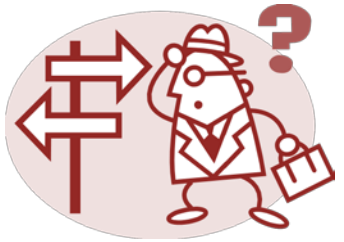
Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future

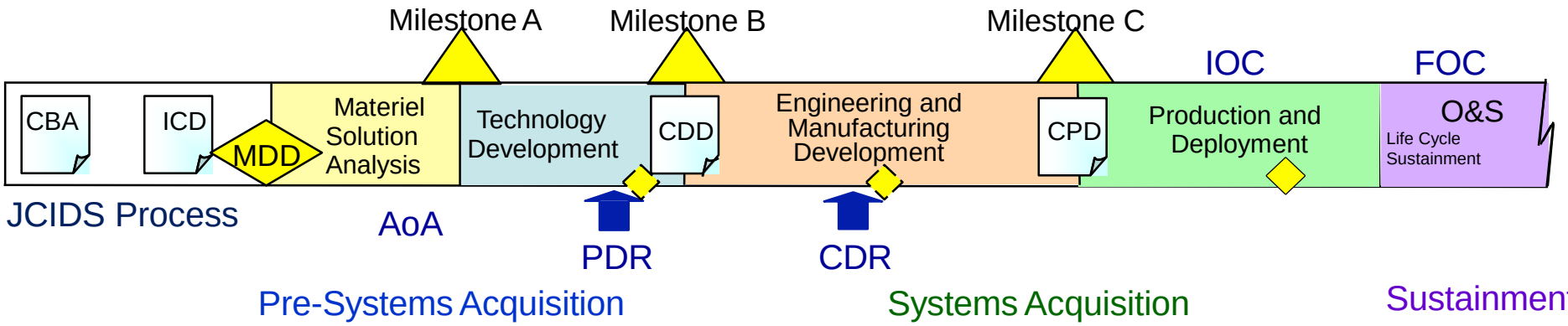
What are your Questions before We Continue?



Back-up Slides

Engineering and Manufacturing Development Phase, Part 1–DoD View (2008)

- **Integrated System Design**
  - **Purpose: Develop the capability defined by the user requirements**
  - **Reduce manufacturing risk; support Milestone C Production decision**
- **Successful "Milestone B" decision to Start**
  - **Senior DoD leader approves development**
  - **Requires significant backup data justifying the development and providing development planning information**
    - **Validated user requirements (Capability Development Document)**
    - **System engineering plan, cost, schedule for development**
    - **Acquisition strategy**
    - **How the proposed system will fit into the DoD enterprise architecture**



Example Tool: Unified Modeling Language *(UML)*

- **Widely used in software development**
- **Includes many diagrams to capture a visual view of system components, their interfaces, and how they interact**
- **Example: Use case depicting a system's functions and their relationships with external entities**

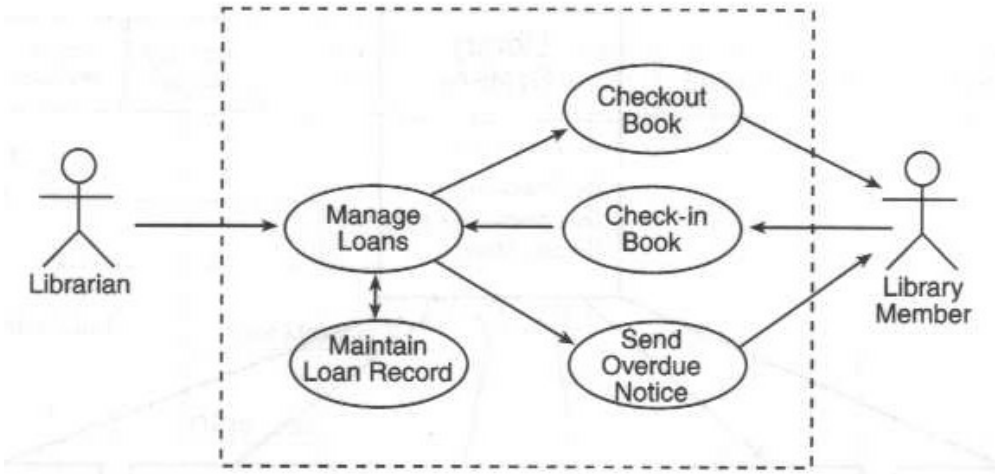
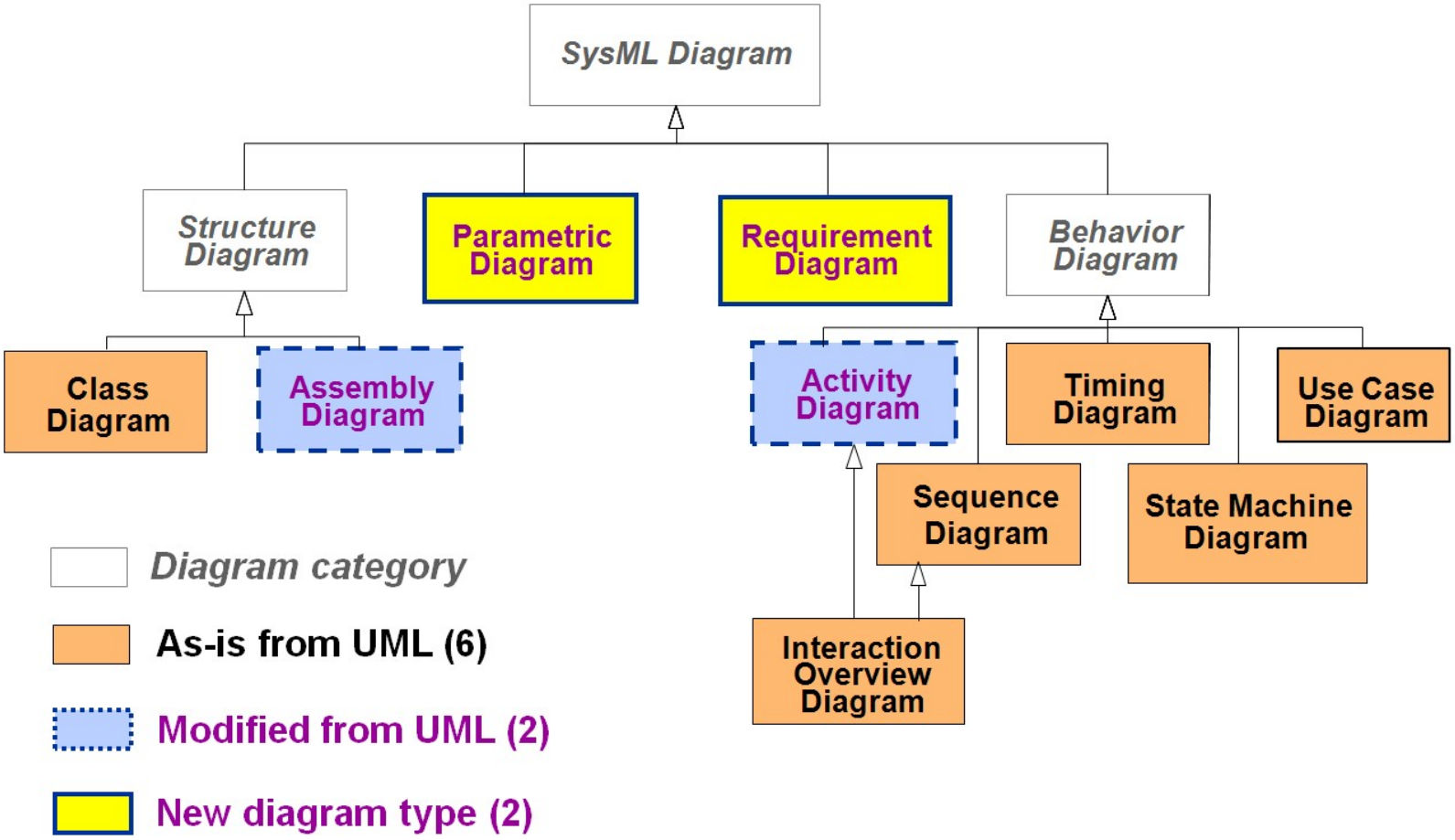
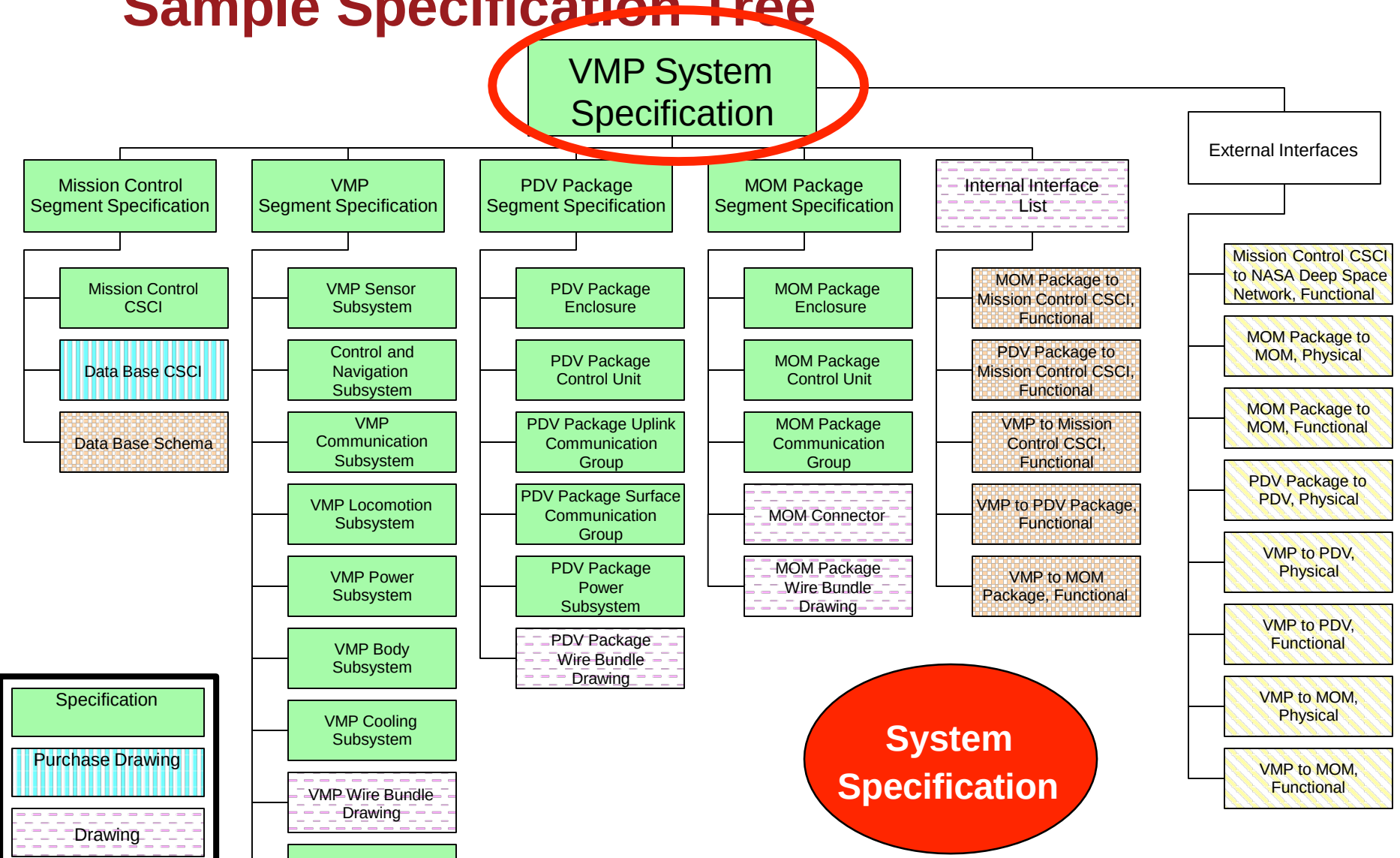


Fig. 13-7 UML use case diagram: library loan management

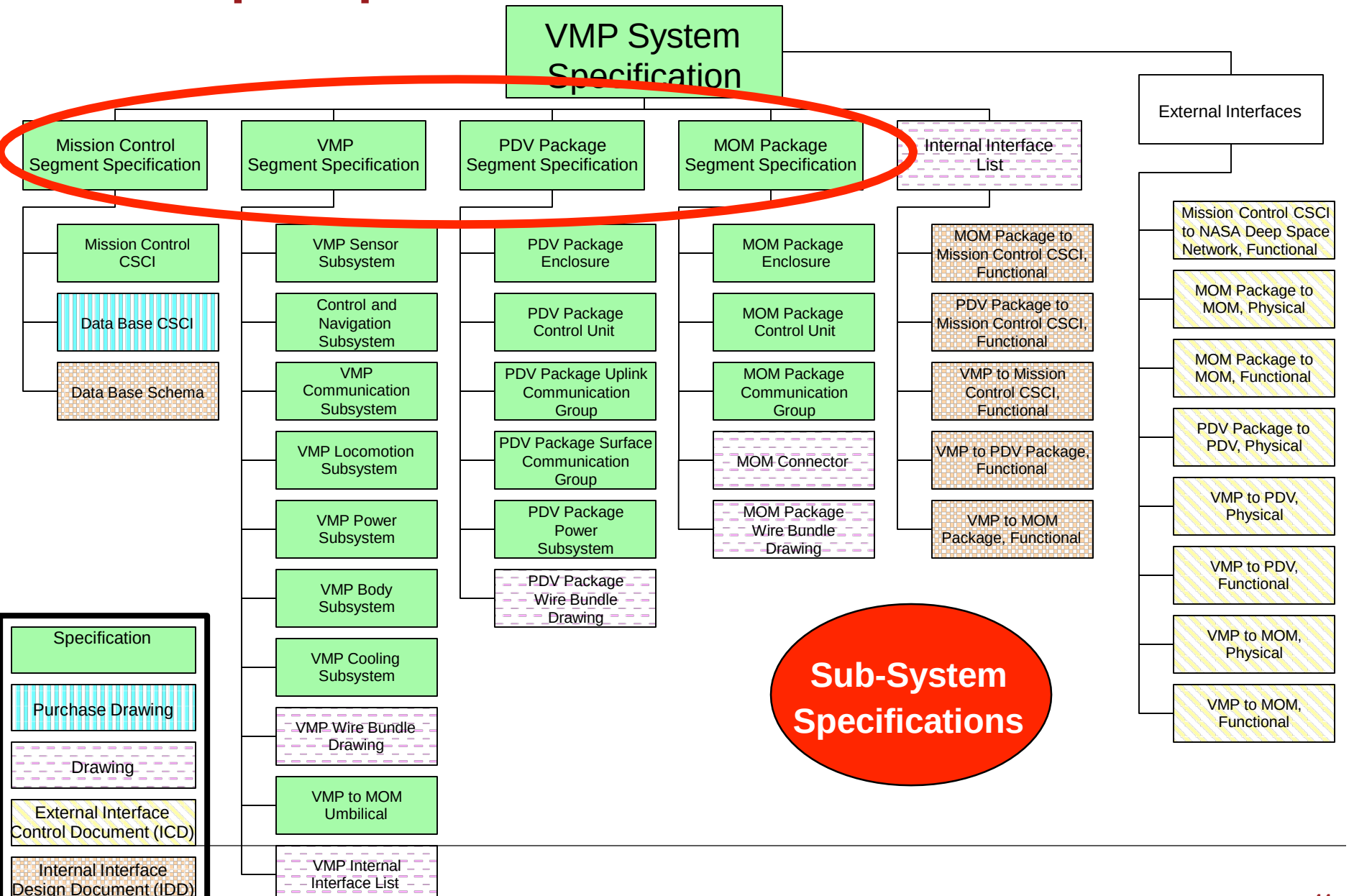
Example Tool: Systems Modeling Language *(SysML))*



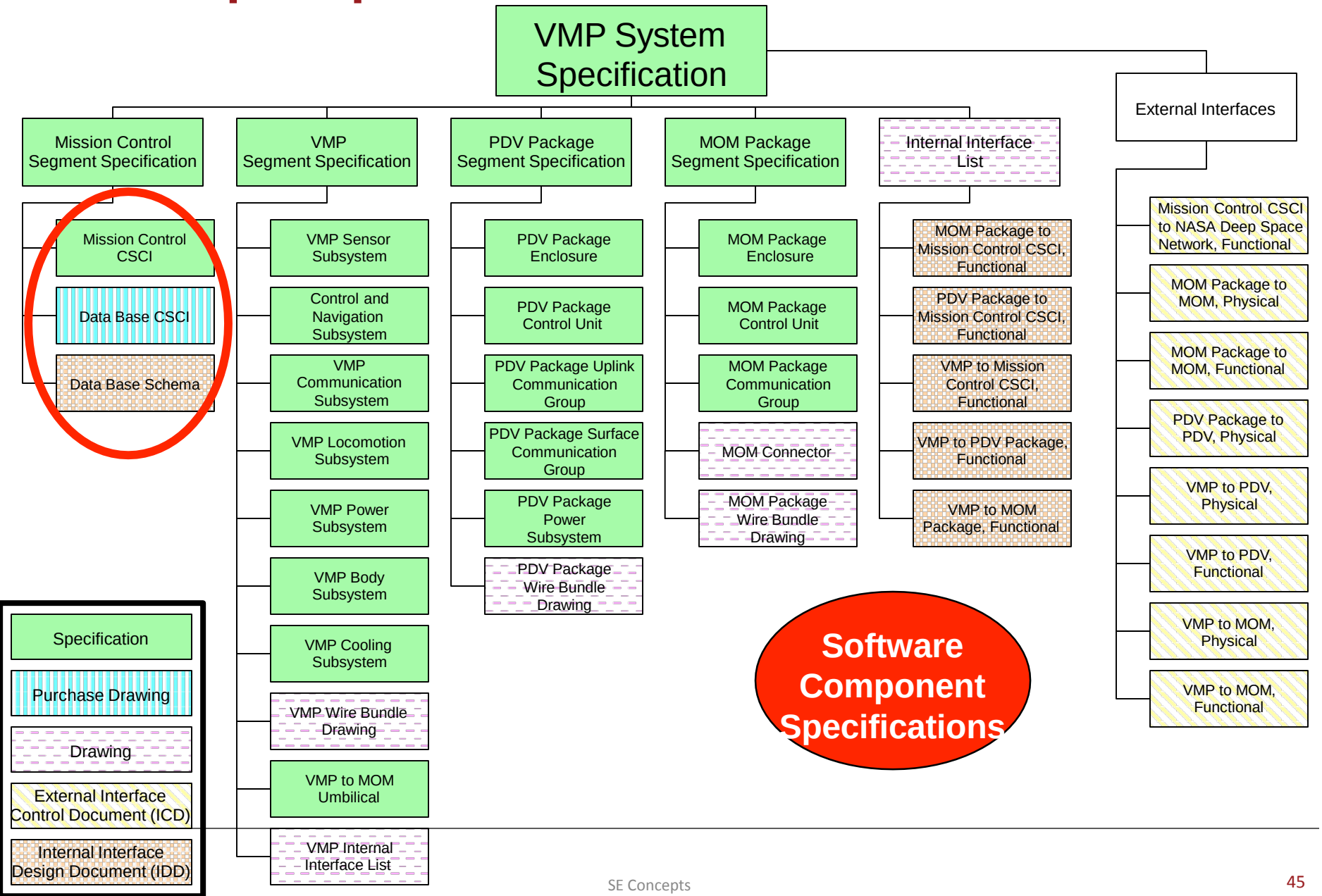
Sample Specification Tree



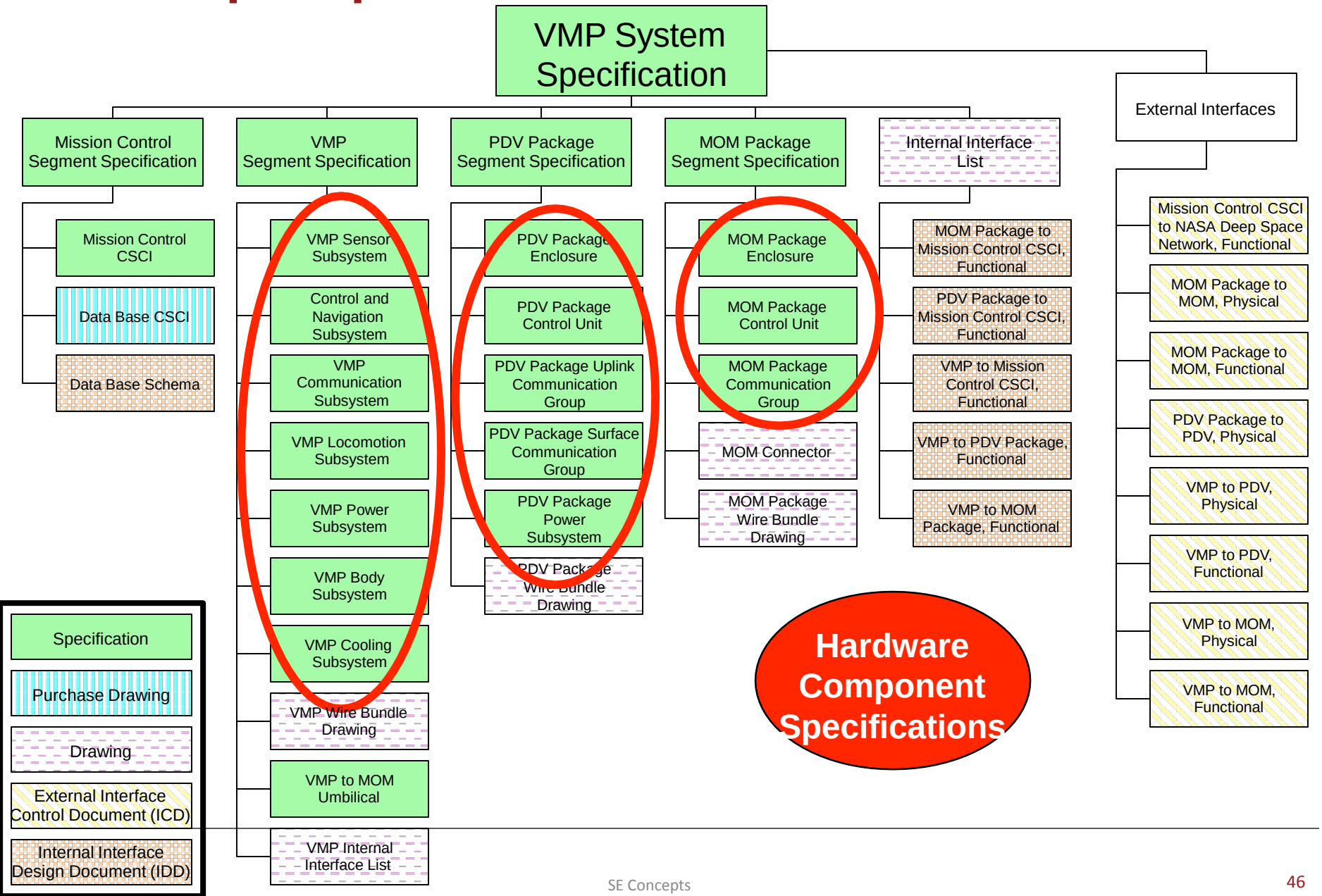
Sample Specification Tree



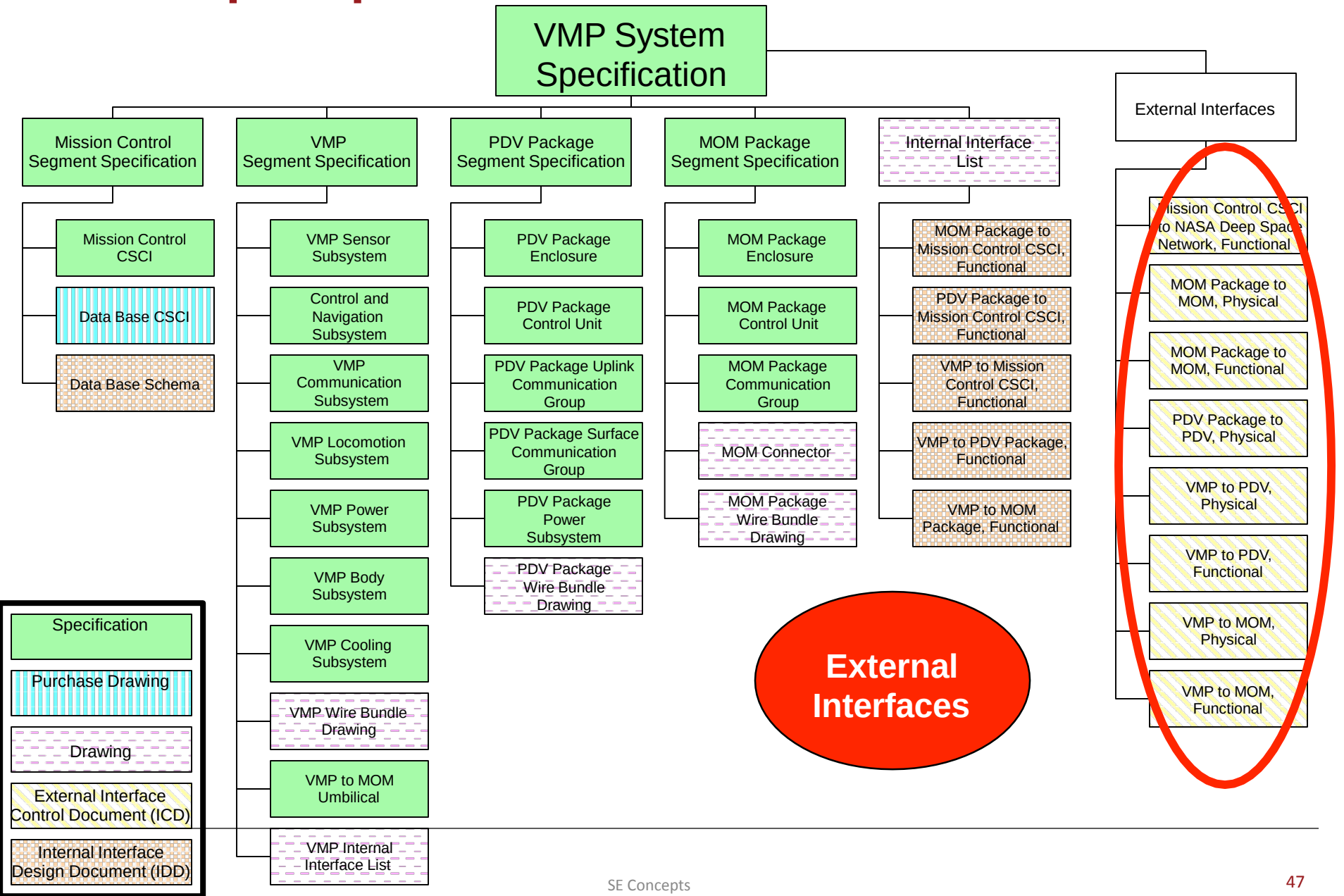
Sample Specification Tree



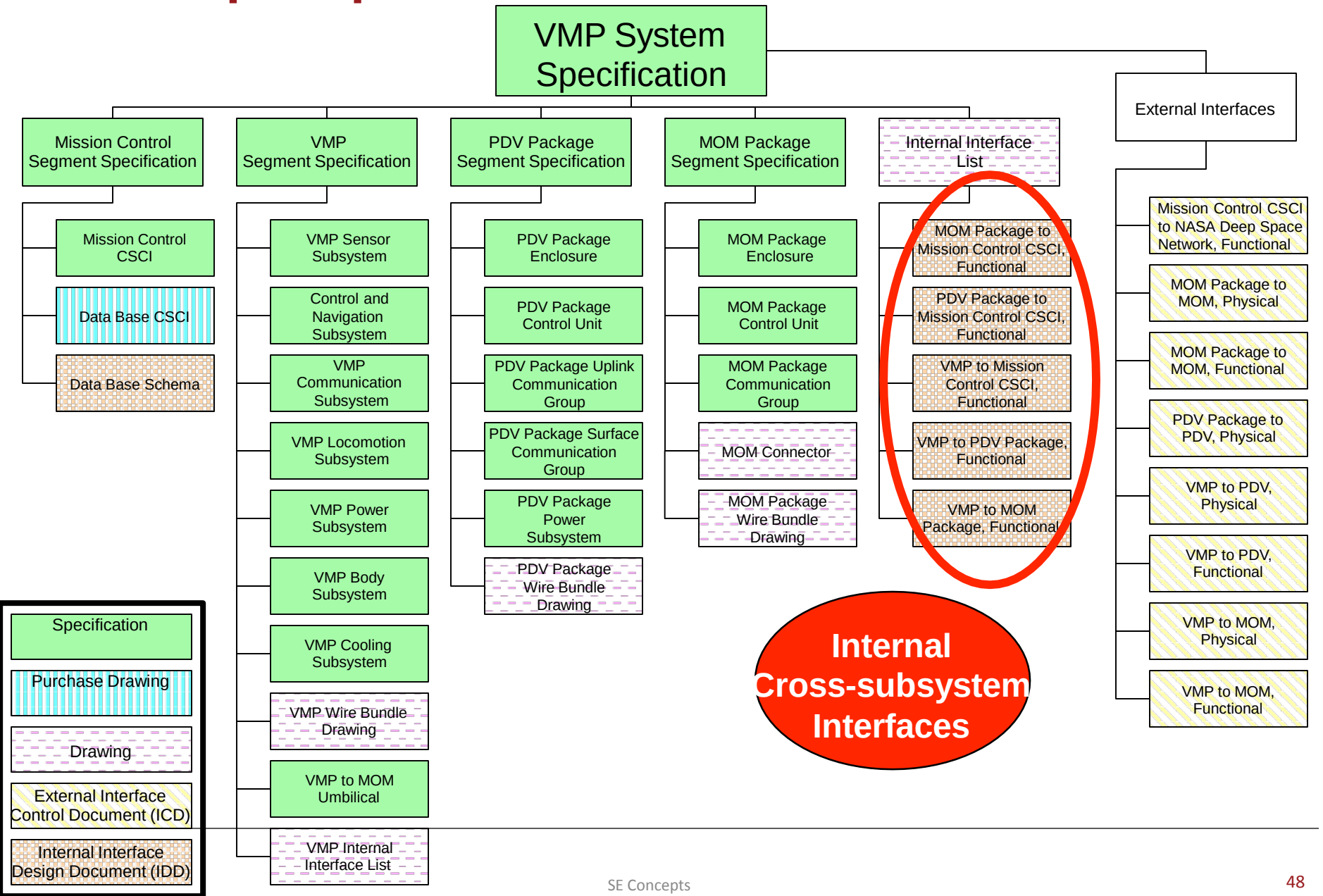
Sample Specification Tree



Sample Specification Tree



Sample Specification Tree



Systems Engineering Technical Short Course

(Session 1)

Lesson 13

Production and Deployment

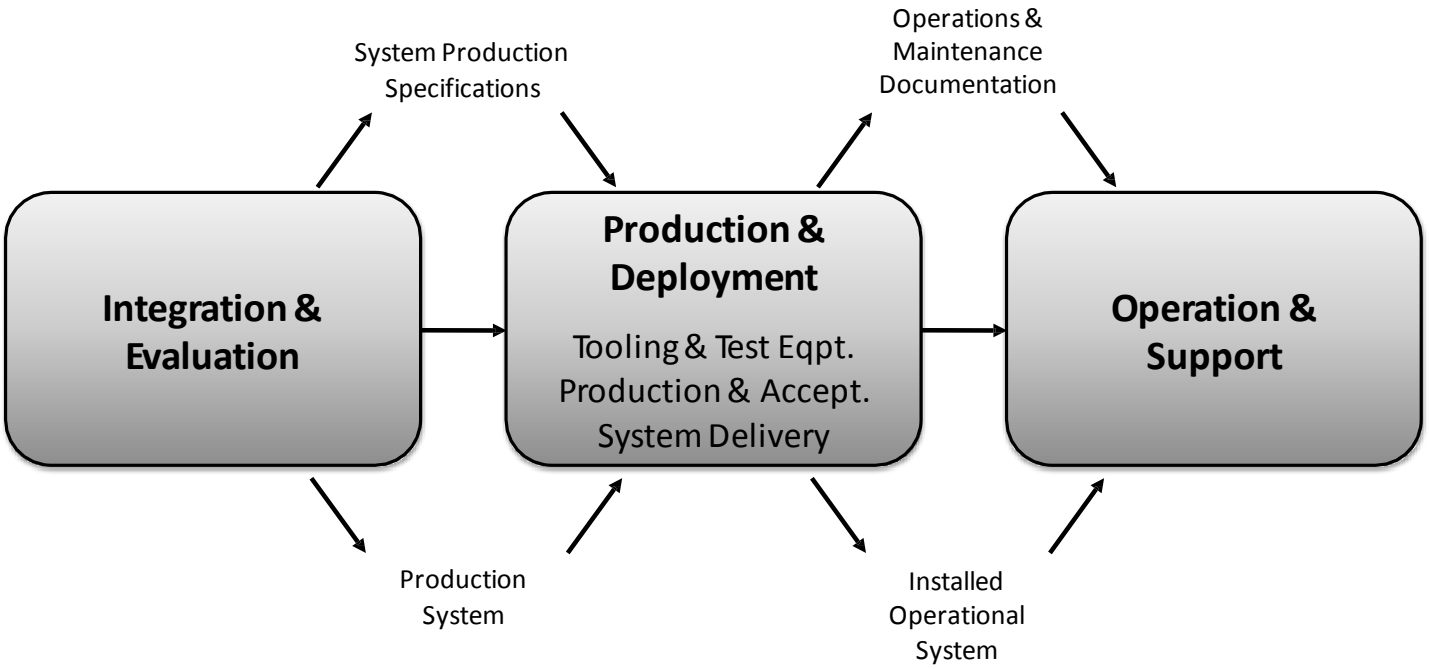
Reference : "Systems Engineering, Principles and Practice," Kossiakoff et. al – Chapter 14

Session 1 Agenda (Jan 27 – 29): Timing

Day	Overview and Concepts of Systems Engineering	Hours*
1	Waters Management kick-off	1/2
	Introduction and overview of entire program (Lesson 1), including Introduction and overview of the projects (15 minutes each)	2
	Presentation and discussion of Concepts of SE (Lessons 2 – 9)	4
	Project assignments, team formations, and working sessions to apply SE concepts to the projects	1
	Day 1 wrap-up: Takeaways and Q&A	1/2
2	Review of day 1 working sessions	1
	Concepts of Systems Engineering (concluded) (Lesson 10 – 14)	4
	Project team working sessions (continued)	2
	Day 2 wrap up and Q&A	1
3	Team working session feedback to class (informal presentations)	1
	Project team working sessions (concluded)	5
	Check-in with class on application of SE concepts to their projects	1
	Session 1 wrap up: Q&A and homework due next session	1

* Times include breaks, and may be changed based on needs of the class

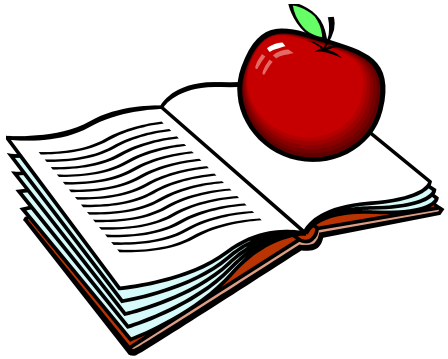
Production and Deployment



….Can our system be efficiently produced and deployed into target environment?

Concepts of Systems Engineering

1. Program Overview
2. What is Systems Engineering?
3. Structure and Environment of Systems
4. Systems Development Process
5. Systems Engineering Processes Overview
6. Systems Engineering in Needs Assessment
7. Risk Management
8. Concept Exploration
9. Concept Definition
10. Advanced Development Stage
11. Engineering Design Stage
12. Integration and Evaluation Stage
13. Production and Deployment
14. Operations, Support, and Disposal



Lesson Content:

- Production
- SE in the phase
- Engineering for production
- Logistic support considerations
- SE Tasks

Plus Appendices

Production:
We have a product, what's left to do?

- **Goals:**
  - **Reproduce what has been designed and proven**
  - **Get the product to the customers**

- **End Result:**
  - **Products**
  - **Happy customers**



Production Phase in the System Life Cycle

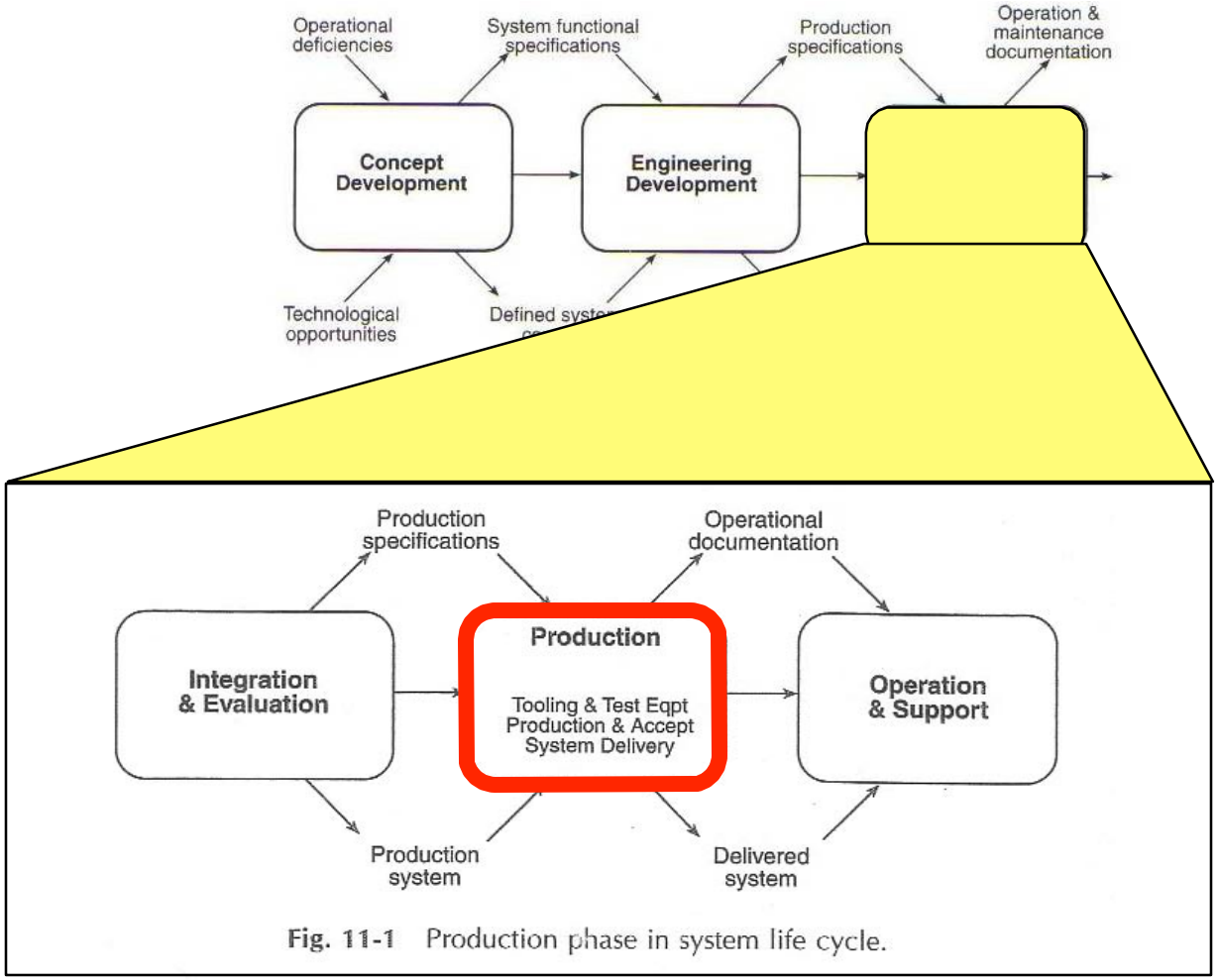
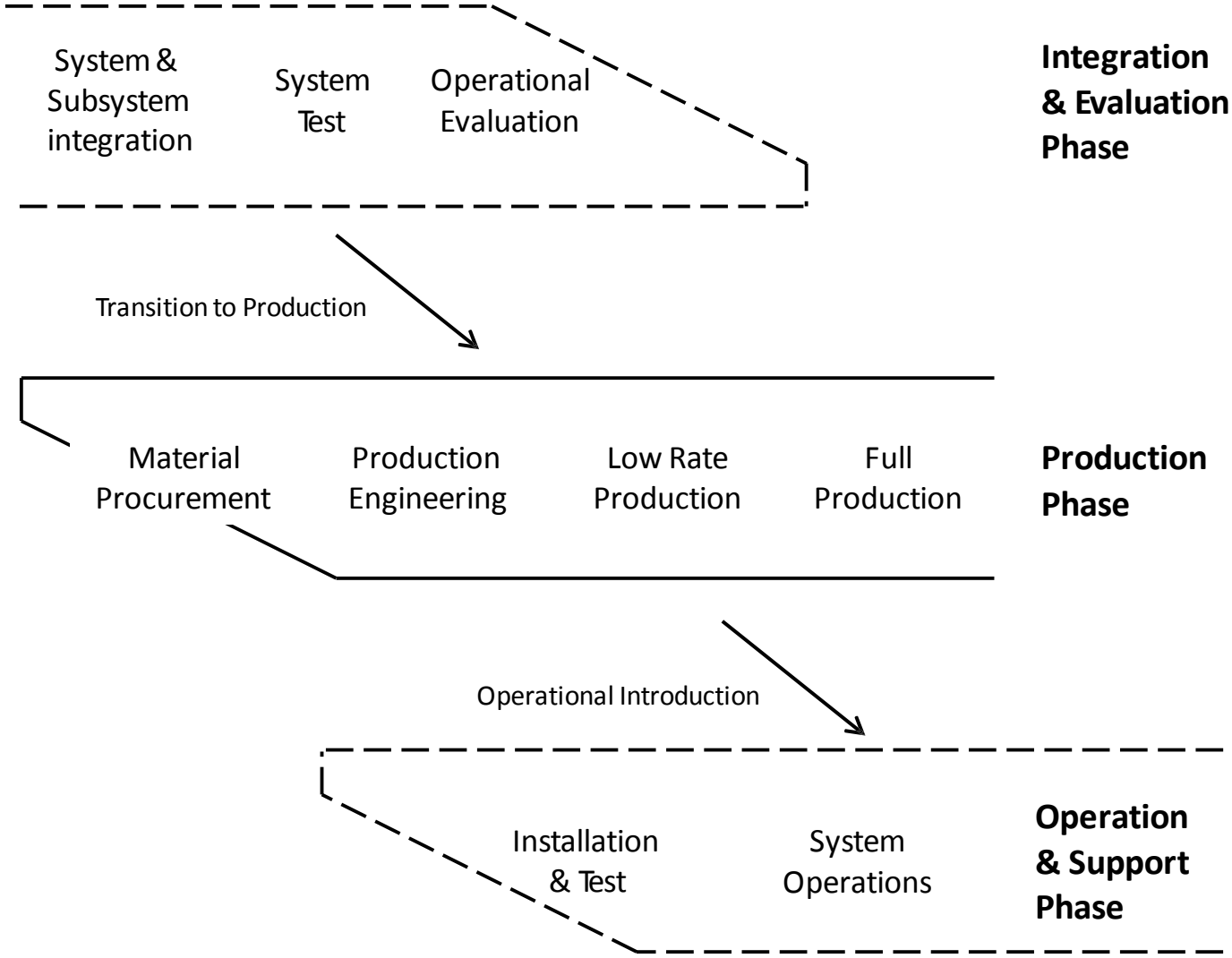


Fig. 11-1 Production phase in system life cycle.

Production Phase Overlap with Adjacent Phases



Starting Points for Production

- **Proven Design**
  - **Prototype(s) that have been tested and shown to meet the specifications**
- **Specifications**
- **Drawings**
- **Documentation**
- **Supportability Plans**

System Engineering Method in Production

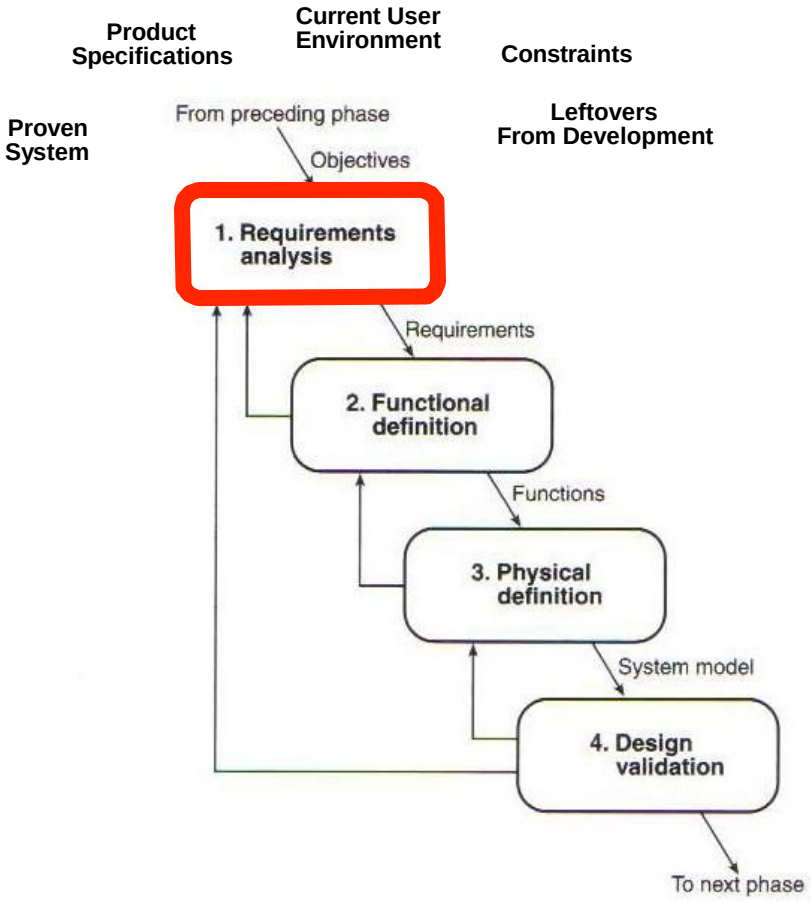


Fig. 3-5 System engineering method top-level flow diagram.

- **Requirements Analysis**
  - **Determine whether the requirements and drawings are ready for production**
  - **Identify whether requirement changes are needed**
  - **Define acceptance of the produced product**

System Engineering Method in Production

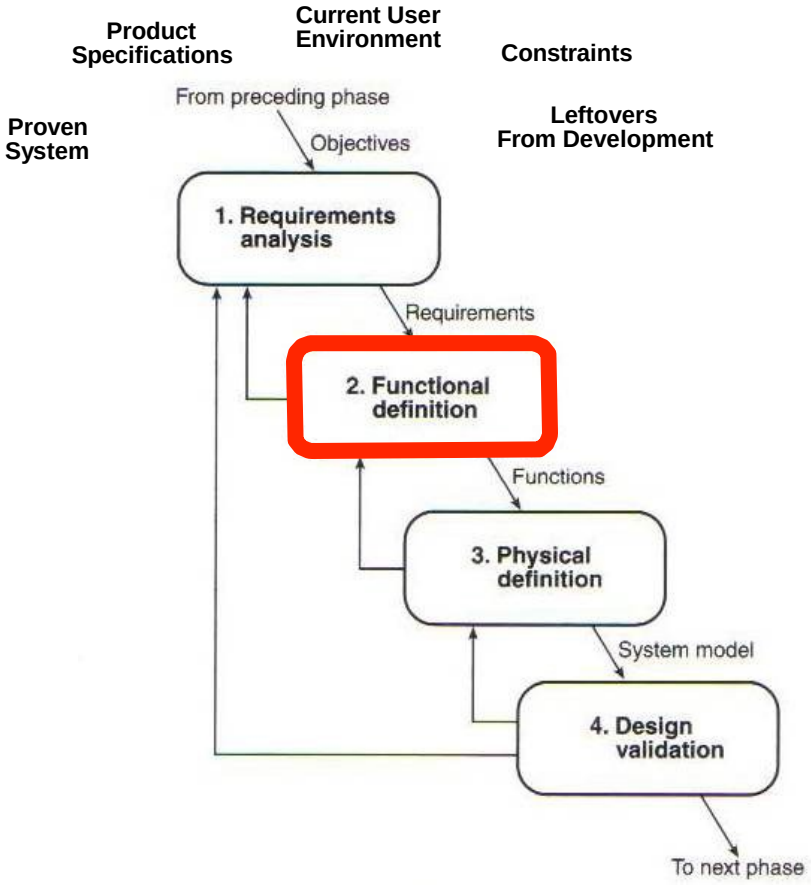


Fig. 3-5 System engineering method top-level flow diagram.

- **Requirements Definition**
  - **Define new requirements and tests needed to verify them**
    - **Done in context at the right level of specification**
      - **System**
        - **Subsystems**
          - **Components**

System Engineering Method in Production

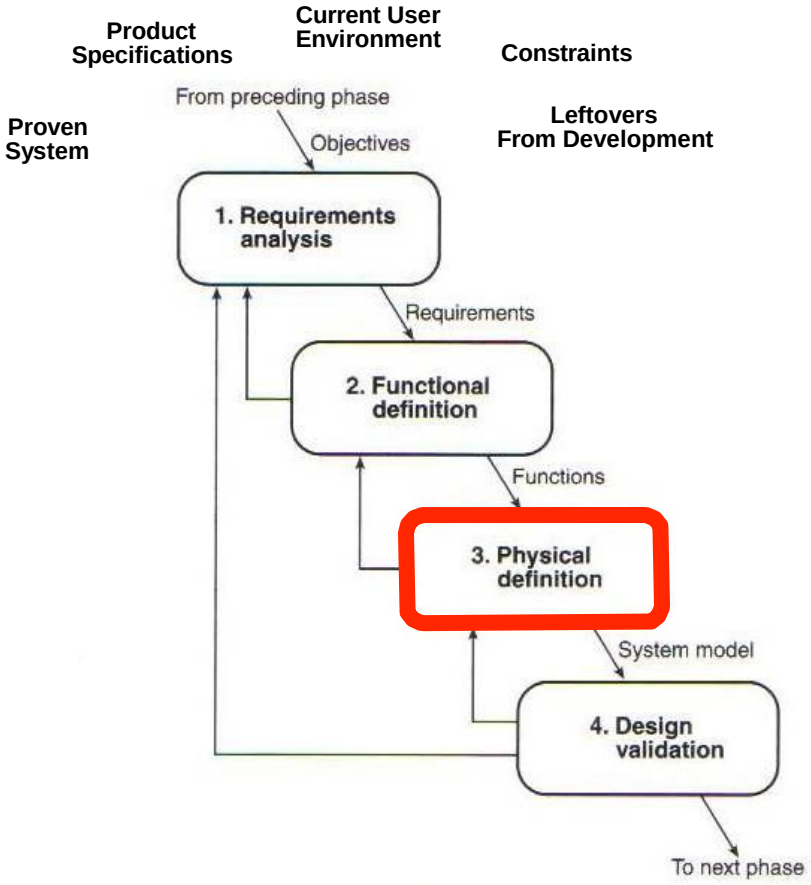


Fig. 3-5 System engineering method top-level flow diagram.

- **Physical Definition**
  - **Configuration management**
  - **Manage changes to the production articles**

System Engineering Method in Production

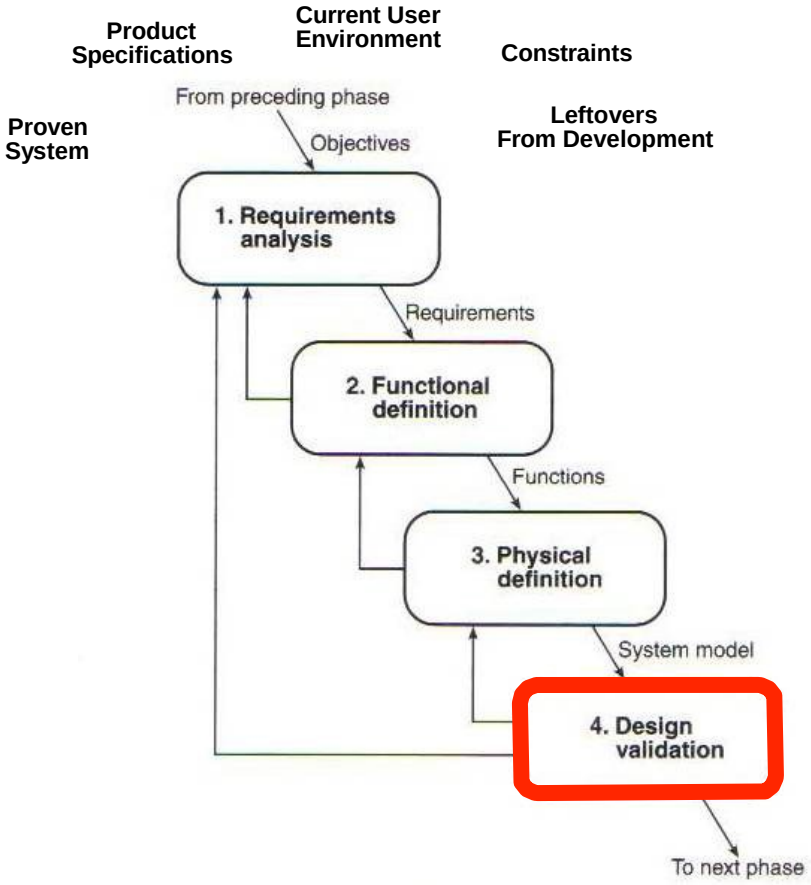


Fig. 3-5 System engineering method top-level flow diagram.

- **Validate Requirements**
  - **Ensure end product meets the acceptance criteria for delivery**
  - **Ensure changes are verified against original/revised requirements**

Engineering For Production

- ***How*** **you build something is as important as** ***what*** **you are building**
  - **If you can't reproduce it, it may be useless**
- **Many considerations…**
  - **Number of systems to build**
  - **Manufacturing considerations**
    - **Procedures**
    - **Processes**
    - **Equipment**
  - **Deployment/distribution considerations**
  - **Logistics support considerations**
  - **Configuration and document control considerations (e.g., engineering changes)**
- **These should be considered from the** ***beginning*** **of the project starting with the needs analysis!**

Manufacturing Considerations (1/3)

- **Case 1: The development system is the production system**
- **Case 2: Copies of the development system need to be produced**
  - **2A: Few copies**
  - **2B: Many copies**
  - **2C: Software**

Which case applies to the Waters Projects?

Manufacturing Considerations (2/3)

- **Limited Production Line Resources**
  - **Vendor facilities & personnel**
  - **Limited facilities worldwide**
    - **Advanced integrated circuits**
    - **Exotic materials**
    - **Large components**

- **Long Lead Components**
  - **Some parts take from months up to years to buy**
    - **Specialty products**
    - **Obscure connectors**
    - **Custom integrated circuit products**

Manufacturing Considerations (3/3)

- **Tooling**
  - **Building 1 unit may be acceptable to build hand-crafted**
  - **May not be economical to build ten units that way**
    - **Develop and build special tooling**

- **Factory Test Equipment**
  - **Same as tooling – may want to build if producing in quantity**

Logistics Support Considerations

- **Maintenance**
  - **By the manufacturer/vendor or the user?**
    - **Affects documentation that needs to be produced and delivered**
    - **Affects training to be developed – in-house or user self-train?**
  - **If user maintenance…**
    - **Generally more documentation needed for user self support**
    - **User may need special test equipment**

- **Spare Parts**
  - **Initially delivered with a system**
  - **Built and stored for future needs**
    - **Depends on product lifetime, time committed to support product**

- **Long term plan for obsolescence**
  - **Proactive or reactive**

Logistics Maintenance Approach Effects Production

- Throwaway item?
  - Need to assume that more spare assemblies will be purchased
- Fixed Item?
  - May need different repair parts to fix the assemblies
  - Also need to consider what test capabilities are needed where
    - How many repair centers?
- Whether to allow for repair or broken components and location(s) for doing the repair are usually based on a life cycle cost estimate
  - What is the cheapest approach?

Transition from Development to Production

- **Continuity of operations is critical during transition to Production**
  - **System production plan is the "blueprint" for the transition**
  - **Engineering and manufacturing facilities are usually separate and independent**
  - **Responsibilities transferred from engineering personnel to manufacturing specialists**
  - **Configuration management process must keep step with engineering change process**

- **Systems engineers play a key role in facilitating a smooth transition**
  - **Facilitate communication between engineering and production personnel**
  - **Ensure the integrity of the system design**
  - **Resolve unexpected problems with component incompatibilities or components using advanced technologies**

Other Tasks for the Systems Engineer in Production

- Estimate risk
- Provide inputs to estimate the needed schedule
- Provide inputs to estimate the cost
- Address user feedback
- Recommend who is needed to support the project
- Maintain good system engineering processes
  - i.e., Configuration management, etc.
- Recommend course of action for technical issues

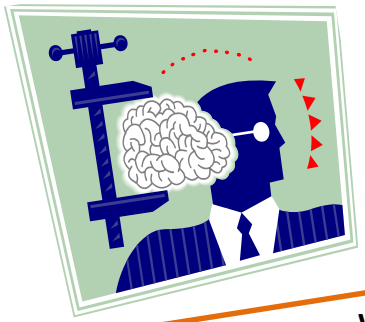
When does systems engineering make their most important contribution to this stage?

System Engineering Products in Production

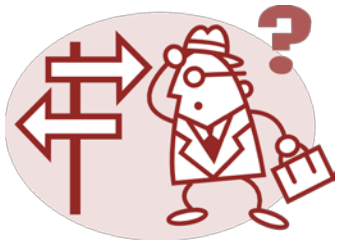
- **Recommendations for technical approaches to resolve required changes**
  - **Solution approach**
  - **Regression test plans**
- **Information for the program manager**
  - **Candidate actions/plans**
  - **Schedule**
  - **Risk**
  - **Cost**
- **Recommendations for product improvements**

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



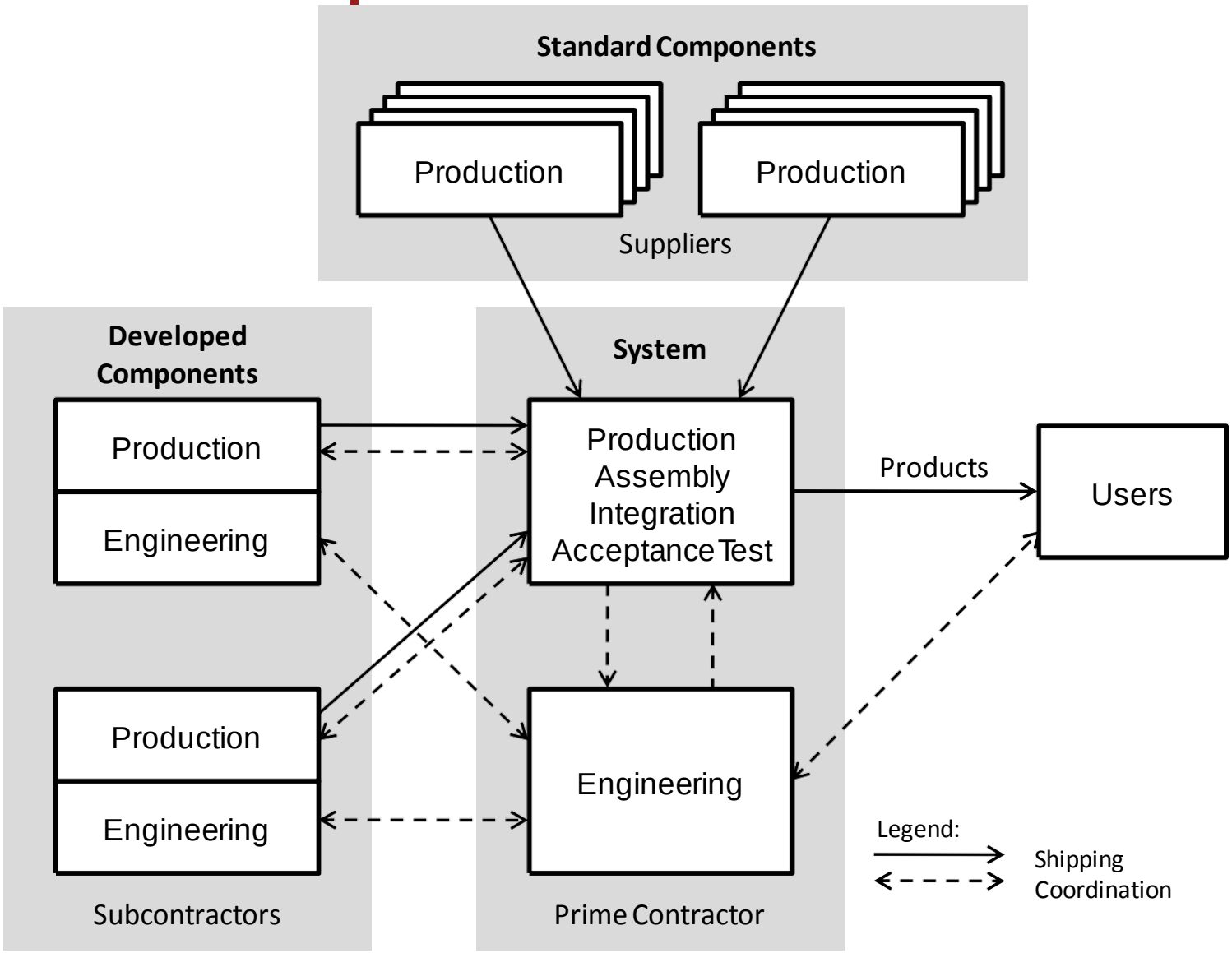
What are your Questions before We Continue?

Back-up Slides

Production Operations

- **System Production Plan – the 'blueprint" for system production and operations … key elements include:**
  - **Delivery schedule for each major component/sub-assembly**
  - **Engineering release schedule**
  - **Fabrication requirements**
  - **Tooling requirements**
  - **Test equipment/requirements**
  - **Quality control processes**
  - **Assembly**
  - **Acceptance testing**
  - **Packaging and shipping**
  - **Production readiness reviews**
  - **Overall schedule and cost estimates**

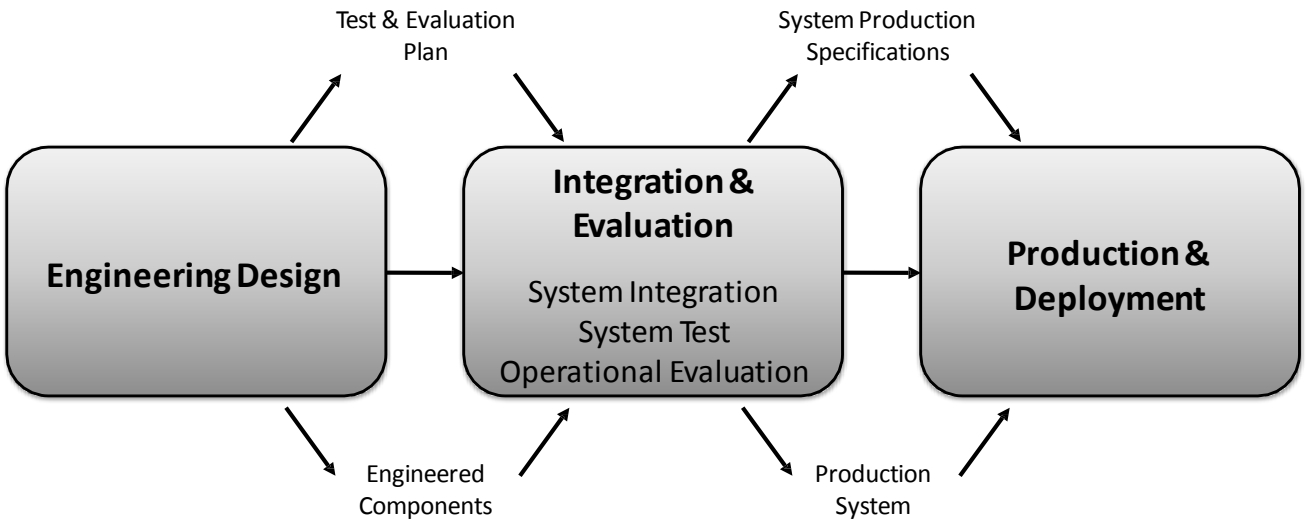
Production Operations



Production Knowledge Base

- **Acquiring knowledge of the suitability of specific components to meet the requirements of specific system applications can be achieved effectively by focusing on**
  - **Components that use advanced technologies, such as**
    - **Electronics Components**
    - **Electro-optical Components**
    - **Electro-mechanical Components**
    - **Mechanical Components**
    - **Thermo-mechanical Components**
  - **Recently developed or changing production processes**
  - **Risk areas that have been previously identified**
    - **Software Components**

Systems Engineering Technical Short Course (Session 1)



Lesson 12: Integration and Evaluation Stage

Reference : "Systems Engineering, Principles and Practice," Kossiakoff et. al – Chapter 13

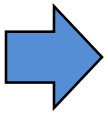
Session 1 Agenda (Jan 27 – 29): Timing

Day	Overview and Concepts of Systems Engineering	Hours*
1	Waters Management kick-off	1/2
	Introduction and overview of entire program (Lesson 1), including Introduction and overview of the projects (15 minutes each)	2
	Presentation and discussion of Concepts of SE (Lessons 2 – 9)	4
	Project assignments, team formations, and working sessions to apply SE concepts to the projects	1
	Day 1 wrap-up: Takeaways and Q&A	1/2
2	Review of day 1 working sessions	1
	Concepts of Systems Engineering (concluded) (Lesson 10 – 14)	4
	Project team working sessions (continued)	2
	Day 2 wrap up and Q&A	1
3	Team working session feedback to class (informal presentations)	1
	Project team working sessions (concluded)	5
	Check-in with class on application of SE concepts to their projects	1
	Session 1 wrap up: Q&A and homework due next session	1

* Times include breaks, and may be changed based on needs of the class

Concepts of Systems Engineering

1. Program Overview
2. What is Systems Engineering?
3. Structure and Environment of Systems
4. Systems Development Process
5. Systems Engineering Processes Overview
6. Systems Engineering in Needs Assessment
7. Risk Management
8. Concept Exploration
9. Concept Definition
10. Advanced Development Stage
11. Engineering Design Stage
12. Integration and Evaluation Stage
13. Production and Deployment
14. Operations, Support, and Disposal

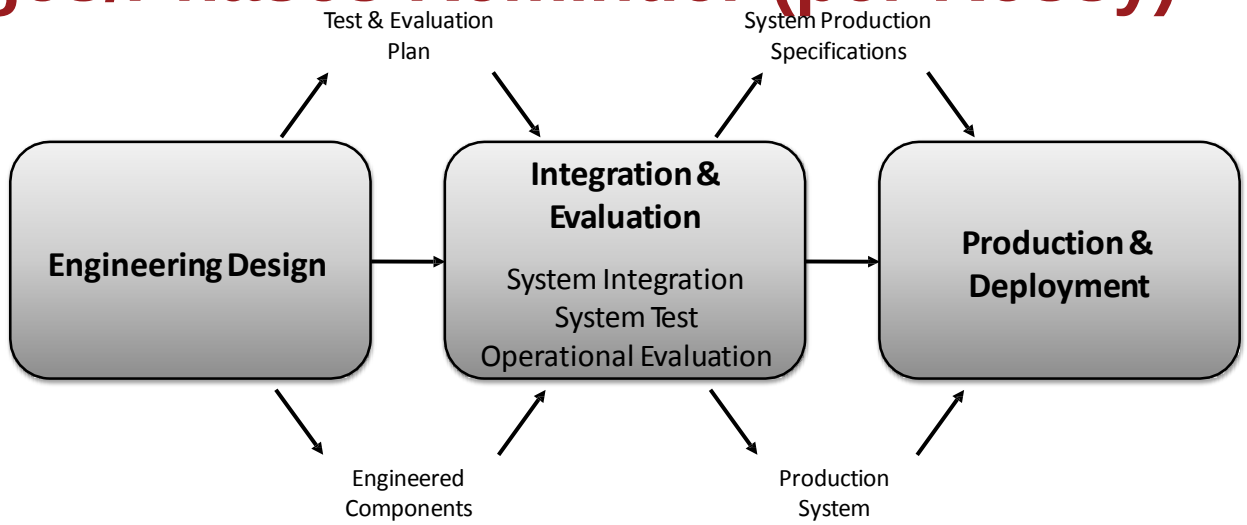


Lesson Content:

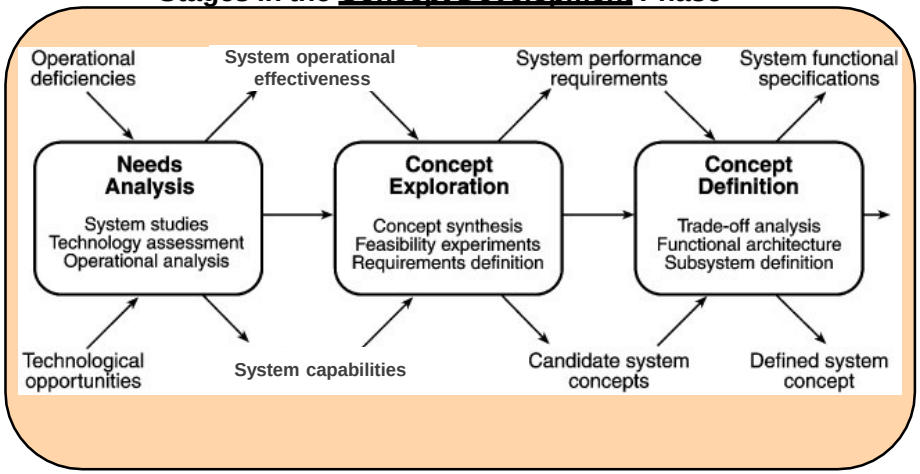
- Integration and evaluation stage
- Process activities
- TEMP
- Development testing
- Operational testing
- Qualification methods

Plus Appendices

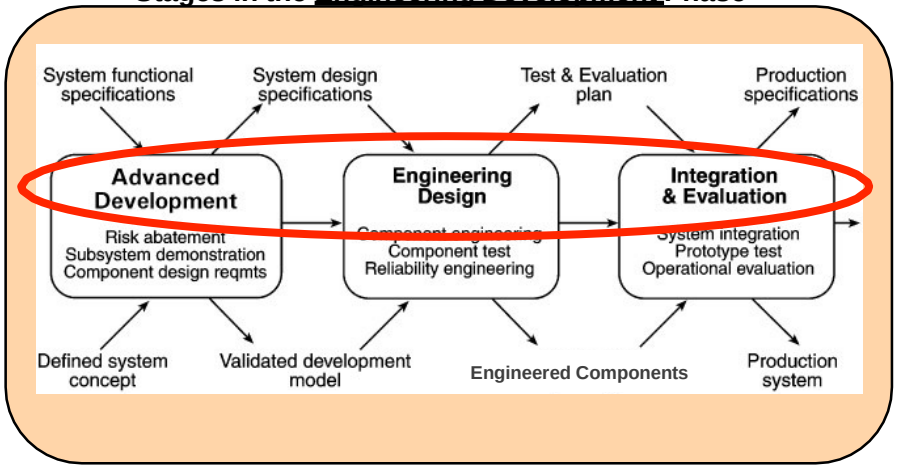
Stages/Phases Reminder (per Kossy)



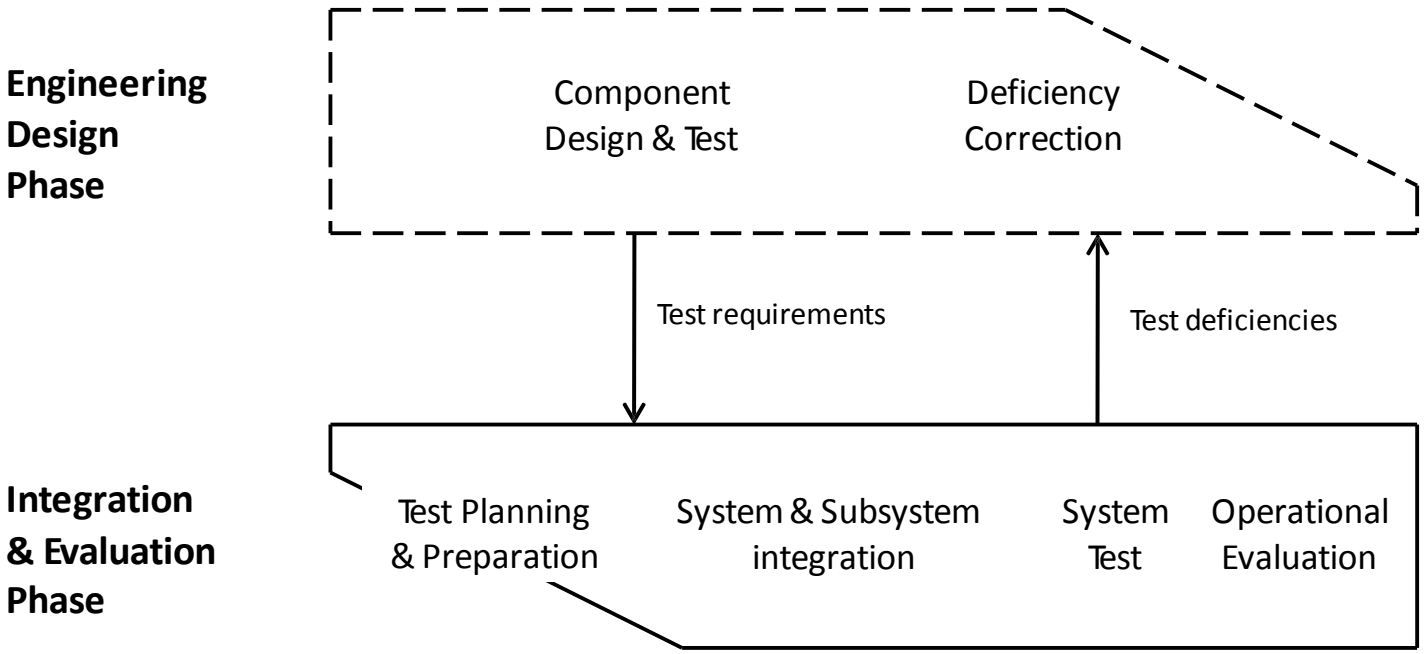
Stages in the Concept Development Phase



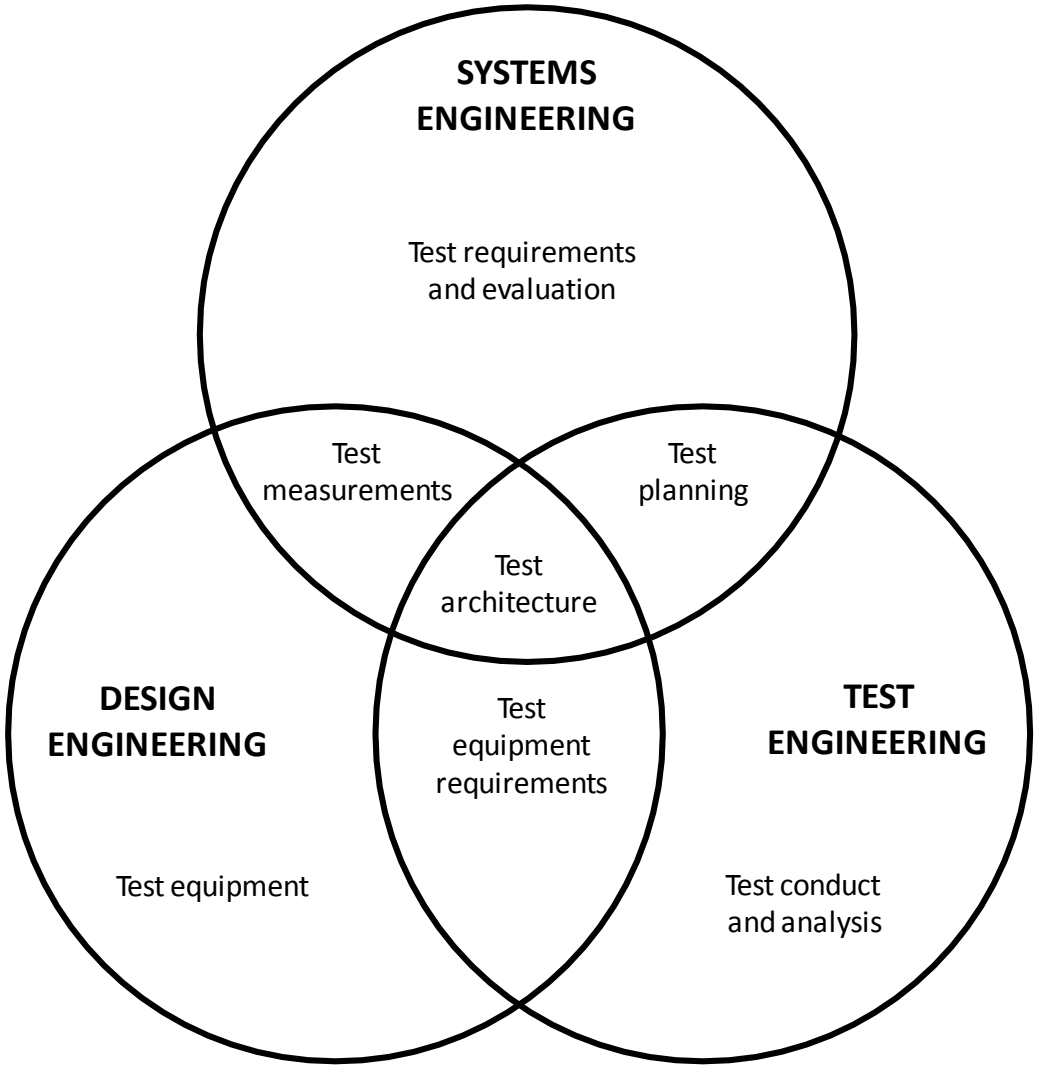
Stages in the Engineering Development Phase



Integration, Test, and Evaluation Phase Relative to Engineering Design Phase



System Test and Evaluation Team



Status of System Materialization at the Integration and Evaluation Phase

Level \ Phase	*Concept Development*			*Engineering Development*		
	Needs Analysis	Concept Exploration	Concept Definition	Advanced Development	Engineering Design	Integration & Evaluation
System	Define system capabilities & effectiveness	Identify, explore and synthesize concepts	Define selected concept with specifications	Validate Concept		**Test & Evaluate**
Subsystem		Define requirements & ensure feasibility	Define functional & physical architecture	Validate subsystems		**Integrate & test**
Component			Allocate functions to components	Define specifications	Design & test	**Integrate & test**
Sub-Component	*Visualize*			Allocate functions to subcomponents	Design	
Part					Make or buy	

System Integration, Test, and Evaluation Processes

Integration Level	Environment	Objective	Process
System	Real operational environment	Demonstrated operational performance	Operational test and evaluation
System	Simulated operational environment	Demonstrated compliance with all requirements	Developmental test and evaluation
System	Integration facility	Fully integrated system	System integration and test
Subsystem	Integration facility	Fully integrated subsystems	Subsystem integration and test
Component	Component test equipment	Verified component performance	Component test

System Integration, Test, and Evaluation Processes – Typical Activities

- **Test Planning and Preparation**
  - **Review system requirements**
  - **Define detailed plans for integration and system test**
  - **Define test requirements and functional architecture**
- **System Integration**
  - **Integrate tested components into subsystems**
  - **Aggregate subsystems into operational systems**
- **Developmental System Testing**
  - **Perform system-level tests over entire operating regime**
  - **Develop test scenarios which exercise all system operating modes**
  - **Identify and eliminate all performance deficiencies**
- **Operational Test, and Evaluation**
  - **Perform system performance tests in realistic operational environment using independent test agents**
  - **Evaluate system readiness for full production and operational deployment**

System Test and Evaluation Master Plan

*(TEMP)**

Major Components

- **System Introduction**
  - ***Describes the system, its mission, operational environment and measures of effectiveness (MOEs)***
- **Integrated Test Program Summary**
  - ***Lists the test program schedule and participating organizations***
- **Developmental Test, and Evaluation**
  - ***Describes the developmental test objectives, methods of approach and principal events***
- **Operational Test, and Evaluation**
  - ***Describes the operational test objectives, configurations, events and scenarios***
- **Test, and Evaluation Resource Summary**
  - ***Lists the test articles, sites, instrumentation and support operations***

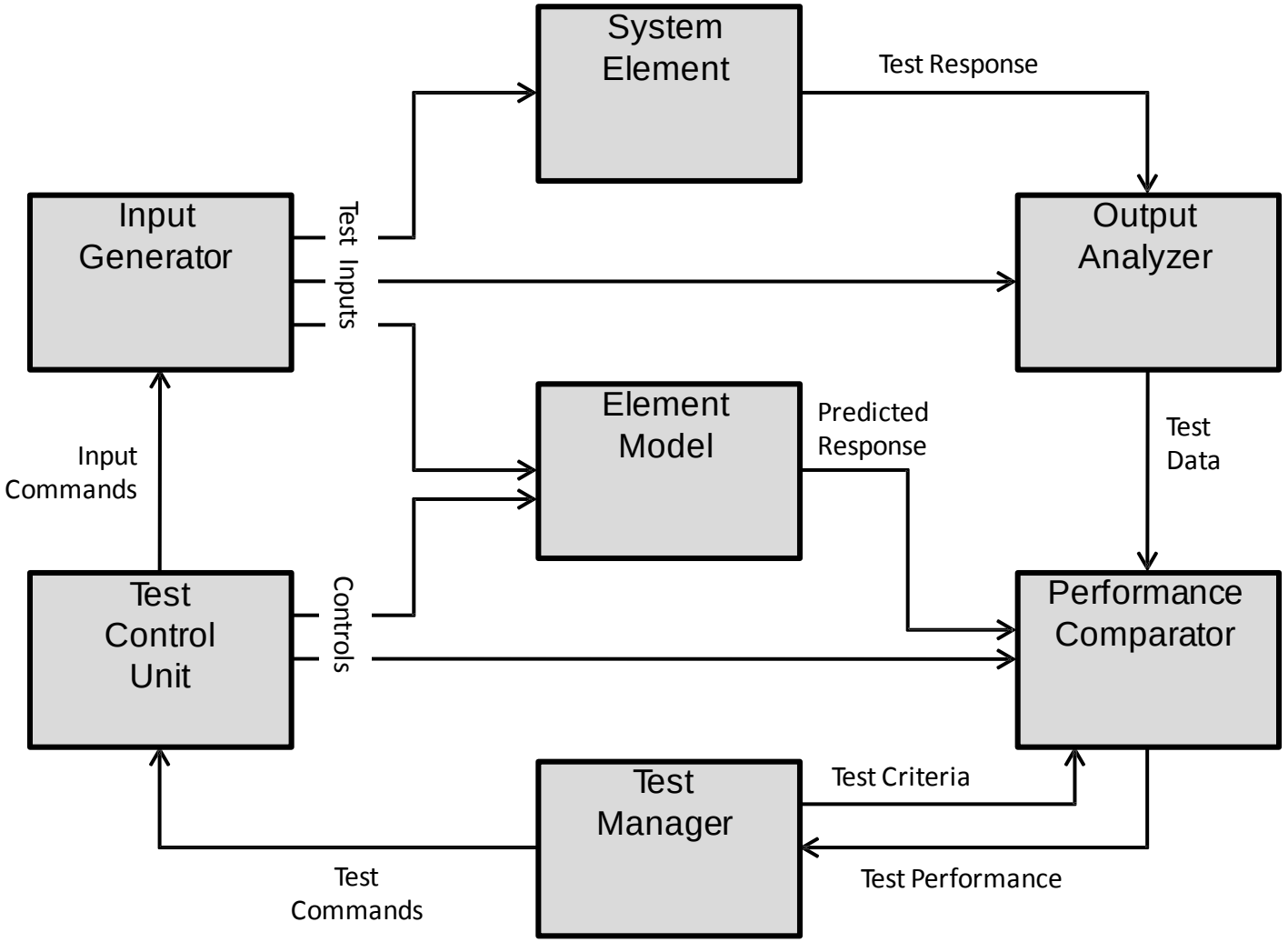
* See Chapter 10 in text for more details.

System Test and Evaluation versus System Development

System Development	Test and Evaluation Planning
NEED: Define the capability to be fielded	OBJECTIVE: Determine the degree of sophistication required of the test program
SYSTEM CONCEPT: Analyze trade-offs between performance, schedule and cost to develop a system concept	TEST CONCEPT: Evaluate trade-offs between test approaches, schedule and cost to develop a test concept
FUNCTIONAL DESIGN: Translate functional requirements into two level specifications for the (sub)system(s)	TEST PLAN: Translate test requirements into a description of each test event and the resources required
DETAILED DESIGN: Design the various components which comprise the system	TEST PROCEDURES: Develop detailed test procedures and test tools for each event

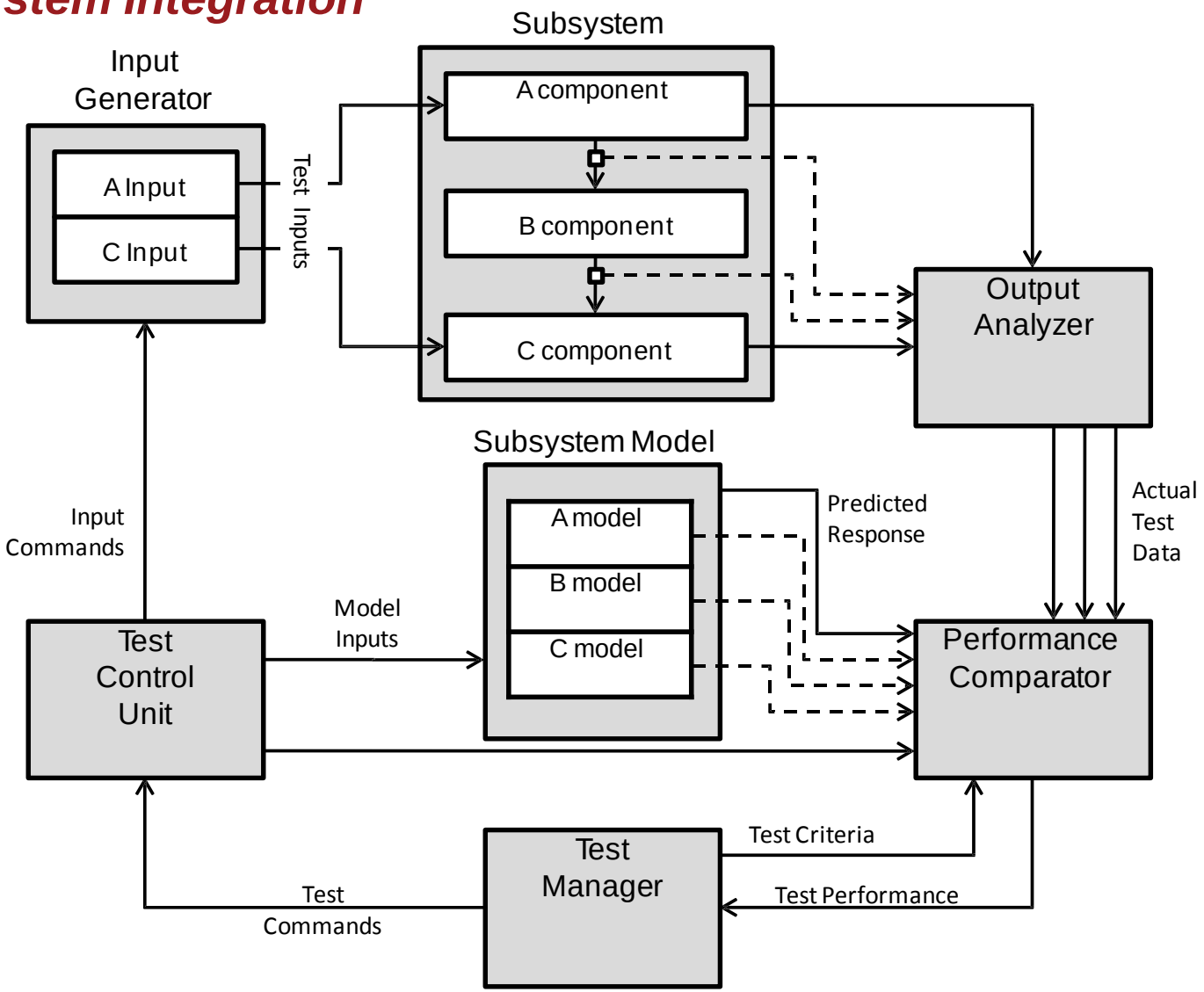
System Integration:

Physical Test Configuration



System Integration:

Subsystem Integration



System Integration:

Total System Integration

- **Uses same principles of integration as those for subsystem integration**
- **Significant increase in scale, complexity and diagnosis of faults**
- **System Integration Test Facility**
  - **Gradually built during system development as a total system "test bed"**
  - **May utilize subsystem test facilities**
  - **Must support extraction of test data from internal subsystem boundaries as well as system outputs**
  - **Must be flexible to accommodate system updates**

Developmental System Testing:

*Major Activities**

- **Developmental Test Planning**
  - **Define specific parameters to be measured**
  - **Summarize test events, scenarios and test design concept**
  - **List all models and simulations to be used**
  - **Describe how the system environment will be represented**
- **System Test Configuration**
- **Test Scenarios**
- **Test Analysis and Evaluation**

* Based, in part, on TEMP (Chapter 8).

Developmental System Testing:

*Major Activities**

- Developmental Test Planning

- **System Test Configuration**
  - **Must be designed to subject the system to all operational system inputs and environments**
  - **Measure all significant responses and operating functions required of the system using system outputs and internal test points**
  - **Ensure system robustness under extreme operating conditions and test for "graceful" degradation, if required.**

- Test Scenarios

- Test Analysis and Evaluation

* Based, in part, on TEMP (Chapter 8).

Developmental System Testing:

*Major Activities**

- Developmental Test Planning
- System Test Configuration
- **Test Scenarios**
  - **Define a structured series of tests which exercise the system over its entire operating regime**
  - **Exercise all system operating modes and stress all critical system functions**
  - **Ensure that the observed system performance meets all requirements**
    - **End-to-end system requirements**
    - **Subsystem and/or component requirements**
- Test Analysis and Evaluation

* Based, in part, on TEMP (Chapter 8).

Developmental System Testing:

*Major Activities**

- Developmental Test Planning
- System Test Configuration
- Test Scenarios
- **Test Analysis and Evaluation**
  - **Must resolve deviations from expected system operation and/or performance**
  - **Sources of deviations include faults in**
    - **Test equipment**
    - **Test procedures**
    - **Test execution**
    - **Test analysis**
    - **System under test**
    - **Excessive performance requirement**
  - **Judicious placement of internal test points within the system can rapidly narrow the search for the cause of the deviation**

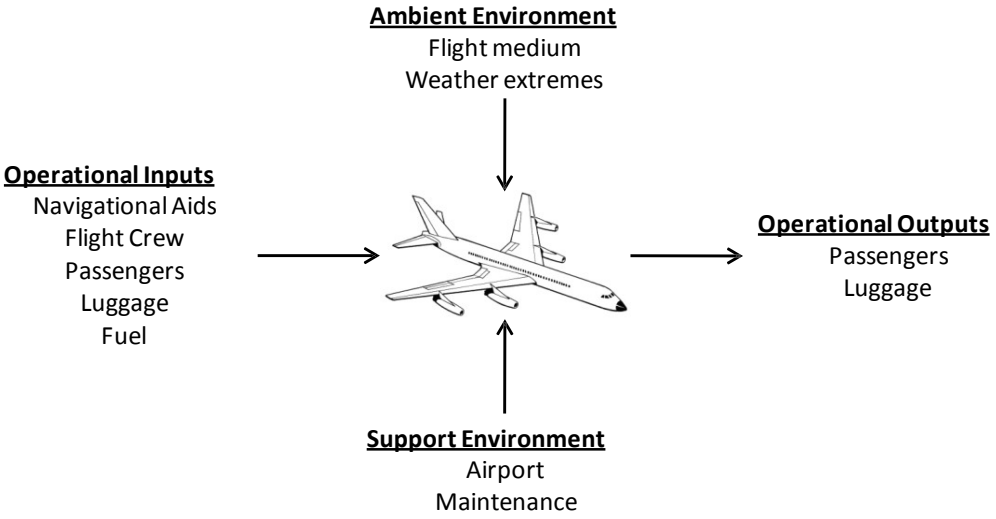
* Based, in part, on TEMP (Chapter 8).

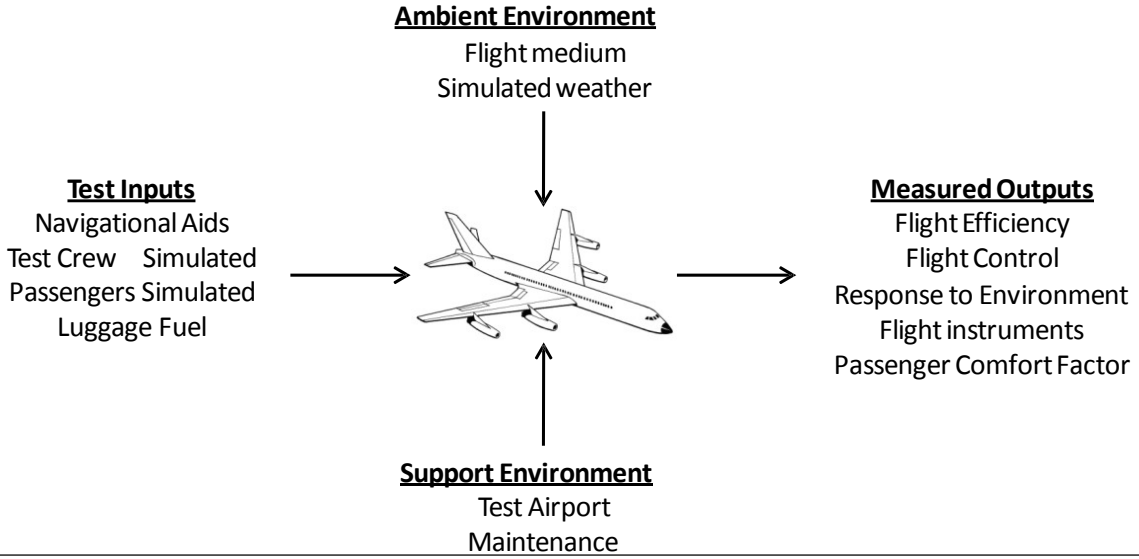
Operational Test and Evaluation

- **High Priority Areas for Operational Testing:**
  - **New Features**
    - **Fixes for previous system deficiencies**
    - **Subsystem upgrades**
    - **New capabilities**
  - **Environmental Susceptibility**
    - **Severe Operational Environments**
  - **Interoperability**
    - **Compatibility with external equipment/interfaces**
    - **Non-standard communication protocols**
  - **User Interfaces**
    - **Effectiveness of human-system interfaces**
    - **Training requirements and aids**

Operational Test and Evaluation:

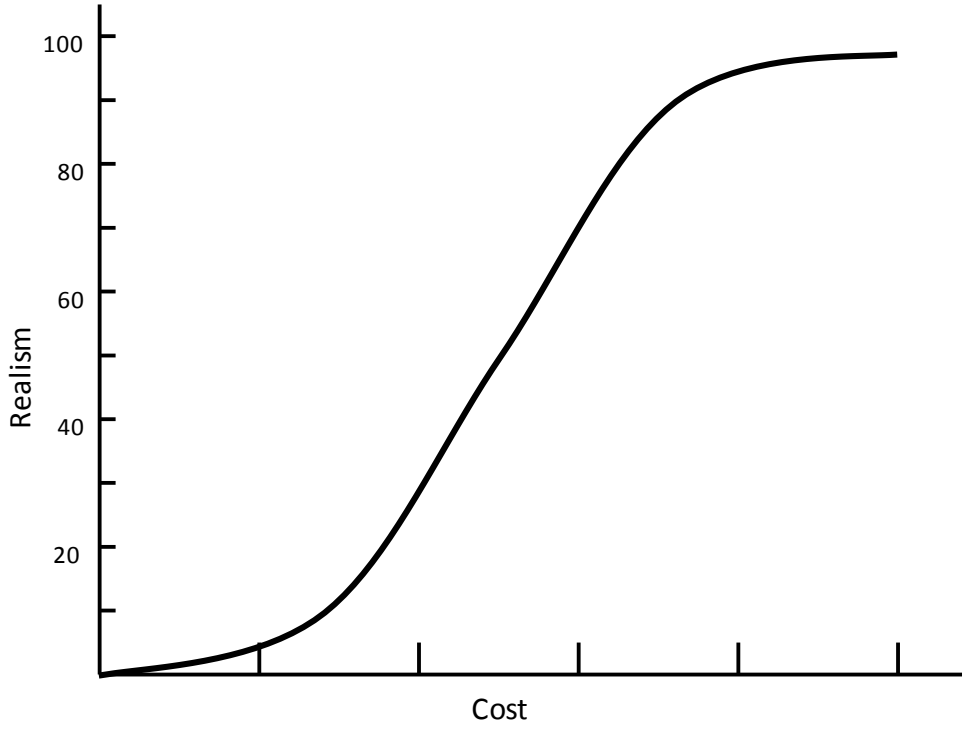
An Example: Passenger Airliner





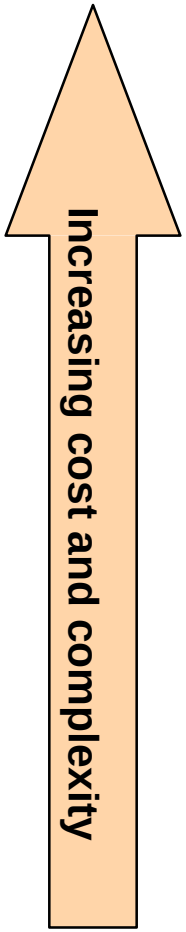
Operational Test and Evaluation

(Notional)



Qualification Methods

Qualification method	Definition
Test	The operation of the system, or a part of the system, using instrumentation or other special test equipment to collect data for later analysis.
Demonstration	The operation of the system, or a part of the system that relies on observable functional operation not requiring the use of instrumentation, special test equipment, or subsequent analysis.
Analysis	The processing of accumulated data obtained from other qualification methods or mathematical modeling of the system using design and reference data. Examples are data reduction, interpolation, extrapolation of test results, reliability assessments based on MIL-HBDK-217E, and finite point stress modeling.
Inspection	The visual examination of system components, drawings, documentation, etc.
Special qualification methods	Any special qualification methods for the system, such as special tools, techniques, procedures, facilities, acceptance limits, use of standard samples, preproduction or periodic production samples, pilot models, or pilot lots.



Source: DI-IPSC-8143A. Data Item Description. System/Subsystem Specification (SSS). Approval Date: 2000-01-10

Operational Test and Evaluation:

Test Reports

- **Quick-Look Reports**

  Provide preliminary test results immediately following a significant test event

- **Status Reports**

  Provide periodic (monthly) reports of specific test events

- **Final Evaluation Reports**

  Provide detailed

  - **Test findings**
  - **Evaluations of the findings relative to the system requirements and specifications**
  - **Recommendations regarding the system's operational capabilities and readiness for deployment**
  - **Recommendations for addressing system deficiencies identified during testing**

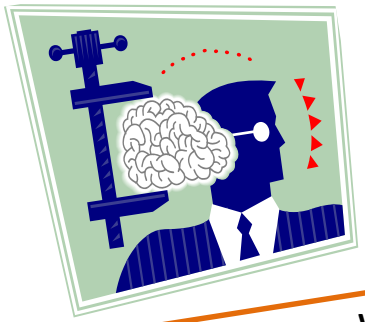
Exercise: Who Does the Testing for Your Projects?



Instructor will provide directions for the exercise

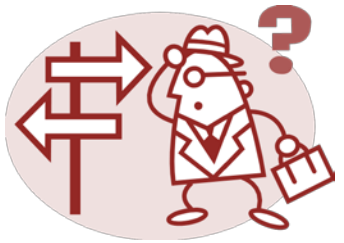
Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future

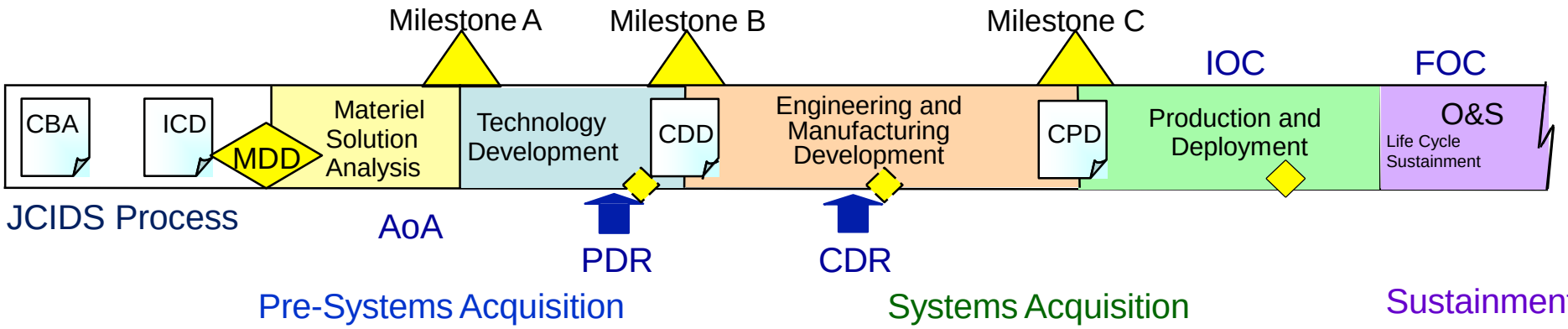
What are your Questions before We Continue?



Back-up Slides

Engineering and Manufacturing Development Phase, Part 2–DoD View (2008)

- **System Capability and Manufacturing Process Demonstration**
  - **Completed Integrated System Design**
  - **Review the design with the Government at the Critical Design Review (CDR)**
  - **Build the system; Reduce manufacturing risk; support Milestone C Production decision**
  - ***Perform Integration and Test on the system***
  - ***Perform Initial Operational Test, and Evaluation (IOT&E) on the system***
- **Prove the system meets its requirements and will fulfill the user's operational need**
  - **Needed to support the Milestone C decision to go to production and deployment**



Systems Engineering Technical Short Course

(Session 1)

Lesson 14:
Operations and Support,
and Disposal

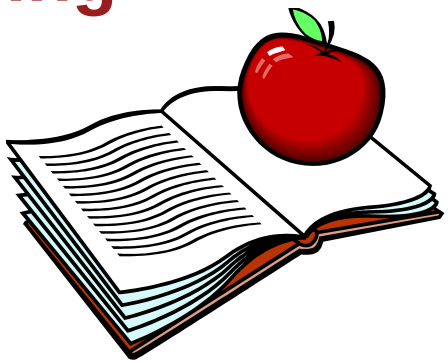
Reference : "Systems Engineering, Principles and Practice," Kossiakoff et. al – Chapter 15

Session 1 Agenda (Jan 27 – 29): Timing

Day	Overview and Concepts of Systems Engineering	Hours*
1	Waters Management kick-off	1/2
	Introduction and overview of entire program (Lesson 1), including Introduction and overview of the projects (15 minutes each)	2
	Presentation and discussion of Concepts of SE (Lessons 2 – 9)	4
	Project assignments, team formations, and working sessions to apply SE concepts to the projects	1
	Day 1 wrap-up: Takeaways and Q&A	1/2
2	Review of day 1 working sessions	1
	Concepts of Systems Engineering (concluded) (Lesson 10 – 14)	4
	Project team working sessions (continued)	2
	Day 2 wrap up and Q&A	1
3	Team working session feedback to class (informal presentations)	1
	Project team working sessions (concluded)	5
	Check-in with class on application of SE concepts to their projects	1
	Session 1 wrap up: Q&A and homework due next session	1

* Times include breaks, and may be changed based on needs of the class

Concepts of Systems Engineering



1. Program Overview
2. What is Systems Engineering?
3. Structure and Environment of Systems
4. Systems Development Process
5. Systems Engineering Processes Overview
6. Systems Engineering in Needs Assessment
7. Risk Management
8. Concept Exploration
9. Concept Definition
10. Advanced Development Stage
11. Engineering Design Stage
12. Integration and Evaluation Stage
13. Production and Deployment
14. Operations, Support, and Disposal

Plus Appendices

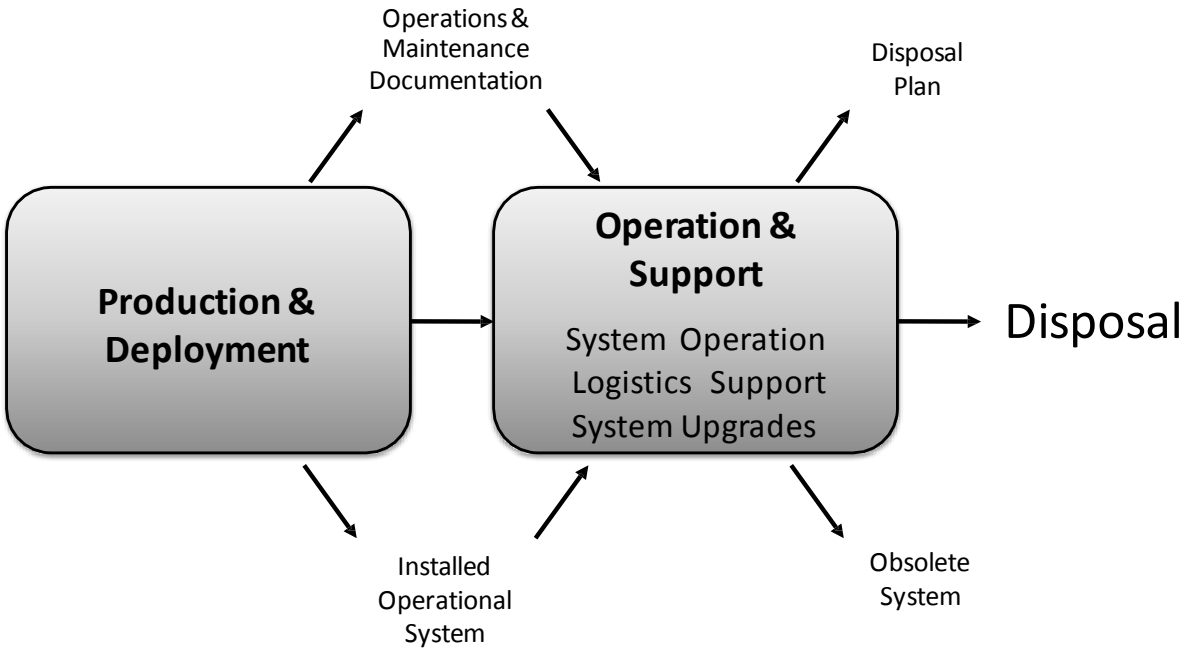
Lesson Content:

- Importance of earlier phases to this phase
- SE in the phase
- Systems installation and test
- Engineering for operations and maintenance
- Disposal considerations

Exercise: Disposal concerns for Waters projects

Appendix C: COTS

Operations and Support in the System Life Cycle – The Users Get Their Product

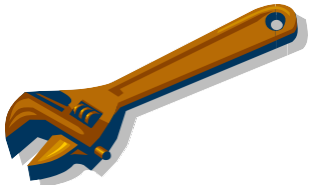


….where the rubber meets the road

Operations and Support
The Engineering is All Done, Right?

- **What about:**
  - **…Fielding the System?**
    - **Unleashing your system on an unsuspecting user**
  - **…Maintaining the product?**
    - **Fixing the system in the field**
  - **…Sustaining the product?**
    - **Keeping a supply of replacement components**
  - **…Upgrading the product?**
    - **Providing enhancements to keep the system useful**









Operations and Support Success Reflects Earlier Phase Systems Engineering Efforts

3. When do we define what we build, test, and deploy?

2. When do we build and test what we deploy?

1. What do we do here?

Produce, transition, operate, maintain, support, secure, ……. , dispose

DEFINE	BUILD/TEST	DEPLOY
Concept Development Phase Needs analysis, concept exploration, concept definition, Requirements engineering, and Architecture	Engineering Design Phase Design and development, System integration, Test and evaluation	Post-development Phase System implementation, Operations & Maintenance, and Transition

4. When do we:
- spend the most money and time?
- have the most leverage?

5. What is the focus of:
- Your organization's projects?
- Your current project(s)?

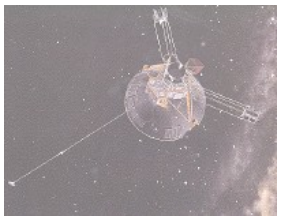
What does this suggest about our systems engineering efforts?

Complex Systems Can Be In Use for a Long Time

- Autos
  - Median age on the road is 8.9 years (2005 data)



- Pioneer 10
  - Launched 1972
  - Last successful data transfer - 2002



- B-52 Stratofortress
  - The youngest aircraft was delivered in 1962

  - still in service

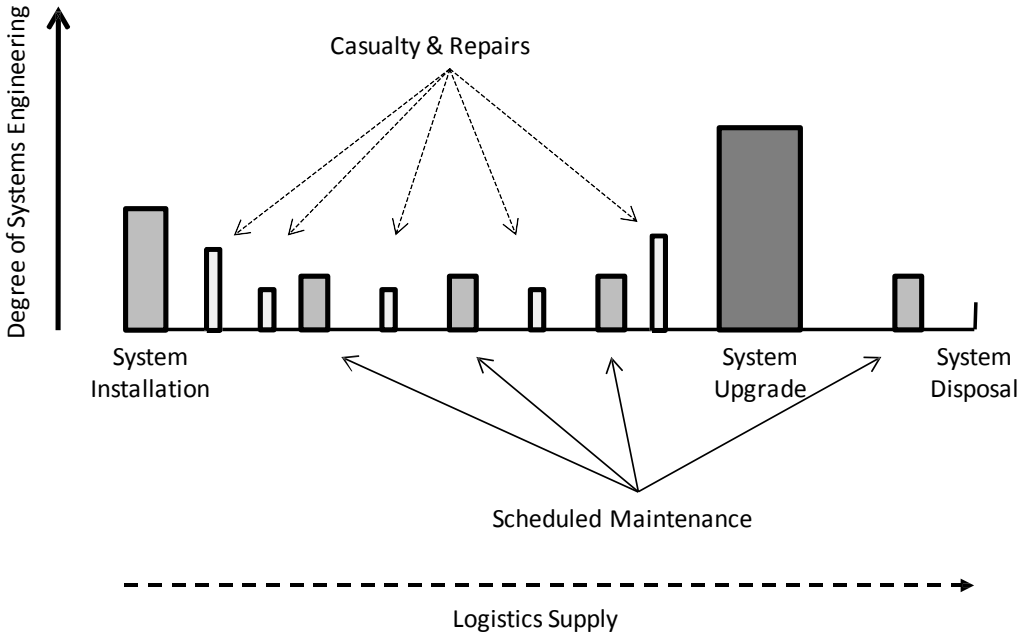


- Software systems: Microsoft Windows XP
  - Introduced in 2001
  - Some embedded applications still based on DOS (1981



Systems Engineering in Operations and Support

- For routine system deliveries, operations and support
  - Often less of a direct system engineer role
  - Occasional routine efforts
- Higher level of effort while initially fielding



Typical SE Tasks

- Update risk assessment
- Address latent design defects or installation/operation problems
- Make recommendations
  - How should a part that is no longer available be addressed?
  - Does system still meet the user need?

Fielding the System

- There are many elements to delivering the system and making it operational
  - Infrastructure at the site to support the actual delivered system
    - Buildings, space, power, air conditioning, cabling
  - Installation & test (to make sure it works)
  - Documentation
    - Not just design specs and drawings →→User & maintenance manuals
  - Training the personnel who will use it
  - Coordinating the integration of the new system with other already existing systems
    - Sudden switchover
    - Phase in of capabilities
- These can all affect the requirements and implementation of what is delivered

The Systems Engineer should ensure that there is a plan

Delivery Considerations

- Complete delivered end item product
- New system "Kit" includes everything new
- System developer install
- Assembled on-site
- "New" system is a modification to an existing item
- User install

Many large-scale real-world systems are a combination of the above

System Installation and Test

- **Difficulty affected by amount of physical and functional integration accomplished at the production facility**
  - **High degree of integration: Largely self contained at delivery**



Vehicle – Saturn Vue



Major Software Application – One set of install disks

  - **Low degree of integration at delivery: Assembled in the field**



AN/FPS-115 Pave Paws Radar

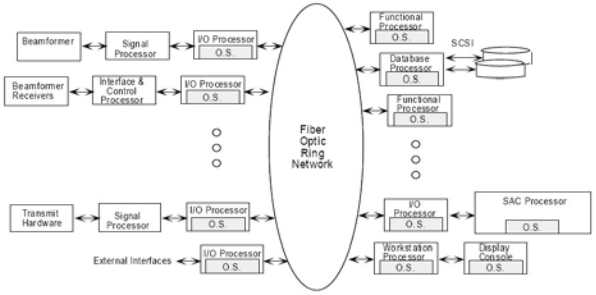


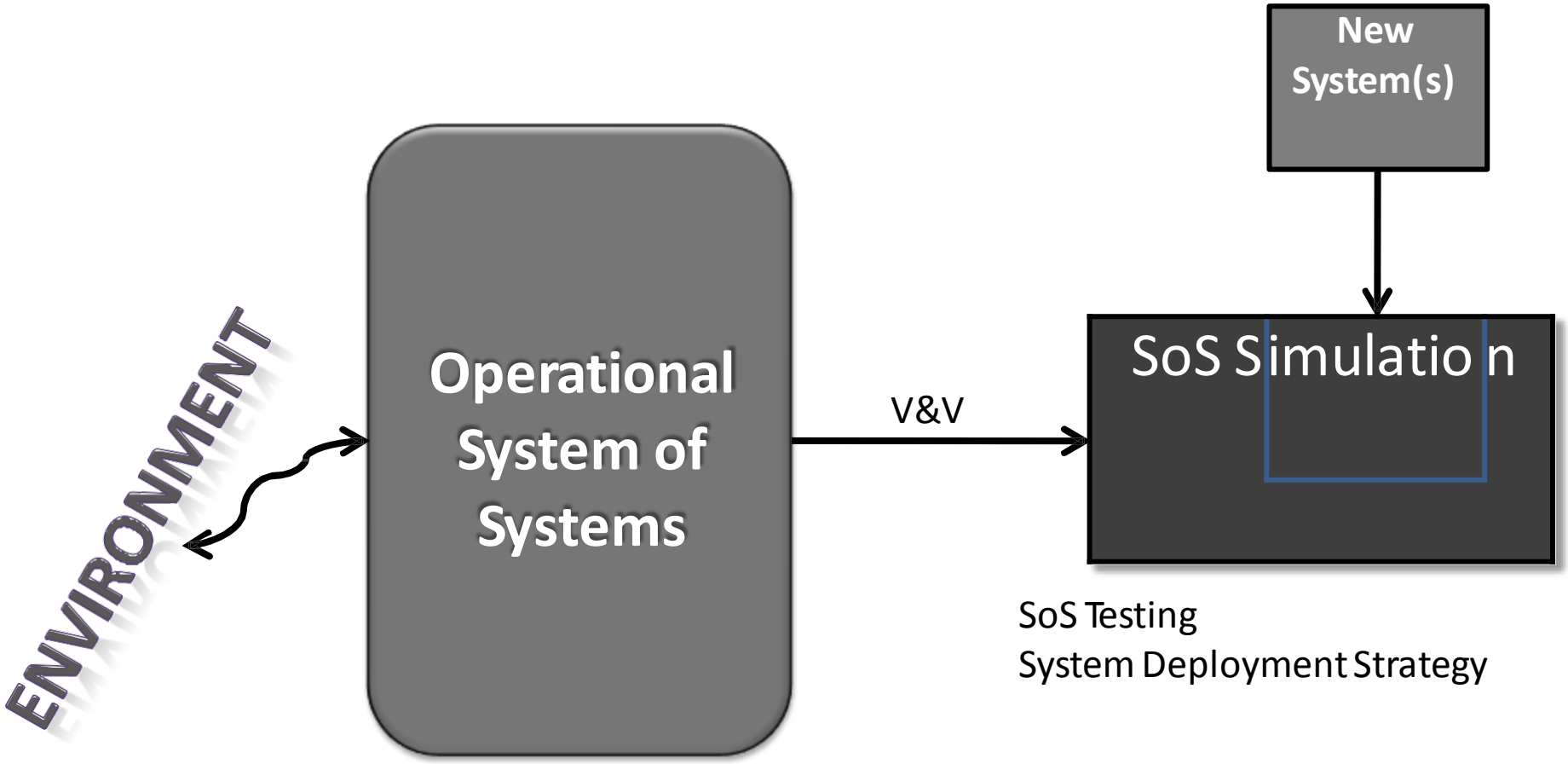
Figure 1. BSY-2 System Network Topology

BSY-2 Submarine Command & Control System Over 100 nodes

System Installation and Test

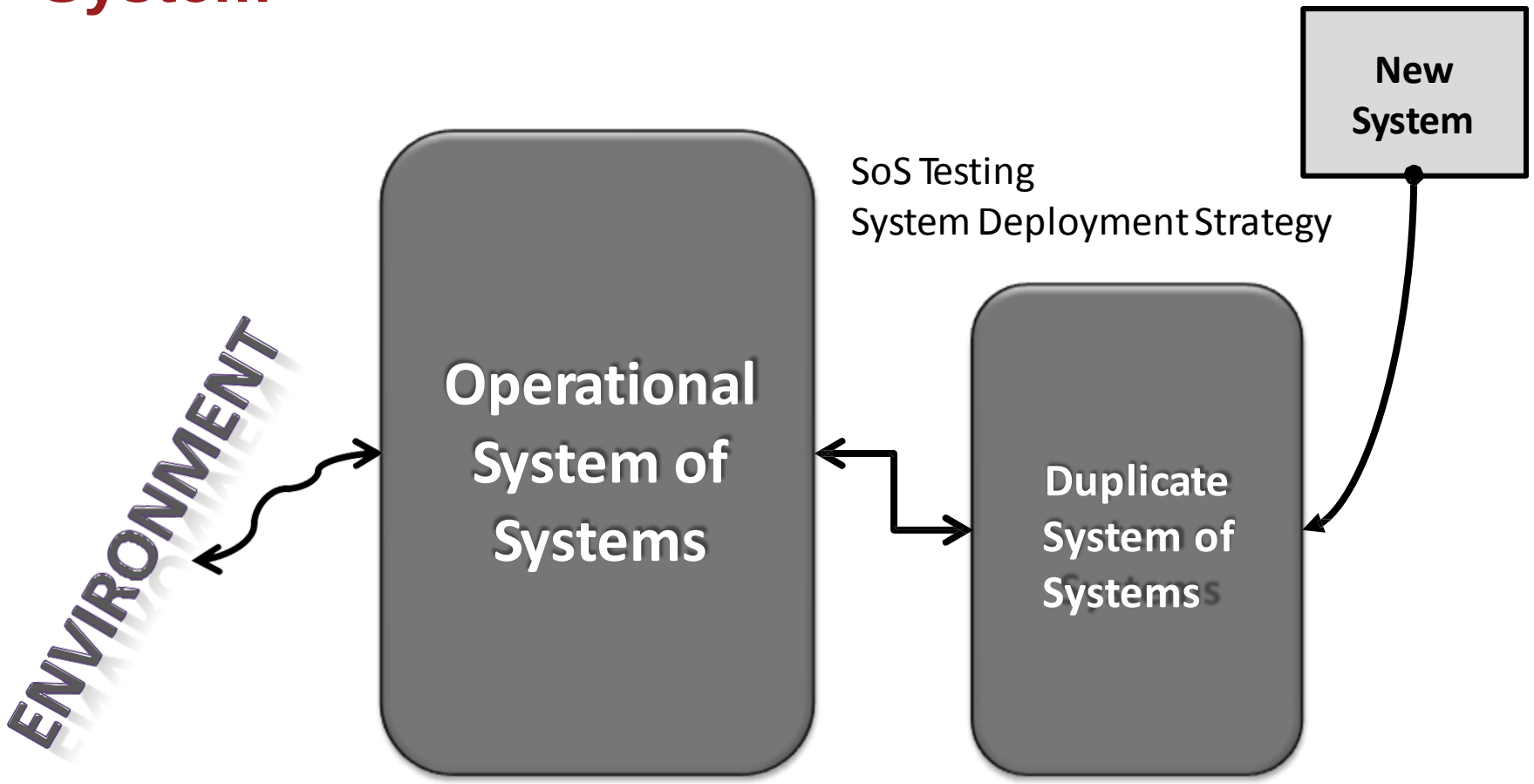
- Difficulty affected by the number and complexity of interfaces at the operating site
- Internal interfaces vs. external interfaces
  - External interfaces have dependencies outside the delivered system
    - Are the other ends of the interface ready?
  - Internal interfaces are (usually) tested in development
    - Need to check that installation is correct
- Installation Challenges
  - Physical interfaces
    - Mating holes/mounting can be wrong, cables can be built incorrectly, components are assembled incorrectly, etc.
  - Functional interfaces
    - Command or data transfers are corrupted in transfer and do not work, software versions do not match in the delivered components, environmental air conditioning can be underpowered, etc.

Non-disruptive Installation via Simulation



Non-disruptive Installation via Simulation

Non-disruptive Installation via Duplicate System



Non-disruptive Installation via Duplicate System

Logistics Support Considerations

- **Maintenance**
  - **By the manufacturer/vendor or the user?**
    - **Affects documentation that needs to be produced and delivered**
    - **Affects training to be developed – in-house or user self-train?**
  - **If user maintenance…**
    - **Generally more documentation needed for user self support**
    - **User may need special test equipment**
- **Spare Parts**
  - **Initially delivered with a system**
  - **Built and stored for future needs**
    - **Depends on product lifetime, time committed to support product**
- **Long-term plan for obsolescence**
  - **Reactive or "proactive"**

Engineering For Operation and Support

- **Modularity**
  - **Good selection for Subsystems/Components**
  - **Good selection for interfaces**
- **Many considerations**
  - **Number of systems to build**
  - **Accessibility for repair**
- **These need to be considered from the beginning of the project starting with the Needs Analysis!**

Support Equipment

- Support elements of the system can be quite substantial
- Example elements
  - Field maintenance and test equipment/test tools
  - Intermediate maintenance and test equipment/test tools
  - Facilities
    - Enclosures/buildings
    - Intermediate maintenance facility
- Examples
  - RF test equipment
  - Antenna test ranges
  - Docking port/dry dock/terminal dock/launch pad
    - Or are these subsystems of the operational system?

Ongoing Maintenance

- **Systems Engineer is not usually involved in ordinary maintenance**
  - **May be brought in to diagnose hard problems**
  - **Especially if problem is between components or systems**

- **However, maintenance is affected by the Systems Engineer's decisions in prior phases of the life cycle**
  - **Reliability requirements**
    - **Is it possible (or needed)?**
    - **Scheduled maintenance vs. fix on breakage**
    - **Mean time between system failures**
    - **Mean time between component failures**
  - **Maintainability requirements and features**
    - **How difficult is it to fix?**

Special Consideration: Commercial Off The Shelf Equipment (COTS)

- **COTS – Components or equipment that you purchase from a commercial company that is not custom designed for your application**
  - **Sold to the general public as "a catalog item"**
  - **Government has a long definition**

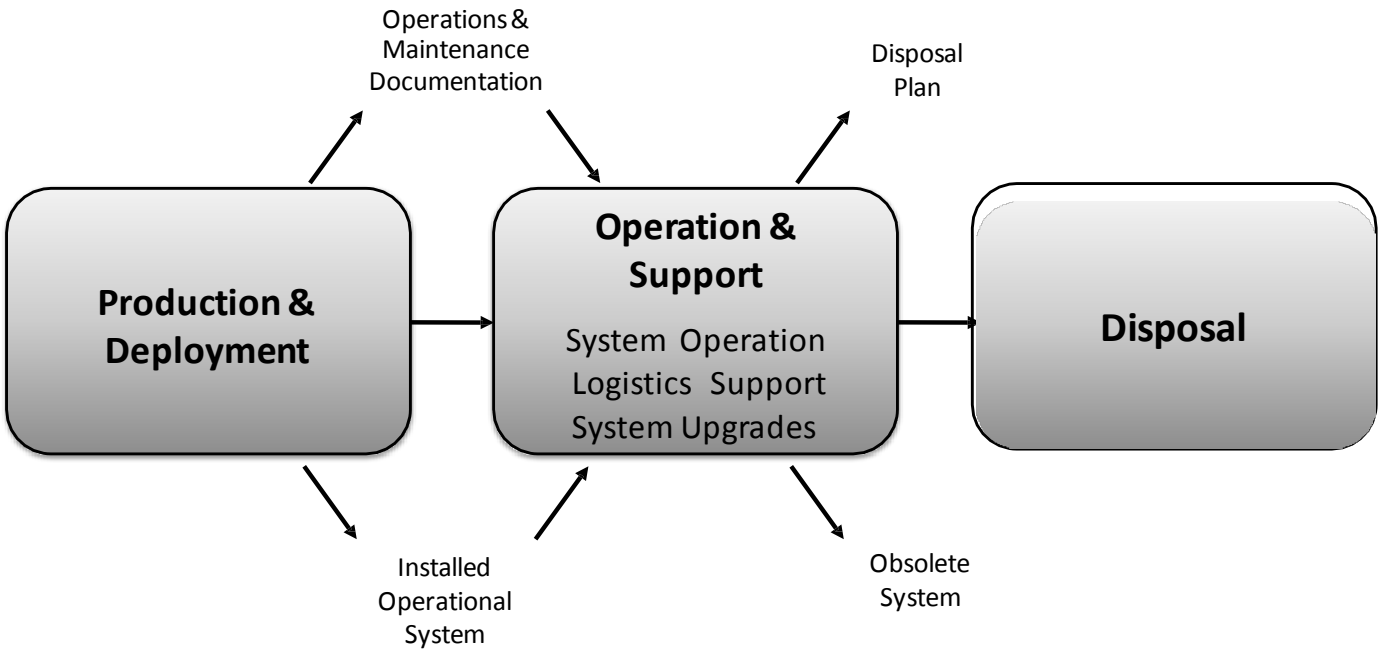
"The entire process of system development – requirements, design, implementation, testing, maintenance – will undergo many radical changes when a system incorporates multiple COTS components. By committing to just one part of the "new paradigm" (i.e., simply buying some commercial products), it is vital to realize that this will usually demand a commitment to all of it."

From "Quotations from Chairman David, A Little Red Book of Truths to Enlighten and Guide on the Long March Toward the COTS Revolution," Carnegie Mellon Software Engineering Institute, 1998

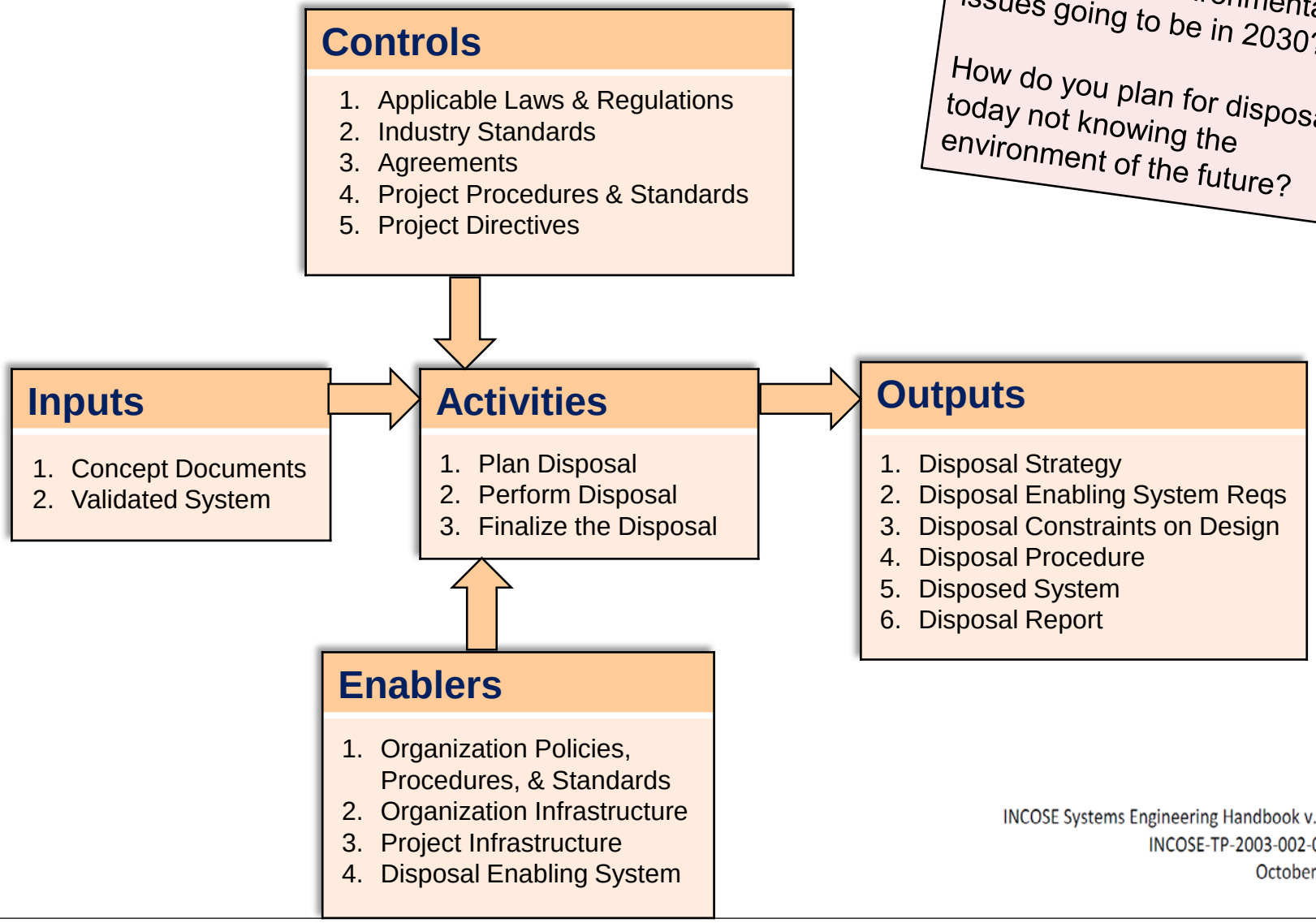
Upgrades

- Treat like new system development – Define the requirement
  - Replace a component for DMS
  - Add functionality or improve performance to meet the user need
- Biggest difference is the degree of emphasis on the existing design
  - Tendency is to constrain the concepts to provide a solution
    - Existing system external interfaces
    - Existing system decomposition into subsystem/components
- The systems engineer should consider all solutions
  - Fit solution into the existing system
  - Change other components to provide the right solution
  - Recommend an alternate solution:
    - Doctrine, Organization, Training, Materiel, Leadership and Education, Personnel and Facilities

Disposal in the System Life Cycle



DISPOSAL



What are the environmental issues going to be in 2030?

How do you plan for disposal today not knowing the environment of the future?

INCOSE Systems Engineering Handbook v. 3.2.2
INCOSE-TP-2003-002-03.2.2
October 2011

Disposal Strategies

- **Basic to Requirements & Solution Development**
- **Strategies for disposal are:**

1) **Recycle/reuse**
2) **Reprocess**
3) **Dispose**
4) **Wait and See**

Base disposal strategies on the best assumptions available

Summary (1/2)

- The systems engineer needs to consider the range of topics for making a system work in the field
  - Readying the support infrastructure
  - Delivery
  - Installation
  - Initial user training
  - Activating the system and beginning operations
  - Initial logistics support
  - Plan for long term maintenance and sustainment
  - Assigning responsibilities

- A fielding plan should be prepared
  - Often ends up being led by a logistics team
  - Systems Engineers should participate

Summary (2/2)

- For routine system deliveries, operations and support
  - Often less of a direct systems engineer role
  - Mainly addresses latent design defects or production problems
- Choices made while developing the system show their effect
  - Reliability
  - Maintainability
  - Choices for defining subsystems, components and interfaces
- Systems Engineers need to understand the operations and support portion of the life cycle to make good decisions in development
- Disposal considerations must be factored into systems engineering work in the earlier phases

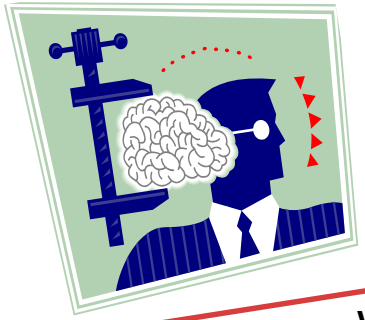
Exercise: Disposal Concerns for Waters Projects



Instructor will provide directions for the exercise

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?

Session 1 Agenda (Jan 27 – 29): Timing

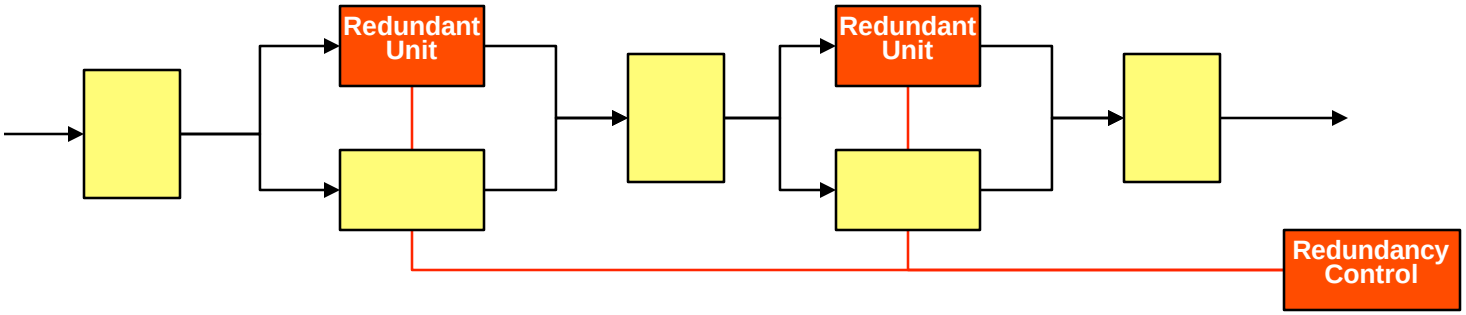
Day	Overview and Concepts of Systems Engineering	Hours*
1	Waters Management kick-off	1/4
	Introduction and overview of entire program (Lesson 1)	3/4
	Presentation and discussion of Concepts of SE (Lessons 2 – 9)	5
	Project assignments, team formations, and working sessions to apply SE concepts to the projects	1
	Day 1 wrap-up: Takeaways and Q&A	1
2	Review of day 1 working sessions	1
	Concepts of Systems Engineering (concluded) (Lesson 10 – 14)	3
	Project team working sessions (continued)	3
	Day 2 wrap up and Q&A	1
3	Team working session feedback to class (informal presentations)	1
	Project team working sessions (concluded)	5
	Check-in with class on application of SE concepts to their projects	1
	Session 1 wrap up: Q&A and homework due next session	1

* Times include breaks, and may be changed based on needs of the class

Back-up Slides

High Reliability May Add Cost

- **Redundancy**
  - **Provides additional spare components built into the system so it keeps operating even if a component fails**
  - **Adds more parts that fail, more complexity**



- **High Reliability Parts**
  - **Tighter tolerances – more difficult to manufacture**
  - **Burn in – run components before installing in the system to weed out early failures**

Considering Logistics: Spare Parts

- **Tradeoff:**
  - **Cost of purchasing/storing lots of spare components/parts**
  - **Cost and time to get broken components/parts repaired**
  - **Effects of system being unavailable if a spare part is unavailable**
- **Caution – Components/parts with long lead times**

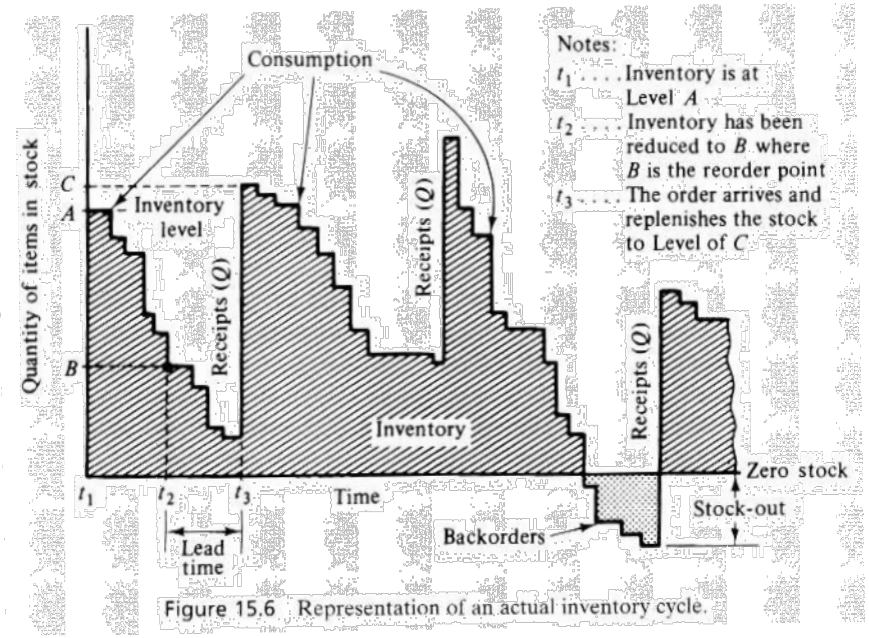


Figure 15.6 Representation of an actual inventory cycle.

- **System Requirements**
  - **System Reliability**
  - **System Availability**
  - **Storage/transportation of components**
- **Considerations/Constraints**
  - **Development Cost**
  - **Operations Cost**
  - **Repair facilities**

Diminishing Manufacturing Sources (DMS)

- When a part or component is no longer available to support the operational life of the system
  - Original vendor no longer makes the part or component
    - No business base
    - Obsolete technology
  - Source of repair parts has insufficient quantity
- Maintaining an active DMS monitoring program is important to prevent repair part shortages
  - Provides advance notice to allow time for a solution
- This is one place where good choice of components and interfaces pays off
  - Poor selection means that unusually large (expensive) components must be replaced or redesigned

Resolving DMS

- **Lifetime Buy**
  - **Ask the vendor for one last order**
  - **Buy enough parts and put them in a warehouse to last the expected lifetime of the system**
- **Qualify an Alternate Item**
  - **Find a substitute part – another vendor may make an identical or sufficiently close part**
  - **Develop a plan to test the substitute**
  - **Update specs, drawings, manuals as needed to allow the alternate part as a substitute**
- **Have a 3rd Party Build More Parts to the Original Drawings**
  - **Need to ensure that you have the right documentation**
  - **Need to ensure the newly produced item does what it is supposed to do**

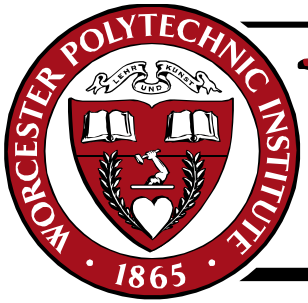
Resolving DMS (continued)

- **Design a New Part**
  - **Design and qualify a new part**
  - **Use the appropriate specification for a new design**
    - **Hopefully, you also have good interfaces to design the new part to interoperate with the rest of the parts correctly**

In all cases, the systems engineer should be involved with the solution

- **Assess risk**
- **Perform a trade study**
- **There are testing considerations**
- **It may be an opportunity to address system shortcomings**





WPI

Systems Engineering Technical Short Course

(Session 1)

Appendix C:
Lesson 14 Supplement:
COTS Considerations

Reference : "Systems Engineering, Principles and Practice," Kossiakoff et. al – Chapter 15

Special Consideration: Commercial Off The Shelf Equipment (COTS)

- **COTS – Components or equipment that you purchase from a commercial company that is not custom designed for your application**
  - **Sold to the general public as "a catalog item"**
  - **Government has a long definition**

"The entire process of system development – requirements, design, implementation, testing, maintenance – will undergo many radical changes when a system incorporates multiple COTS components. By committing to just one part of the "new paradigm" (i.e., simply buying some commercial products), it is vital to realize that this will usually demand a commitment to all of it."

From "Quotations from Chairman David, A Little Red Book of Truths to Enlighten and Guide on the Long March Toward the COTS Revolution," Carnegie Mellon Software Engineering Institute, 1998

COTS – The Good

- **Reduces development time**
  - **Functionality can be immediately purchased**
- **Lower initial cost**
  - **Get the advantage of quantity production**
  - **Someone else has already done the design and development**
- **Competitive environment**
  - **Stimulates improving performance**
  - **Contributes to lower initial purchase cost**

COTS – The Bad

- Vendor-unique product

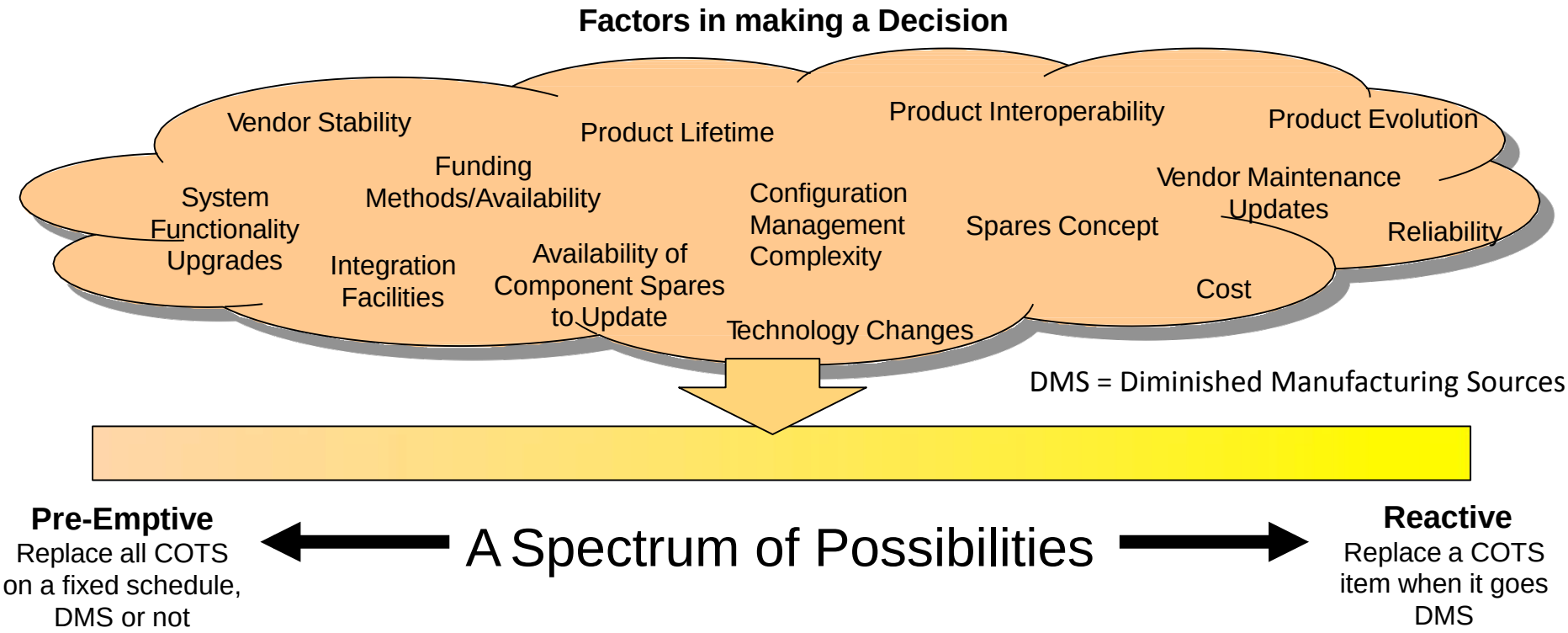
- Vendor can change their products at their whim
  - They abandon a product for a new one
    - Contributes to DMS problems
  - They change the product to lower their production costs
    - e.g., swap a part like a memory module
    - "Minor changes" can have surprising side effects

- Most likely you will not have good documentation
  - Vendor product manuals and sales brochure sheets
  - Few detail specifications
  - No drawings

COTS – The Ugly!

- You get what you get – Be sure to test what you really need
  - Vendors sometimes claim to meet a specification they really do not
    - Do not implement lesser used parts of the specification
  - Bugs & glitches
- You usually have to live with the vendor's choices
  - If a vendor does not sell it, you will have to develop it yourself
  - Riskiest approach: Modify a product from a COTS vendor and use it as your own component
    - It is now your product, the original vendor will not support it
    - You probably do not have the needed documentation if there is a problem
- Resolving DMS on a COTS component often has side effects
  - New generations of equipment change may also change interfaces
  - Example: New CPU hardware may need new driver or OS upgrade

Approaches to COTS Management



- No single answer is obviously the best
- Solution is often a mix of answers depending on subsystem/component
- Need an approach the system's user community can support
- Initial system concept and requirements sets the wheels in motion



Systems Engineering Technical Short Course

(Session 1)

Appendix A:
Lesson 2 Supplementary Topics

Origins of Systems Engineering

- **Rapid growth of technology** *…. stimulated by WWII and investments in technology for military advantage*
- **Growth of Automation** *… stimulated by the advent of the digital computer and software technology*
- **Competition** *… driving innovation and more advanced system solutions …*
- **Specialization** *… partitioning complex systems into specialized components*

Each of these is discussed in a subsequent slide

Origins of Systems Engineering:

Rapid Growth of Technology

- **Radar technology**
- **Guided missile technology**
- **High performance aircraft technology**
- **Nuclear technology**

Origins of Systems Engineering:

Growth of Automation

- **Digital Computers**
- **Software Technology**
- **Air Traffic Control**
- **Computer-aided Design**
- **Computer-aided Manufacturing**
- **Industrial Automation**

Origins of Systems Engineering:

Competition

- **Innovation increases risks**
  - **Use of new technology may have *unintended consequences***
  - **Characteristics of new materials, processes and components *not well understood***
  - **Unanticipated issues may arise during system development that could**
    - **Increase costs**
    - **Delay program delivery**

- **Conservative approach also risky**
  - **Competitive advantage relies on bringing new products and technologies to market ahead of your competitors**
  - **Inferior products become obsolete quickly**

Risk Management ⇔ *the process of dealing with calculated risks through a process of analysis, development, testing and engineering oversight*

Origins of Systems Engineering:

Specialization

- **Partitioning complex systems into modular, disparate building blocks that work together to form an efficient operational system requires:**
  - **Modularity** ***… a measure of the degree of mutual independence of the individual system building blocks***
  - **Decomposition techniques** ***… to form highly modular functional and physical building blocks that can be re-used efficiently.***
  - **Integration processes** ***… which ensure that the disparate building blocks "fit' together perfectly to provided the desired response required for the operational system***
  - **Interfaces** ***… which define boundaries between the modular components***
  - **Interactions** ***… which define the behaviors between and among modular components***

Systems Engineering as a Profession

CNNMoney
A Service of CNN, Fortune & Money

FORTUNE | Money

Home | Video | Business News | Markets | Term Sheet | Economy | Tech | Personal Finance

BEST JOBS IN AMERICA *Money/PayScale.com's list of great careers* 2009

Full List | High Pay | Job Growth | Quality of Life | Sectors

1. Systems Engineer 1 of 50 Next

Top 50 rank: 1
Sector: Information Technology

What they do: They're the "big think" managers on large, complex projects, from major transportation networks to military defense programs. They figure out the technical specifications required and coordinate the efforts of lower-level engineers working on specific aspects of the project.

Why it's great: Demand is soaring for systems engineers, as what was once a niche job in the aerospace and defense industries becomes commonplace among a diverse and expanding universe of employers, from medical device makers to corporations like Xerox and BMW.



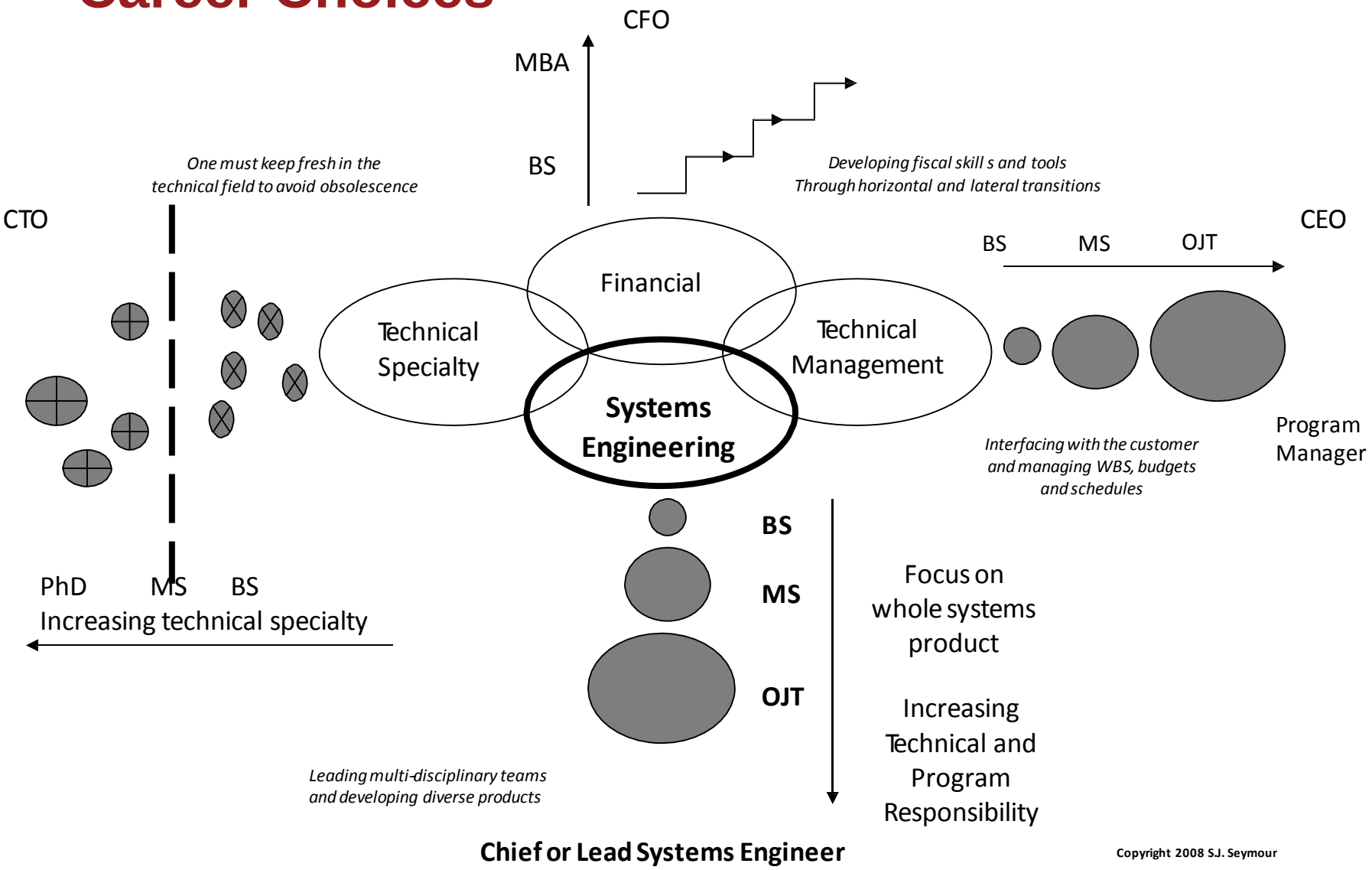
PHOTO: BRAD DECECCO

Anne O'Neil, a chief systems engineer at the N.Y.C. Transit Authority, is one of five female senior managers in a department of 1,500.

Pay can easily hit six figures for top performers, and there's ample opportunity for advancement. But many systems engineers say they most enjoy the creative aspects of the job and seeing projects come to life. "The transit system I work on really makes a tangible difference to people," says Anne O'Neil, chief systems engineer for the New York City Transit Authority.

- **Gaining popularity in the commercial sector**
- **Not just the federal space anymore**
- **Net centric systems and the complexity they contain are primary driver**
- **Telecommunications and Unified Communications are a great example of highly complex commercial systems**

Career Choices

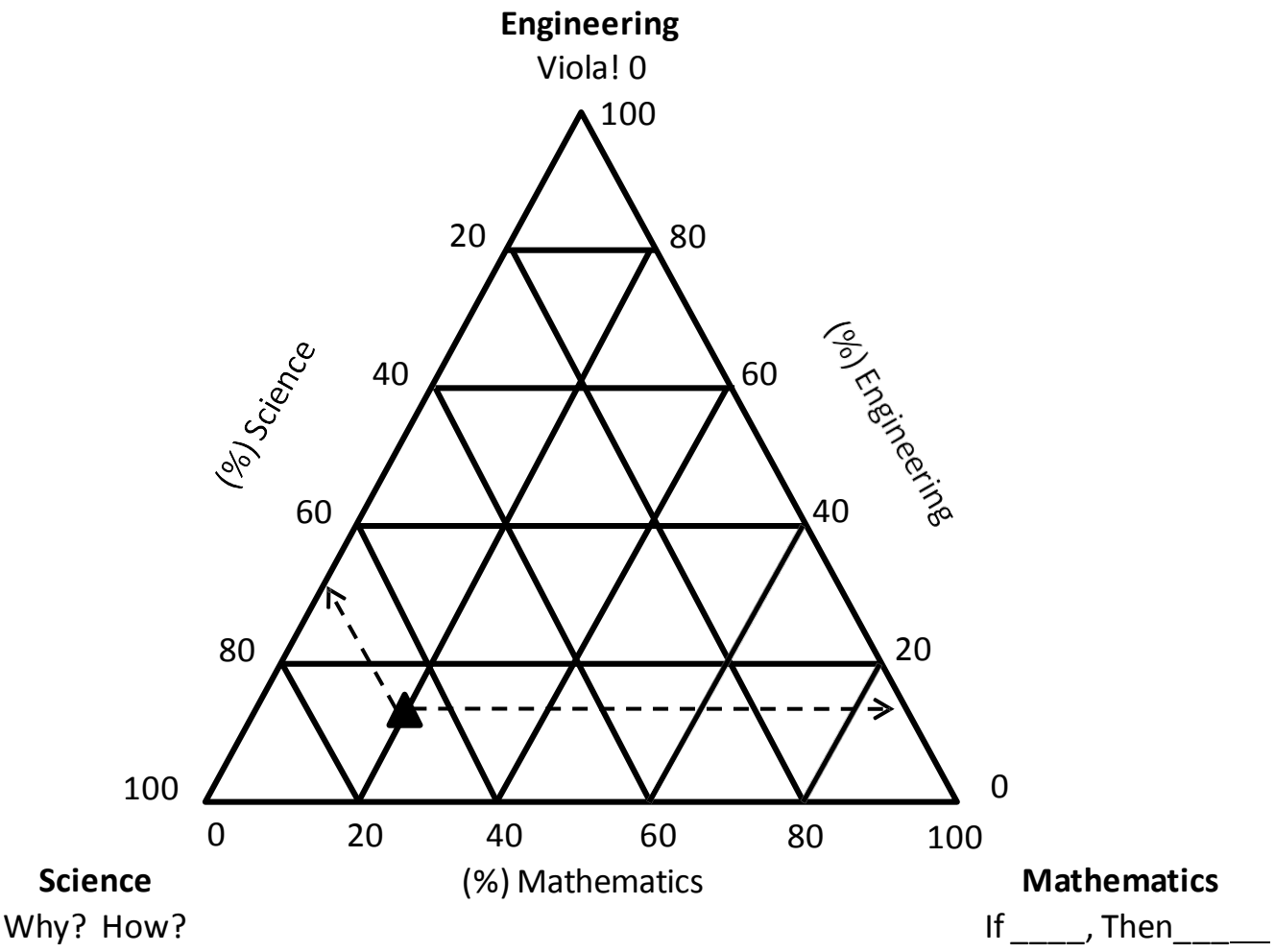


Copyright 2008 S.J. Seymour

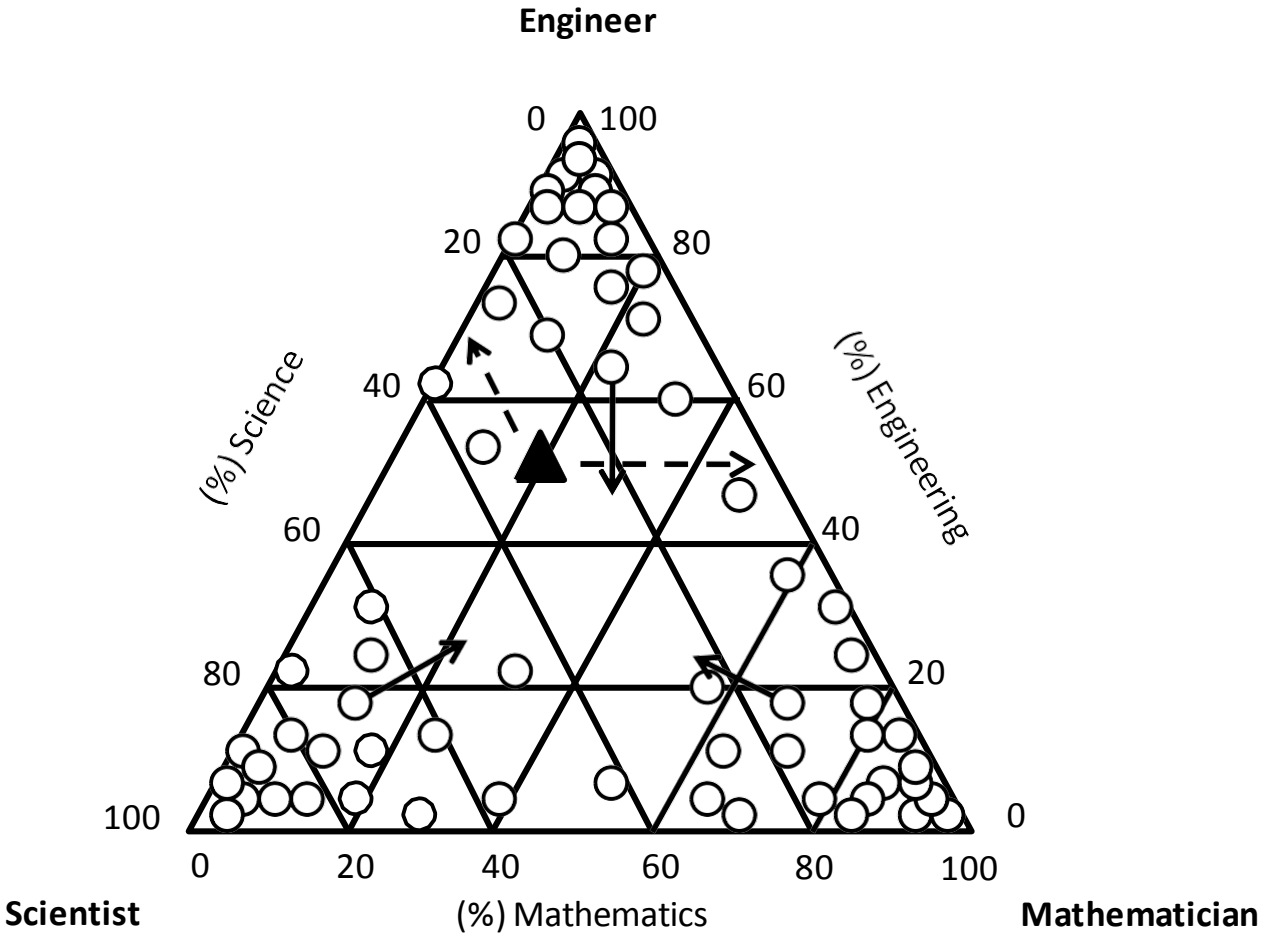
Orientation of Technical Professionals

- **The Engineer – Viola!**
- **The Mathematician – IF A, THEN B**
- **The Scientist – Why? How?**
- **Polarization (competitive nature) into distinct disciplines exacerbates the need for systems engineers**
- **Certain unique individuals resist polarization, cross the boundaries and enjoy taking on the bigger and more complex problem**

Technical Orientation Phase Diagram



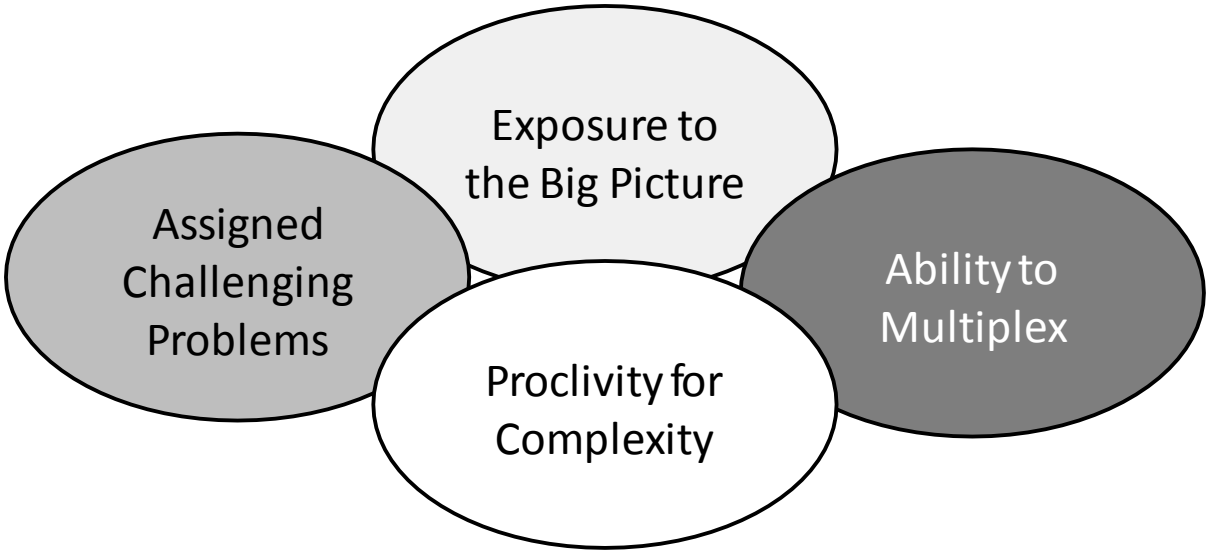
Technical Orientation Population Density Distribution



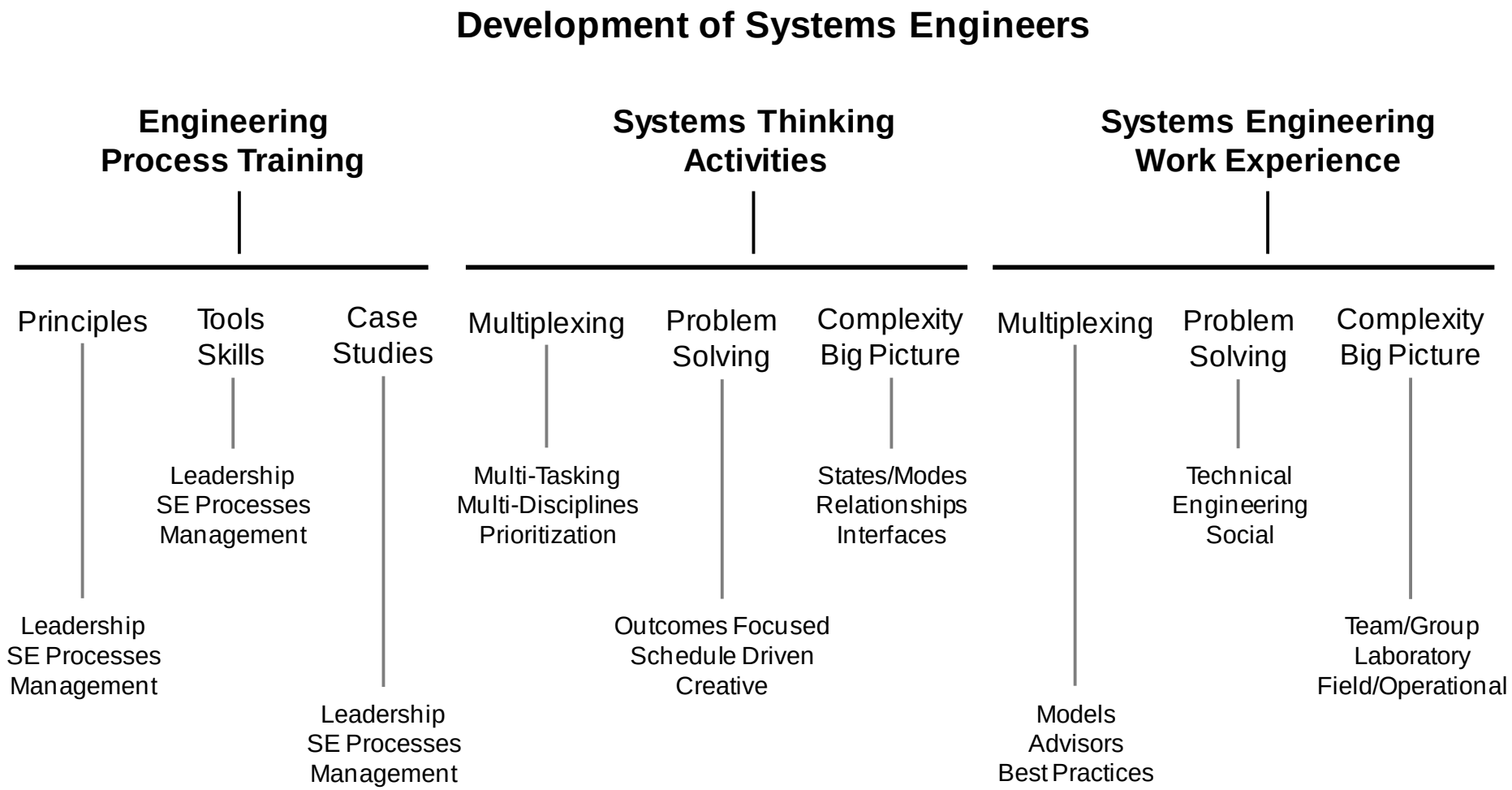
Attributes and Motivations

- Enjoy learning new things and solving problems
- Like challenges
- Skeptical of unproven assertions
- Open minded to new ideas
- A solid background in science and engineering
- Demonstrated technical achievement in a specialty area
- Knowledgeable in several engineering areas
- Pick up new ideas and information quickly
- Good interpersonal and communication skills

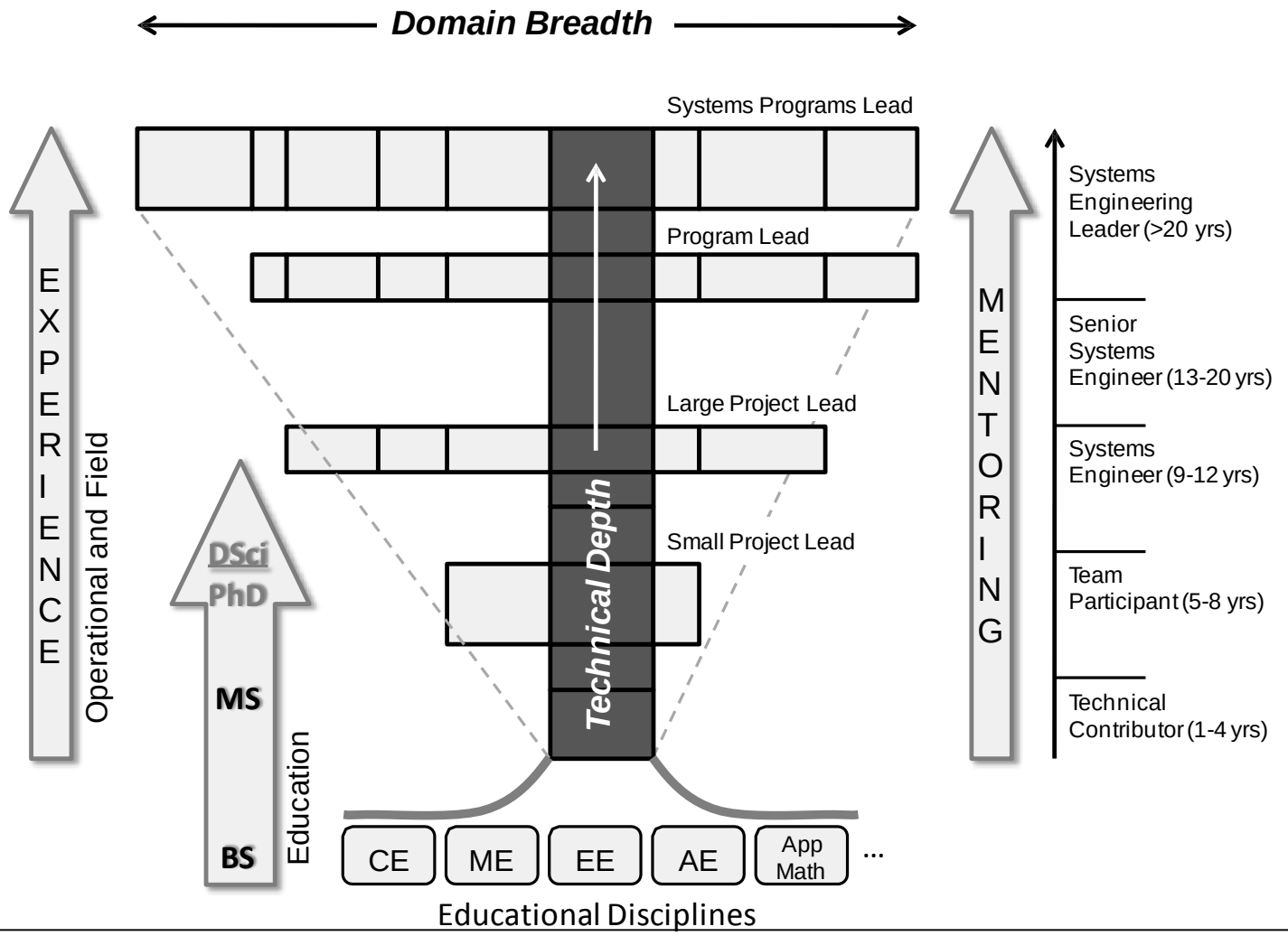
Career Development Model



Career Development Components



T Model for Systems Engineering Career Development



INCOSE (www.incose.org)

The International Council on Systems Engineering



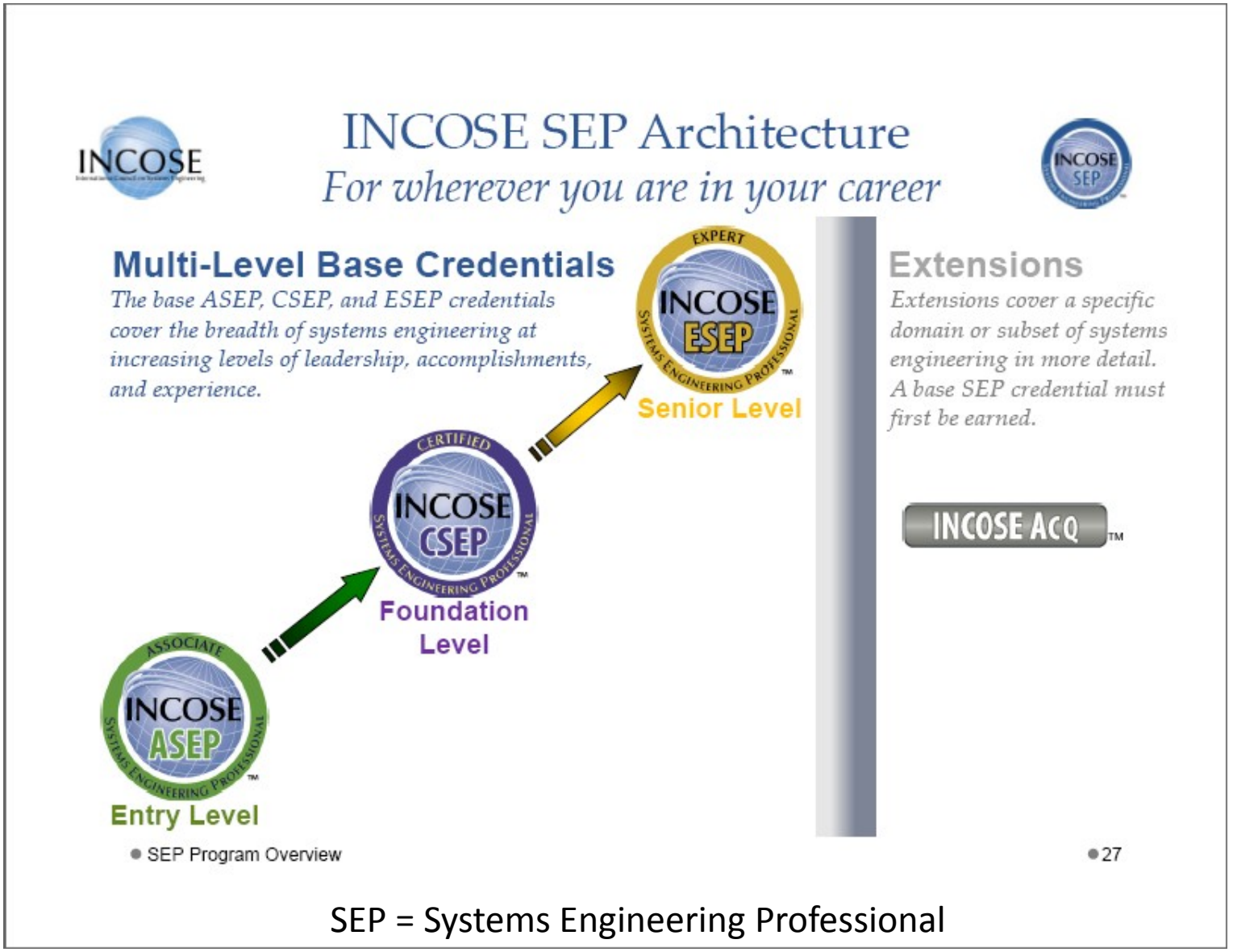
INCOSE Membership

- **Regular Membership dues are $135**
- **Student Membership dues are $35**
  - **Members qualify for the Student category if their course load is at least 3/4 of full time in an engineering or related field**
- **Systems Engineering Handbook**
  - **WPI Student and Members may download a softcopy at *no charge***
  - **As a privilege you have access to the INCOSE INSIGHT Library and Products Area. The link to the site is https://connect.incose.org/products/default.aspx**

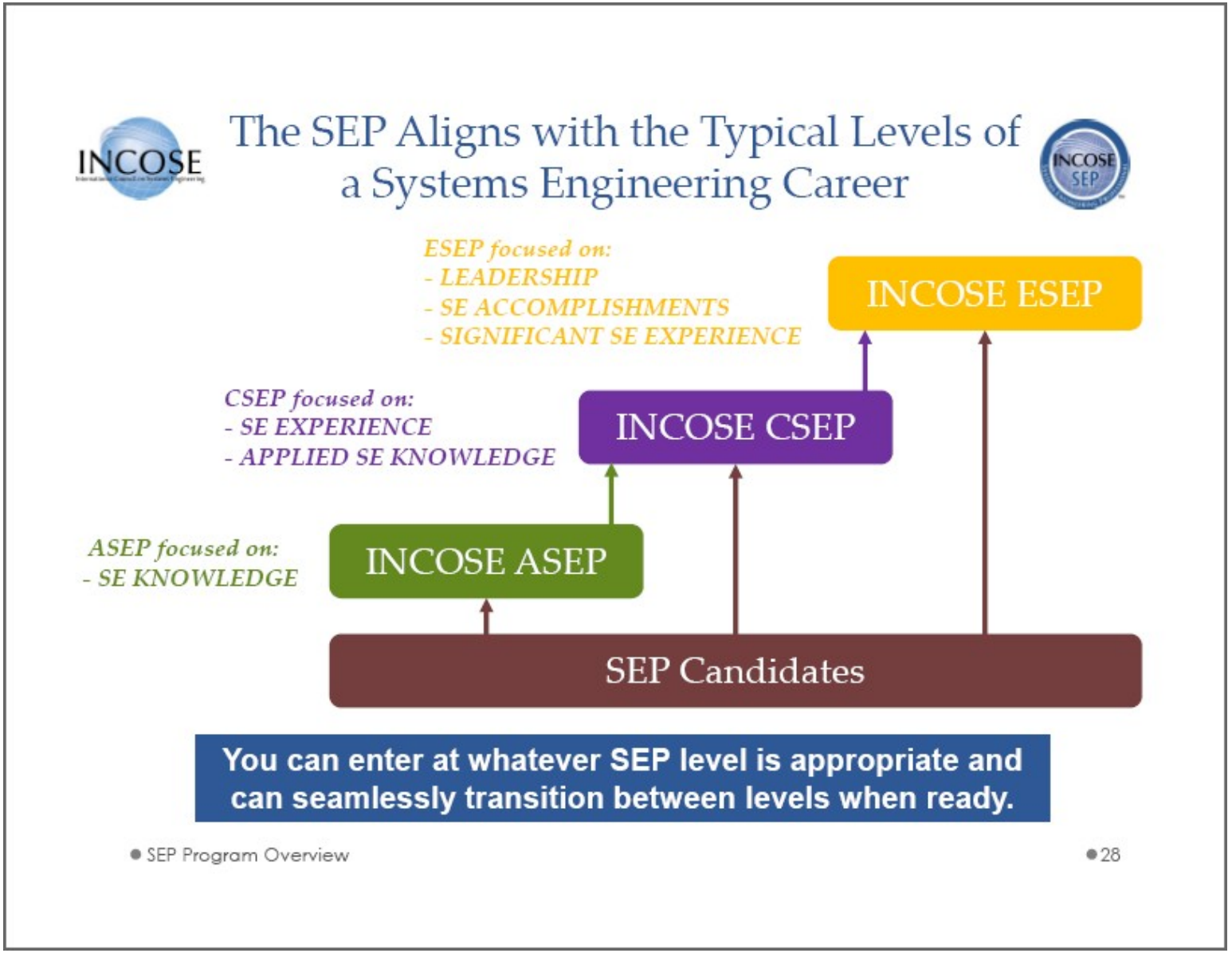
    The username is: worcesterpolytechnic

    The password is: wpicpe

*** All prices are as of June 2013**



SEP = Systems Engineering Professional


The SEP Aligns with the Typical Levels of a Systems Engineering Career
ESEP focused on:
- LEADERSHIP
- SE ACCOMPLISHMENTS
- SIGNIFICANT SE EXPERIENCE
INCOSE ESEP
CSEP focused on:
- SE EXPERIENCE
- APPLIED SE KNOWLEDGE
INCOSE CSEP
ASEP focused on:
- SE KNOWLEDGE
INCOSE ASEP
SEP Candidates
You can enter at whatever SEP level is appropriate and can seamlessly transition between levels when ready.
SEP Program Overview
28

Acq Extension

US DoD Acquisition Extension

- Targeted towards systems engineers who support or work in a US Department of the Defense acquisition environment
- Candidates must first become an ASEP, CSEP, or ESEP
- Acquisition knowledge items tested through both the core ASEP/CSEP exam and an additional Acq exam (Note: ESEPs must also take both exams)
- Extension renewed concurrently with the corresponding base certification
- Available since 2008

INCOSE ACQ™



Systems Engineering Technical Short Course

(Session 1)

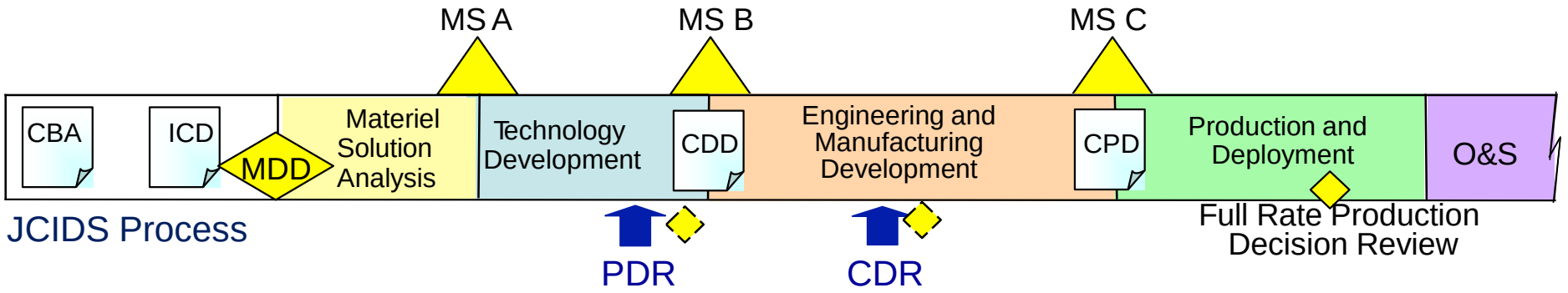
Appendix B:

Supplement to Lesson 4
DoD Instruction 5000.02
8 December 2008Systems

DoD Instruction 5000.02
8 December 2008

Operation of the Defense Acquisition System
Statutory and Regulatory Changes

Defense Acquisition Management System

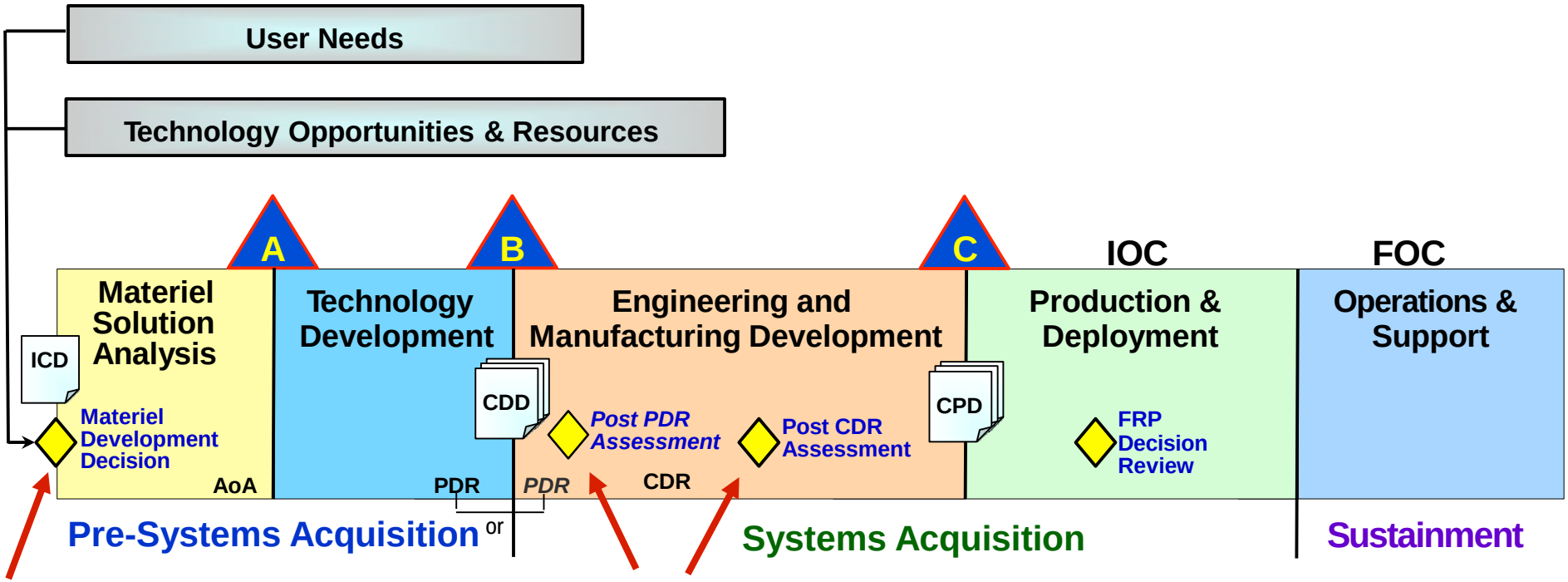


Some Terms

MDAP: Major Defense Acquisition Program
MS: Milestone
JCIDS: Joint Capability Integration Development System
CBA: Capabilities Based Assessment
ICD: Initial Capabilities Document
MDD: Materiel Development Decision
CDD: Capability Development Document
CPD: Capability Production Document
O&S: Operations and Support
PDR: Preliminary Design Review
CDR: Critical Design Review

DISTRIBUTION STATEMENT A -- Cleared for public release by OSR on 11 Jan 2010 -- SR case number 10-S-0779 refers.

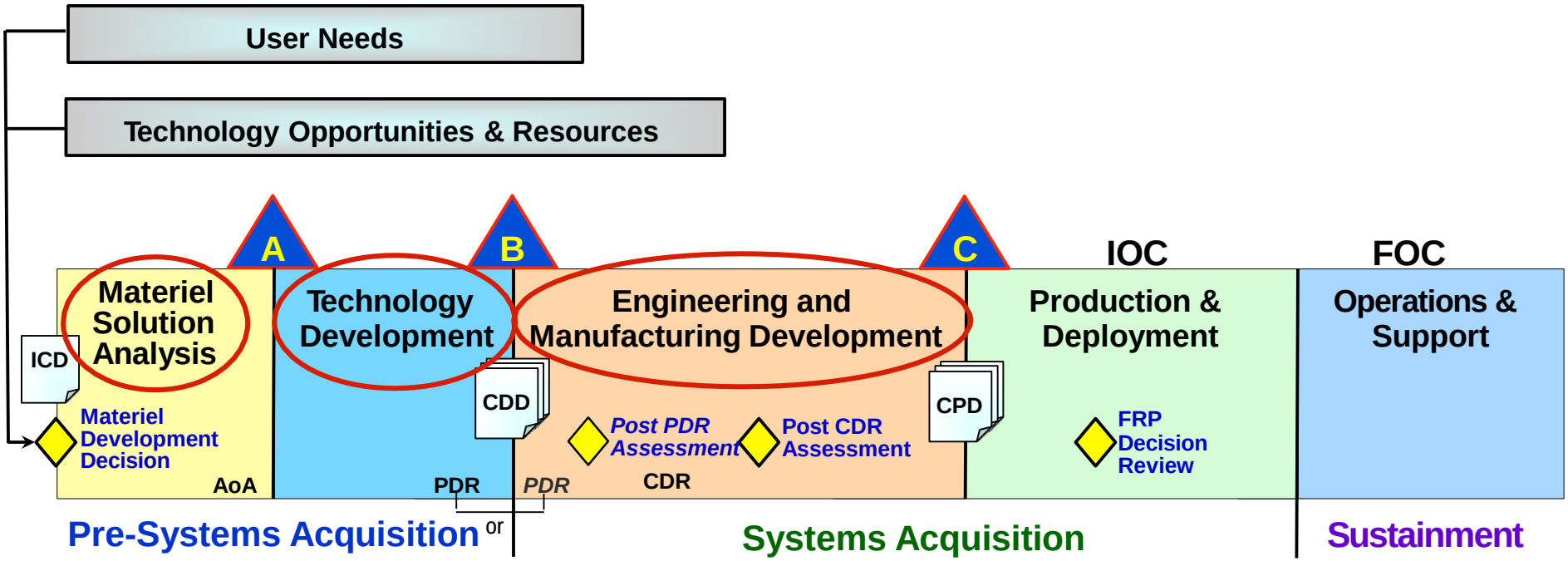
The Defense Acquisition Management System 2008



Changes to Decision Points

Old (2003)	New (2008)	Change from 2003
Concept Decision (CD)	Materiel Development Decision (MDD)	MDD required prior to entering the process at any point
N/A	*Post-PDR Assessment*	*MDA's assessment of PM's PDR Report (if PDR after MS B)*
Design Readiness Review DRR	Post-CDR Assessment	MDA's assessment of PM's CDR Report

The Defense Acquisition Management System 2008

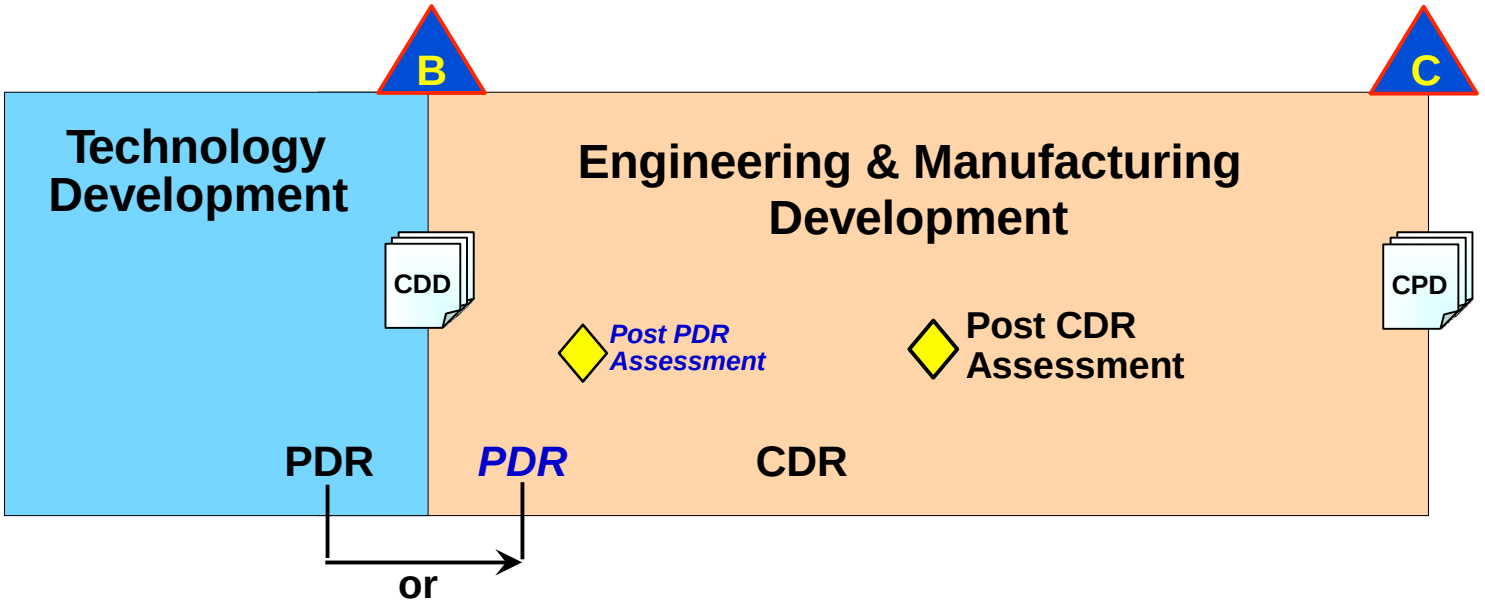


Changes to Phases

Old (2003)	New (2008)	Change from 2003
Concept Refinement (CR)	Materiel Solution Analysis	More robust AoA (result of changes to JCIDS)
Technology Development (TD)		Competitive prototyping
Systems Development & Demonstration (SDD)	Engineering & Manufacturing Development (EMD)	More robust system engineering

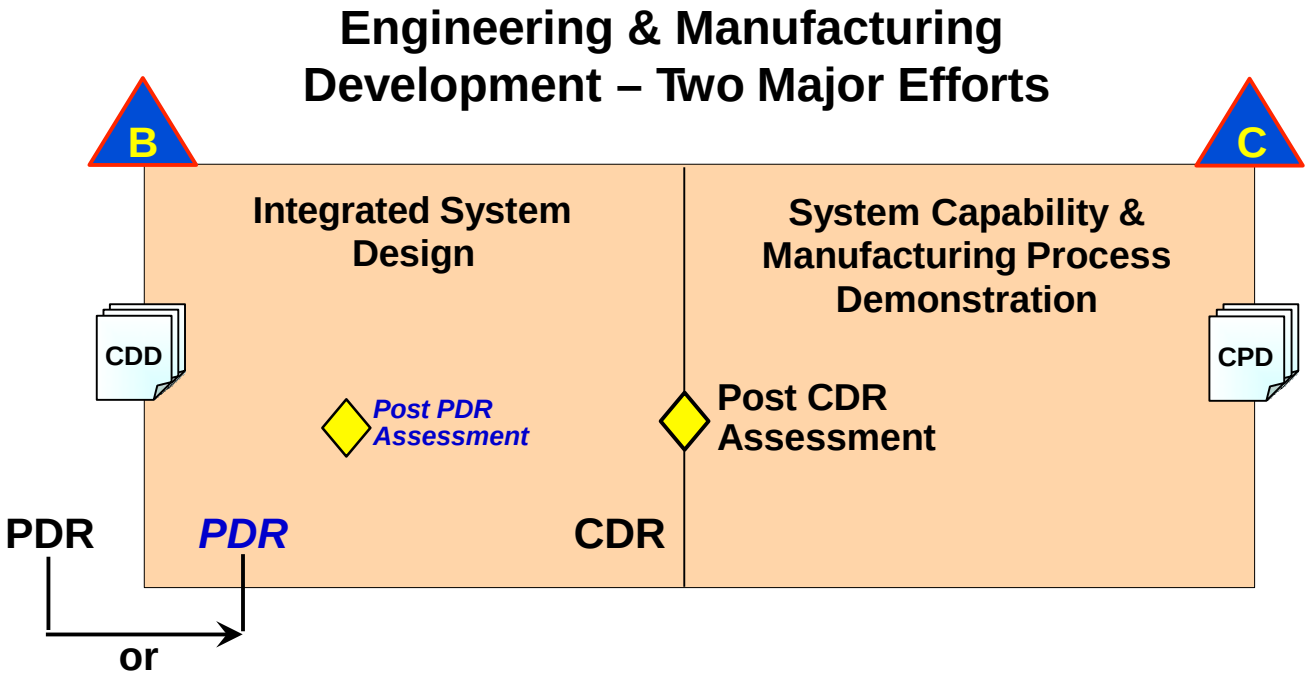
The Defense Acquisition Management System 2008

Preliminary Design Review



PDR Before Milestone B	PDR After Milestone B
• Planned for in Technology Development Strategy • PDR Report provided to MDA at MS B • Includes recommended requirements trades	• Planned for in Acquisition Strategy • PDR Report provided to MDA prior to Post PDR Assessment • Reflects requirements trades • At Post PDR Assessment, MDA considers PDR report; determines action(s) required to achieve APB objectives and issues ADM

The Defense Acquisition Management System 2008

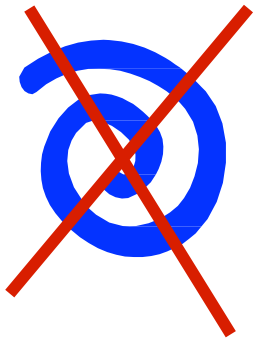


Old (2003)	New (2008)	Change from 2003
System Design	Integrated System Design	Establishment of Product Baseline for all Configuration Items
System Demonstration	System Capability & Manufacturing Process Demonstration	Manufacturing processes effectively demonstrated; production-representative article(s) demonstrated in intended environment; T&E assesses improvements to mission capability and operational support based on user needs.

Evolutionary Acquisition

From two processes… → **To one process…**

- Incremental Development: End-state is known; requirements met over time in several increments
- Spiral Development: End-state is not known; requirements for increments dependent upon technology maturation and user feedback.



- Capability delivered in increments, recognizing up front need for future capability improvements
- Each increment:
  - depends on mature technology
  - is a militarily useful and supportable operational capability
  - Successive Technology Development Phases may be needed to mature technology for multiple increments

Encl 12, Systems Engineering (New)

- **Systems Engineering Plan (SEP) required at each milestone**
- **Milestone Decision Authority (MDA) is approval authority for the SEP**
- **For programs where USD(AT&L) is MDA, and programs on the DT-only portion of OSD T&E Oversight List, SEPs must be submitted to Director, Systems and Software Engineering 30 days prior to DAB/ITAB review**
- **Program Executive Offices (PEOs) must have lead systems engineer – oversees SE across PEOs portfolio; reviews SEPs; assesses performance of subordinate systems engineers with PEO and program manager (PM)**
- **Event-driven technical reviews required – with SMEs independent of program, unless waived by MDA**
- **Requires configuration management to establish and control product attributes and the technical baseline**
- **Spectrum Supportability determination required**

Encl 12, Systems Engineering (New)
Continued…

- **Environmental Safety and Occupational Heath (ESOH) risk management required to be integrated with overall SE process; Programmatic ESOH Evaluation (PESHE) required of all programs regardless of ACAT**
- **NEPA and EO 12114 (Environmental Effects Abroad of Major Federal Actions) analysis required of PM, approved by CAE**
- **Addresses PM support of Mishap Accident Investigations**
- **Requires Corrosion Prevention Control Plan for ACAT I programs at MS B and C**
- **Requires PMs to employ modular open systems approach to design**
- **Data Management Strategy (DMS) required to assess long-term technical data needs of the system – included in Acquisition Strategy**

Systems Engineering Technical Short Course

(Session 1)

Appendix: C
Lesson 14 Supplement: COTS Considerations

Reference : "Systems Engineering, Principles and Practice," Kossiakoff et. al – Chapter 15

Special Consideration: Commercial Off The Shelf Equipment (COTS)

- **COTS – Components or equipment that you purchase from a commercial company that is not custom designed for your application**
  - **Sold to the general public as "a catalog item"**
  - **Government has a long definition**

"The entire process of system development – requirements, design, implementation, testing, maintenance – will undergo many radical changes when a system incorporates multiple COTS components. By committing to just one part of the "new paradigm" (i.e., simply buying some commercial products), it is vital to realize that this will usually demand a commitment to all of it."

From "Quotations from Chairman David, A Little Red Book of Truths to Enlighten and Guide on the Long March Toward the COTS Revolution," Carnegie Mellon Software Engineering Institute, 1998

COTS – The Good

- **Reduces development time**
  - **Functionality can be immediately purchased**
- **Lower initial cost**
  - **Get the advantage of quantity production**
  - **Someone else has already done the design and development**
- **Competitive environment**
  - **Stimulates improving performance**
  - **Contributes to lower initial purchase cost**

COTS – The Bad

- Vendor-unique product

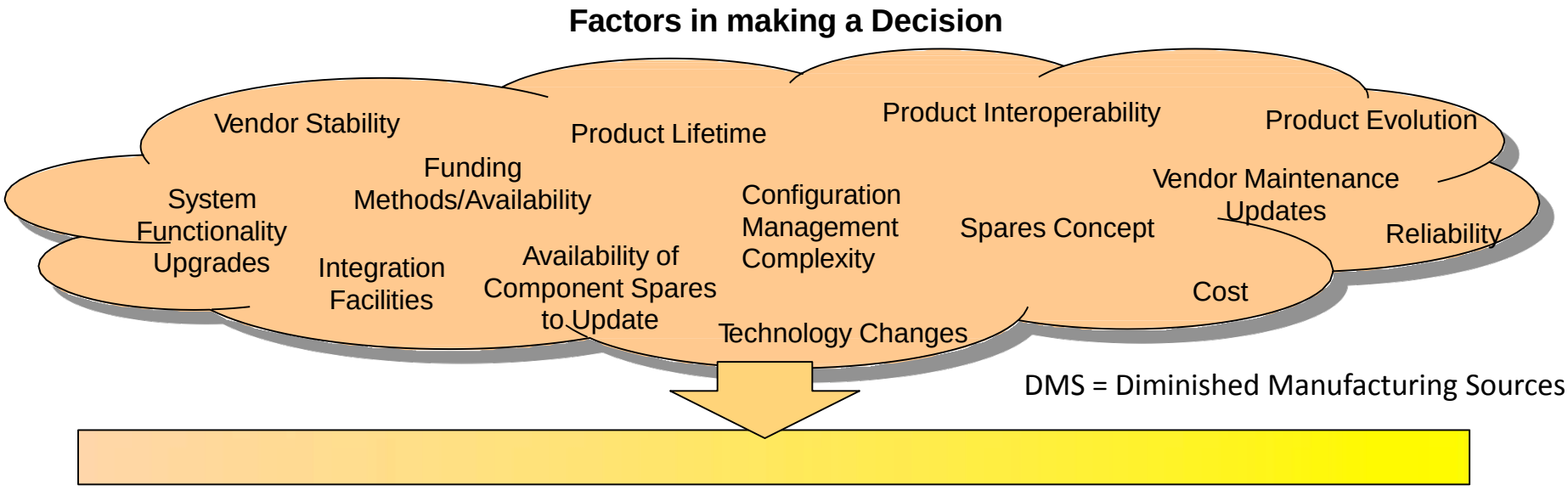
- Vendor can change their products at their whim
  - They abandon a product for a new one
    - Contributes to DMS problems
  - They change the product to lower their production costs
    - e.g., swap a part like a memory module
    - "Minor changes" can have surprising side effects

- Most likely you will not have good documentation
  - Vendor product manuals and sales brochure sheets
  - Few detail specifications
  - No drawings

COTS – The Ugly!

- You get what you get – Be sure to test what you really need
  - Vendors sometimes claim to meet a specification they really do not
    - Do not implement lesser used parts of the specification
  - Bugs & glitches
- You usually have to live with the vendor's choices
  - If a vendor does not sell it, you will have to develop it yourself
  - Riskiest approach: Modify a product from a COTS vendor and use it as your own component
    - It is now your product, the original vendor will not support it
    - You probably do not have the needed documentation if there is a problem
- Resolving DMS on a COTS component often has side effects
  - New generations of equipment change may also change interfaces
  - Example: New CPU hardware may need new driver or OS upgrade

Approaches to COTS Management



Pre-Emptive
Replace all COTS on a fixed schedule, DMS or not

A Spectrum of Possibilities

Reactive
Replace a COTS item when it goes DMS

- No single answer is obviously the best
- Solution is often a mix of answers depending on subsystem/component
- Need an approach the system's user community can support
- Initial system concept and requirements sets the wheels in motion

Systems Engineering Technical Short Course

Appendix D
Course Supplement

This supplement contains course material and information that was not available when the course materials were printed



Appendix D Contents

- Link to Video explaining how to use the WPI course website
- Communication
- Additional course references
- Others may be added as the course progresses

How to Access WPI Course Website

WPI has set up a site dedicated to the Waters Corporation Technical Short Course in Systems Engineering.

This site contains the course materials and other useful information and tools for your use during the course.

The link below is to a video that explains how to access and use the course site

Video Tutorial for myWPI

An Essential of Project Management - Communication

Forsberg and Mooz identify five essentials of good project management:

1. **Organizational Commitment – includes culture, charter, resources needed, and tools and training**
2. **Communication – language and techniques used to achieve mutual understanding across the team – based on a common vocabulary**
3. **Teamwork – Effectively working together towards a common goal, with acknowledged interdependency and trust, and acceptance of a common code of conduct, and with a shared reward**
4. **Project cycle – related to the system lifecycle discussed throughout the course**
5. **Management Elements – essentially the processes that are applied situationally through the life cycle**

Adapted from one of our texts: Visualizing Project Management, by Forsberg et al

Communication

The object of communication is mutual understanding

Effective communication is based upon:

1. **Theory of communication properly applied**
   - While beyond the scope of this course, we will discuss some aspects of this when we deal with stakeholders in session 2
   - You will probably touch on this in your collaboration training
2. **A communication plan – identifying the key players, and their communication roles (i.e., who are the official points of contact, and what is the scope of their official communication)**
3. **A common vocabulary – and a commitment to use it**
   - Glossary of SE terms
   - Glossary of project terms
4. **A team trained in the use of appropriate tools – such as modeling tools, SysML, and others**

Adapted from one of our texts: Visualizing Project Management, by Forsberg et al

Communication in Context

Systems Engineering must communicate:

- ideas
- concepts
- requirements
- solutions

Across organizational:

- boundaries
- levels
- disciplines
- environments

To achieve:

- mutual understanding
- mutual agreement
- stakeholder satisfaction
- a solution with system integrity

Modeling and models are an essential communication tool to achieving mutual understanding and agreement.

We use models and simulations in all of their forms. How do models contribute to systems thinking?

Modeling is the language of Systems Engineering

Primary Reference Textbooks (shown earlier)

Systems Engineering Principles and Practice (2nd ed.), Kossiakoff, A., Sweet, W. N., Seymour, S., and Biemer, S. M., John Wiley & Sons, Inc., 2nd Edition (May, 2011), ISBN-13: 9780470405482.

Discovering Requirements: How to Specify Products and Services, Alexander, Ian and Beus-Dukic, Ljerka, John Wiley & Sons, Inc., 2009, ISBN-13: 9780470712405.

The Engineering Design of Systems: Models and Methods, Buede, Dennis M., John Wiley & Sons, Inc., 2nd Edition (February, 2009), ISBN: 0-470-16402-6.

The Art of Systems Architecting, Third Edition, Rechtin, Eberhardt and Maier, Mark W., CRC, 3rd Edition (January, 2009), ISBN: 1-420-07913-1.

Visualizing Project Management: Models and Frameworks for Mastering Complex Systems, Forsberg, Mooz, and Cotterman, Wiley, 3rd edition (September 1, 2005), ISBN-13: 978-0471648482

Selected Other References (1 of 8)

Systems Engineering Handbook – A Guide for System Life Cycle Processes and Activities, Version 3.2.2, The International Council on Systems Engineering, 2011

NASA Systems Engineering Handbook, NASA/SP-2007-6105, Revision 1

Handbook of Systems Engineering and Management, Sage and Rouse (editors), John Wiley and Sons, 1999

Systems Analysis, Design and Development – Concepts, Principles, and Practices, Charles S. Wasson, Wiley-Interscience, 2006

Systems Requirements Analysis, Jeffrey O. Grady, Elsevier Academic Press, 2006

Writing Better Requirements, Ian F. Alexander and Richard Stevens, Addison-Wesley, 2002

Selected Bibliography (2 of 8)

Introduction to Systems Engineering, Sage and Armstrong, Second edition, Wiley-Interscience, 2011

Augustine's Laws, 6th edition, Norman R Augustine, AIAA, 1997

The Bishop's Boys: A Life of Wilber and Orville Wright, Tom Crouch, W.W. Norton and Co, Inc. 1989

Communicating Project Management, Hal Mooz, Kevin Forsberg, Howard Cotterman, John Wiley & Sons, 2003.

Requirements Engineering, 3rd edition, Elizabeth Hull, Ken Jackson, and Jeremy Dick, Springer, 2011

Selected Bibliography (3 of 8)

Credibility: How Leaders Gain and Lose It, Why People Demand It, James S. Kouzes, and Barry Z. Posner, Jossey-Bass, Inc. 1993

The Critical Chain, Eliyahu. M Goldratt, North River Press, 1997

The Design of Everyday Things, Donald A. Norman, Doubleday, 1990

The Dilbert Principle, A Cubicle's-Eye View of Bosses, Meetings, Management Fads and Other Workplace Afflictions, Scott Adams, Harper Business, 1996.

Effective Requirements Practices, 3rd printing, Ralph R. Young; Addison-Wesley, 2004.

The Fifth Discipline: The Art and Practice of The Learning Organization, Revised edition, Peter M. Senge, Doubleday, 2006

Selected Bibliography (4 of 7)

A Guide to the Project Management Body of Knowledge, 4th Edition, Project Management Institute, Inc. Four Campus Boulevard, Newtown Square, Penn, 19073-3299, 2008

Leadership Secrets of Attila the Hun, Wess Roberts, Warner Books, inc. 1271 Avenue of the Americas, New York, New York 10020, 1985

Mastering the Requirements Process, 2nd edition, Suzanne Robertson, James Robertson; Addison Wesley, 2011.

Mastering Virtual Teams, Deborah L. Duarte and Nancy Tennant Snyder, Jossey Bates, 1999.

Necessary But Not Sufficient, Eliyahu. M Goldratt, North River Press, 1998

Partnership Pays: Project Management the Øresund Way, Helena Russel, route One Publishing Ltd, Horizon House, Azalea Drive, Swanley, Kent BR8 8JR, UK, 2000.

Selected Bibliography (5 of 8)

A Framework for Information Systems Architecture, John Zachman, IBM Los Angeles Scientific Center, 1986

The Path Between the Seas: The Creation of the Panama Canal 1870 – 1914, David McCullough, Simon and Schuster1977

People Styles at Work, Robert Bolton and Dorothy Grover Bolton Ridge Associated, Inc. 1996

The Project Surgeon, A Troubleshooter's Guide to Business Crisis Management, Boris Hornjak, Project Management Institute, 2001

Rescuing Prometheus, The Story of the Mammoth Projects-SAGE, ICBM, ARPANET/Internet, and Boston's Central Artery/Tunnel-That Created New Styles of Management, New Forms of Organization, and a New Vision of Technology, Thomas P. Hughes Pantheon Books, 1998

Selected Bibliography (6 of 8)

Ship of Gold in the Deep Blue Sea, Gary Kinder, Atlantic Monthly Press 1998

Skunk Works, Ben Rich with Leo Janos, Little-Brown, Inc, 1994.

Software Engineering, A Practitioner's Approach, 7th edition, Roger S. Pressman, 2010

The Soul of a New Machine, Tracy Kidder, Random House, 1981

Structures: Or Why Things Don't Fall Down, J.E. Gordon, DaCapo Press, 1981

Success in Project-Driven Organizations People, Processes and Politics, Joan Knutson, John Wiley and Sons, 2001

System Engineering Management, Benjamin Blanchard, John Wiley & Sons, Inc., 1998

Systems Analysis, Design and Development – Concepts, Principles, and Practices, Charles S. Wasson, 2006

Selected Bibliography (7 of 8)

Success in Project-Driven Organizations People, Processes and Politics, Joan Knutson, John Wiley and Sons, 2001

Systems Architecting: Creating & Building Complex Systems, Eberhard Rechtin, PTR Prentice-Hall 1991

Systems Engineering: Coping with Complexity, Richard Stevens et. al, Prentice-Hall, 1998

Systems Engineering and Analysis, 5th edition, Benjamin S. Blanchard and Wolter J. Fabrycky, Prentice Hall, 2011

NASA Systems Engineering Handbook, Rev 1, NASA/SP-2007-6105, 2007

Systems Engineering Guidebook, James N. Martin, CRC Press, 1996

Selected Bibliography (8of 8)

System Integration, Jeffrey O. Grady, CRC Press, 1994.

Systems Requirements Analysis, Jeffrey O. Grady, Academic Press, 2006

Systems Thinking and Learning, Stephen G. Haines, HRD Press 1998

System Verification – Proving the System Design Solution Satisfies the Requirements, Jeffrey O. Grady, Academic Press, 2007

To Engineer is Human: The Role of Failure in Successful Design, Henry Petroski, Vintage Books, 1992

Why Buildings Fall Down: How Structures Fail, Mattys Levy and Mario Salvadori, W. W. Norton and Company, 1992

Why Buildings Stand Up: The Strength of Architecture, Mario Salvadori, W W Norton and Company, 1990

Systems Engineering Technical Short Course for Waters Corporation

This course planned to be conducted at the Waters facilities as follows:

Session 1:	**January 27 – 29, 2015:**	**Overview and Concepts of Systems Engineering**
Session 2:	**February 16 – 17, 2015:**	**Stakeholder Needs**
Session 3:	**March 16 – 17, 2015**	**System Requirements**
Session 4:	**April 13 – 14, 2015**	**System Architecture and Design**
Session 5:	**May 13 – 14, 2015**	**System Integration and Test**
Session 6:	**June 10, 2015**	**Collaboration Training with WLI. (not part of WPI material)**
	June 11, 2015	**Capstone presentations**

Robert B. Wheeler, ESEP, PMP
Thomas F. Gannon, Ph.D.

This course is developed by Worcester Polytechnic Institute (WPI) for Waters Corporation. Course materials remain the property of WPI. WPI and the course developer (Robert Wheeler of Robert B Wheeler & Associates for sessions 1, 2, and 3) retain the right to use, reuse, and modify this content in future course deliveries.



Systems Engineering Technical Short Course

Session 2: Stakeholder Needs

February 16 – 17, 2015

Robert B. Wheeler, ESEP, PMP



Systems Engineering Technical Short Course

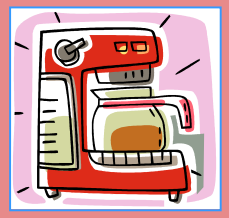
(Session 2: Stakeholder Needs)

Lesson 1:
Program Review Update

Welcome and Opening Remarks……

Logistics:

- **Facility**
- **Safety & emergencies**
- **Schedule & breaks**
- **Parking lot**
- **Other?**









Code of Conduct:

- On time
- Mutual respect
- Machines off & closed unless we are using them
- Participation!
- Other?

Remember from Session 1 – During this course you will: (1/2)

All Sessions: Apply the Systems Engineering principles and practices to your assigned projects

Sessions 1 through 3 (Bob Wheeler, instructor):

- **Develop and use a common vocabulary**
- **Apply systems thinking and holism to your projects**
- **Articulate the Problem to be Solved by your project**
- **Develop a Technical Risk Management Plan**
- **Elicit and express stakeholders expectations and requirements**
- **Develop User Concept of Operations (CONOPS), solution selection criteria, and acceptance criteria**
- **Conduct system functional and performance analyses**
- **Define and express system requirements**
- **Manage requirements through the system lifecycle**

Session 1	This Session 2	Session 3

Remember from Session 1 – During this course you will: (2/2)

Sessions 4 & 5 (Tom Gannon, instructor):

- **Develop creative alternative system architectures**
- **Conduct analyses of your architectural alternatives to meet stakeholder and system requirements**
- **Define a candidate architecture and design for your solution**
- **Develop strategies and plans for system integration, validation and verification testing**
- **Manage requirements through the system lifecycle (continued)**
- **Document the results of your work in appropriate artifacts**

Session 6:

- **Present a status report on your project to Waters senior management**

Schedule



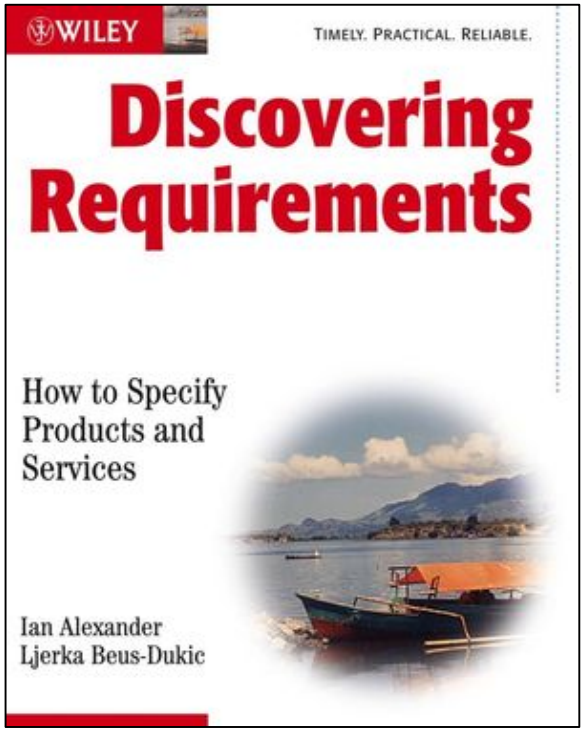


Session	Dates	Topic
1.	January 27 – 29	Introduction – Overview and Concepts of SE (US)
2.	February 16 – 17	Stakeholder Needs*
3.	March 16 – 17	Systems Requirements*
4.	April 13 – 14	Systems Architecture and Design (UK)*
5.	May 13 – 14	Systems Integration and Test*
6.	June 10	Collaboration Training with WLI
	June 11	Capstone Presentations (US)

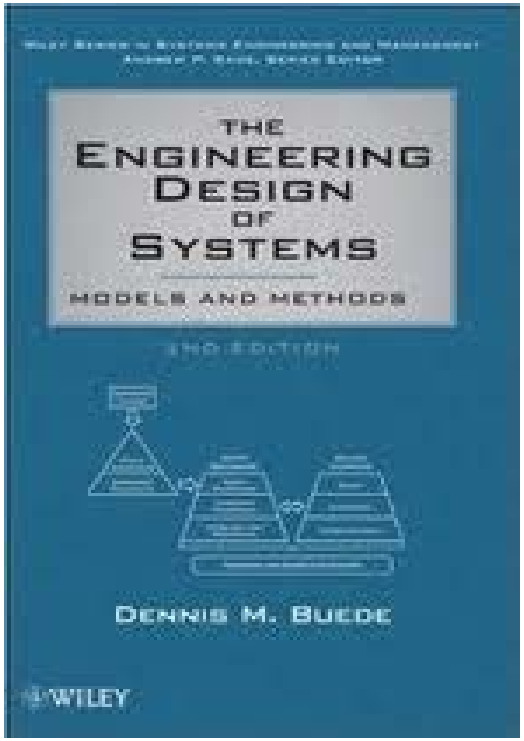
Notes:

- Denotes sessions held for some participants via video-teleconference.
- Class times and room locations are shown later

Primary Requirements Course Texts



Ian Alexander and Ljerka Beus-Dukic, *Discovering Requirements*, Wiley, 2009



Dennis M. Buede, The Engineering Design of Systems, 2nd Edition, New Jersey: Wiley, 2009

Please see Session 1 Appendix D for the other three primary course texts, and many other applicable references

What Happens if I Miss a Class?

It is important to your own learning and to the group process to attend all the class sessions.

However, it is understood that some absences cannot be avoided.

If you miss a class, please take the responsibility to:

- **Review the missed lectures**
- **Contact the instructor with any questions you may have as a result of reviewing the lecture materials**
- **Provide a brief outline of the missed material, identifying what you believe to be the key learning points, to the instructor.**
- **Contact your team project leader to ensure that you are aware of any project work completed, or action items assigned, during your absence.**

All lecture material will be posted on the WPI course site:

- **You will be told how to use the WPI course site, and how to access the lecture material, early in session 1**

Video Tutorial for myWPI

Room Assignments

Milford Rooms

Date	Time	Primary Location	Break-out Rooms			
27-Jan	8:00 - 4:00 pm	C4		N/A	N/A	N/A
28-Jan	8:00 - 4:00 pm	C4		N/A	N/A	N/A
29-Jan	8:00 - 4:00 pm	C4		N/A	N/A	N/A
16-Feb	7:00 - 4:00 pm	Atriums 2 & 3	1:00 - 4:00 pm	mfg conf	Atlantis	Global Ops
17-Feb	7:00 - 4:00 pm	C4	1:00 - 4:00 pm	mfg conf	Atlantis	Acquity
16-Mar	7:00 - 4:00 pm	Atriums 2 & 3	1:00 - 4:00 pm	mfg conf	Atlantis	Global Ops
17-Mar	7:00 - 4:00 pm	C4	1:00 - 4:00 pm	mfg conf	Atlantis	Acquity
13-Apr	7:00 - 4:00 pm	Atriums 2 & 3	1:00 - 4:00 pm	mfg conf	Atlantis	Global Ops
14-Apr	7:00 - 4:00 pm	C4	1:00 - 4:00 pm	mfg conf	Acquity	Atlantis
13-May	7:00 - 4:00 pm	C4	1:00 - 4:00 pm	mfg conf	Acquity	Atlantis
14-May	7:00 - 4:00 pm	C4	1:00 - 4:00 pm	mfg conf	Acquity	Atlantis
10-Jun	8:00 - 4:00 pm	WTC 2&3		N/A	N/A	N/A
11-Jun	8:00 - 1:00 pm	Director's Room		N/A	N/A	N/A

Wilmslow Rooms

Date	Time	Primary Location	Break-out Rooms			
16-Feb	12:00 - 18:00	Hydrogen	14:00 to 17:00	Helium	Neon	Silicon
17-Feb	12:00 - 18:00	Hydrogen	12:30 to 16:30	Helium	Oxygen	Silicon
16-Mar	12:00 –18:00	Hydrogen	14:00 to 17:00	Helium	Oxygen	Silicon
17-Mar	12:00 - 18:00	Hydrogen	12:30 to 16:30	Helium	Oxygen	Silicon
13-Apr	12:00 - 18:00	Hydrogen	14:00 to 17:00	Helium	Oxygen	Silicon
14-Apr	12:00 - 18:00	Hydrogen	12:30 to 16:30	Helium	Oxygen	Silicon
13-May	12:00 - 18:00	Hydrogen	14:00 to 17:00	Helium	Oxygen	Silicon
14-May	12:00 - 18:00	Hydrogen	12:30 to 16:30	Helium	Oxygen	Silicon

What are your Questions before We Look at Assignment 1 results?



Sessions 2 through 5 Agenda

- **Day 1**
  - **Review project team assignments from previous Session**
  - **Presentation and discussion of Session topic**
  - **Project team working sessions on the application of the Session topics to their projects**
  - **Day 1 wrap-up and Q&A**

- **Day 2**
  - **Review Day 1 topic and project working sessions**
  - **Presentation and discussion of Session topic (concluded)**
  - **Project team working sessions (concluded)**
  - **Check-in with class on the application of the Session topic to their projects**
  - **Session wrap-up: Q&A and homework due next Session**

Timing will be similar to session 1, and will be announced at beginning of each session

Session 1 Introduction: Learning Objectives

- **Develop and use a common vocabulary**
- **Explain systems engineering processes (both hard and soft) in the context of a system lifecycle model**
- **Plan and implement a risk management process**
- **Use a Systems Engineering Management Plan to plan the technical work on the project**

Session 1 Introduction: Homework Assignment: Due at beginning of Session 2*

Develop the following deliverables:

- Communication Plan, including common vocabulary approach and draft
- System Lifecycle Model for your project
- Risk Analysis emphasizing technical risks (consider doing a draft Risk Management Plan)
- Draft Systems Engineering Management Plan (SEMP)
- Draft Configuration Management Plan showing how you will control the evolving baseline

Update the previously developed deliverables:

- Project Description

* Each project team must decide which of, and the extent to which, each of these deliverables will be done

Assignment 1 Results: Reports from each Project Team

WPI SYSTEMS ENGINEERING PROJECT GROUPING

SYSTEMS CONTROLLER	NEXUS	SMART CONSUMABLES	WHISTLER
Sponsor: Allen Caswell	**Sponsor: Joe Luongo**	**Sponsor: Tom Walter**	**Sponsor: John Chan**
Cindy Miller	Abhijit Tarafder	Jacob Fairchild	Stephen Thornton
Richard Moulds	Wade Leville	Jason Hill	David Gordon
Richard Chapman	Jim Henriksen	Mathew DeLano	John Heden
Steve Binney	Ian Trivett	Jan Driesen	Phillip Young
Justin Deary	Daniel Kenny	Alex Hooper	
	Jim Usowicz	Mark Pickering	
	Joshua Scanlon	Senthil Bala	

Assignment 1 Results: Reports from each Project Team

WPI SYSTEMS ENGINEERING PROJECT GROUPING

SYSTEMS CONTROLLER	NEXUS	SMART CONSUMABLES	WHISTLER
Sponsor: Allen Caswell	**Sponsor: Joe Luongo**	**Sponsor: Tom Walter**	**Sponsor: John Chan**
Cindy Miller	Abhijit Tarafder	Jacob Fairchild	Stephen Thornton
Richard Moulds	Wade Leville	Jason Hill	David Gordon
Richard Chapman	Jim Henriksen	Mathew DeLano	John Heden
Steve Binney	Ian Trivett	Jan Driesen	Phillip Young
Justin Deary	Daniel Kenny	Alex Hooper	
	Jim Usowicz	Mark Pickering	
	Joshua Scanlon	Senthil Bala	

Assignment 1 Results: Reports from each Project Team

WPI SYSTEMS ENGINEERING PROJECT GROUPING

SYSTEMS CONTROLLER	NEXUS	SMART CONSUMABLES	WHISTLER
Sponsor: Allen Caswell	**Sponsor: Joe Luongo**	**Sponsor: Tom Walter**	**Sponsor: John Chan**
Cindy Miller	Abhijit Tarafder	Jacob Fairchild	Stephen Thornton
Richard Moulds	Wade Leville	Jason Hill	David Gordon
Richard Chapman	Jim Henriksen	Mathew DeLano	John Heden
Steve Binney	Ian Trivett	Jan Driesen	Phillip Young
Justin Deary	Daniel Kenny	Alex Hooper	
	Jim Usowicz	Mark Pickering	
	Joshua Scanlon	Senthil Bala	

Assignment 1 Results: Reports from each Project Team

WPI SYSTEMS ENGINEERING PROJECT GROUPING

SYSTEMS CONTROLLER	NEXUS	SMART CONSUMABLES	WHISTLER
Sponsor: Allen Caswell	**Sponsor: Joe Luongo**	**Sponsor: Tom Walter**	**Sponsor: John Chan**
Cindy Miller	Abhijit Tarafder	Jacob Fairchild	Stephen Thornton
Richard Moulds	Wade Leville	Jason Hill	David Gordon
Richard Chapman	Jim Henriksen	Mathew DeLano	John Heden
Steve Binney	Ian Trivett	Jan Driesen	Phillip Young
Justin Deary	Daniel Kenny	Alex Hooper	
	Jim Usowicz	Mark Pickering	
	Joshua Scanlon	Senthil Bala	

Session 2 Stakeholder Needs: Learning Objectives

- **Identify the “problem to be solved” and the “context”**
- **Elicit and express stakeholders expectations and requirements**
- **Develop a user Concept of Operations (CONOPS)**
- **Develop solution selection criteria and acceptance criteria**

Session 2 Stakeholder Needs: Homework Assignment due at beginning of Session 3*

Develop the following deliverables:

- **Statement of the Problem to be Solved for your project**
  (A summarizing sentence, plus an explanatory paragraph that communicates clearly the problem you are trying to solve. Do this for the work project, and then for the course project; You may include a diagram)
- **Stakeholder Analysis (A table listing key stakeholders and their type, perceived expectations, and priority. Also an explanation of your assumptions, process used, and what you think are your key results)**
- **User CONOPS (Context diagrams, examples of operational and support scenarios, description of operational and support environments)**
- **Stakeholder Expectations (list of initially identified S/H expectations)**
- **Stakeholder Requirements and an initial cut at Traceability**
  (List of S/H requirements, trace to their source, and explanation of your requirements rationale and process used)

Update the previously developed deliverables

* Each project team must decide which of, and the extent to which, each of these deliverables will be done

What's next?

- **Day 1**
  - **Review project team assignments from previous Session**
  - **Presentation and discussion of Session topic**
  - **Project team working sessions on the application of the Session topics to their projects**
  - **Day 1 wrap-up and Q&A**

- **Day 2**
  - **Review Day 1 topic and project working sessions**
  - **Presentation and discussion of Session topic (concluded)**
  - **Project team working sessions (concluded)**
  - **Check-in with class on the application of the Session topic to their projects**
  - **Session wrap-up: Q&A and homework due next Session**

Timing will be similar to session 1, and will be announced at beginning of each session

Session 2 - Stakeholder Needs

Lessons

1. **Program Overview Update**
2. **Introduction to Requirements**
3. **Stakeholders and their Expectations**
4. **Stakeholder Requirements**
5. **Special Topics (if time permits)**

Appendices*

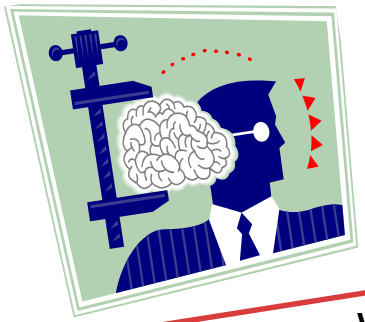
E. **Session 1 Summary**
F. **A Modeling Exercise**
G. **Other Appendices TBD**

*Appendices A, B, C, and D were contained in Session 1 of the course; remember that Appendix D has a complete course bibliography. The course will reference, but will not cover, the appendices

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		

All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future





What are your Questions before We Continue?



Systems Engineering Technical Short Course

(Session 2: Stakeholder Needs)

Lesson 2: Introduction to Requirements

Robert B. Wheeler, Instructor

Sessions 2 through 5 Agenda

- **Day 1**
  - Review project team assignments from previous Session
  - Presentation and discussion of Session topic
  - Project team working sessions on the application of the Session topics to their projects
  - Day 1 wrap-up and Q&A

- **Day 2**
  - Review Day 1 topic and project working sessions
  - Presentation and discussion of Session topic (concluded)
  - Project team working sessions (concluded)
  - Check-in with class on the application of the Session topic to their projects
  - Session wrap-up: Q&A and homework due next Session

Timing will be similar to session 1, and will be announced at beginning of each session

Session 2 - Stakeholder Needs

Lessons

1. Program Overview Update
2. **Introduction to Requirements**
3. Stakeholders and their Expectations
4. Stakeholder Requirements
5. Special Topics (if time permits)

Lesson Content

- **Why are we here?**
- Introduction to Requirements Engineering
- Three Process Models for Requirements Engineering

Appendices*

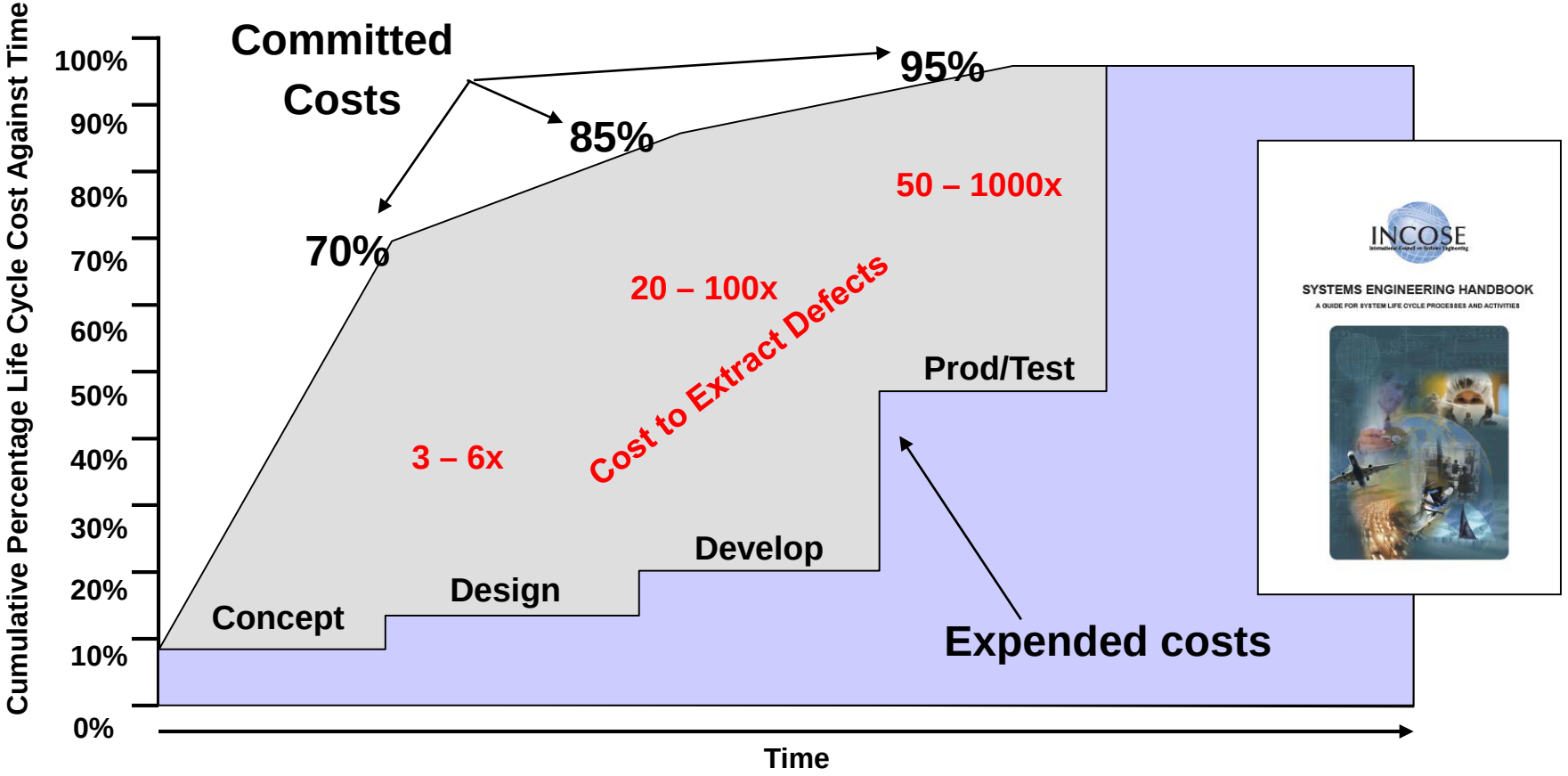
E. Session 1 Summary
F. A Modeling Exercise
G. Other Appendices TBD

*Appendices A, B, C, and D were contained in Session 1 of the course; remember that Appendix D has a complete course bibliography. The course will reference, but will not cover, the appendices

Why Are We Here? (1/3)

- Your expectations
  - See our exercise results from Session 1
- Add to these the instructor's expectations:
  - To have you all learn the material
  - To have you all succeed in the class
  - To meet your expectations. . . . . .
  - To have you be able to take the course concepts and make a positive difference to the performance of your jobs and the success of your projects.
- And, we are here because of the state of requirements engineering in the World today……..

Why Are We Here? (2/3)



Work and decisions made early in the project set the stage for project success or failure. In which of these stages are requirements developed?

Why Are We Here (3/3) – What Constitutes System Success?

- ✓Delights customers and stakeholders
- ✓Satisfies specified requirements
- ✓Solves the "problem-to-be-solved" in the target context
- ✓Does what it is supposed to do efficiently and effectively
- ✓Meets budget, schedule, and technical requirements
- ✓Is producible, testable, sustainable, supportable, maintainable, secure, operable, safe, legal, ……..

These are typical results of the Session 1 discussion exercise

Are any of these related to requirements quality

How Are We Doing?

- A survey of over 8000 projects found that the top three reasons that projects were delivered late, over budget, and with less functionality than desired revolved around requirements management practices: (1)
  - Lack of stakeholder input
  - Loss of control over changing requirements
  - Incomplete/misunderstood requirements
- 35% of 107 projects failed to capture the necessary requirements. This was caused by: (2)
  - Lack of time (61%)
  - Insufficient resources (22%)
  - Poor access (17%)

More recent results show improvement, but the most recent got worse!

1. The Standish Group, 1994, "Charting the Seas of Information Technology," Dennis, MA
2. Chatzoglou, P. and McCauley, L.A., 1995, "Journal of Computer Applications in Technology"

How Are You Doing in Your Environment? Exercise or Discussion (1/2)



Depending on time available, and the extent to which we did the Knowledge Café in Session 1, the instructor might modify this exercise

Exercise or Discussion (2/2)

- You may remember that we did a session 1 exercise to examine trends and issues in your system development environment.
- How might you answer the same questions specifically about your requirements development environment?

- The questions:
  1. What are the trends and issues in your requirements engineering environment?
  2. What improvements do you need to make in the performance of requirements engineering?
  3. What do you need from others (your customers, management , contractors, and subcontractors) to improve your organization's requirements engineering?
  4. What "tools" are you and your organization using to perform requirements engineering?

Instructor will explain the exercise process



Session 2 - Stakeholder Needs

Lessons

1. Program Overview Update
2. **Introduction to Requirements**
3. Stakeholders and their Expectations
4. Stakeholder Requirements
5. Special Topics (if time permits)

Lesson Content

- Why are we here?
- **Introduction to Requirements Engineering**
- Three Process Models for Requirements Engineering

Appendices*

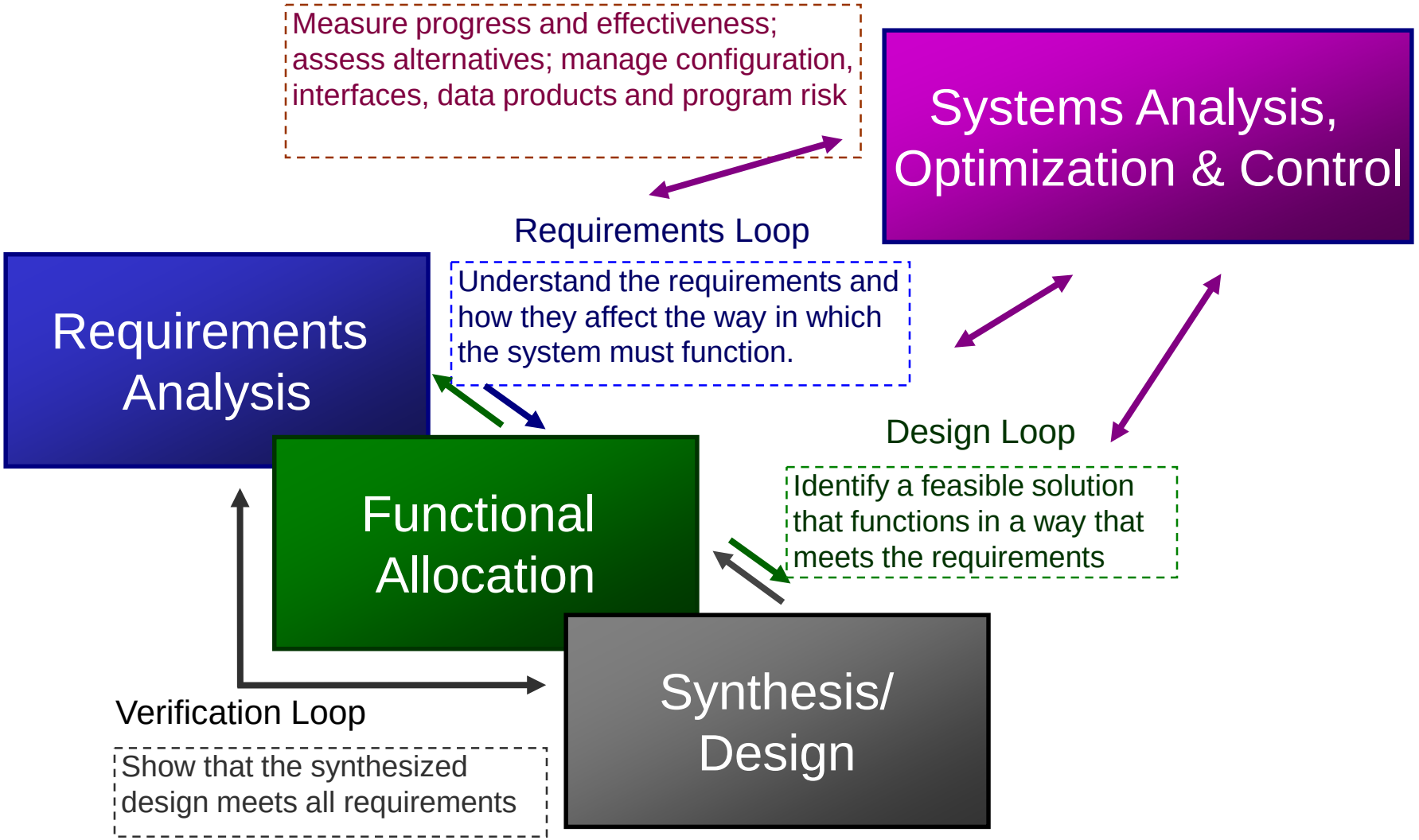
E. Session 1 Summary
F. A Modeling Exercise
G. Other Appendices TBD

*Appendices A, B, C, and D were contained in Session 1 of the course; remember that Appendix D has a complete course bibliography. The course will reference, but will not cover, the appendices

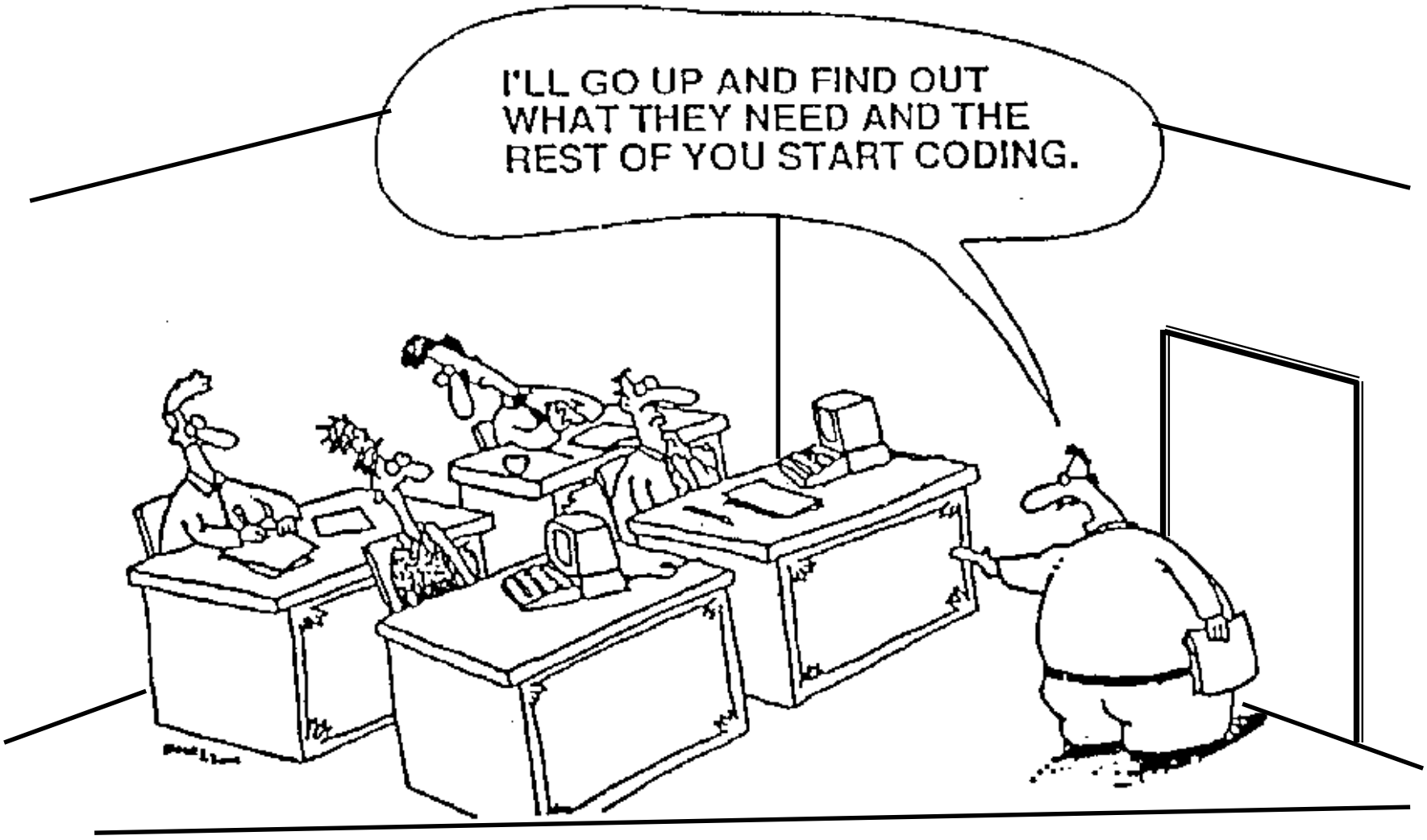
Class Philosophy on Requirements

- Sessions 2 and 3 focus on the processes and tools appropriate to requirements engineering.
- There are many approaches to requirements engineering and development. The “best” approach is context-dependent, and varies based on many factors such as project size, complexity, and risk and customer and internal organizational standards.
- Requirements engineering is often practiced in an environment of ambiguity; it is the SE’s job to resolve that ambiguity as much as possible using the processes and tools discussed in this course.
- A key tenet of this course is that the assumptions and rationale on which a requirement is based must be understood, recorded, and traced to that requirement.
- In like manner, you should make it a practice to record, and trace your results to, the assumptions, rationale, and processes used in your project work.

A Traditional View of the Systems Engineering Process Begins with Requirements Analysis

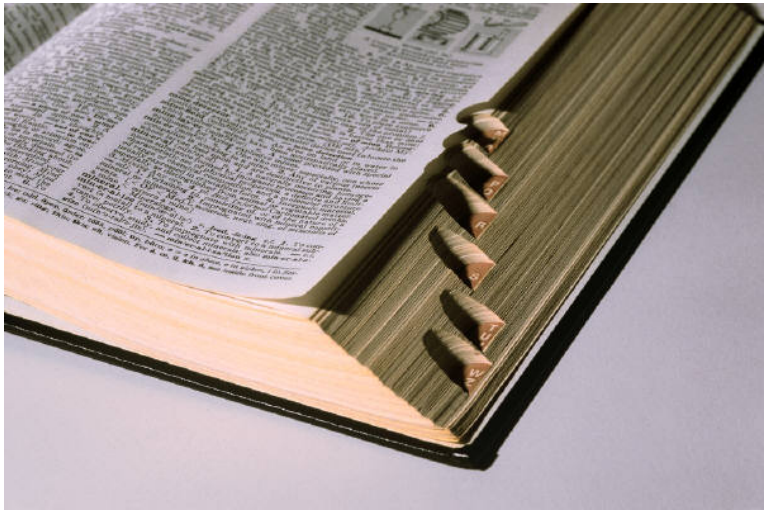


How the Other Folks Do It…….

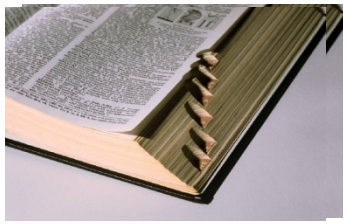


Key Definitions (1/8)

- The next several slides contain definitions of terms that are important to the performance of requirements engineering
- Definitions from multiple sources are shown.
- Other definitions may be added in subsequent lessons.



Key Definitions (2/8)



- **Stakeholders:**
  - **Buede:**
    - **Owner and/or bill payer, developer, producer or manufacturer, tester, deployer, trainer, operator, user, victim, maintainer, sustainer, product improver, and de-commissioner. Each stakeholder has a significantly different perspective of the system and the system's requirements**
  - **INCOSE:**
    - **A party having a right, share, or claim in a system or in it's possession of characteristics that meet that party's needs and expectations**
  - **Any person or organization that can affect, or be affected by, the results of a project (or the resulting system)**

Key Definitions (3/8)



- **Requirement:**
  - **INCOSE:**
    - **A statement that identifies a system, product, or process' characteristic or constraint, which is unambiguous, , clear, unique, consistent, stand-alone (not grouped), and verifiable, and is deemed necessary for stakeholder acceptability**
  - **Alexander:**
    - **(Narrow sense) Verifiable condition of a (class of) product or system, initially false, that must be made to become true by the development project. Traditionally written in a style such as "the system shall do XYZ", for inclusion in a commercial contract. Contrast with assumption; goal. Subclasses include: function; quality; constraint; interface**
    - **(Broad sense) Any requirement element; or as "the requirements", equivalent to a specification.**

Key Definitions (4/8)



- **Requirement (Continued):**
  - **A characteristic that a system or software item is required to possess in order to be acceptable to the acquirer**
  - **A binding statement . . . requirements are expressed using the word "shall"**
  - **See EIA/IEEE J-STD-016**
- **Requirements can be solution-independent, or solution-dependent**

Key Definitions (5/8)



- **Requirement element:**
  - **Alexander: A work product of the process of discovering requirements. Subclasses include: stakeholder analysis; goals; scenarios; rationale; assumptions; definitions; priorities; measurements; qualities; constraints.**
- **Stakeholder requirement:**
  - **Alexander: (Set of) requirements stating desired results in the world, before "optioneering" (i.e., without assuming any knowledge of the product's design). (*Concept- or solution-independent)***
- **Artifact:**
  - **Any representation of a system or of an aspect of a system, such as a document or model. Examples of requirements artifacts include specifications, context diagrams, traceability matrixes, CONOPS, or other documents and models (paraphrased from several sources)**

Key Definitions (6/8)



- **Goal (or expectation):**
  - **(Functional or nonfunctional) statement (a requirement element) of a stakeholder's desired target for a project. May not be completely realizable or verifiable. Nonfunctional goals can often be decomposed into functional goals (Alexander)**
  - **Requirement that has not yet received an output priority; even if strongly desired by stakeholders, it's inclusion in a (phase of) project is uncertain (Alexander)**
  - **An expectation**

- **Criterion (or selection criteria):**
  - **Alexandra: A measurable goal against which design options can be compared in trade-off analyses (*sometimes referred to as "selection criteria"*)**

Key Definitions (7/8)



Needs/Requirements/Capabilities

- Needs are the user's perspective
  - "A lack of something wanted or deemed necessary"
  - "A necessity arising from the circumstance of a situation or case"
- Requirements are engineering-speak on describing what a system must do, not how the system will do it.
  - Intermediate step between the user needs and the design
  - Defining requirements means to surface understand, and document every required capability to meet the user needs.
- A capability is the ability to achieve a desired effect under specified standards and conditions through combinations of means and ways to perform a set of tasks
  - Capability descriptions should be general enough so as not to prejudice decisions in favor of a particular means of implementation, but specific enough to evaluate alternative approaches to implement the capability

Key Definitions (8/8)

Term	More Definitions from *"Discovering Requirements"*
Design	The process of stating how a product is to be constructed, as opposed to requirement which states what and why
Key (requirement)	Approach to setting priorities in which only a small number of requirements (e.g., 7±2) may be designated key
Specification	Description of a product or a part of one, defining what is to be built. Naturally assumes some knowledge of chosen design. (Concept-dependent or solution-dependent)

Characteristics of Good Requirements (1/4)

Requirements shall be

- Necessary
- Implementation independent
- Clear and Concise
- Complete
- Consistent
- Achievable
- Traceable
- Verifiable
- Follow language use best practices

- **Necessary**
  - Only those that contribute and are needed to solve the problem to be solved in the target context
  - Not unnecessary specification of design
  - Not redundant, not covered elsewhere

Why is this important?

- **Implementation Independent**
  - Deals with the entity being specified as a black box
  - Says "what" should be done at each level, not "how"

Does the customer have the right to specify design details?

Adapted from INCOSE Systems Engineering Handbook, v3.2, pages 76-77

Characteristics of Good Requirements (2/4)

Requirements shall be

- Necessary
- Implementation independent
- **Clear and Concise**
- **Complete**
- Consistent
- Achievable
- Traceable
- Verifiable
- Follow language use best practices

- **Clear and Concise**
  - **Must convey what is to be done to next the next level of development**
  - **One and only one interpretation**
  - **Terms defined/ glossary used**
  - **One "shall" per requirement (no "ands", "ors", commas)**

Are your requirements clear and concise?

- **Complete**
  - **Must not need further amplification at the level for which it is intended**
  - **Provides sufficient capability or characteristics**

Is a requirement complete if it needs amplification at the next lower level?

Adapted from INCOSE Systems Engineering Handbook, v3.2, pages 76-77

Characteristics of Good Requirements (3/4)

Requirements shall be

- Necessary
- Implementation independent
- Clear and Concise
- Complete
- **Consistent**
- **Achievable**
- **Traceable**
- Verifiable
- Follow language use best practices

- **Consistent**
  - **Must not conflict with other requirements**
  - **Includes consistency with applicable government standards, interface standards, and other requirements**
- **Achievable**
  - **Must be possible within the available technology, time, and cost**

How can you be sure that a requirement is achievable?

- **Traceable**
  - **Must trace to higher level requirements and, ultimately, to the problem to be solved**

How do you achieve requirements traceability? What tools do you use?

Adapted from INCOSE Systems Engineering Handbook, v3.2, pages 76-77

Characteristics of Good Requirements (4/4)

Requirements shall be

- Necessary
- Implementation independent
- Clear and Concise
- Complete
- Consistent
- Achievable
- Traceable
- **Verifiable**
- **Follow language use best practices**

- **Verifiable ***
  - **Each specification requirement must be verified at some level**
  - **Verification is by test, inspection, demonstration, or analysis**
  - **No requirement is complete without identifying the associated verification requirement**

When, and to what level, must the verification requirements be defined?

- **Follow language use best practice**
  - **More later**

***Concept-independent requirements must also be able to be validated**

Adapted from INCOSE Systems Engineering Handbook, v3.2, pages 76-77; see also Buede Table 6.3

Roles of Requirements

- Defines the problem/performance → System View
- Constitutes agreement between user and developer → Contract
- Metrics for design alternatives → Trade Space
- Checklist for design decisions → Audit
- Forms the basis of testing and approval → Acceptance Criteria

Levels of Requirements

▪ User Needs	▪ My house must be comfortably cool, even on the hottest days
▪ Operational Requirement	▪ The system shall keep the house cooled down to 74 degrees F throughout the year
▪ System Requirement	▪ The system shall be capable of cooling a 2600 sq ft house down to 74 degrees F in a climatic environment of up to +120 deg F
▪ Subsystem Requirement	▪ The Chiller subsystem shall dissipate up to 22,000 BTU per hour
▪ Component Requirement	▪ The Compressor motor shall develop at least 2 HP.

Good Requirement Examples

- The system shall be capable of transporting a minimum of 20 passengers.
- The system shall be capable of a maximum speed of at least 200 km/hr.
- The system shall determine its location to within an accuracy of 10 meters.
- The system shall detect failures of its components and display them to the operator.
- The system shall operate in ambient air temperatures between -40 deg C and +51 deg C.
- The system shall have a mean time between failures of no less than 200 hours.

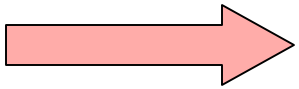
Poor Requirement Examples

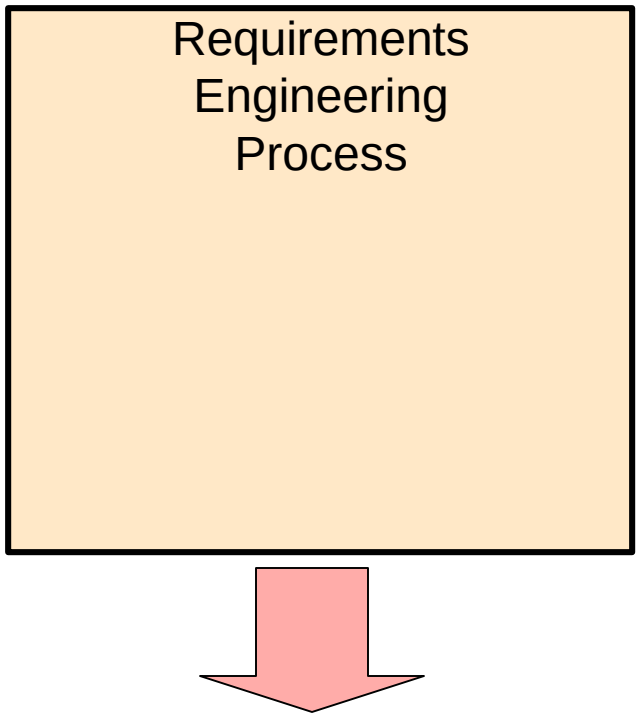
- **The land transport system will be capable of travelling up to 1600 km/hr.**
  - ***"Will" is a statement of intent. "Shall" is a binding requirement per standard practice and in legal contracts.***
  - ***The required velocity performance is beyond current technology and will turn the development into a research program.***
- **The system shall require minimal maintenance.**
  - ***"Minimal" is subjective. How minimal is good enough?***
- **The system shall be compatible with external power sources.**
  - ***Ambiguous statement. Are there specific power sources that the system must be compatible with or does this mean all possible sources?***
- **The system shall not cause harm to operators.**
  - ***The phrase "shall not" is a very poor term to use in a requirement as it is often difficult to prove a negative. What test could you create for this?***
  - ***Also, "cause harm" is a subjective term.***

Requirements Engineering

Inputs and Outputs

- Existing System Requirements
- Stakeholder Needs
  - Customer(s)
  - Users
  - Maintainers
  - Security
  - Management
- Organizational Standards
- Regulations
- Domain Information
- Type of Software System





- Agreed-Upon Requirements
- System Specifications
- System Models

Discussion: What comes First ? (1/3)

- Requirements?
- Design?
- Architecture?
- Expectations?





Exercise: What is the Status of Requirements for your Projects?



The instructor will provide directions and timing

Session 2 - Stakeholder Needs

Lessons

1. Program Overview Update
2. **Introduction to Requirements**
3. Stakeholders and their Expectations
4. Stakeholder Requirements
5. Special Topics (if time permits)

Lesson Content

- Why are we here?
- Introduction to Requirements Engineering
- **Three Process Models for Requirements Engineering**

Appendices*

E. Session 1 Summary
F. A Modeling Exercise
G. Other Appendices TBD

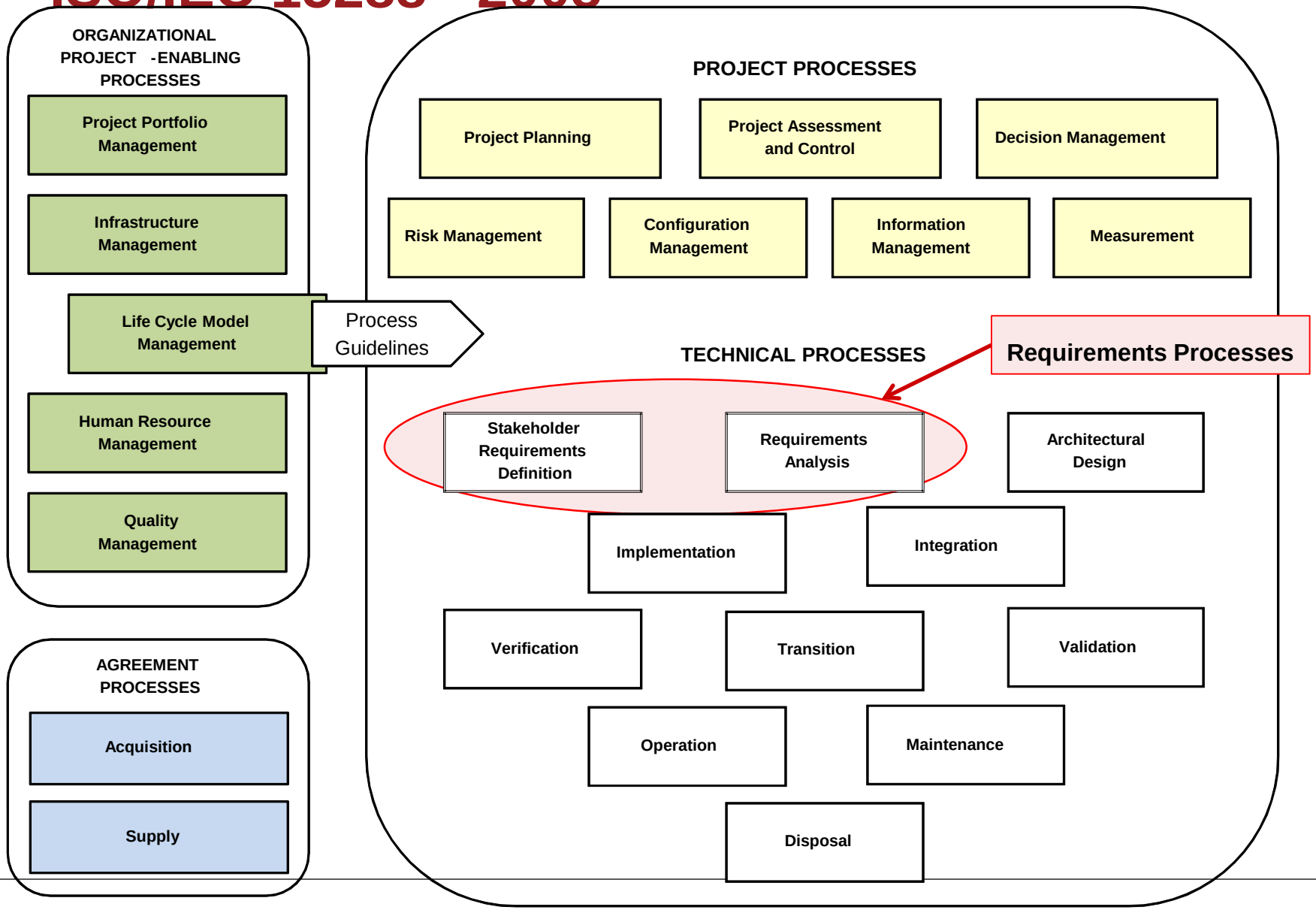
*Appendices A, B, C, and D were contained in Session 1 of the course; remember that Appendix D has a complete course bibliography. The course will reference, but will not cover, the appendices

Three Process Models for Requirements Engineering

- Many process models exist
- This course uses three models to understand and communicate requirements engineering
  - INCOSE Systems Engineering Handbook
    - Based on ISO/IEC 15288
  - The Vee Model
    - Per the course text by Buede
    - Also discussed in the INCOSE Systems Engineering Handbook
  - The Discovery Model
    - Per the course text by Alexander

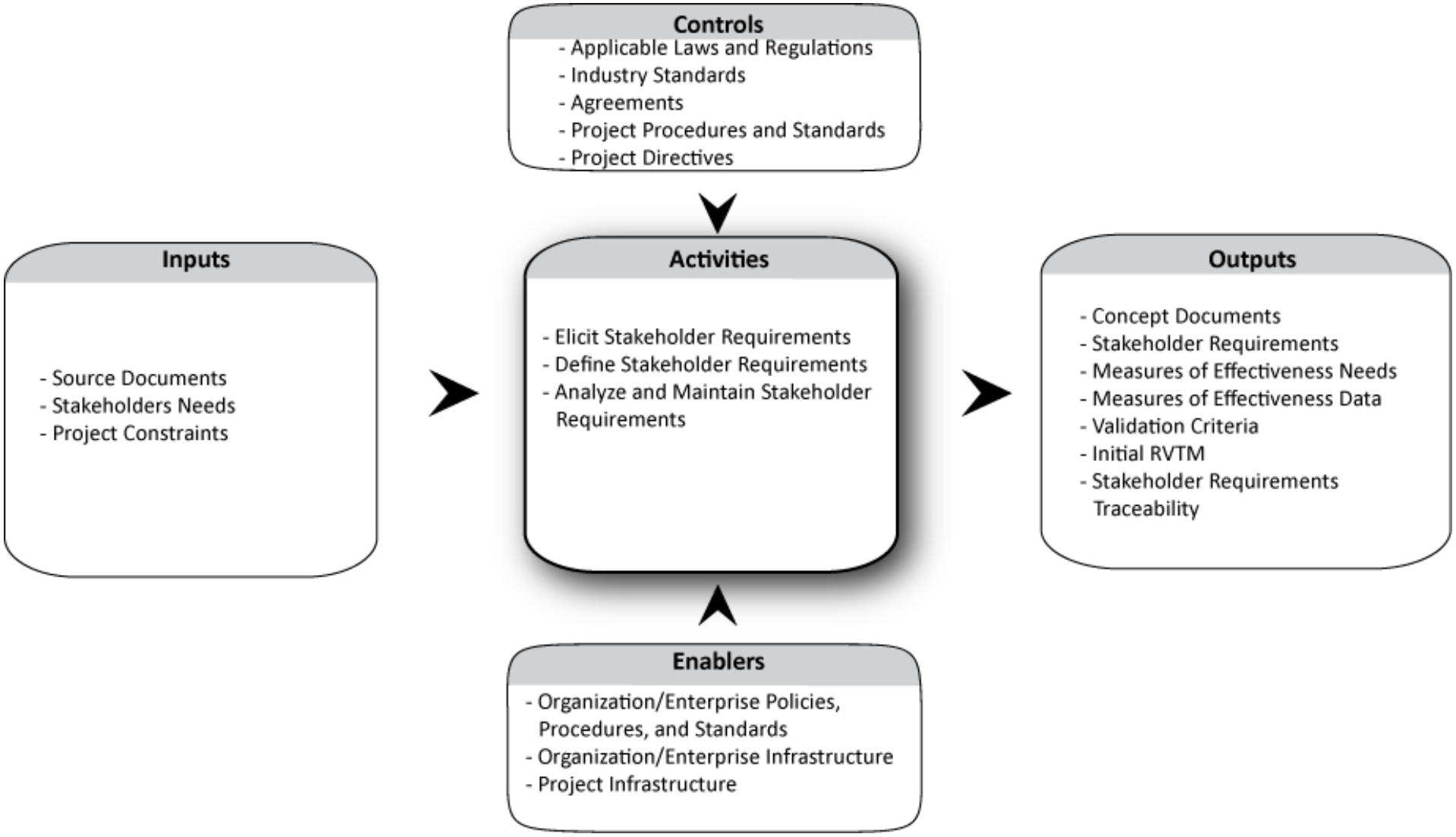
At the end of this lesson a composite of these models will be established as the context model for session 2 & 3 of this course

INCOSE SE Process Model based on ISO/IEC 15288 - 2008



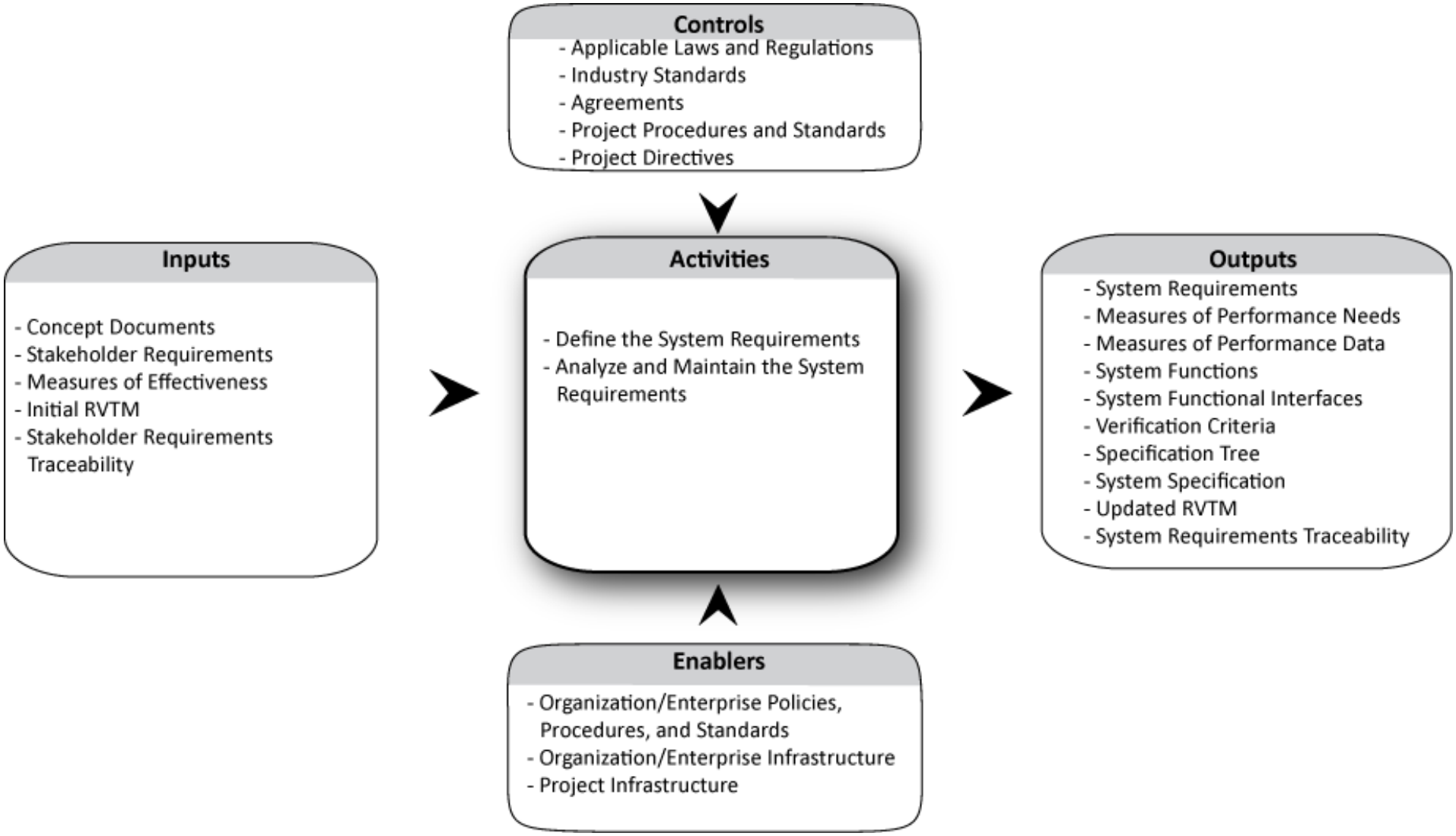
Source: INCOSE Systems Engineering Handbook, version 3.2.2, Figure 1-1

Stakeholder Requirements Definition Process Context Diagram (INCOSE)



Source: INCOSE Systems Engineering Handbook, version 3.2.2, Figure 4-2

Requirements Analysis
Process Context Diagram (INCOSE)



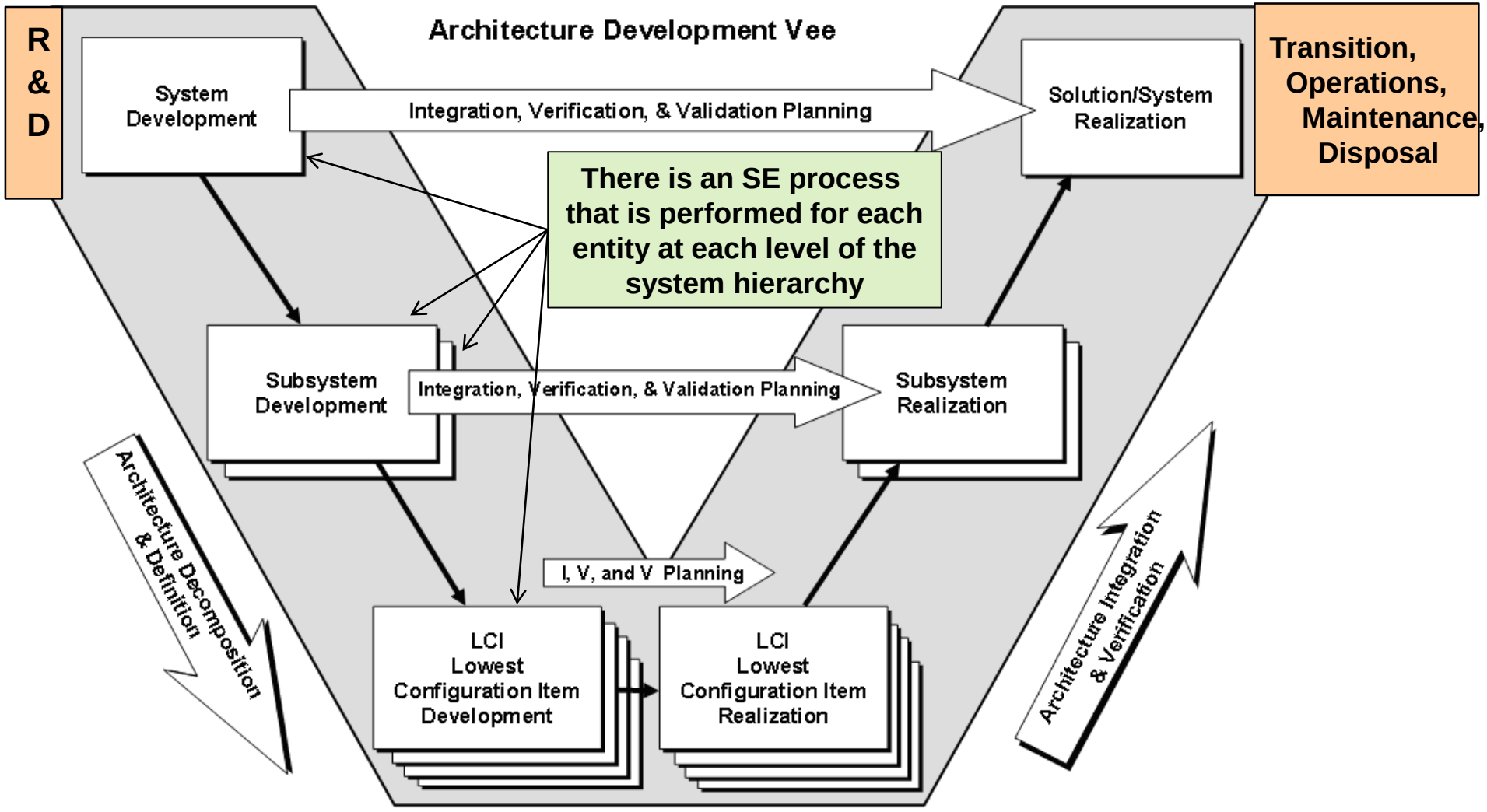
Source: INCOSE Systems Engineering Handbook, version 3.2.2, Figure 4-4

Using the Vee to Model the Requirements Engineering Process

- The Vee is particularly valuable in illustrating the systems engineering process
- The Vee model consists of three parts:
  - The Architectural Vee that illustrates the system hierarchy and the relationship between the levels in the hierarchy
  - The Systems Design Process used for every entity at every level of the system hierarchy (Left side of the Vee)
  - The Systems Integration Process used for every entity at every level of the system hierarchy (right side of the Vee)

Source: Although the slides illustrating the Vee have been adapted from INCOSE and Buede, much of the defining work on the Vee was done by Forsberg and Mooz

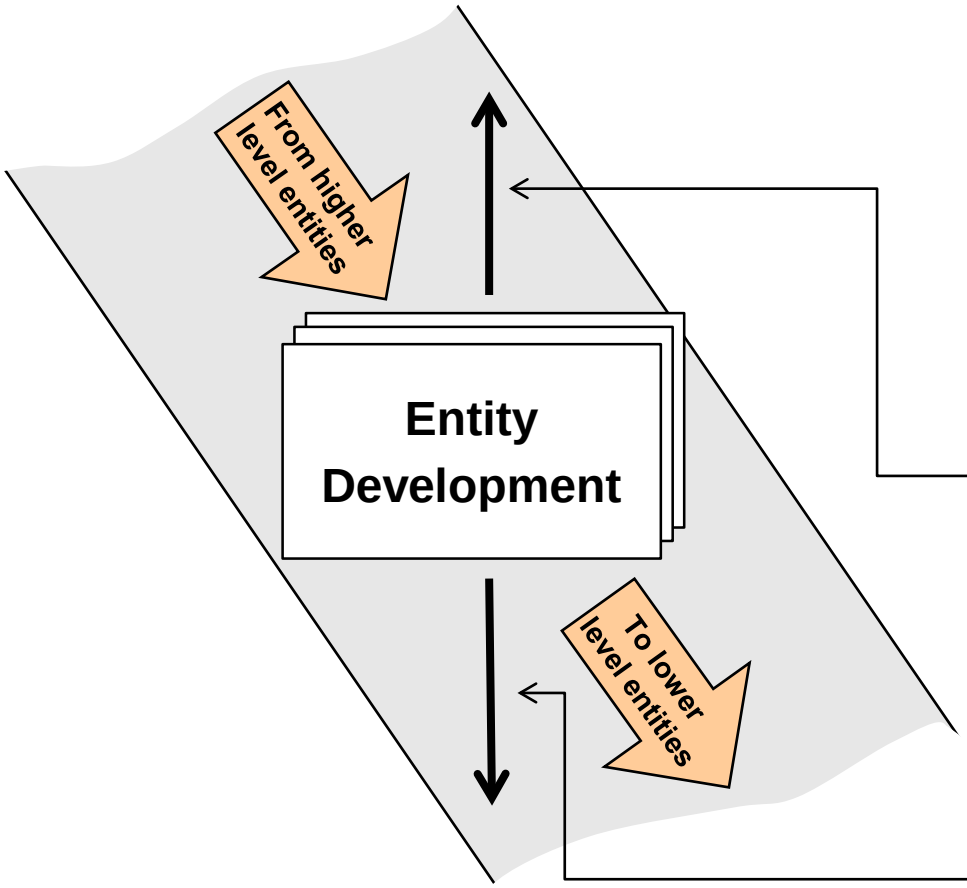
Architectural Vee Process Model



The next two slides show the two processes used on the left side and on the right side of the Vee respectively

Adapted from INCOSE SE Handbook, Version 3.2.2, Figure 3-4

The Vee Model Incorporates Risk Management into the SE Process



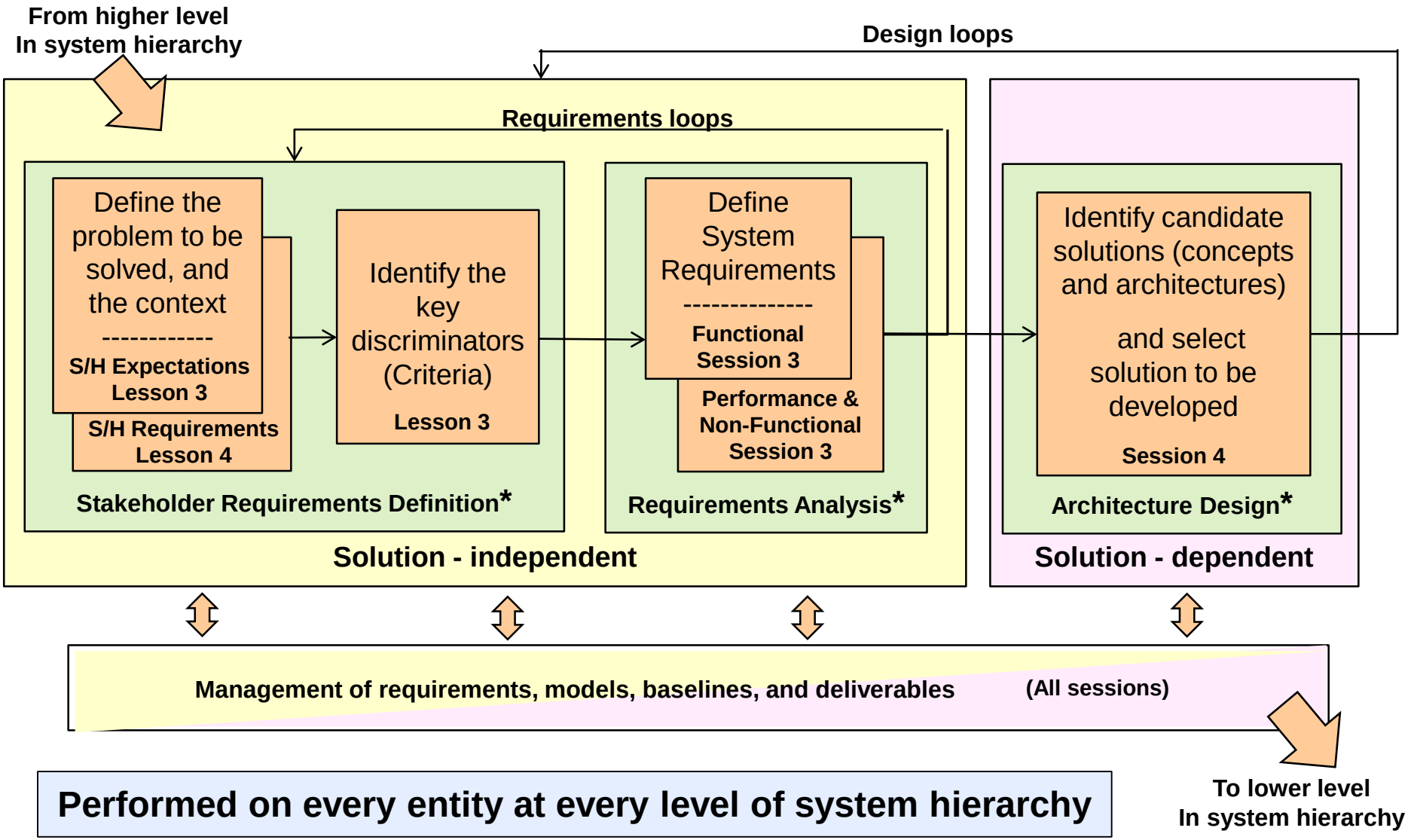
- Entity development progresses down the "core" of the left side of the Vee, from system to subsystem to lower level entities
- Information gained while developing a lower level entity might require revisiting an upper level entity

  This is called "off core" in-process validation
- To manage risk, It may be necessary to work at a lower level prior to completing the higher level development

  This is called "off core" risk management

Can you think of examples of "off core" activities?

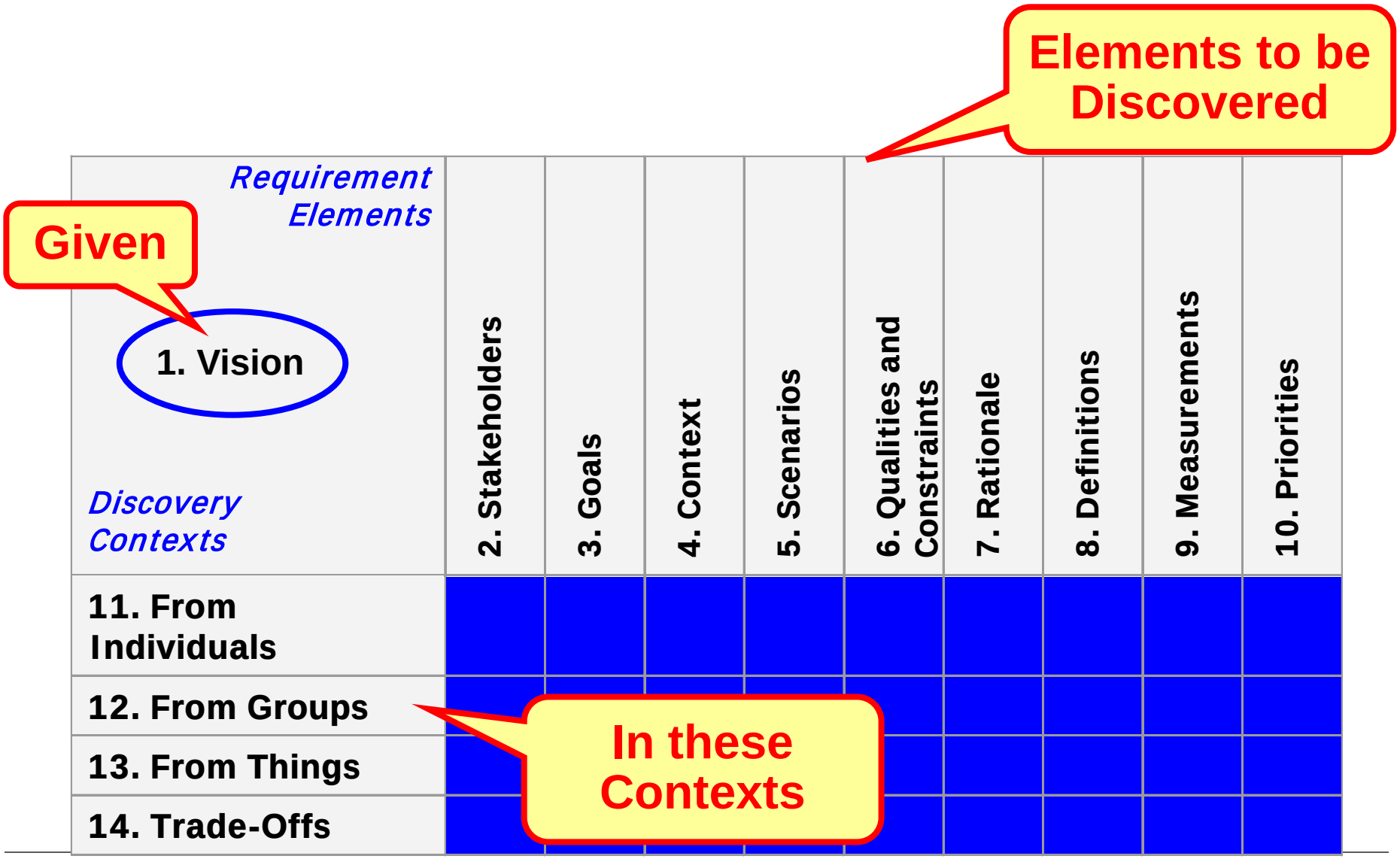
The Requirements Engineering Process Context for this Course (Left Side of Vee)

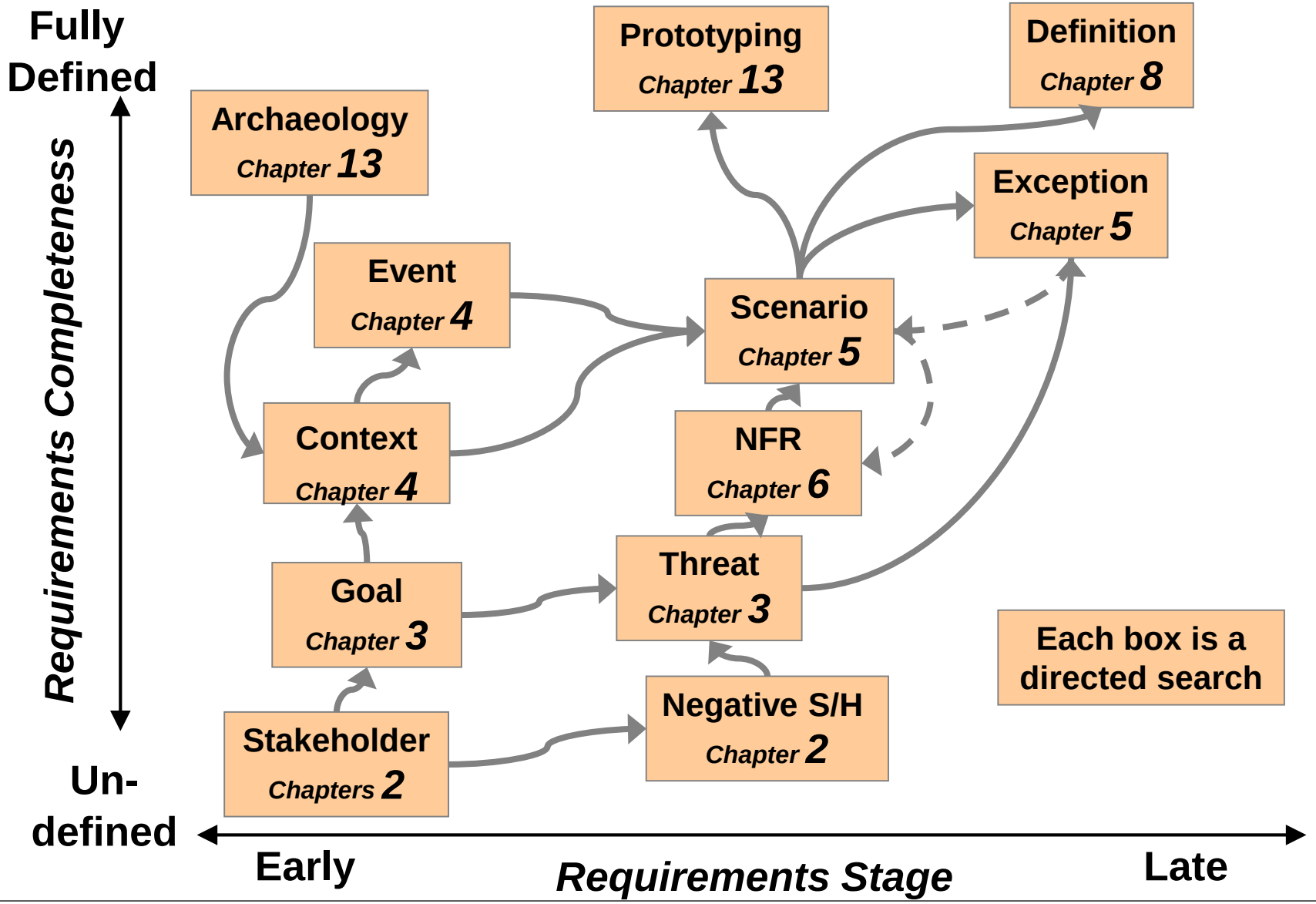


Performed on every entity at every level of system hierarchy

* Indicates INCOSE SE Handbook process name

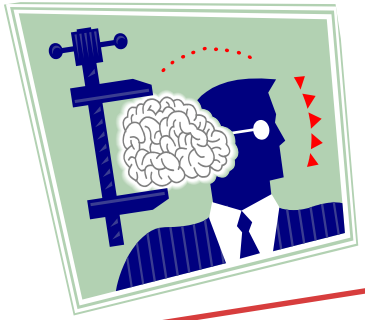
Using "Discovery" to Model the Requirements Engineering Process




Fully Defined
Requirements Completeness
Un-defined
Archaeology Chapter 13
Prototyping Chapter 13
Definition Chapter 8
Exception Chapter 5
Event Chapter 4
Scenario Chapter 5
Context Chapter 4
NFR Chapter 6
Goal Chapter 3
Threat Chapter 3
Each box is a directed search
Stakeholder Chapters 2
Negative S/H Chapter 2
Early
Requirements Stage
Late

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?

Back-up Slides

Discussion: What comes First ? (1/3)

- Requirements?
- Design?
- Architecture?
- Expectations?





What Comes First? (2/3)

- Before we attempt to design the system, we need to understand the "problem to be solved" and the "context" in which it is to be solved
- This leads to a set of "solution independent" requirements:
  - Requirements that must be satisfied regardless of what solution is ultimately selected
  - Normally captured in models and "requirements documents"
  - Must be validated through the system lifecycle
  - Help ensure the right solution is ultimately built
- As these solution independent requirements evolve, we begin to look for multiple solution concepts that will solve the problem in the proper context:
- This leads to a definition of each solution concept sufficient to "select" which best satisfies the solution-independent requirements

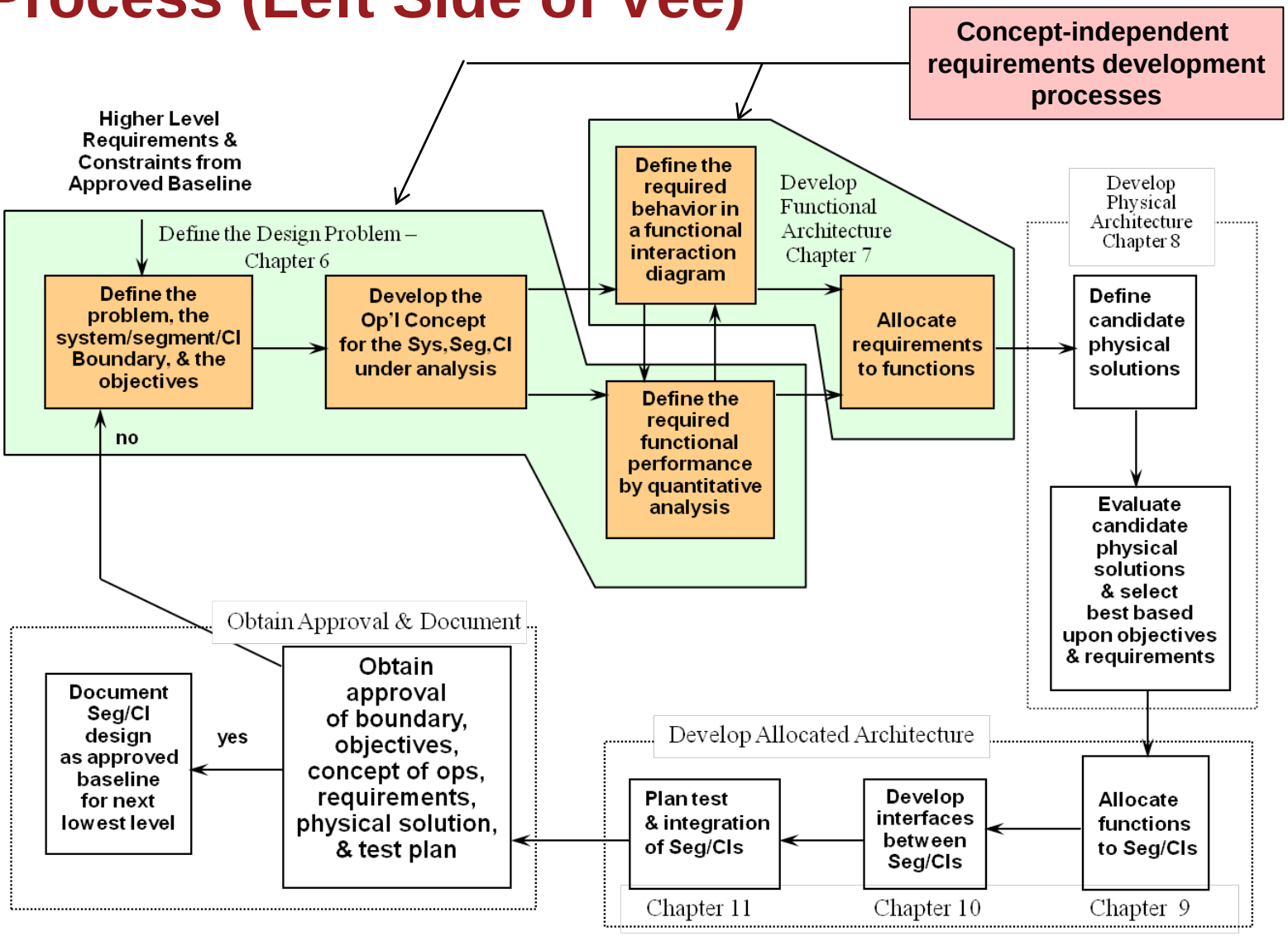
What Comes First? (3/3)

- The selected solution is defined via a set of models and design artifacts, including specifications. These specifications include the "solution-dependent" requirements that must be ultimately satisfied:
  - Requirements that must be satisfied by the selected solution
  - Normally captured in models and "specifications"
  - Must be verified through tests, demonstrations, inspections and analyses.
  - Help ensure the solution is built right
- The solution-dependent requirements must be traceable back to the solution-independent requirements, and the "problem to be solved"
- What comes first is less important than congruency of the artifacts and the consistency of the evolving system model

The requirements engineering process must find be performed iteratively with all the other Systems Engineering Processes, and recursively through the system hierarchy

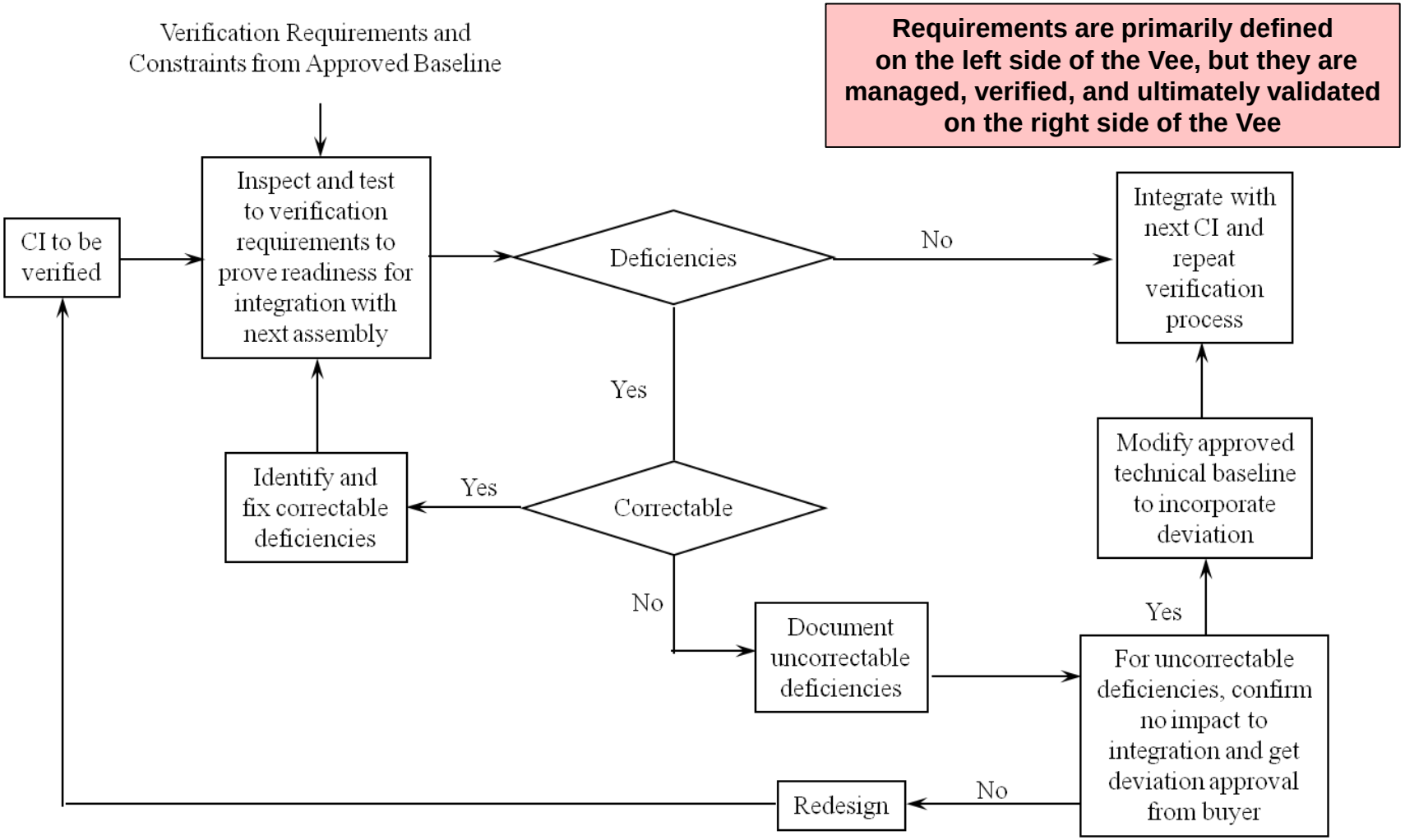
Alexander, Chapter 14, discusses this eternal question in more depth

The Systems Engineering Design* Process (Left Side of Vee)



Source: Adapted from Buede text, Figure 1.20

The Systems Engineering Integration Process (Right Side of Vee)



Source: Adapted from Buede text, Figure 1.21



Systems Engineering Technical Short Course

(Session 2: Stakeholder Needs)

Lesson 3: Stakeholders and their Expectations

Robert B. Wheeler, Instructor

Sessions 2 through 5 Agenda

- **Day 1**
  - **Review project team assignments from previous Session**
  - **Presentation and discussion of Session topic**
  - **Project team working sessions on the application of the Session topics to their projects**
  - **Day 1 wrap-up and Q&A**

- **Day 2**
  - **Review Day 1 topic and project working sessions**
  - **Presentation and discussion of Session topic (concluded)**
  - **Project team working sessions (concluded)**
  - **Check-in with class on the application of the Session topic to their projects**
  - **Session wrap-up: Q&A and homework due next Session**

Timing will be similar to session 1, and will be announced at beginning of each session

Session 2 – Stakeholder Needs

Lessons

1. Program Overview Update
2. Introduction to Requirements
3. **Stakeholders and their Expectations**
4. Stakeholder Requirements
5. Special Topics (if time permits)

Lesson Content



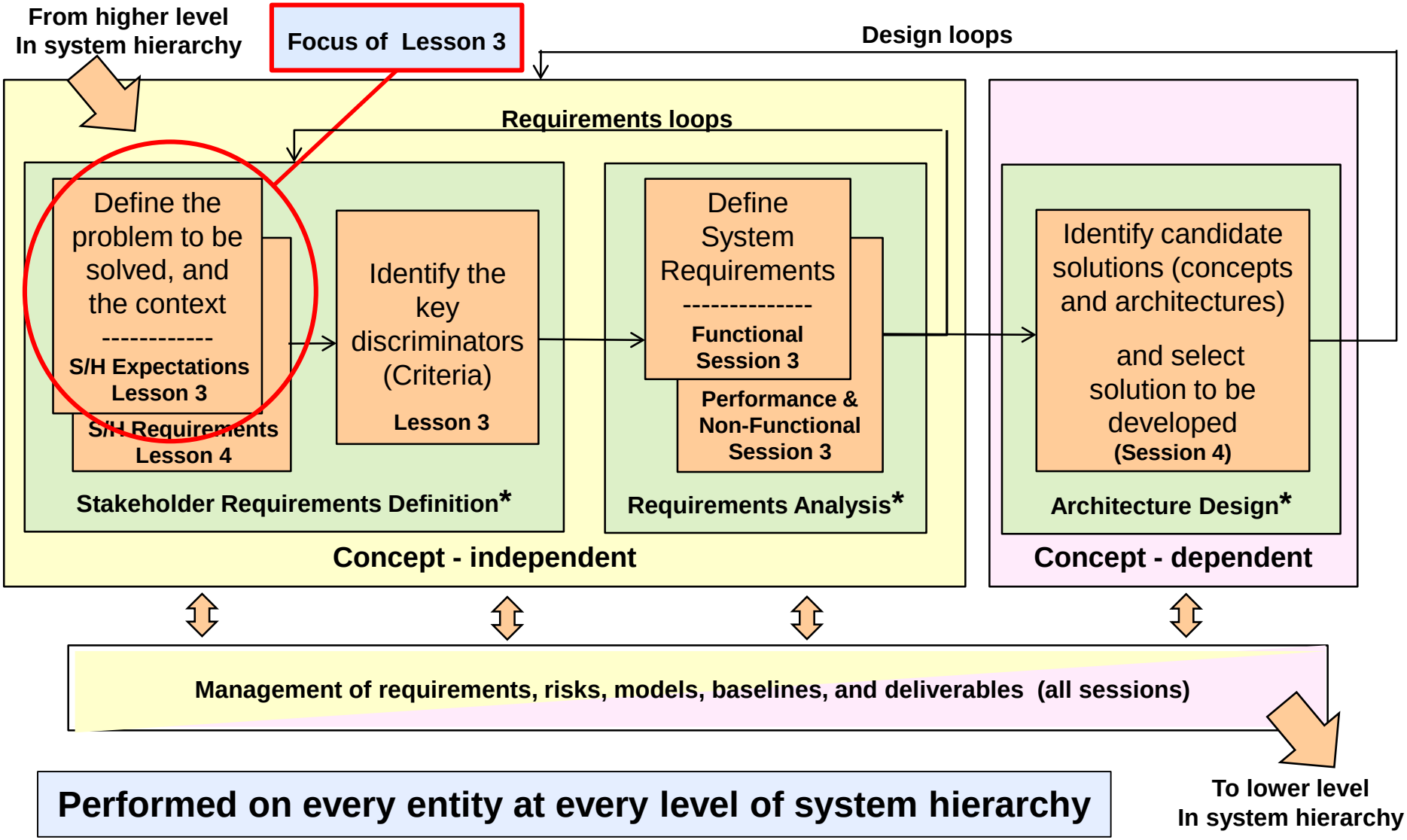
- **The problem to be solved**
- Identifying stakeholders
- Eliciting stakeholders expectations and involvement

Appendices*

E. Session 1 Summary
F. A Modeling Exercise
G. Other Appendices TBD

*Appendices A, B, C, and D were contained in Session 1 of the course; remember that Appendix D has a complete course bibliography. The course will reference, but will not cover, the appendices

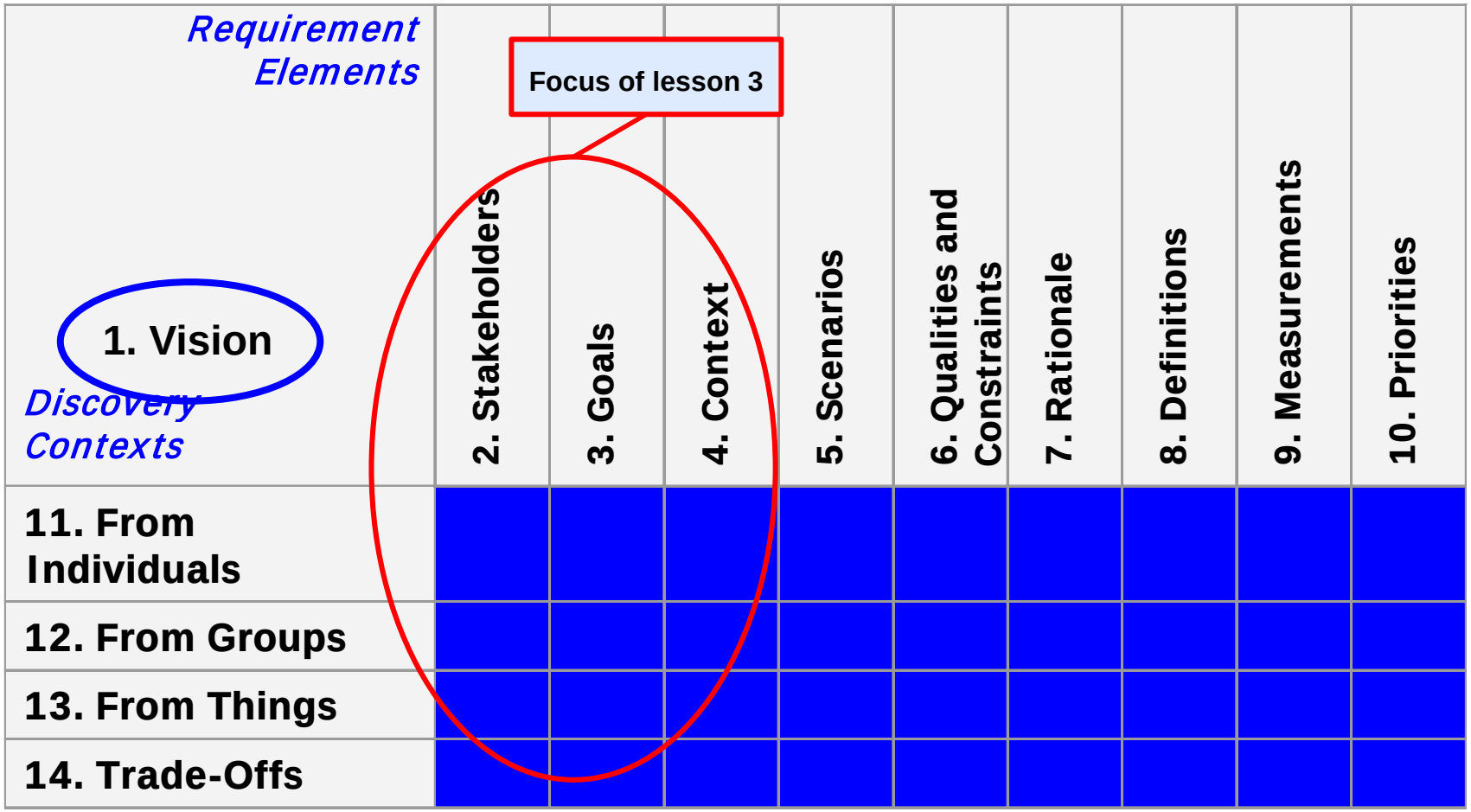
The Requirements Engineering Process Context for this Course (Left Side of Vee)



Performed on every entity at every level of system hierarchy

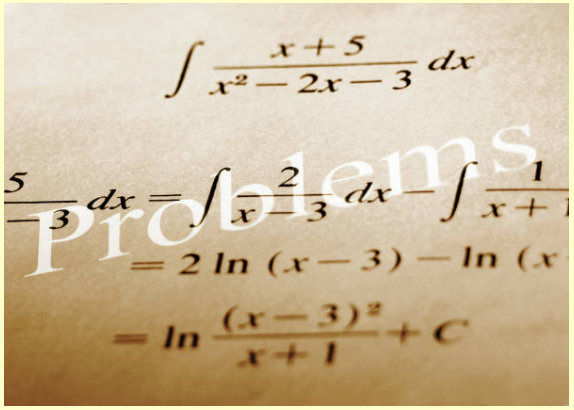
* Indicates INCOSE SE Handbook process name

Using "Discovery" to Model the Requirements Engineering Process



The Origin of all Requirements (1/2)

The Problem to be Solved





The Context

The Origin of all Requirements (2/2)

Projects come from stakeholders (usually customers or users) with problems to solve in a particular context

1. **What is the problem to be solved?**
   - **The fundamental reason for your project**
   - **What the system must provide**
   - **Understood through user involvement**
2. **What is the target context?**
   - **The environment in which the problem must be solved and the system will exist**
   - **Understood through stakeholder involvement**
3. **Who are the stakeholders?**

Requirements Engineering starts with these three questions;

Let's look at them next

The Problem to be Solved

- What is the problem to be solved?
  - The fundamental reason for your project
  - The key top level need for the system to be provided

- Typically called
  - "top-level requirements"
  - "operational need",
  - "mission need"
  - "business objective"



- User involvement is key to understanding the problem to be solved

The requirements engineering process must ensure that everything we do, especially requirements we define, directly supports the problem to be solved

The Nature of The Problem to be Solved

- **Existing systems problems:**
  - **Performance shortfalls**
  - **Obsolescence**
  - **No longer effective or efficient**
  - **Unmitigated risk is too high**
- **New or changing**
  - **Mission**
  - **Threat**
  - **Intelligence need**
  - **Operational or support environment**
- **Opportunity**
  - **Emerging technology**
  - **Opportunity for innovation**



What is the nature of this problem?

Typically, the Problem to be Solved is recorded in a User Needs Statement

The Problem to be Solved - Some Examples

- Become more market-driven in everything we do
- Improve I/T system customer satisfaction while reducing operating costs
- Prevent damage and injury to personnel from IEDs
- Reduce risk of loss in conducting global cross currency exchange transactions
- Prevent terrorists attack using commercial airliners
- …before this decade is out, of landing a man on the moon and returning him safely to Earth”

Can you categorize these examples per the list on the previous chart?

Do you know the “problem to be solved” on your project

The Problem Context

- **What is the context in which the problem is to be solved:**
  - **What is the nature of the environment in which the problem exists?**
  - **What is the rationale for the project?**
  - **Will our solution be part of a super system?**
  - **With what external systems must our system interface?**
  - **What environmental and other constraints exist**
  - **What are the results of any earlier relevant studies?**
  - **What are the environments in which the system will be:**
    - **Operated**
    - **Maintained**
    - **Supported**
    - **Secured**
    - **Developed**
    - **Produced**
    - **Verified & Validated**
    - **Trained**
    - **Retired**
    - **Are there others?**
  - **Answers the "who, what, when, where, why, and how" questions**



Typically, the Problem Context is recorded in a User CONOPS

Context Example

Problem: Reduce risk of loss in conducting global cross currency exchange transactions

What	**Specific currencies ($, £, €, ¥)**
When	**Operate 24x7x365, initial operating capability in 2 years**
Where	**London - main office; Portsmouth – test; Brussels - development; Major cities throughout the World - offices**
Who	**International banking community**
Why	**To reduce the risk of loss due to failure to settle**
How	**Must use specified legacy systems**

Initial answers to the who, what, when, where, why, and who questions suggest more detailed questions, and lead naturally to the full context

Discussion:

For the problem to be solved and the problem context:

- **Who is responsible for defining it?**
- **Who typically defines it on your projects?**
- **Where is it recorded**



Scenario: You have just been assigned as the lead SE for a new project to provide a new system to the customer

Situation A: the problem to be solved and the context are poorly defined.

What should you and your team do?

Situation B: the problem to be solved and the context are summarized at a high level, but a preferred solution is sufficiently well-defined for your team to provide it.

What should you and your team do?

Session 3 – Stakeholder Needs

Lessons

1. Program Overview Update
2. Introduction to Requirements
3. **Stakeholders and their Expectations**
4. Stakeholder Requirements
5. Special Topics (if time permits)

Lesson Content

- The problem to be solved?
- **Identifying stakeholders**
- Eliciting stakeholders expectations and involvement

Appendices*

E. Session 1 Summary
F. A Modeling Exercise
G. Other Appendices TBD

*Appendices A, B, C, and D were contained in Session 1 of the course; remember that Appendix D has a complete course bibliography. The course will reference, but will not cover, the appendices

Stakeholders and Requirements Engineering



- Key stakeholders (the sponsor and others) bring the problem and the context to the project
  - User involvement is key to understanding the problem to be solved
- Requirements Engineering is the process that transforms the problem and context into a set of achievable requirements
  - Stakeholder involvement is essential to this process
- All stakeholders must be identified and engaged according to their importance to (their "stake" in) the project



In the beginning there are only expectations*; requirements engineering transforms these expectations into a set of agreed engineering requirements

* Alexander calls these expectations "goals"

Typical Stakeholders



Sponsor
- Executive
- HRM
- Functional
- Project
- Finance

Customer
- Buyer
- SE/Technical
- Contracting Office
- COTR

Management
- SE/Specialties
- Engineers
- Programmers
- Production

Users
- Operators
- Maintainers
- Supporters
- Administrators

Providers

Other
- Legacy systems
- Interfacing systems
- Regulators
- Standards
- Other

A stakeholder is any person or organization that can affect, or can be affected by, the results of the project

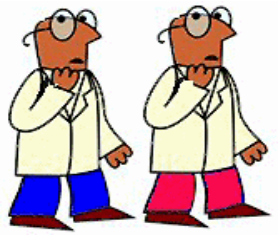
Common Stakeholder Roles



Political Beneficiaries



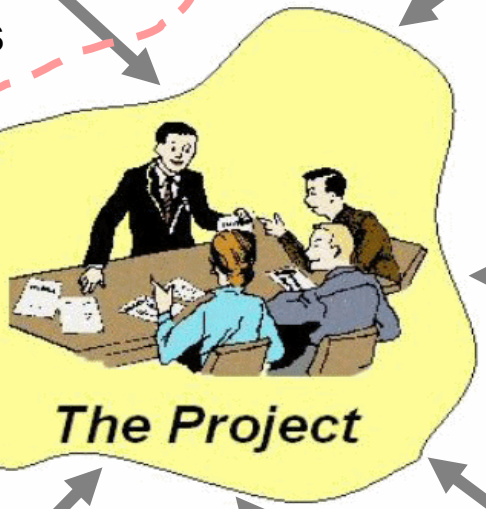
Financial Beneficiaries

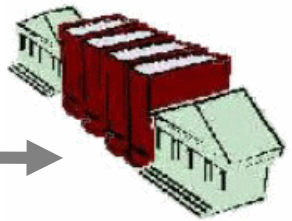


Experts



Functional Beneficiaries





Regulators

Beneficiaries – it's for them!



Social Beneficiaries



Operational Stakeholders



Interfacing Stakeholders

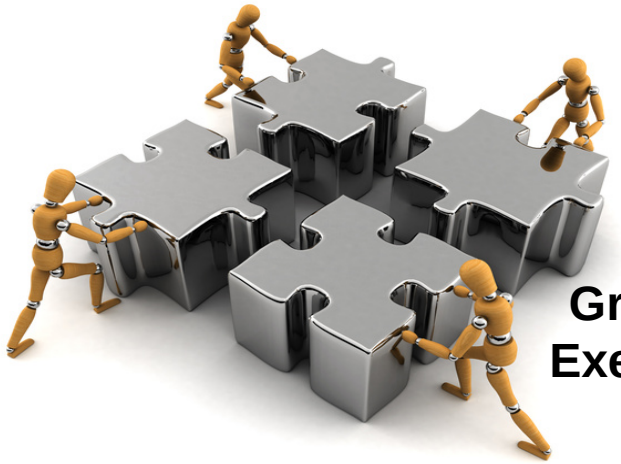


Negative Stakeholders

Exercise: Five Minute Manager or Knowledge Café”

Form three groups; each group will brainstorm answers to one of the questions below. A member of each group will present results to the class

- **How do you identify your stakeholders?**
- **How do you involve your stakeholders?**
- **What problems do you have getting stakeholders involved?**



Group Exercise

Typical Stakeholders - Sponsor or Champion

Definitions:

- **Sponsor - provides development funding for the project**
- **Champion - provides political support for the system**
- **These roles are sometimes combined, and often confused.**

Importance

- **These are your most important stakeholders**
- **Without these stakeholders your project is doomed to failure**

Who is your project champion?

Who is your project sponsor?





Typical Stakeholders - Users and Operators

Type or Role	Work Performed	Example
Operator	Interacts with system to deliver results	Vehicle driver
Maintenance operator	Interacts with system to perform planned and unplanned maintenance	Vehicle mechanic
Support operator	Provides help and co-ordination to keep other operators productive	Parts supplier, Trainer
Functional beneficiary users	Uses the results of the system, product or service to receive some benefit	Passengers, Fleet operator

Typical Stakeholders - Providers

- Providers are individuals and organizations that develop, test, manufacture, and deliver the product or service
- Includes development team, provider management, subcontractors, and other provider organization members
- Examples include:
  - Systems engineering
  - Engineering specialties
  - Design engineering
  - Testers
  - Manufacturing, production
  - Facilities
  - Requisite domain experts
  - Can you think of others?



Integrated Product Teams (IPTs) and Integrated Product and Process Development (IPPD) help ensure involvement of these stakeholders

Typical Stakeholders - Beneficiaries

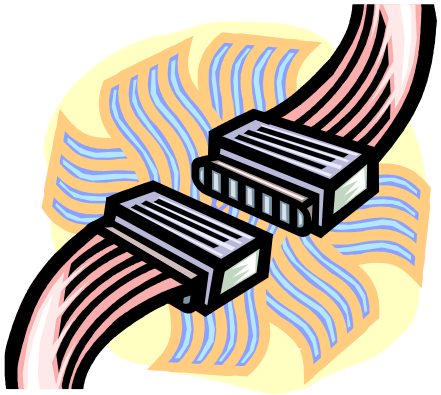
- **Beneficiaries benefit is some way from the project, system, product, or service**
- **Beneficiaries include:**
  - **Functional – typically users**
  - **Financial:**
    - **Customers**
    - **Shareholders**
    - **Communities**
  - **Political**
  - **Social**



Typical Stakeholders - Interfacers and Legacy System Owners

Managers and members of the organizations that own:

- **Systems that will interface with your system, product, or service, including:**
  - **existing systems (these are also legacy systems)**
  - **systems being developed in parallel with yours**
- **Legacy systems that will:**
  - **operate in conjunction with yours (these are also interfacing systems)**
  - **become part of your system**
  - **be replaced by your system**
  - **Offer insight into the development of your system**



Which of these stakeholders will be least cooperative

Which of these offer the biggest challenge to systems engineering?

Typical Stakeholders - Regulators and Standard Bearers

Types	Examples
International standards	ISO 15288,….
National standards	EPA standards
Industry best practices	CMMI, 3-Sigma, Baldridge,….
Professional societies	PMI, IEEE, INCOSE, ASTM,
Organizational standards (Government)	FAR, DODAF, NASA NPR7123, FAA, many others
Organizational standards (Commercial)	Most commercial companies have their own standards
Project standards	WPI template for this course

The project team must identify which standards apply, and specifically how each standard impacts the project.

Typical Stakeholders - Negative Stakeholders

Some stakeholders will oppose or obstruct your project; examples include:

- **Competitors (inside and outside your organization)**
- **Security treats including thieves, vandals, hackers, disgruntled (ex) employees, virus and malware producers, …**
- **Military: enemies and terrorists**
- **A portion of the general public**
- **Special interest groups**
- **Others**

Can you give examples of negative stakeholders?

How did they impact you?

What did you do about them?

Typical Stakeholders - Surrogate Stakeholders

- Stakeholders who are acting on behalf of other stakeholders
- At some level, we are all surrogates
  - We act on behalf of those that pay us
- Examples include:
  - Employees
  - Representatives (e.g. to IPTs, to change boards)
  - Consultants, hired experts
  - Members of focus groups
- Surrogacy has advantages and disadvantages:
  - Surrogates are paid to get it right
  - They help ensure stakeholder input
  - They might not correctly represent the real stakeholder
  - They might bring their own agenda



SEs must often surrogate for stakeholders who will not or cannot be involved

Techniques for Identifying Your Stakeholders

- Ask sponsor or other key stakeholders already identified
- Look at organization charts
- Brainstorm
- Techniques discussed in next set of slides:
  - Checklist, taxonomy, or lists from similar projects
  - Onion model
  - Super-system diagram
  - Context diagram
- Other methods exist
  - Some may have been identified in the previous class discussion
  - Methods can be used in combination

Stakeholders must be identified by name

Using Checklists to Identify Stakeholders

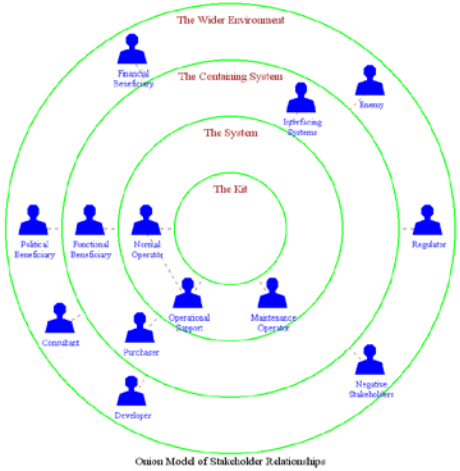
- **Sources of checklists:**
  - **Academic and textbook**
    - ***Alexander, Figure 2.4***
  - **INCOSE and other professional organizations**
  - **Other organizations processes and best practices**
  - **Your organizational processes and best practices**
    - **Standard**
    - **Tailored**
      - **To your types of projects**
      - **Based on experience with specific projects like your project**
- **Checklists can be in many forms, including sample results from the other techniques:**
  - **For example, an onion diagram tailored to your project type**



Does your organization use checklists to identify stakeholders?

Using an Onion Model to Identify Stakeholders (1/2)

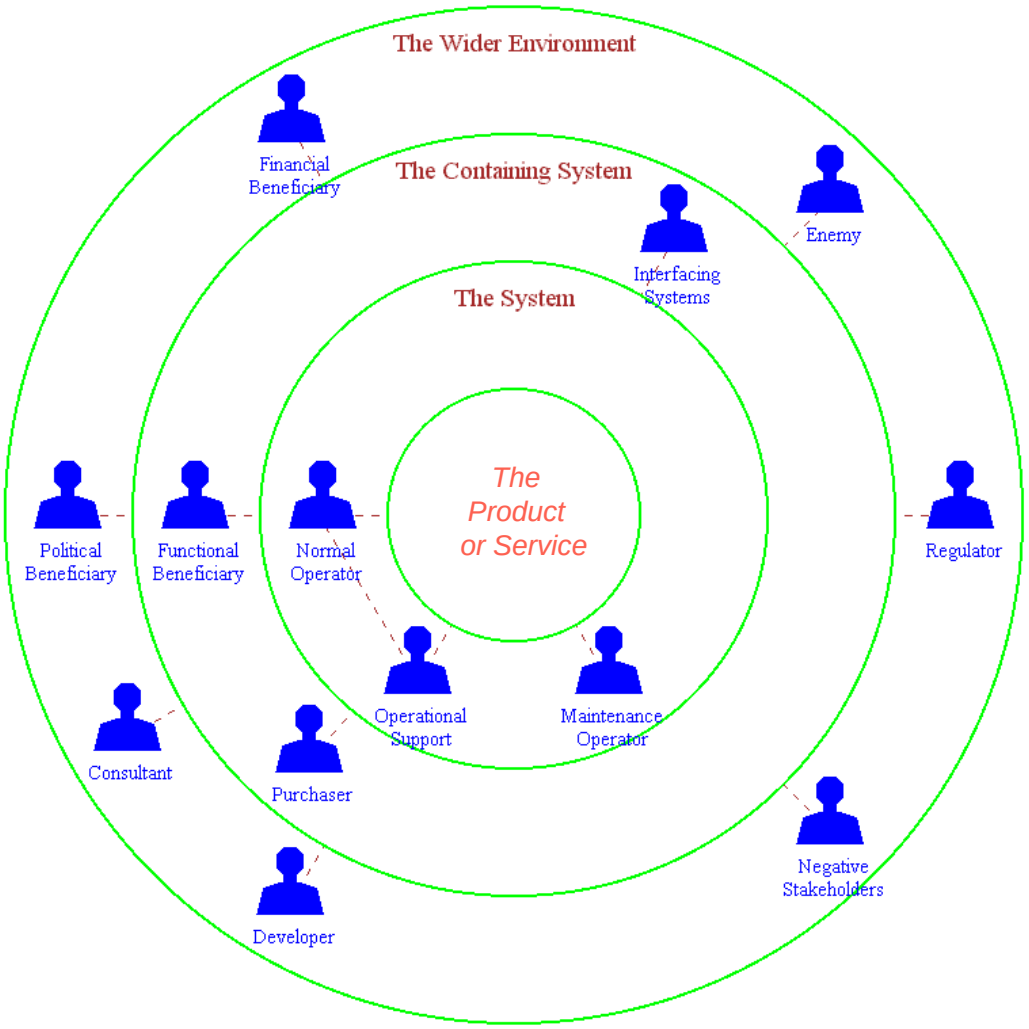
- Onion model is a template technique
  - A graphical form of a checklist
  - Rings indicate the degree of separation from the system
  - Identifies typical stakeholders
  - Must be tailored to your project
  - Other stakeholders may be added as appropriate
- Additional onion models may be drawn for subsystems or subprojects to reflect different viewpoints
  - For example, the system vs. the development system
- Onion model benefit: helps emphasize the importance of operators and other users



Onion Model of Stakeholder Relationships

See text: Alexander, Figure 2.1

Using an Onion Model to Identify Stakeholders (2/2)

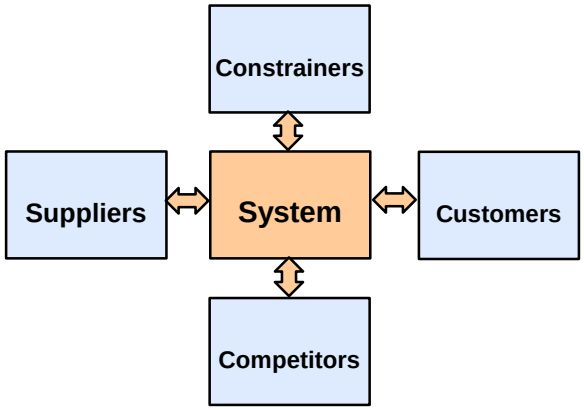


Onion Model of Stakeholder Relationships

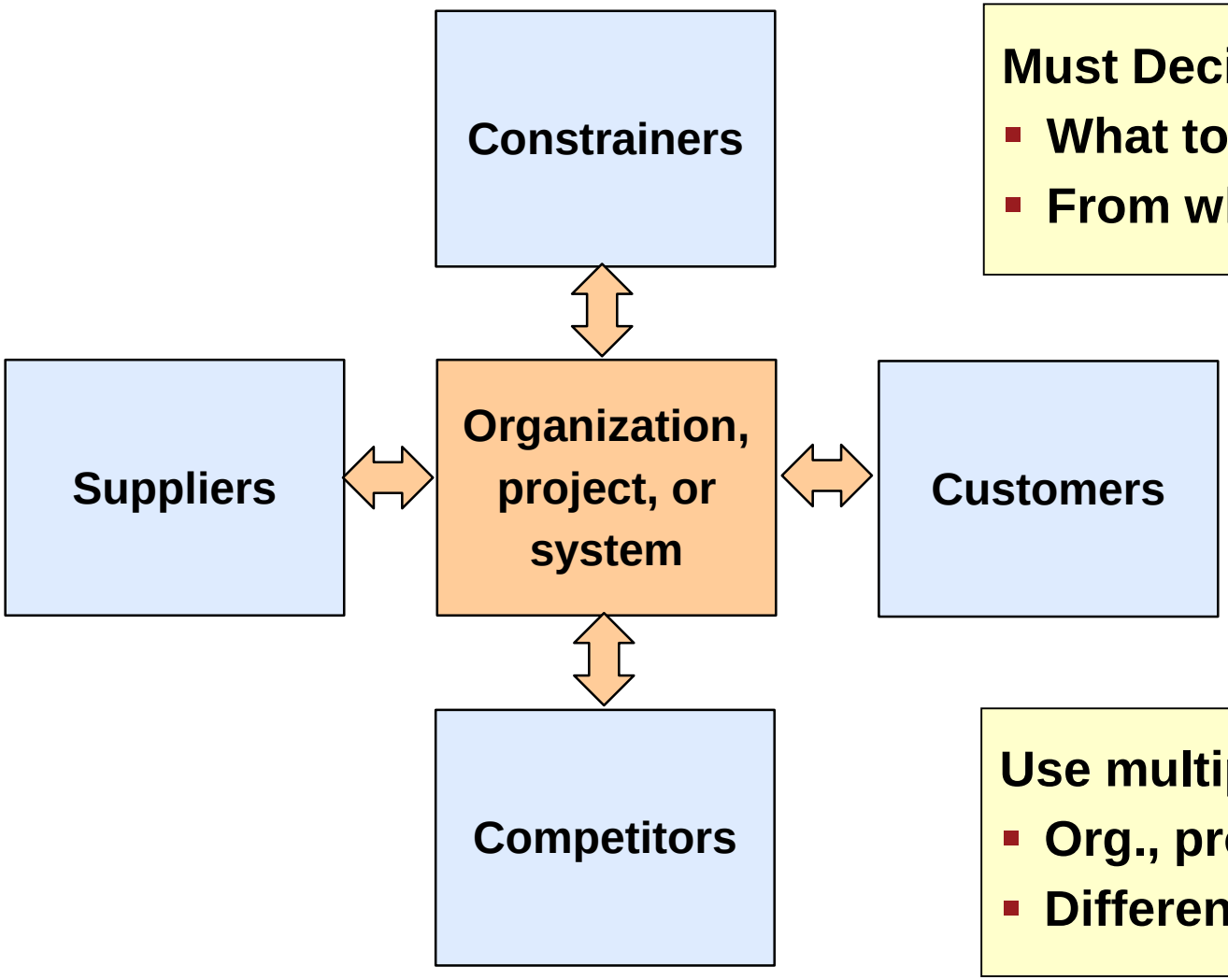
Adapted from Ian F Alexander, *A Taxonomy of Stakeholders*, International Journal of technology and Human Interaction, Vol 1, 1, 2005, pages 23-59 ; also similar to Figure 2.1 from Alexander text.

Using a Supersystem Model to Identify Stakeholders (1/4)

- Supersystem model may be done for your project, system, or subset of either
- Treats your system or project as a black box
- Views all stakeholders as belonging to four types:
  - Suppliers to your project or system
  - Constrainers of your project or system
  - Customers, including users, of your project or system
  - Competitors (alternate providers) for the results of your project or system
- Identifies relationships between the stakeholders and the system
- Should be done from different viewpoints (e.g. you, your customer, key users)



Using a Supersystem Model to Identify Stakeholders (2/4)



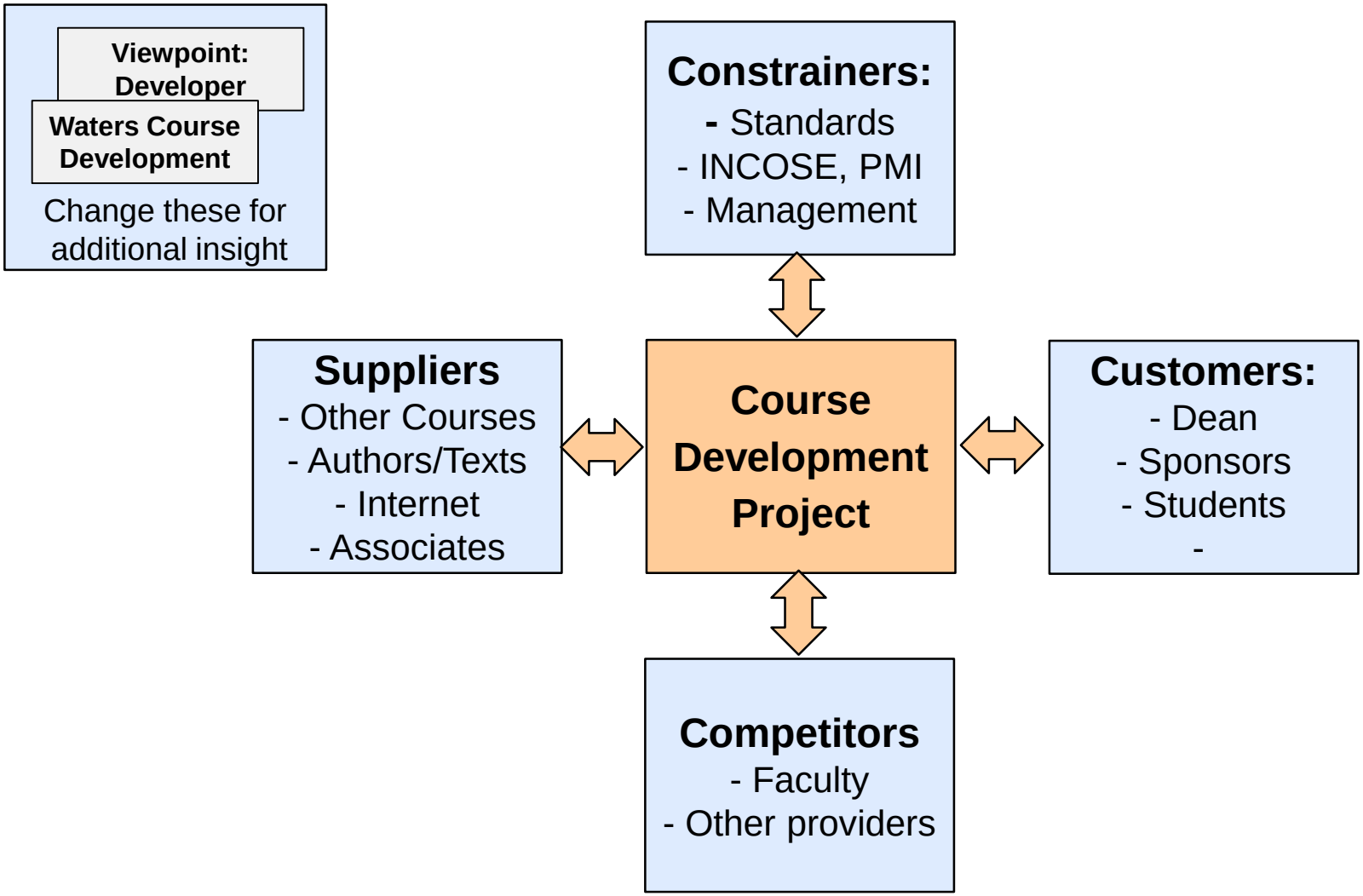
Must Decide:

- What to model?
- From whose viewpoint?

Use multiple models:

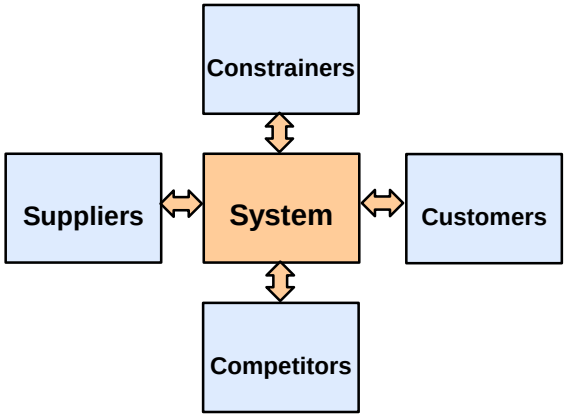
- Org., project, system….
- Different viewpoints

Using a Supersystem Model to Identify Stakeholders (3/4): Simple Example



Using a Supersystem Model to Identify Stakeholders (4/4): The Process

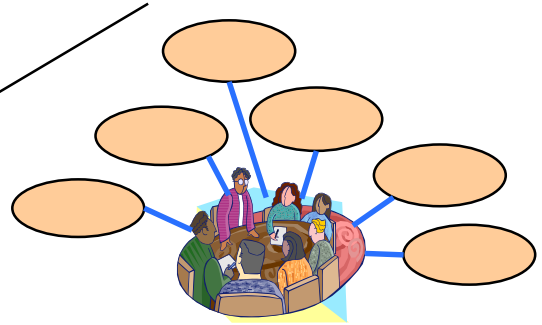
- Decide what you will model
  - Your organization, project, or system
  - Initially, try the organization or project
  - Later the system and subsets are appropriate
- Look outward towards external world
- Decide whose viewpoint to use
  - Later other viewpoints can be done for added insight
- Identify the stakeholders in each of the four categories
- Identify the relationships
- Establish the boundary (scope) by asking what is inside and what is outside

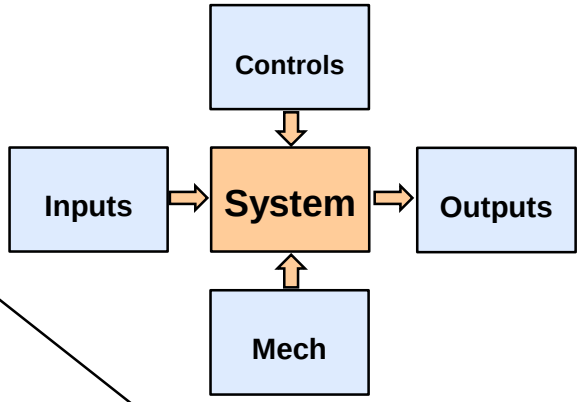


Seek clarification using the elicitation techniques in the next section

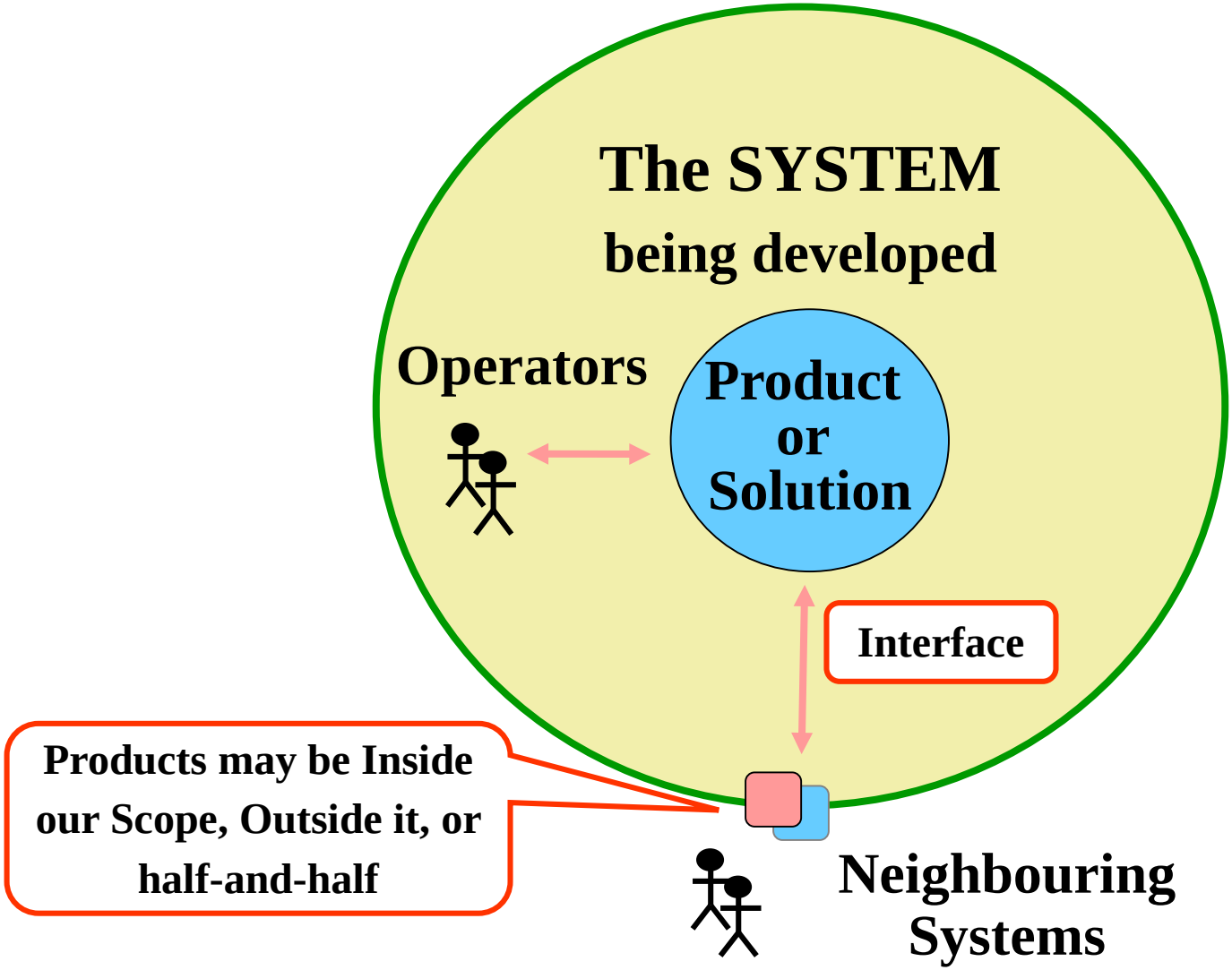
Using a Traditional Context Diagram to Identify Stakeholders (1/2)

- A context diagram can be used for identifying stakeholders and understanding boundaries
- Context diagrams come in many forms
  - Simplest is a circle representing the system surrounded by all the stakeholders (similar to earlier slide on "typical stakeholders")
  - IDEF0 models can be used to represent system context (next slide)
- Context diagrams, as their name implies, help us understand the problem context



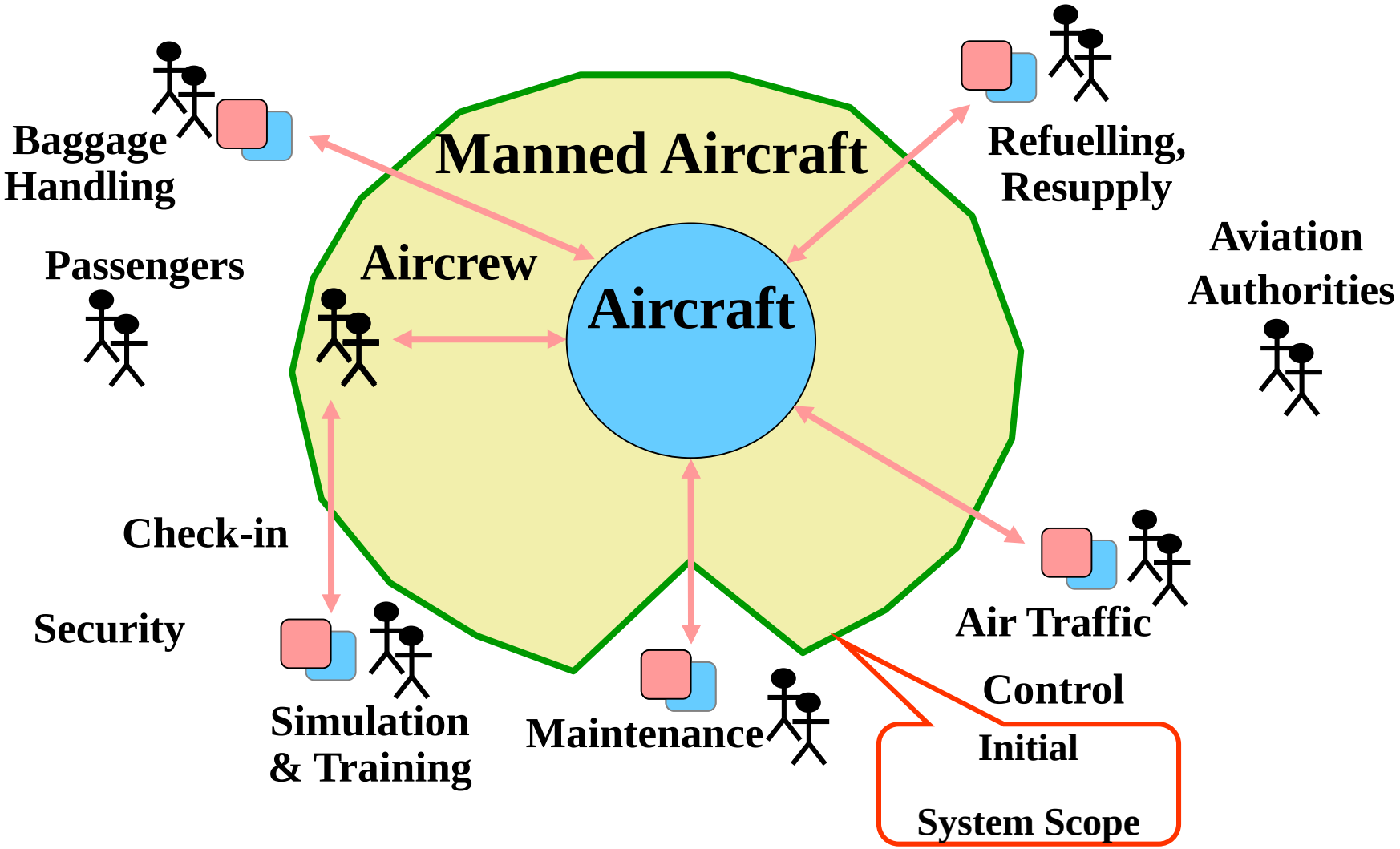


Using a Traditional Context Diagram to Identify Stakeholders (2/2)



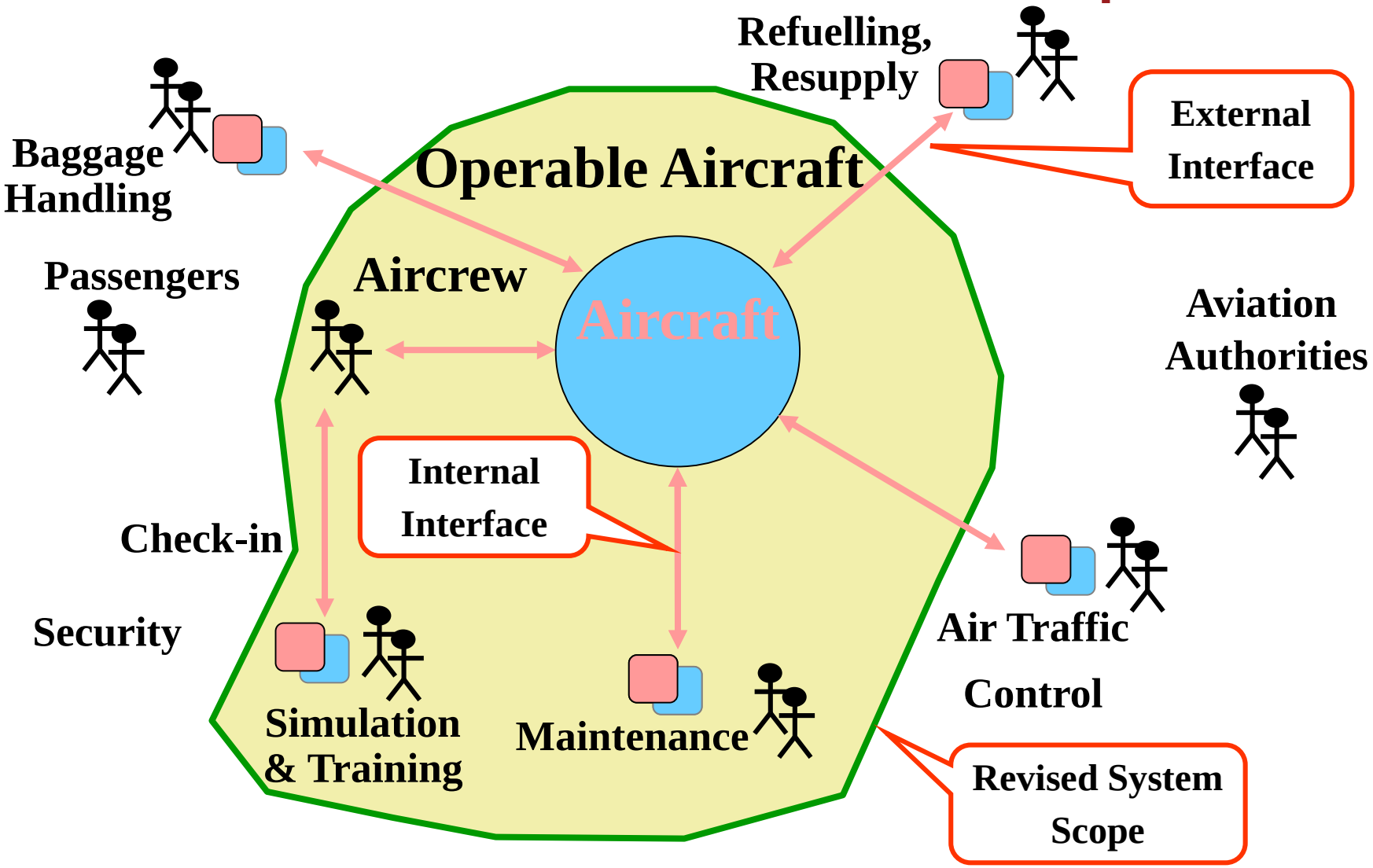
Source: adapted from Alexander text, Figure 4.3

Using a Context Diagram to identify Stakeholders – Manned Aircraft Example (1/3)



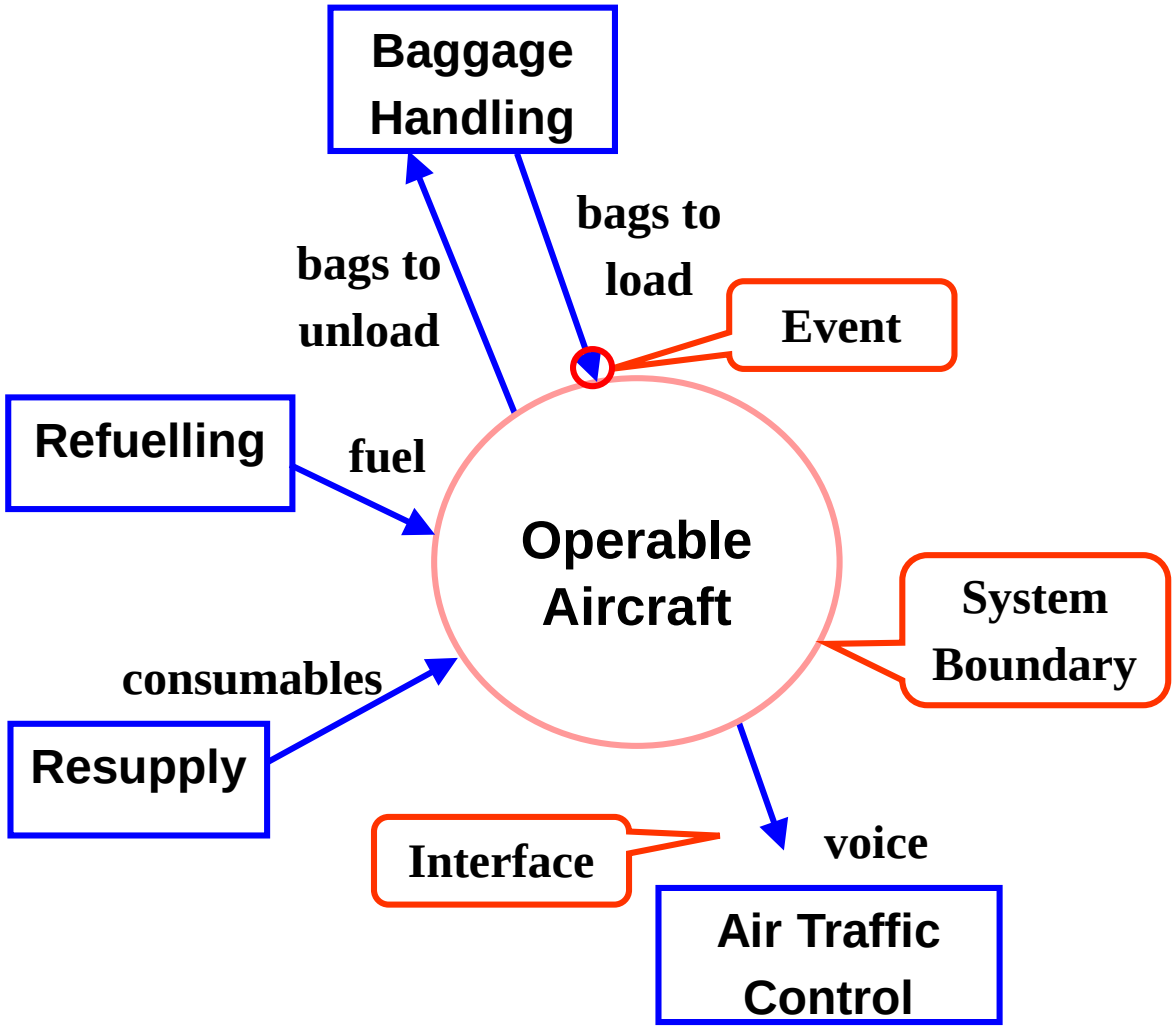
Source: adapted from Alexander text, Figure 4.4

Using a Context Diagram to identify Stakeholders – Manned Aircraft Example (2/3)



Source: adapted from Alexander text, Figure 4.5

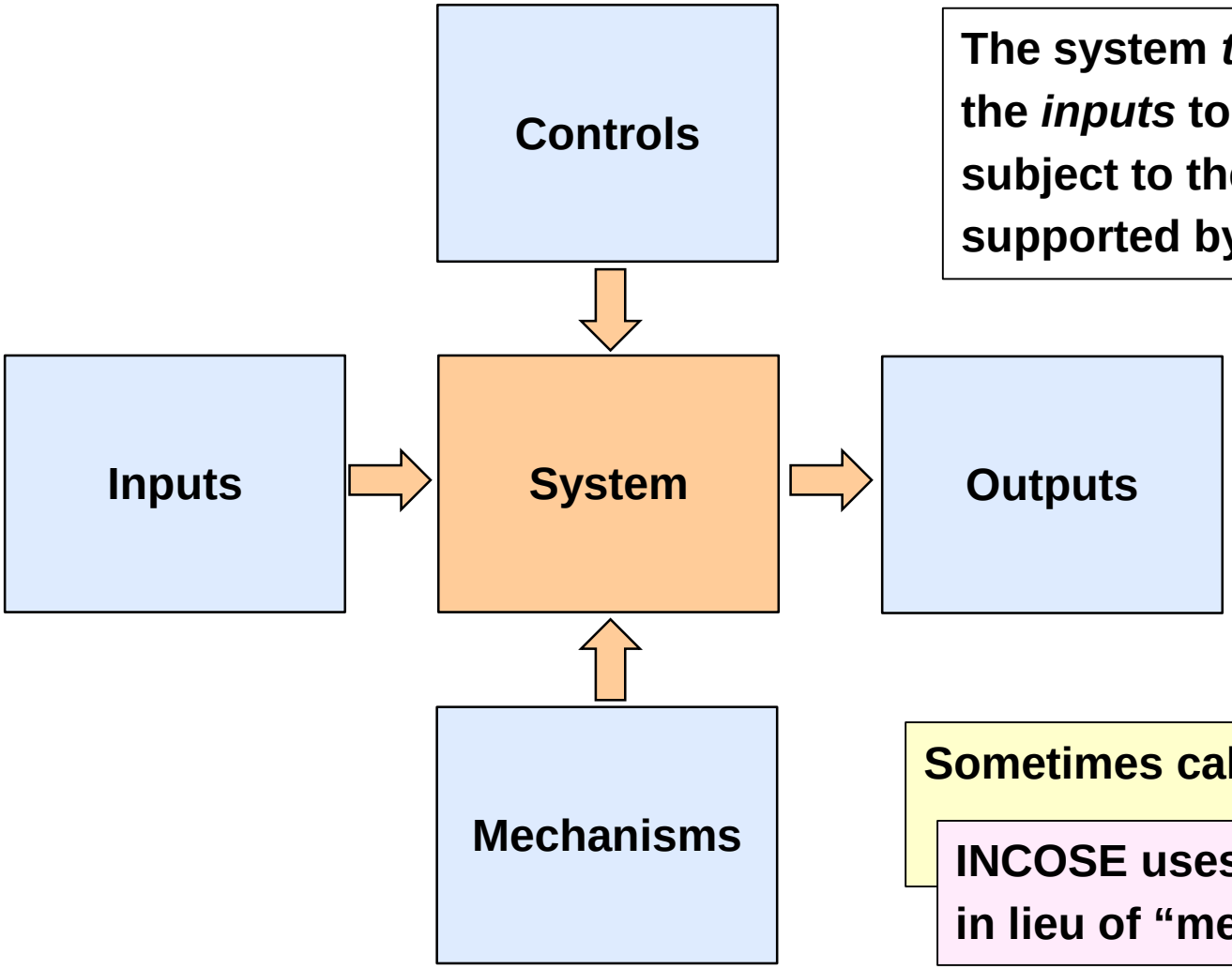
Using a Context Diagram to identify Stakeholders – Manned Aircraft Example (3/3)



Event / Task	In/Out
Load Bags	In Scope
Unload Bags	In Scope
Refuel	In Scope
Check-in Passengers	Out of Scope
Radio Air Traffic Control	In Scope
Provide Airport Security	Out of Scope

Source: adapted from Alexander text, Figure 4.6

Using a Context Diagram to Identify Stakeholders - IDEF0 Example (1/3)



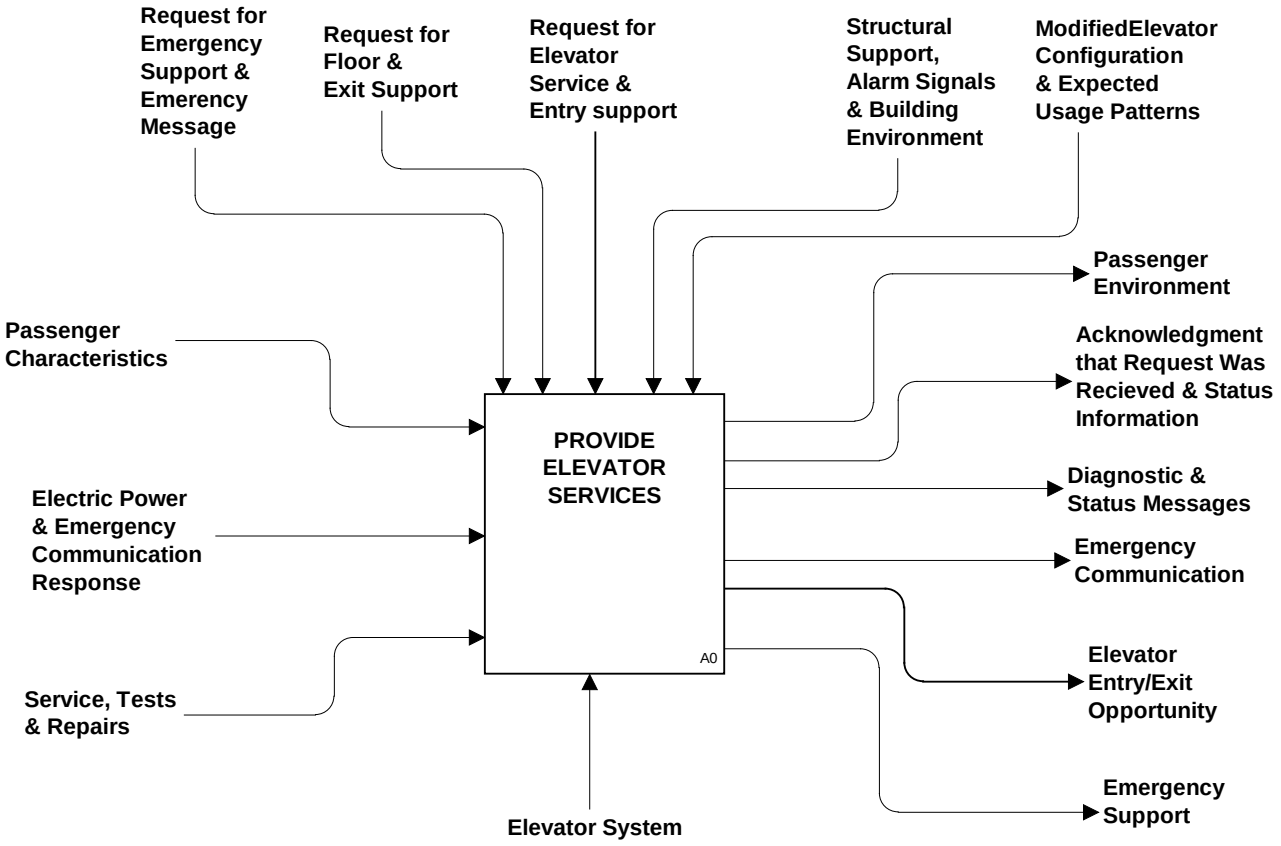
The system *transforms* the *inputs* to the *outputs*, subject to the *controls*, and supported by the *mechanisms*

Sometimes called an "ICOM"

INCOSE uses "enablers" in lieu of "mechanisms"

Using a Context Diagram to Identify Stakeholders - IDEF0 Example (2/3)

USED AT: George Mason Univ.	AUTHOR: Dennis Buede PROJECT: Elevator Case Study NOTES: 1 2 3 4 5 6 7 8 9 10	DATE: 05/24/99 REV:		WORKING / DRAFT / RECOMMENDED / PUBLICATION	READER DATE	CONTEXT: Top



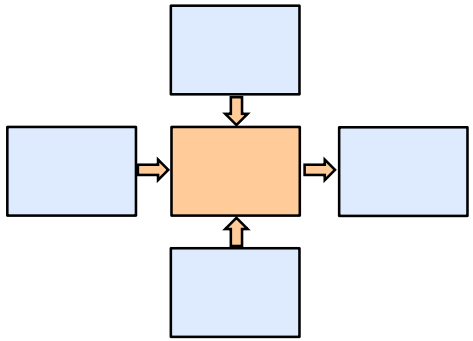
NODE: A-0	TITLE: Provide Elevator Services	NUMBER: P. 2

Source: Buede text, Figure 2.5

Using a Context Diagram to Identify Stakeholders - IDEF0 Example (3/3)

- IDEF0 is a formal method
  - Formal structure, informal text
  - Supported by tools (e.g. CORE)
  - Can be used to model the project, system (product or service) or a process
    - Previous slide models the service (provide elevator service)
    - Buede text appendix B uses IDEF0 to model the systems engineering process.

- An IDEF0 context diagram can be a first step in using IDEF0 models for functional analysis.
- IDEF0 will be discussed later in the course



Stakeholder Identification Techniques also Help Define System Boundaries

- **Boundaries** *… defines that which is inside the system and that which is outside the system.*
  - **Inside – the product that is to be developed and delivered**
  - **Outside – the environment in which the system operates, including its operators**
  - **Interfaces – provide the mechanisms by which the system can interact with its environment**

- **(Boundaries of truly complex systems may be difficult to define)**

- **Environmental Interactions** *… define the interfaces and nature of interacts between the system and its surroundings.*

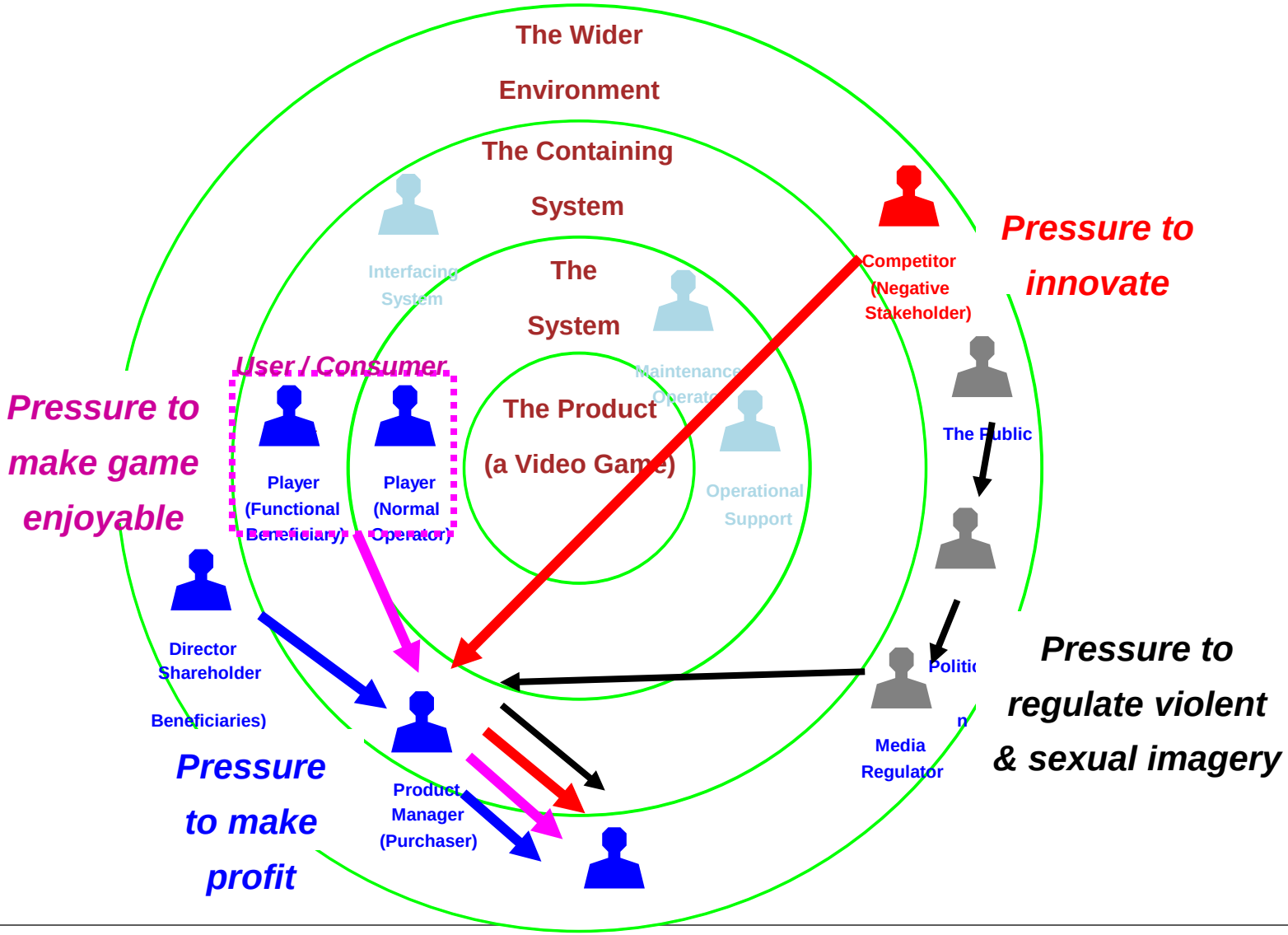
- **(Interactions of truly complex systems are key)**

Managing Stakeholders

- First they must be identified, by name
- A plan for their involvement is essential
  - Include formal (control gates) and informal activity
  - Include in the Project Plan and Systems Engineering Plan
  - Get stakeholder agreement and approval of the plan
  - Include in the WBS
- Engage them per the plan
  - Get their approval of the evolving requirements through in-process and formal reviews
  - Practice in-process validation
- Manage ongoing involvement
  - Encourage and reward involvement
  - Escalate non-involvement
  - Be alert to changes in stakeholders

What do you do when a new stakeholder replaces an old one?

Influences Diagram: Influences on a Product Manager



Exercise: Stakeholder Identification

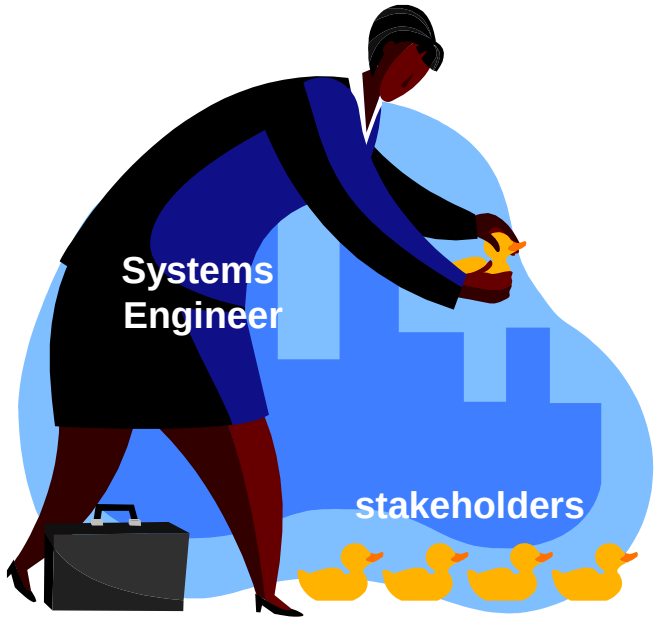
- The instructor will identify the project to be used for this exercise
- Identify the stakeholders for this project:
  - Develop one of the context diagrams illustrated in either of your two textbooks
  - Be sure to include as many of the stakeholders as possible.
- Instructor will provide timing and other directions



The project to be used for this exercise will be identified at the beginning of the exercise

Next Steps

Once we have identified our stakeholders, we can engage them to understand the problem and the context



Session 3 – Stakeholder Needs

Lessons

1. Program Overview Update
2. Introduction to Requirements
3. **Stakeholders and their Expectations**
4. Stakeholder Requirements
5. Special Topics (if time permits)

Lesson Content

- The problem to be solved?
- Identifying stakeholders
- **Eliciting stakeholders expectations and involvement**



Appendices*

E. Session 1 Summary
F. A Modeling Exercise
G. Other Appendices TBD

*Appendices A, B, C, and D were contained in Session 1 of the course; remember that Appendix D has a complete course bibliography. The course will reference, but will not cover, the appendices

Expectations

In the beginning there are only expectations; requirements engineering transforms these expectations into a set of agreed engineering requirements



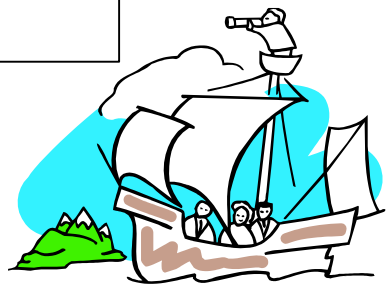
Your text (Alexander) uses the term "goals" in lieu of "expectations"

Expectations are not requirements. We are not yet ready to negotiate. Instead we want to collect all the expectations; Until we uncover all the expectations, we cannot do a good job of defining, much less agreeing to, the requirements.

Elicitation

One model for the early process is “elicitation” This is a proactive process in which stakeholders are engaged for the purposes of (1) obtaining their cooperative participation, and (2) understanding their expectations

Elicitation is a first step in the process your text (Alexander) calls “discovery”. Lesson 4 discusses how we analyze these elicited expectations, and produce a set of agreed concept – independent stakeholder requirements



Expectation Elicitation Techniques

- Analogy to existing or past systems
- Stakeholder Analysis
- Communication and information exchange
- Modeling (prototyping, data modeling, simulation)



Goal is mutual understanding

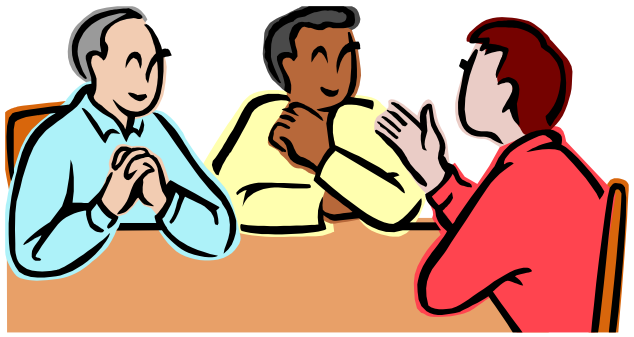
Typically multiple techniques are used on a project

Analogy to Existing or Previous Systems

- Most systems are not revolutionary, but instead are evolutions from what already existed before
- Requirements from the new system are often based at least partly on analogies to previous systems
- Examples:
  - Polaris/ Poseidon => Trident
  - Windows => ….. XP => Vista => Windows 7 => ???
  - Many automobile models (Model "T" => Model "A")
- Alexander covers this in what he calls "Archeology"

Discussion:

1. Can you think of other examples from your experience?
2. What are the pros and cons of this approach?



Stakeholder Analysis

A process to identify expectations from stakeholders, and prioritize their importance

Stakeholder Role	Requirement Type	Discovery technique
Use this form to plan stakeholder engagement (See Alexander Table 2.4)		

Stakeholder	Type	Expectation	Priority
Use this form to identify expectations and prioritize the stakeholders prior to the engagement			

How do we determine stakeholder priority?

Who ultimately decides (arbitrates) stakeholder priority?

Performing a Stakeholder Analysis

To plan how to engage the stakeholders:

- **Use the first template from the previous page**
- **List the stakeholders you have identified**
- **Indicate the type of requirement you expect from the stakeholder (e.g., technical, cost, schedule…)**
- **Identify the elicitation techniques appropriate to each stakeholder**

To identify and prioritize stakeholders and their expectations (in preparation for the engagement):

- **Use the template from previous page**
- **List the stakeholders you have identified**
- **List the stakeholder type (sponsor, customer, user, constrainer, provider, etc.)**
- **For each stakeholder, record what you and your team believe will be their primary expectations**
- **Indicate the perceived priority of the stakeholder**

A Stakeholder Analysis prepares for the engagement, and records the results

Elicitation by Communication and Information Exchange Techniques (1/4)

- Questionnaires and surveys
  - Administer to all key stakeholders
  - Statistical interpretation may be necessary
  - Survey instrument design is difficult
    - Might not give you correct answers or answers you can use



- Comment Forms
  - Useful when there are existing users
  - Barking dog gets the bone?

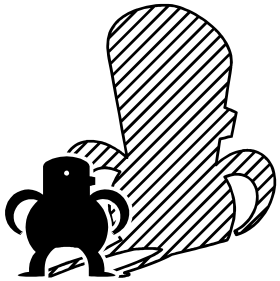


- Focus groups
  - Useful when there are large numbers of stakeholders
  - Requires identifying “representative” stakeholders



Elicitation by Communication and Information Exchange Techniques (2/4)

- **Observing (Shadowing)**
  - **Extremely valuable to observe the context, and how the "problem" is being solved currently**
  - **Helps verify what stakeholders say in other techniques**



- **Apprenticing: A coached observation**

- **Brainstorming**
  - **Always a useful technique for generating creative ideas and viewpoints**
  - **Who should be included?**

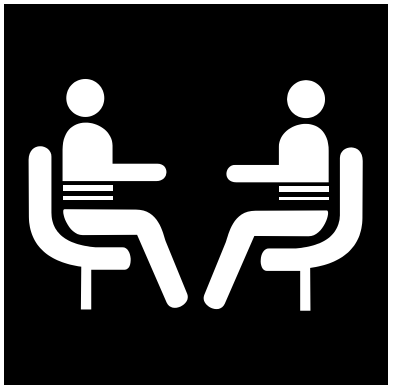


Elicitation by Communication and Information Exchange Techniques (3/4)

- **Documentation reviews:**
  - **Essential for interview preparation**
  - **What documents should be reviewed?**



- **Interviews and workshops**
  - **Individuals are interviewed**
  - **Groups can be included in workshops**
  - **Should be planned, not adhoc**
  - **Results should be documented and agreed at end of meeting**



Elicitation by Communication and Information Exchange Techniques (4/4)

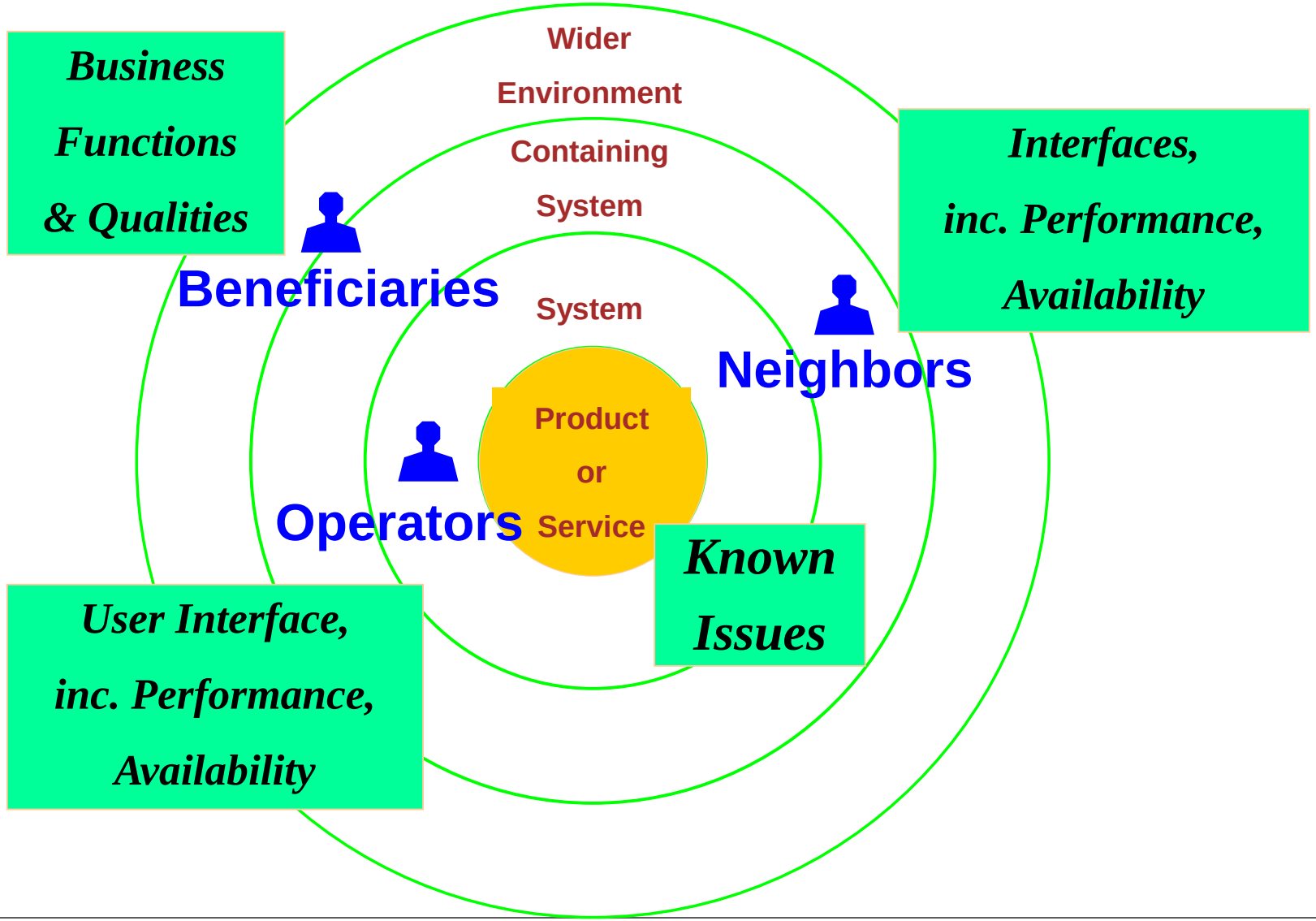
- **Archeology**
  - **Digging through other project histories**
  - **Making analogies to the new system**
- **Expert opinion, consultants**
  - **Can be very valuable**
  - **These are almost always surrogate stakeholders**
  - **Make sure they are experienced in the areas for which you need input**

Discovering Expectations (goals)

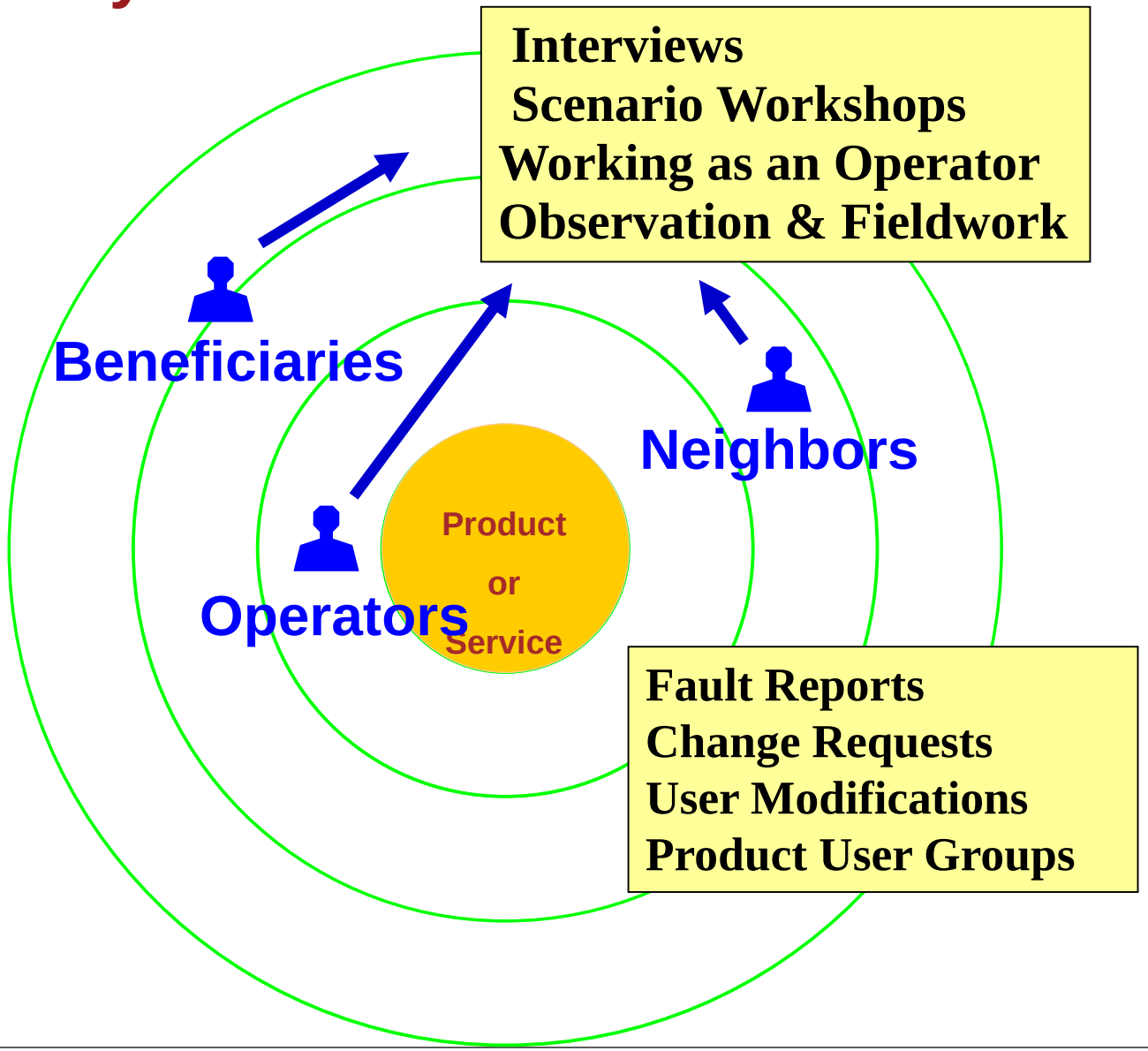
The next several charts discuss Alexander's Discovery approach to eliciting expectations (goals) from stakeholders

- **Discovering from operational roles**
- **Discovery contexts and sources**
- **Discovery from non operational roles**
- **An example**

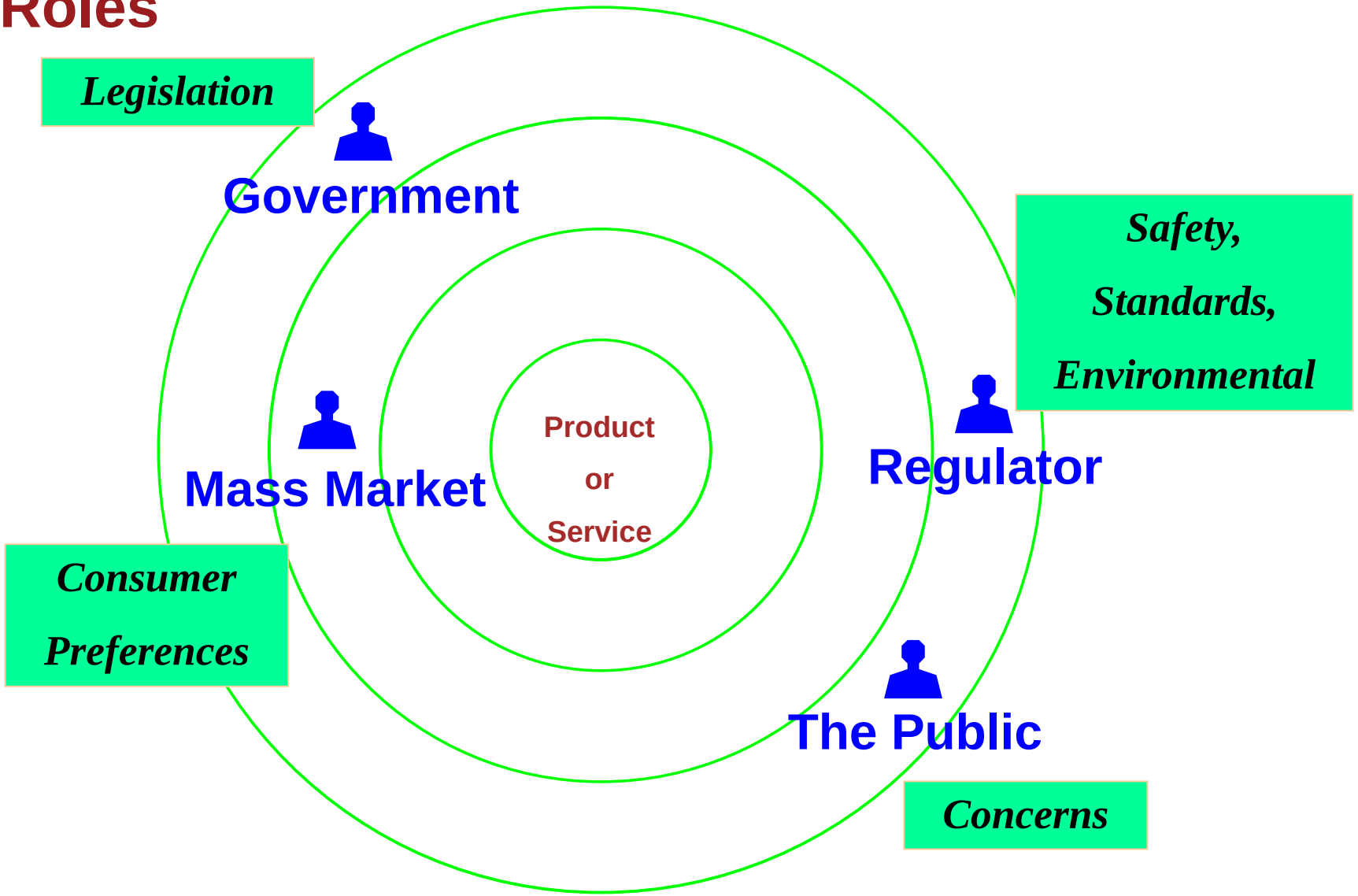
Discovering from Operational Roles



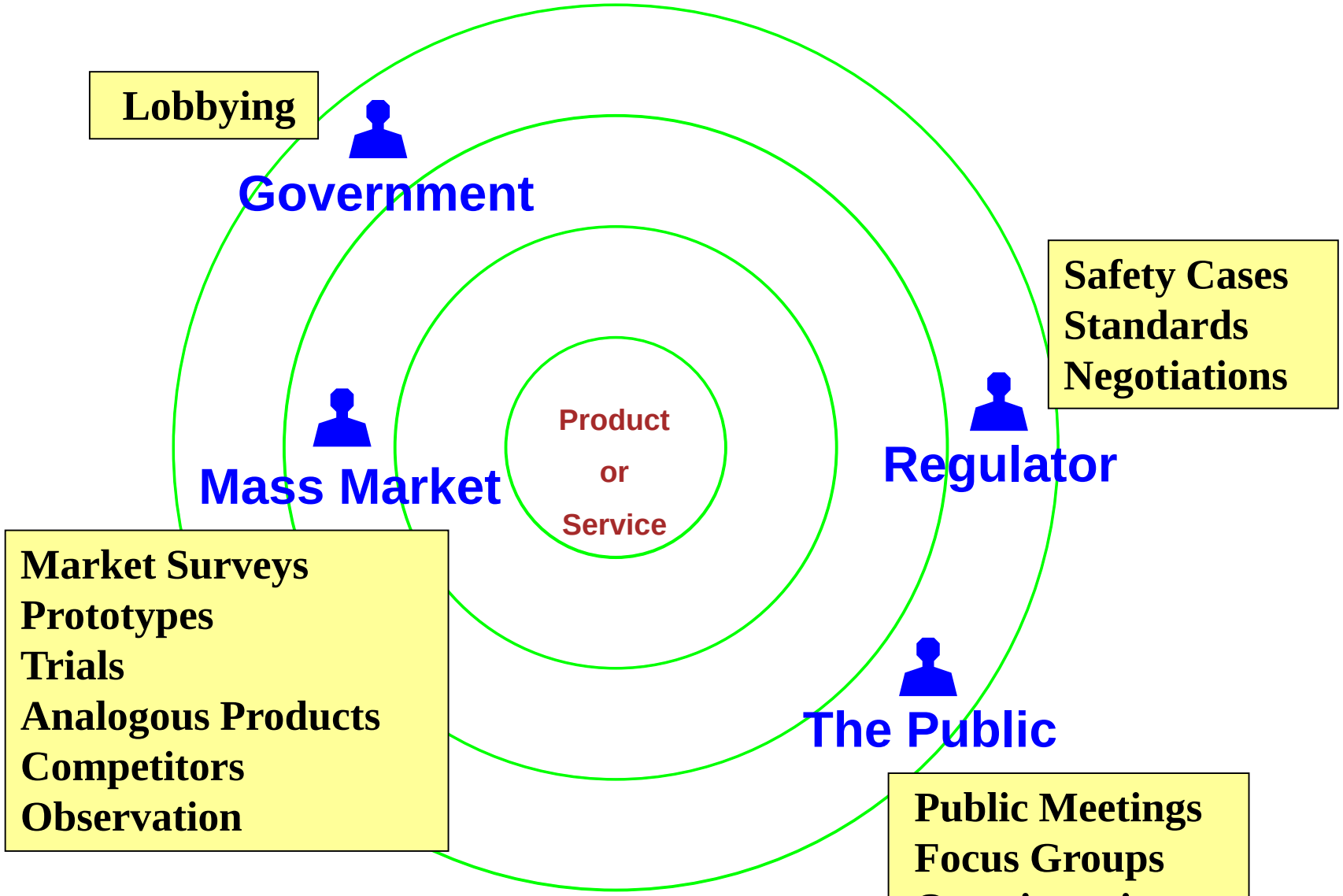
Discovery Contexts and Sources



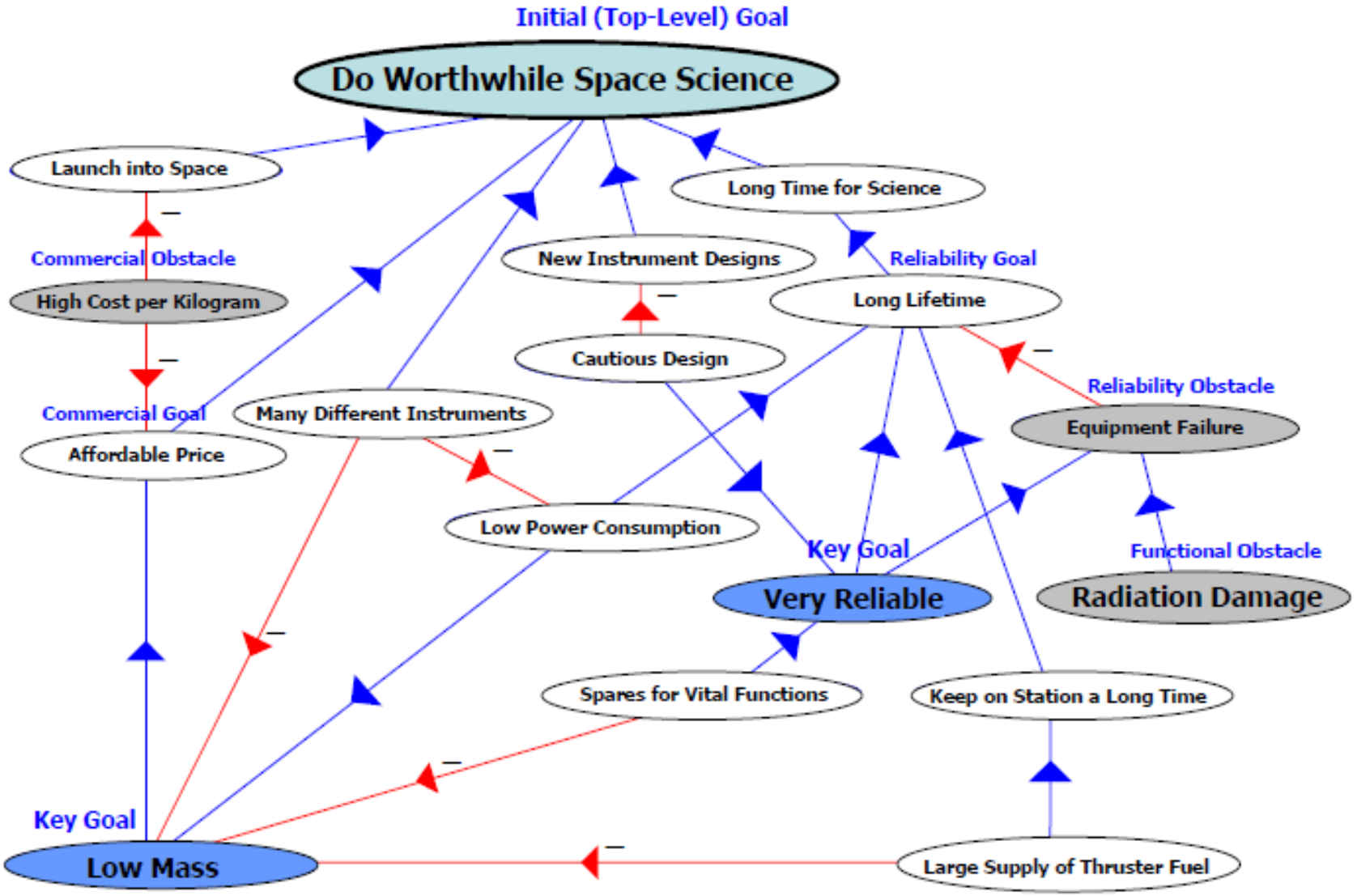
Requirements from Non-Operational Roles



Discovering from Non-Operational Roles



Example Goal Diagram



Alexander, Figure 3.2

Other Considerations

- **Objectives Hierarchy**
- **Modeling**
- **Agile Methods**

Exercise: Stakeholder Analysis

- The instructor will identify the project to be used for this exercise
- Perform a stakeholder Analysis for this problem
  - Two groups will focus on planning the stakeholder engagement
  - Other groups will focus on expectations and priority
- Instructor will provide timing and other directions

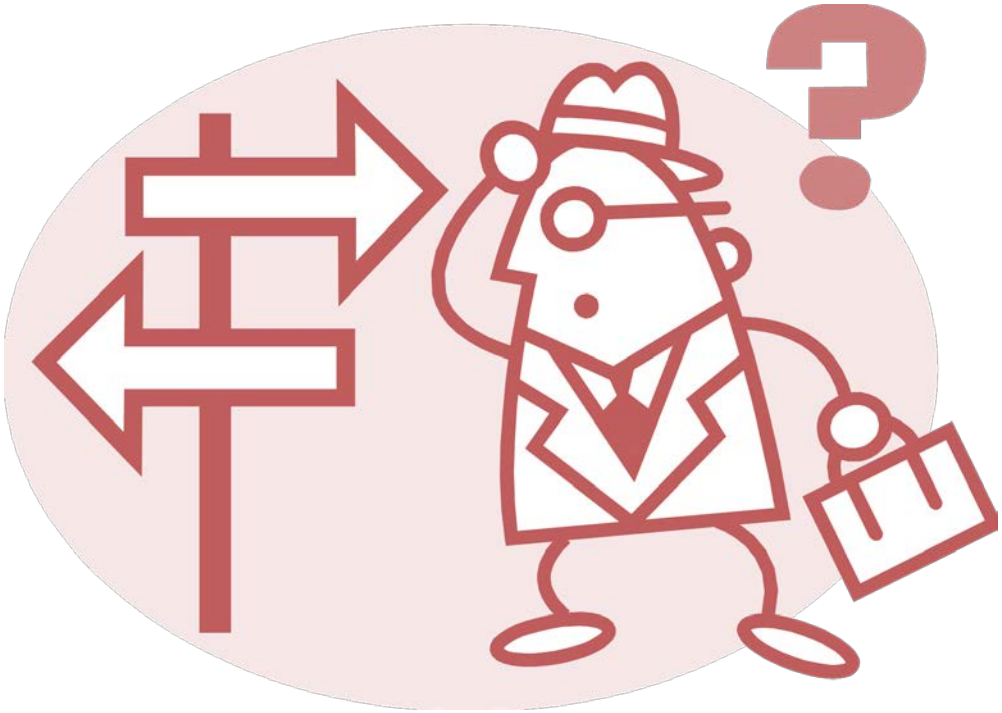


The project to be used for this exercise will be identified at the beginning of the exercise

Looking Forward……

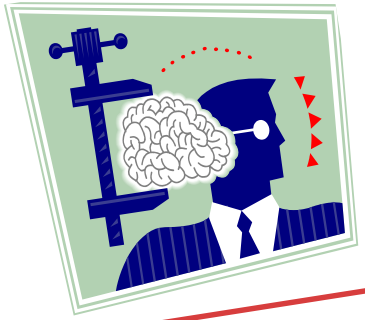
- This lesson looked at eliciting stakeholder expectations
- Next lesson will look at analyzing those expectations to determine what are the real stakeholder requirements

- Are there any questions?



Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?



Systems Engineering Technical Short Course

(Session 2: Stakeholder Needs)

Lesson 5:
Special Topics
(if time permits)

Robert B Wheeler, Instructor

Sessions 2 through 5 Agenda

- **Day 1**
  - Review project team assignments from previous Session
  - Presentation and discussion of Session topic
  - Project team working sessions on the application of the Session topics to their projects
  - Day 1 wrap-up and Q&A

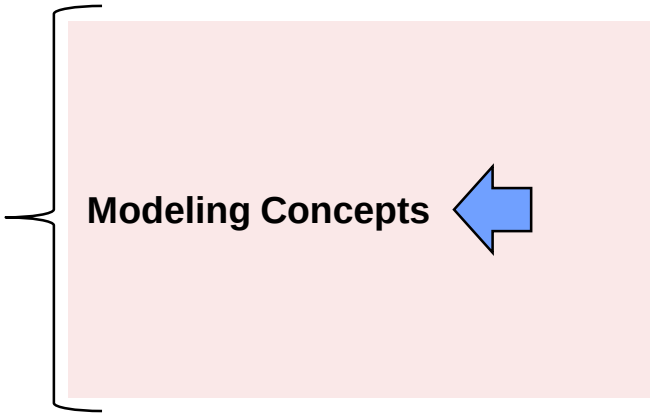
- **Day 2**
  - Review Day 1 topic and project working sessions
  - Presentation and discussion of Session topic (concluded)
  - Project team working sessions (concluded)
  - Check-in with class on the application of the Session topic to their projects
  - Session wrap-up: Q&A and homework due next Session

Timing will be similar to session 1, and will be announced at beginning of each session

Session 2 – Stakeholder Needs

Lessons

1. Program Overview Update
2. Introduction to Requirements
3. Stakeholders and their Expectations
4. Stakeholder Requirements
5. **Special Topics (if time permits)**



Appendices*

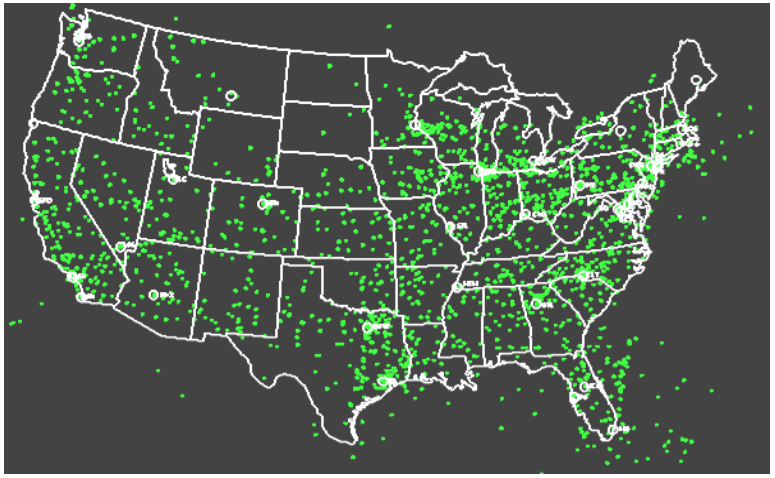
E. Session 1 Summary
F. A Modeling Exercise
G. Other Appendices TBD

*Appendices A, B, C, and D were contained in Session 1 of the course; remember that Appendix D has a complete course bibliography. The course will reference, but will not cover, the appendices

What is a Model?

- An abstraction of reality designed to convey information
  - Defined with respect to some aspect of structure or behavior
  - Necessarily incomplete and approximate
  - A model of the system is *not* the system

- It takes many models to adequately describe a complex system









What is a model?

- A model is an incomplete representation of reality:
  - Simplifies whatever is being modeled.
  - Based on assumptions about the environment.
  - Provide insight into system or entity behavior
  - Provide views of aspects of the system to stakeholders
- Everyone has their own mental models to help understand objects in our environments.
- Models are often used to communicate and understand future realities (e.g., systems to be developed)
- Models sometimes can ultimately define reality!

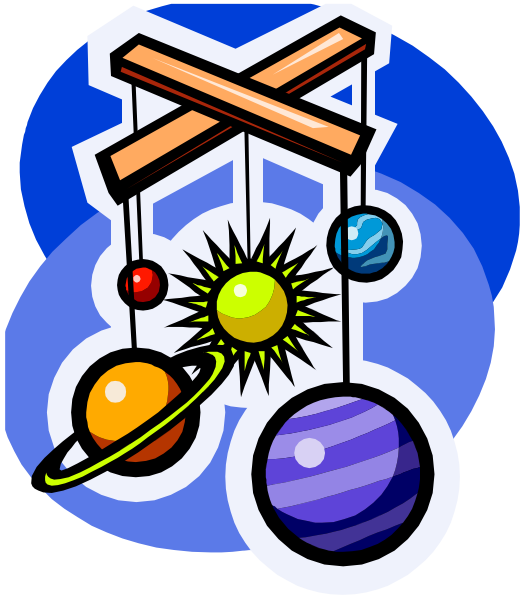
All Models are wrong; some are useful

George Box

What is Modeling?

"Modeling is the creation of abstractions or representations of the system to predict and analyze performance, cost, schedules, and risks, and to provide guidelines for systems research, development, design, manufacturing, and management." (Maier/Rechtin p.11)

Models are used throughout the development cycle to understand and communicate our evolving solution.



Modeling has many Purposes

- Communicate concepts
- Specify concepts
- Test concepts
- Estimate or predict some quantitative measure associated with the system
- Select one design option over another
  - Cost models
  - Performance models
  - Decision models



- Communicate baseline progression
  - to user for validation
  - to solution provider to obtain satisfactory solution

Systems Engineering uses models to transform users needs into a solution satisfying those needs.→

Models Communicate and Facilitate Mutual Understanding

- We perform modeling to create and evolve models that communicate clearly, and facilitate mutual understanding
- Models must be able to be understood by those who will review them, and make system decisions based on them
- Formal models improve communication and mutual understanding IF all parties understand the rules
  - Formal symbols (graphics)
  - Formal syntax definitions (text)

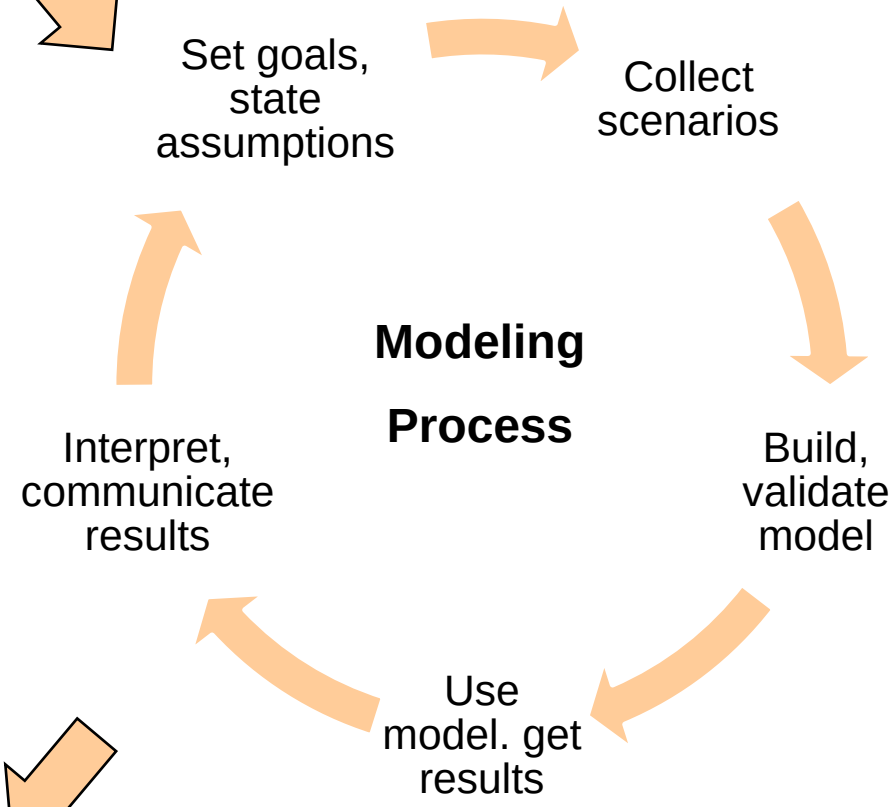
"The purpose of modeling is to relentlessly expose the inevitable consequences of your assumptions."
– Unknown

Modeling Should Be Viewed as a Process

Problem to be Solved, Question to be answered

- **Modeling is a collection of interrelated steps.**
- **At each step of modeling, evaluation of the other steps must be taken.**
  - **Backtracking to earlier steps is often necessary.**
  - **The completion of the modeling steps often results in starting over.**
- **Keeping the modeling goals at the forefront of all modeling activities is the key to successful modeling.**

Answer, decision, or solution



Communication in Context

- **Systems Engineering must communicate:**
  - **ideas**
  - **concepts**
  - **requirement**
  - **solutions**

Across organizational:

- **boundaries**
- **levels**
- **disciplines**
- **environments**

To achieve:

- **mutual understanding**
- **mutual agreement**
- **stakeholder satisfaction**
- **a solution with system integrity**

Modeling and models are an essential communication tool to achieving mutual understanding and agreement.

We use models and simulations in all of their forms. How do models contribute to systems thinking?

Modeling is the language of Systems Engineering

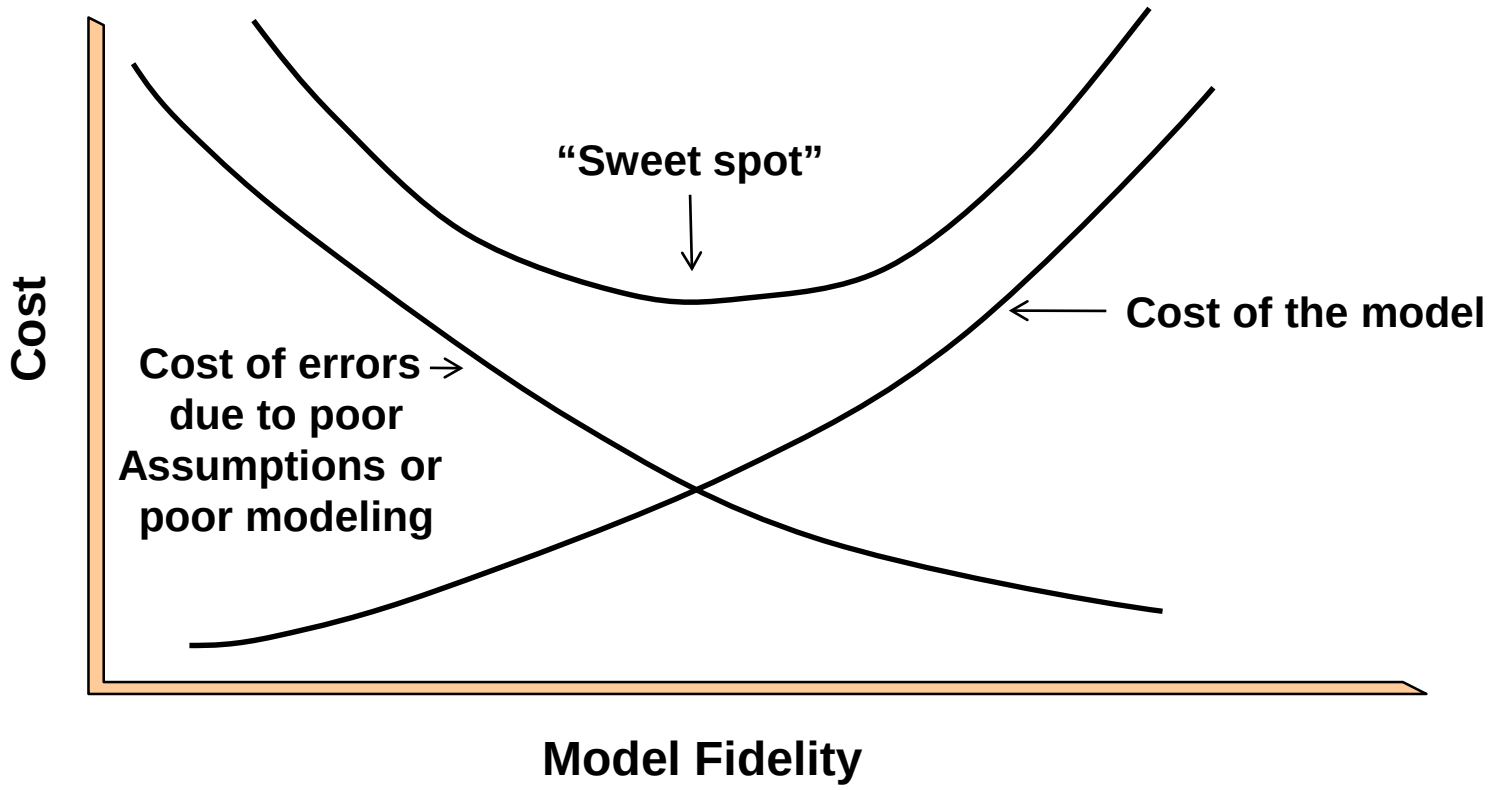
Model Fidelity

- **Definition: The degree to which the model conforms to reality**
- **Increasing model fidelity requires more robust models**
- **Robust models are expensive**

What fidelity is required for a system model?

How robust must our models be?

Model Fidelity Objectives are a Function of Risk

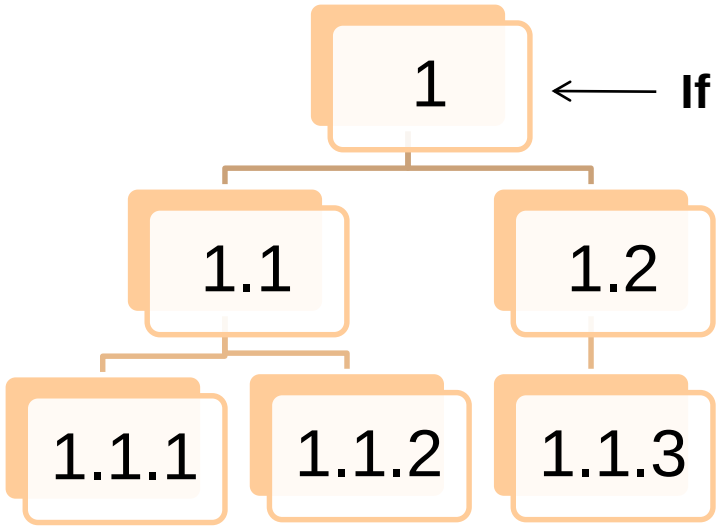


This diagram is "notional". How do we know where the sweet spot is?

The INCOSE Systems Engineering Handbook, Chapter 10, has a nice write-up on "tailoring"

An Illustration of Modeling Fidelity

- **In general, you should model at levels lower than the level at which you are making a decision or answering a question**



If you want to understand at this level…,

…then model at this level

…..and make simplifying assumptions for lower levels

- **Use risk, criticality of the decision, and cost of the modeling to decide the level of fidelity to be achieved**

How do you know if you have enough detail?

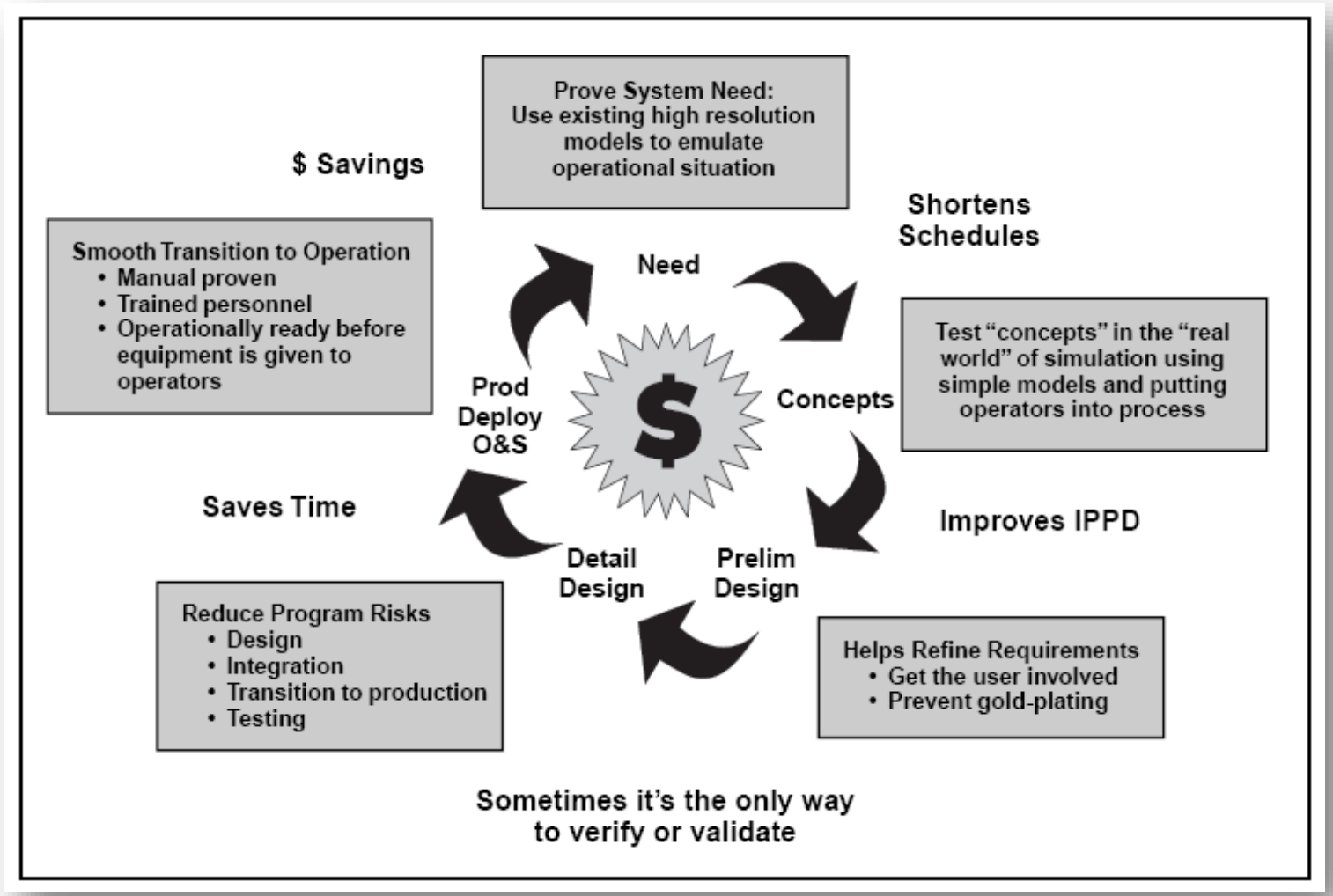
How do you know if you have too much detail?

Model Categories by Form

- **Physical models:**
  - These are geometric equivalents, either as miniatures, enlargements or duplicates to scale.
  - Examples: globes, solar system models, atomic models
- **Mathematical models:**
  - Describe and explain the system they represent using the language of mathematics.
  - May be statistical or deterministic.
  - Examples: radioactive decay, Ohm's law, Newton's laws.
- **Analog models:**
  - Originally, analog computers were used to model phenomena which could be described by differential equations.
  - Now, digital computers implement abstract analogs of myriad phenomena.
- **Schematic models:**
  - A reduction of a state or event to a chart or diagram
  - Examples: a football play, an organizational chart, a process flow chart, a floor plan, an electronic schematic diagram

Benjamin S. Blanchard, Wolter J. Fabrycky, *Systems Engineering and Analysis*, Second Edition, Prentice Hall, 1990, pp. 124-126.

Modeling Offers Many Benefits

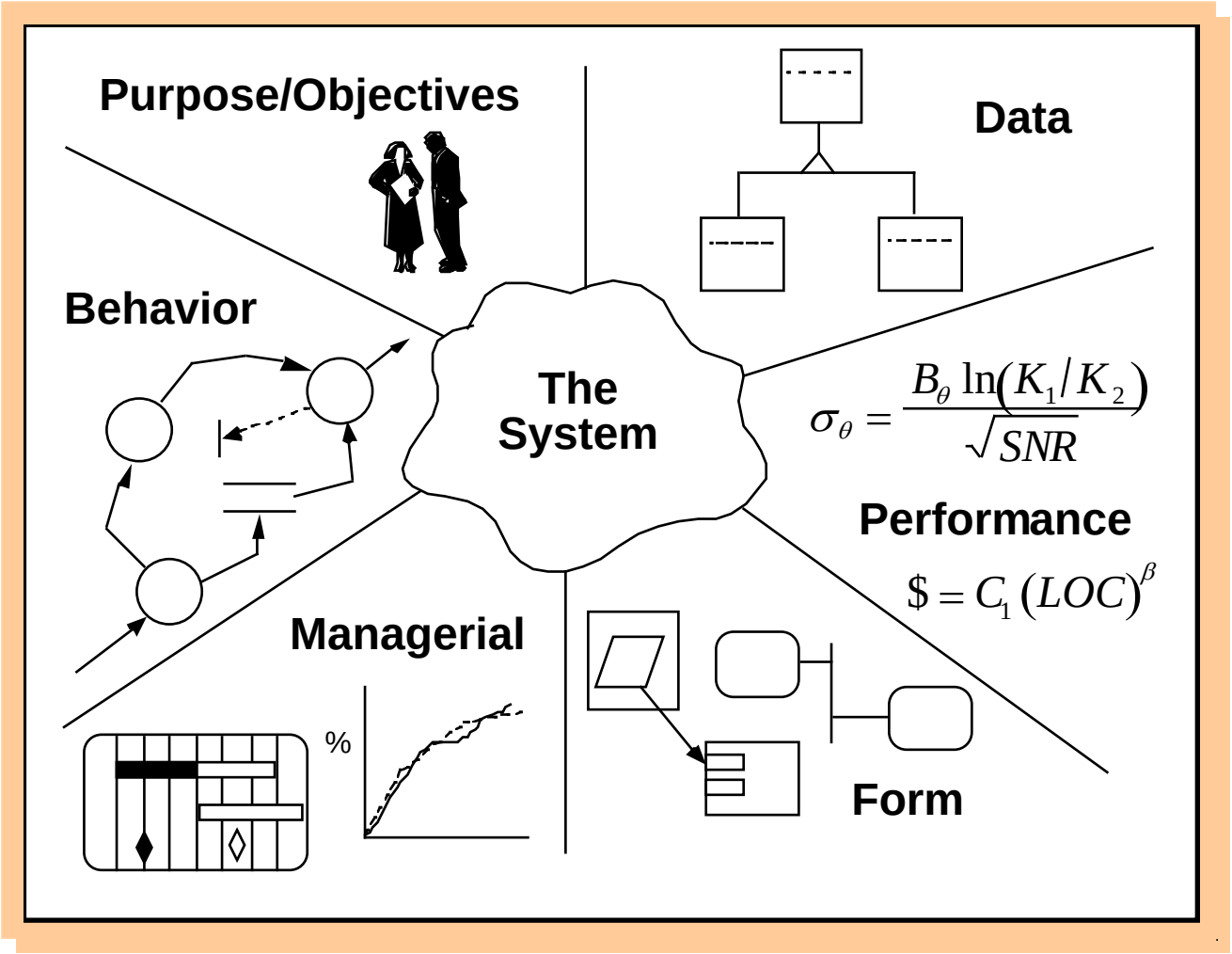


Frameworks, Viewpoints, and Views

- **Framework: A structure for organizing information that defines the scope of the subject and how the areas of the subject relate to each other.**
- **Viewpoint: A viewpoint is a standard or template for constructing a view.**
- **View: A description of the entire system from the perspective of a set of related concerns. A view is composed of one or more models.**



A Six-View Framework



SOURCE: Rechtin and Maier, "*The Art of Systems Architecting*"

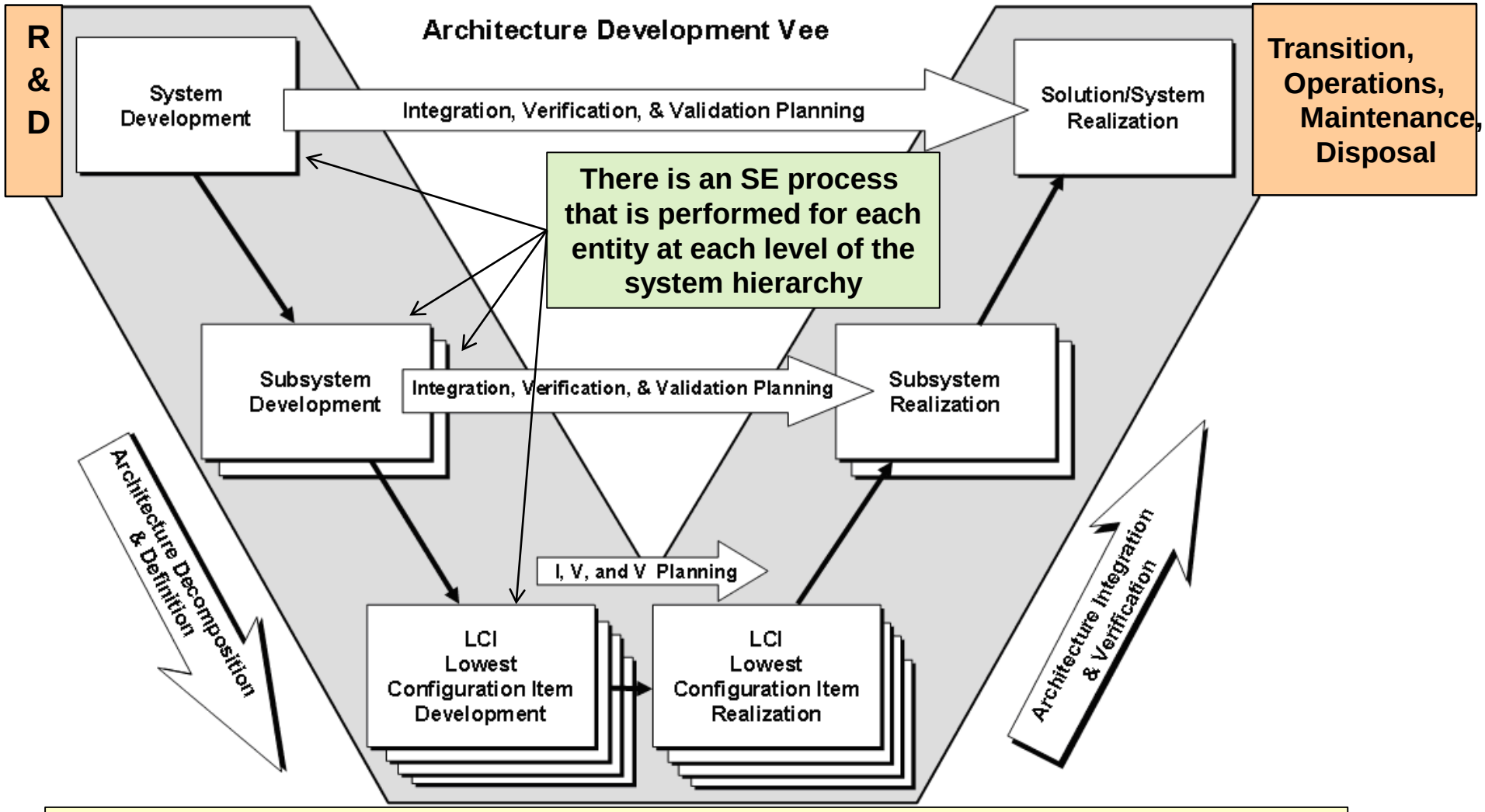
Other Frameworks

- DODAF
- FEA Framework
- MODAF
- Zachman
-
-
-

- Are you familiar with other frameworks?

Architecture frameworks are discussed thoroughly in Session 4, which is based on SYS510, System Architecture and Design

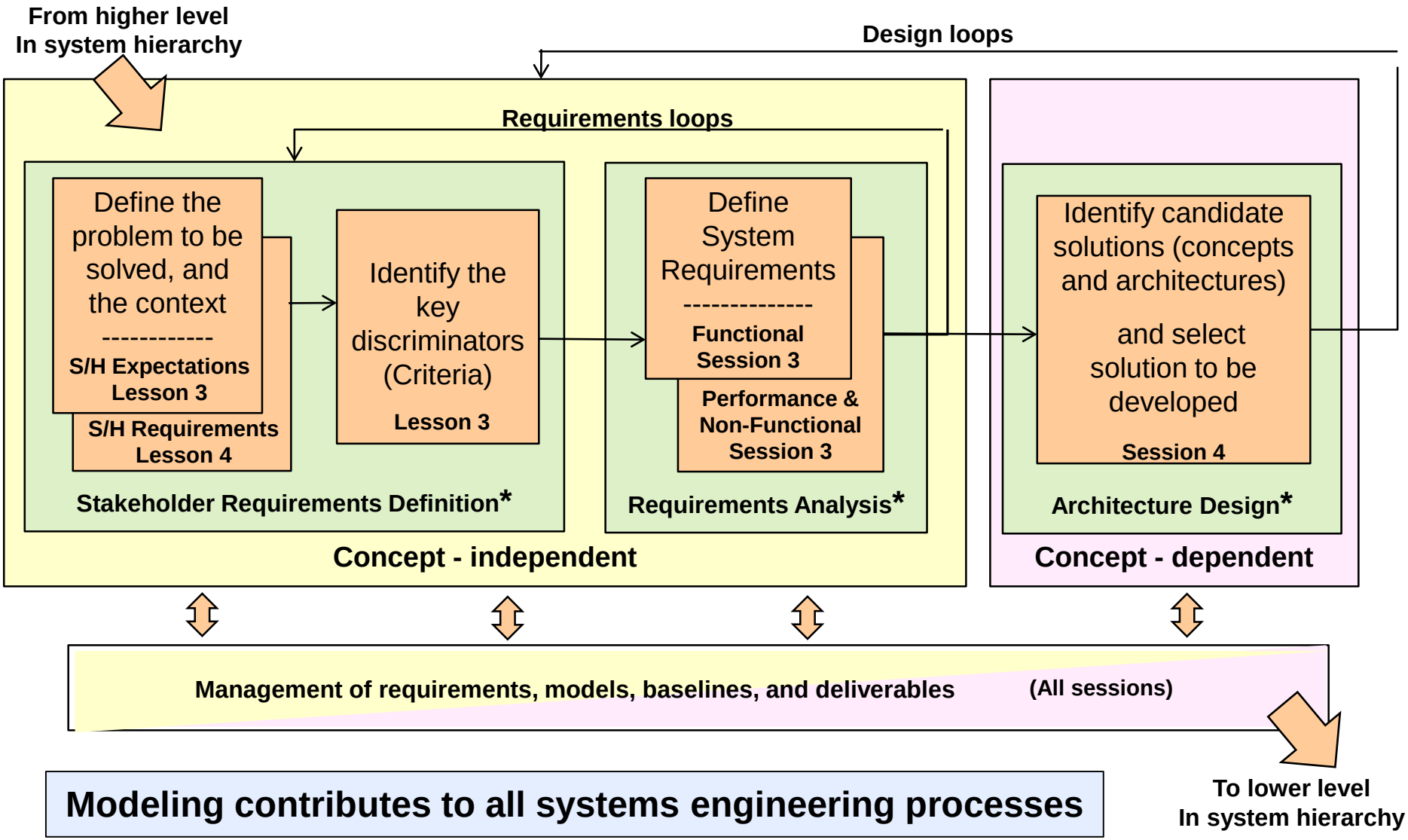
Architectural Vee Process Model



We use models to communicate and understand our processes; Whatever model you use, make sure your team understands it

Adapted from INCOSE SE Handbook, Version 3.2, Figure 3-4

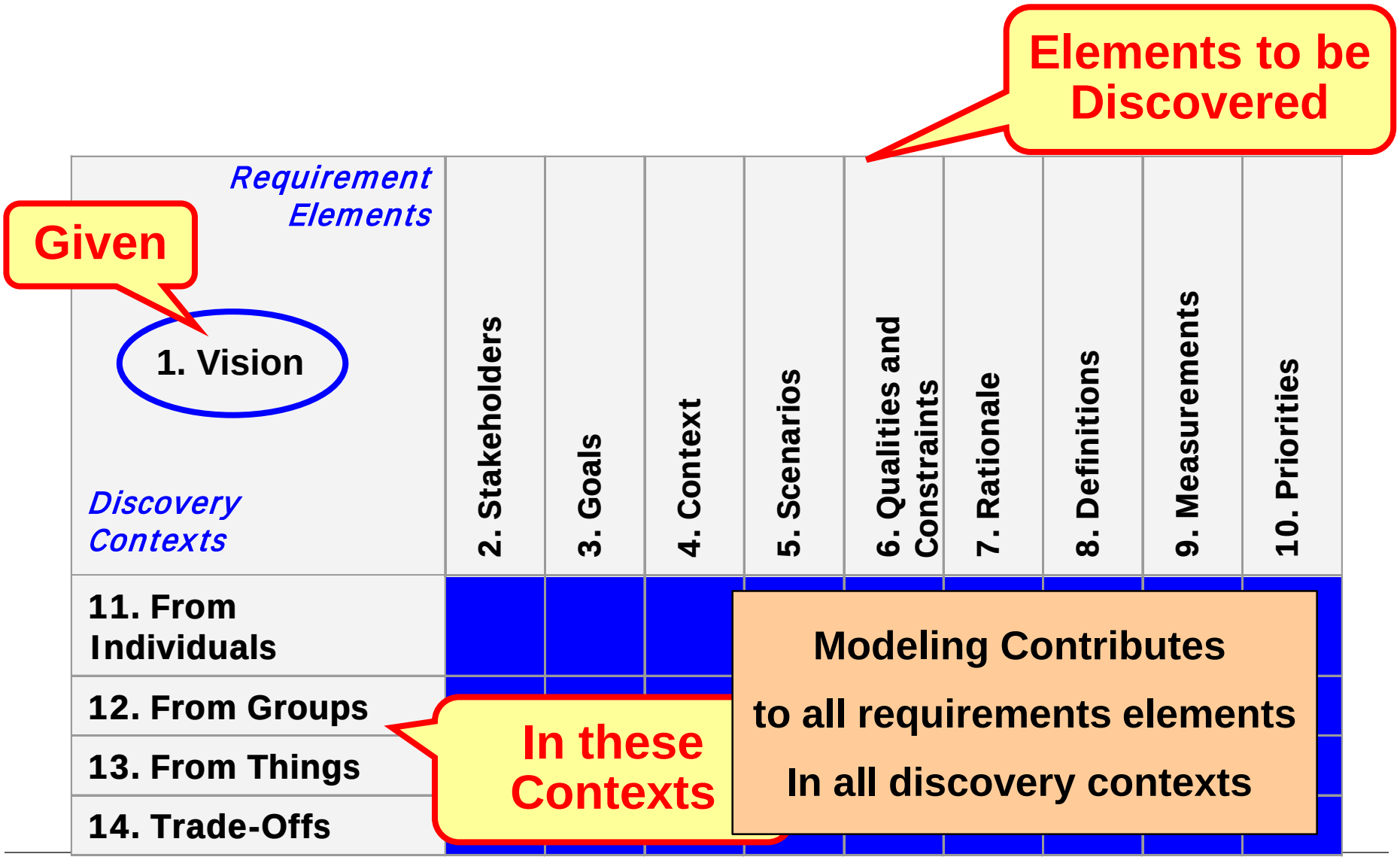
The Requirements Engineering Process Context for this Course (Left Side of Vee)



Modeling contributes to all systems engineering processes

* Indicates INCOSE SE Handbook process name

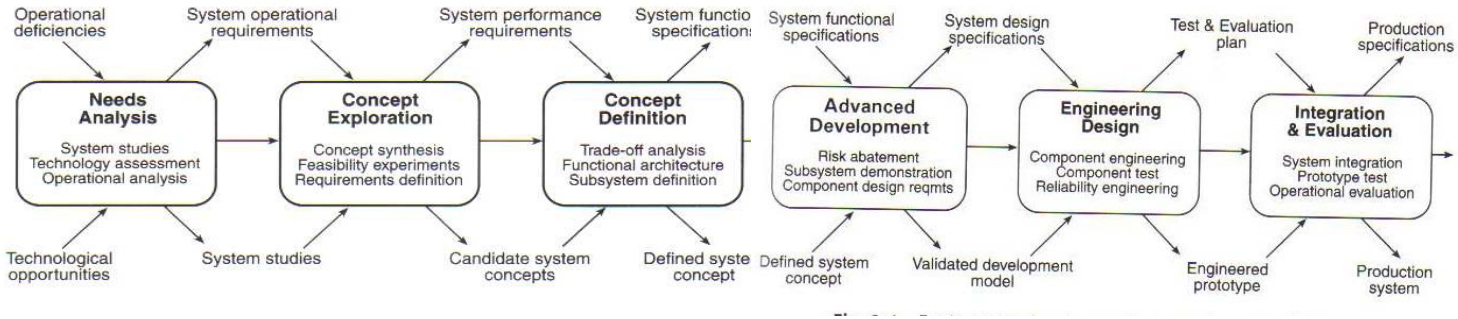
Using "Discovery" to Model the Requirements Engineering Process



Kossiakoff : Build the System Model through the System Life Cycle

Evolving requirements and specifications

⬅======== Concept Development ========➔ ⬅===== Engineering Development =====➔ ⬅ = Post Development =➔



System Production

Operation

Maintenance

Sustainment

Disposal

Evolving system model and system

Every phase/stage results in a system model with increased fidelity

The evolving system model should be subject to configuration control

Slide Adapted from SYS501 (Kossiakoff)

INCOSE'S MODEL BASED SYSTEMS ENGINEERING (MBSE) INITIATIVE

Context for this Course:

- **Focus is requirements engineering**
- **Includes modeling for requirements engineering**
- **Discusses the relationship between requirements engineering and other systems engineering processes**



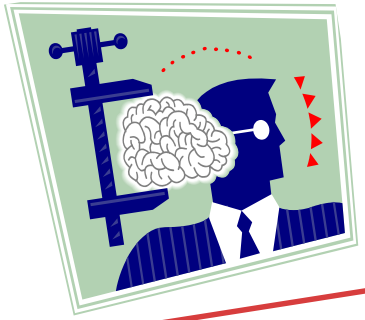
- **MBSE vision for the future is an integrated model of the system, including requirements engineering**

A Summary of Important Modeling Concepts

- Models contribute to communicating and achieving mutual understanding
- A model is a partial representation of reality
- Modeling efforts should be directed to specific goals:
  - Answering specific questions
  - Solving a particular problem
  - Informing a risky decision
- Modeling continues until the goal is effectively and efficiently achieved
- Risk and criticality drive model fidelity and complexity: modeling reduces uncertainty (risk) in the development process
- In the Vee, modeling is usually an “off-core” activity to reduce risk
- The future vision of Systems Engineering is Model Based Systems Engineering (MBSE)

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?



Systems Engineering Technical Short Course

(Session 2: Stakeholder Needs)

Lesson 4: Stakeholder Requirements

Robert B. Wheeler, Instructor

Sessions 2 through 5 Agenda

- **Day 1**
  - Review project team assignments from previous Session
  - Presentation and discussion of Session topic
  - Project team working sessions on the application of the Session topics to their projects
  - Day 1 wrap-up and Q&A

- **Day 2**
  - Review Day 1 topic and project working sessions
  - Presentation and discussion of Session topic (concluded)
  - Project team working sessions (concluded)
  - Check-in with class on the application of the Session topic to their projects
  - Session wrap-up: Q&A and homework due next Session

Timing will be similar to session 1, and will be announced at beginning of each session

Lesson 4 Context:

- Lesson 3 dealt with identifying and engaging stakeholders, and eliciting their expectations
- Lesson 4 extends the lesson 3 activity, but focuses on the analysis of the elicited expectations to form validatable and achievable stakeholder requirements.
- Lessons 3 and 4 together cover the INCOSE Stakeholder Requirements Definition Process
- Performing the activities discussed in lessons 3 and 4 yield a set of solution-independent (concept-independent) stakeholder requirements in user and other stakeholder language
- The resulting stakeholder requirements are the basis for:
  - Developing the technical requirements
  - Session 3 of this Technical Short Course

Session 2 – Stakeholder Needs

Lessons

1. Program Overview Update
2. Introduction to Requirements
3. Stakeholders and their Expectations
4. **Stakeholder Requirements**
5. Special Topics (if time permits)

Lesson Content

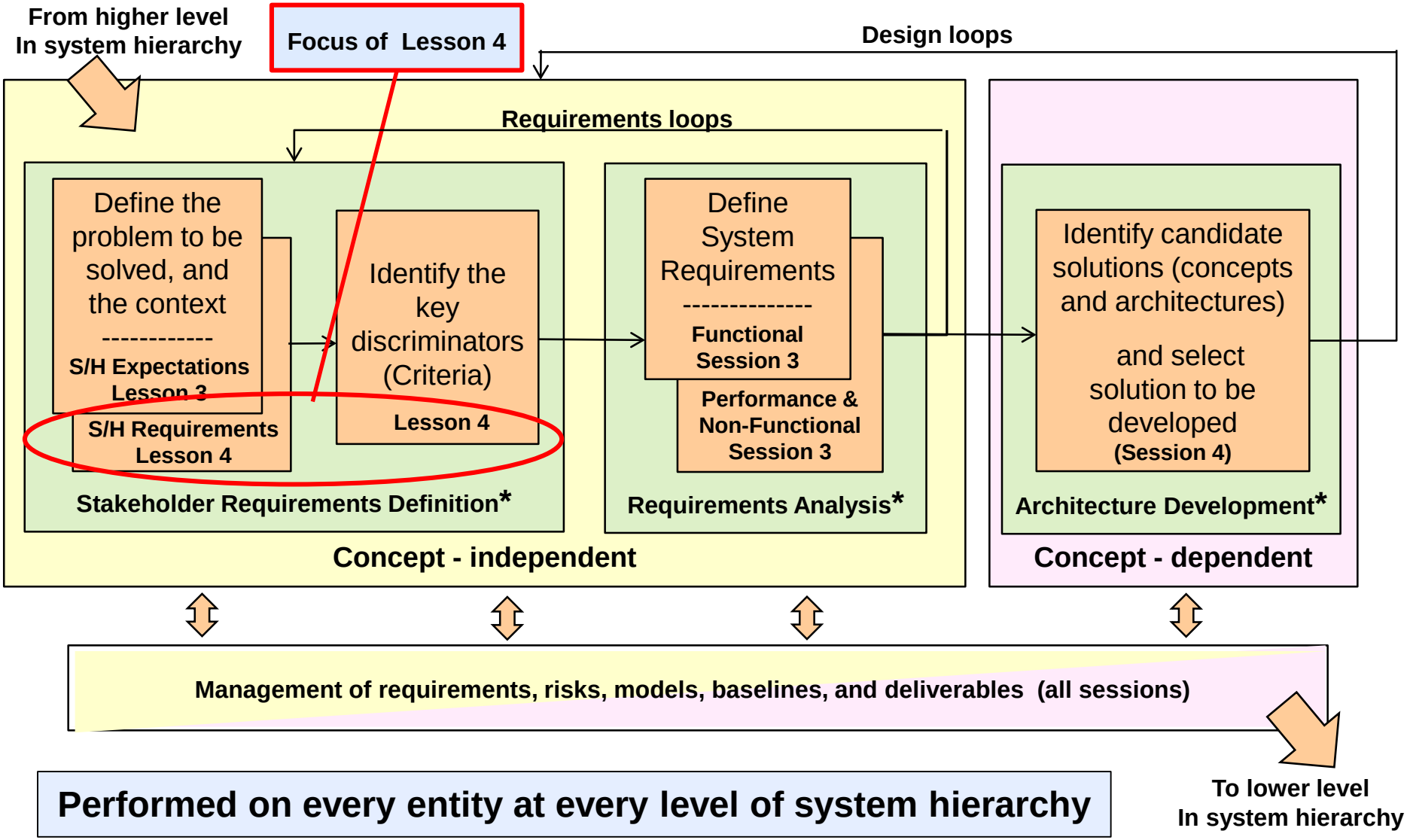
- **Stakeholder Requirements Definition – the process**
- The CONOPS
- Solution Selection Criteria

Appendices*

E. Session 1 Summary
F. A Modeling Exercise
G. Other Appendices TBD

*Appendices A, B, C, and D were contained in Session 1 of the course; remember that Appendix D has a complete course bibliography. The course will reference, but will not cover, the appendices

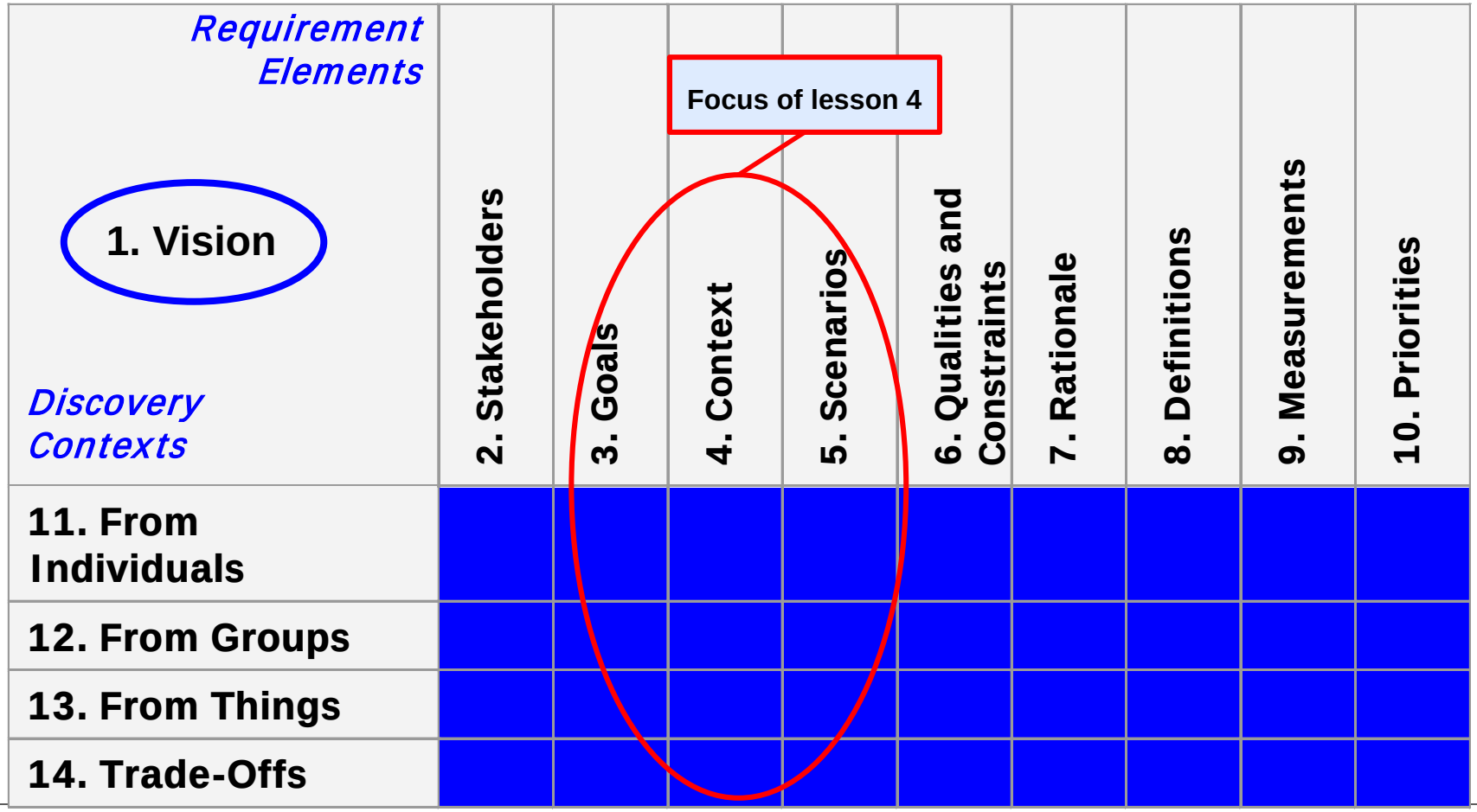
The Requirements Engineering Process Context for this Course (Left Side of Vee)



Performed on every entity at every level of system hierarchy

* Indicates INCOSE SE Handbook process name

Using "Discovery" to Model the Requirements Engineering Process



Discussion: What do Stakeholder Requirements Look Like?

- What do the *expectations* look like?
  - When you first get them?
  - After your elicitation is done?
- Are they readily available? From whom?
- What is the typical format?
- Are they complete? correct?
- Is additional work required to transform these *expectations* into a complete set of stakeholder *requirements*?



Typical "Requirements" Inputs

- A mixture of unlabeled:
  - Expectations
  - Musts and wants
  - Problem, context, and solution
  - Quantified and un-quantified



- Incorrect, incomplete, …..
- Often not written down
- System boundaries unclear
- Not well communicated, and not agreed
- If written, often in textual form rather than graphic
- Seldom in a formal language

This situation can (often does) repeat at each level of system hierarchy

Discussion: What *Should* Stakeholder Requirements Look Like?

- **What should the final result look like?**
  - **The problem to be solved**
  - **The context**
- **What artifacts should exist at the end of this process?**
- **What typically are your results?**



Typical Stakeholder Requirements Outputs

User Requirements

- Operational Need
- Current capability
- Mission
- Support required
- Operators & users
- Criteria for evaluation and selection
- Cost & schedule constraints

Often in a User Requirements Statement

Concept Documents

- Production
- Deployment
- Operations
- Support
- Disposal

Often combined into a User CONOPS

Supporting Data

- Rationale
- Assumptions
- Analysis:
  - Technology
  - Threat
  - Mission
  - Other

Traceability Vehicle:

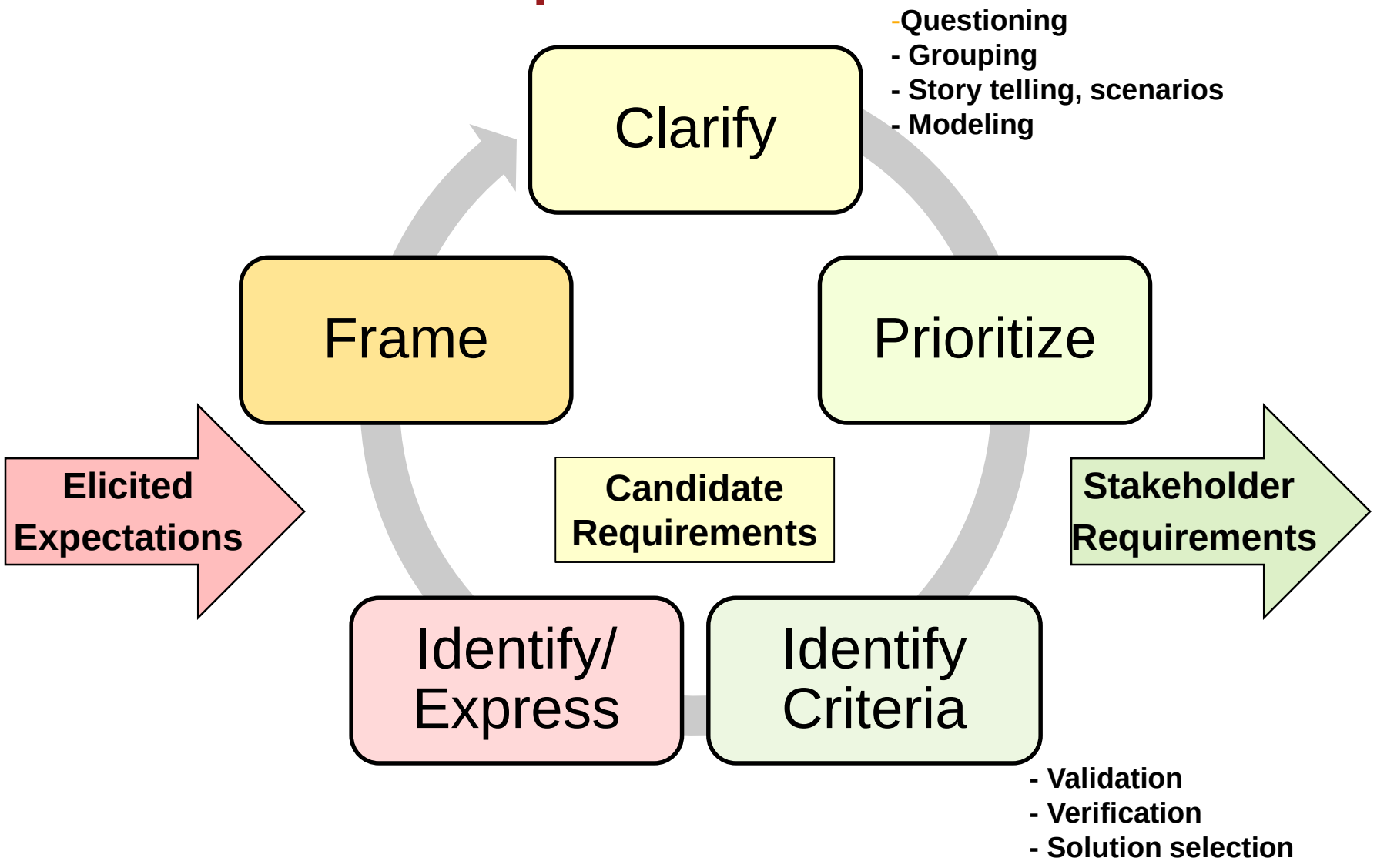
- Traceability to source
- Requirements Verification Traceability Matrix

Remember, these are solution-independent, in stakeholder language

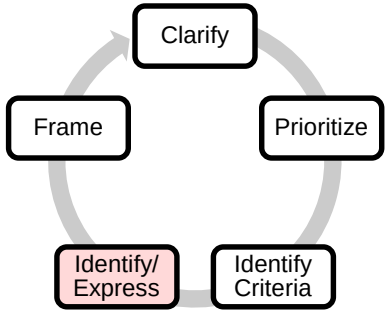
Stakeholder Requirement Artifact Templates

- **Key artifacts include:**
  - **User Requirements Statement**
  - **User CONOPS**
  - **Traceability Matrix (initial)**
- **There are many other artifacts that are**
  - **Equivalent to the above and/or**
  - **That can fit into the above artifacts and make them more useful**

Process to Transform Expectations to Stakeholder Requirements



Identifying and Expressing Requirements



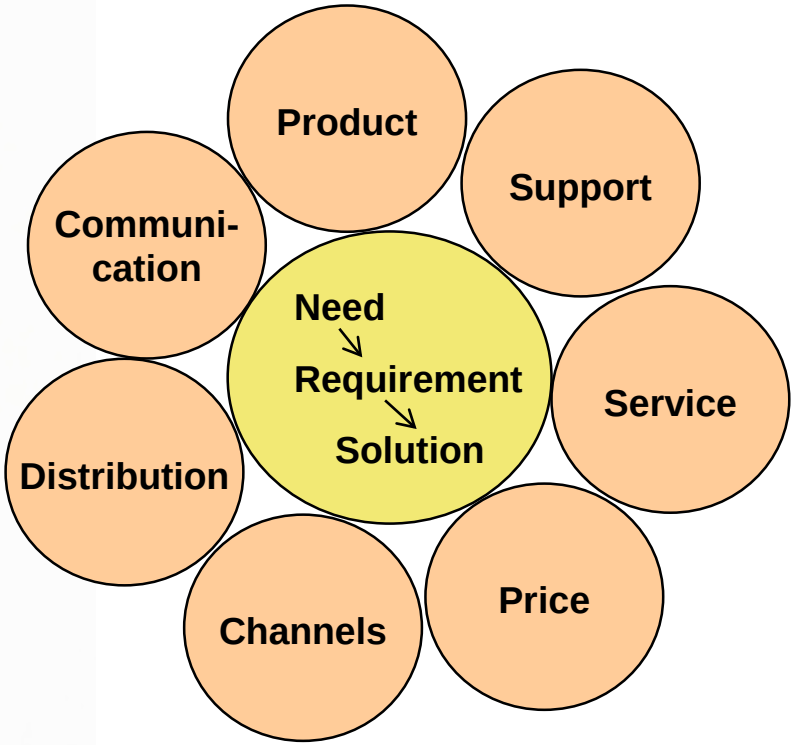
- **Start with the elicited expectations**
- **Remember the characteristics of "good" requirements from lesson #1:**

Requirements shall be

- **Necessary**
- **Implementation independent**
- **Clear and Concise**
- **Complete**
- **Consistent**
- **Achievable**
- **Traceable**
- **Verifiable**
- **Follow language use best practices**

Capture All of the Requirements (1/2)





A daisy is more beautiful if it has all of its petals

You need requirements for every petal on the daisy

You need to track all requirements for every petal on the daisy throughout the system life cycle

IBM used a "Petals on the Daisy" idea for their Enterprise Business Process Reengineering project , circa 1990

Capture All of the Requirements (2/2)

- Start with the elicited expectations
- Do you have inputs from all stakeholders
- Have you considered:
  - Stakeholder, mission, and operational needs
  - Functionality (more in session 3)
  - Non functional requirements (more in session 3):
    - Reliability, maintainability, availability, other “ilities”
    - Integrated logistics support
    - Later phases (e.g., manufacturing, operations and maintenance, disposal)

What are the petals on the daisy for your organization? For your project?

The key to correctly identifying all the requirements is to involve all relevant expertise in requirements capture and analysis; Concurrent engineering, IPTs and IPPDs are best practices

Use a Requirements Template

Unique ID		**Source: Stakeholder:**	
Sys Engineer		**Source: Document:**	
Date:			
Shall Statement			
Rationale			
Validation/Verification strategy			
Traces to			
priority	**Cost**	**Schedule**	**Risk**
Entered by			

Software tools are particularly valuable in providing templates

Traceability Starts Now

- Record the source of every requirement
  - Which stakeholder?
  - What vehicle (email, phone call, meeting, document)?
  - Is there a higher-level "parent"
- Maintain records of the source of every evolving requirement
- Trace all "children" to "parents"

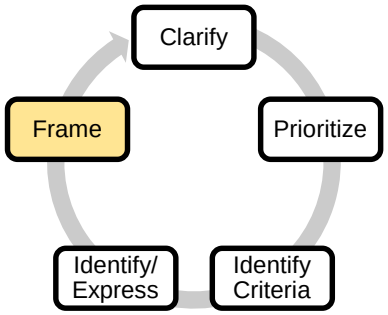


Later traceability should be evolved to facilitate requirements management, validation and verification planning and recording, and change management.

Traceability will be discussed later in more detail.

Exercise (1/2)

Bob, married with three children, needs to replace the family automobile, an aging minivan. An independent consultant, Bob travels two weeks each month by air across country to see and perform work for clients. During these business trips, the minivan is parked at the airport. When not traveling, Bob works from an office in the home. He teaches one night a week when home at a local community college, about 30 miles a way from home. He is active in the community and "on the go" when home. The family uses the minivan for regular vacations and family visits in nearby states. Bob's wife, a local business professional, works locally. She has her own small car, and uses it daily to drive the kids all over the place, and to drive herself to and from work. Bob would prefer to improve on the 18 miles per gallon minivan, but needs the versatility it provides.





Exercise question will be uncovered during the lecture

Exercise (2/2)

Slide will be uncovered during the lecture

Frame the Requirement Properly (1/2)

- **Framing: Setting an approach or query within an appropriate context to achieve a desired result or elicit a precise answer. *(businesswords.com)***
- **For a requirement, this means ensuring the frame does not preclude a legitimate solution**
- **For the exercise just completed, did your group frame the "problem to be solved" such that the instructor-selected solution in the case was precluded?**

Frame the Requirement Properly (2/2)

Select the best frame for your requirements at every level to ensure communicating what you mean, and avoid un-intended implications:

- **Use the language of the stakeholder; avoid developer language**
- **Be aware of the impact of language used on the meaning conveyed**
- **Use alternate words, phrases, models, and examples when discussing the requirement with stakeholders.**
- **Use the clarification techniques discussed later in this lesson:**
  - **The five whys**
  - **The 4 whos**
  - **The who, what, when, where, why, and how questions**

Systems thinking is your friend when framing requirement statements. Why is this so?

Exercise: Framing: Part 2

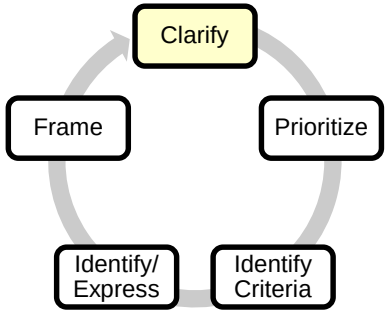
Slide will be uncovered during the lecture



The instructor will provide exercise directions

Clarify Requirements

- In a sense, this whole lesson is about clarifying
- Clarifying includes anything we can do to better understand the requirement from the view of the stakeholders
- In this section we cover:
  - Questioning
  - Grouping
  - Modeling
  - Story telling
  - Scenarios



Questioning (1/3)

- General questions
  - Context-independent
  - Useful in any context
  - Particularly useful when you don't know what else to ask
  - Checklists work well
- Context questions (Next slide)



- Sample General questions
  - Am I on the right track?
  - Are your answers official, or do I need to get approval from someone else?
  - Do you have a particular process or methodology that you like to use?
  - Do we need to draw a picture? OR Can you explain this picture you drew?
  - Can you think of something I should have asked?
  - Do you have questions for me?
  - Can we talk again later?

Questioning (2/3)

- Context - dependent questions
  - Tailored to the specific problem context
  - Framed to better understand the context
  - Many checklists exist
    - Internet
    - Text books
    - Professional organizations
    - Standards (organizational, company, government agency)
  - Often built into tools and methodologies
  - Hierarchical process
    - Questions lead to more detailed questions





The 4 whos



The 5 whys

Questioning (3/3)

Examples of context - dependent questions

- **Who will use (operate, maintain, support, sell,…..) the system?**
- **How will the system be used (operated, maintained, supported, sold, ……..)?**
- **Where will the system be …… (made, tested, operated, maintained, stored, disposed to…….)?**
- **Why is this system needed?**
- **When will this system be ….. (developed, manufactured, deployed, achieve IOC*, deactivated,…..)?**
- **Under what weather conditions will this system be……?**



*** IOC = Initial Operating capability**

Exercise: Develop Questions for the Waters Project Teams

- **The instructor will form 4 exercise teams**
  - **Each exercise team will be co-located**
- **Each exercise team will be assigned to write questions for one of the waters project teams**
  - **The intent is that no one will be writing questions for their own project**

This Exercise Team	Asks Questions about this waters Project
Ex Team 1	1. Systems Controller
Ex Team 2	2. NEXUS
Ex Team 3	3. Smart Consumables
Ex Team 4	4. Whistler

Exercise directions:

1. **Each exercise team meets and comes up with 5 good questions for the project identified in the table on this slide**
2. **Each exercise team will brief at least 1 of their questions to the class**
3. **The resulting questions will be provided to the project team for consideration in their project work**
4. **The receiving project team will provide feedback regarding the value of the received questions to their project (later)**

Grouping

- Grouping is used to better understand the stakeholder requirements
- Can be done many ways:
  - By type
  - By expected physical location
  - By time phase or sequence
  - By who will be the actor

- The value of grouping to requirements engineering is:
  - Improved communication and understanding
  - Enhanced by using more than one way to group requirements
  - Enhanced by performing grouping as a team activity

Grouping Using Affinity Diagrams (1/2)

From Wikipedia, the free encyclopedia:

- **The affinity diagram is a business tool used to organize ideas and data. It is one of the Seven Management and Planning Tools.**
- **The tool is commonly used within project management and allows large numbers of ideas stemming from brainstorming to be sorted into groups, based on their natural relationships, *(affinities)* for review and analysis.**
- **The affinity diagram was devised by Jiro Kawakita in the 1960s[3] and is sometimes referred to as the KJ Method.**

The Seven new Management and Planning Tools have their roots in Operations Research work done after World War II and the Japanese *Total Quality Control* (TQC) research

Grouping Using Affinity Diagrams (2/2)

Process

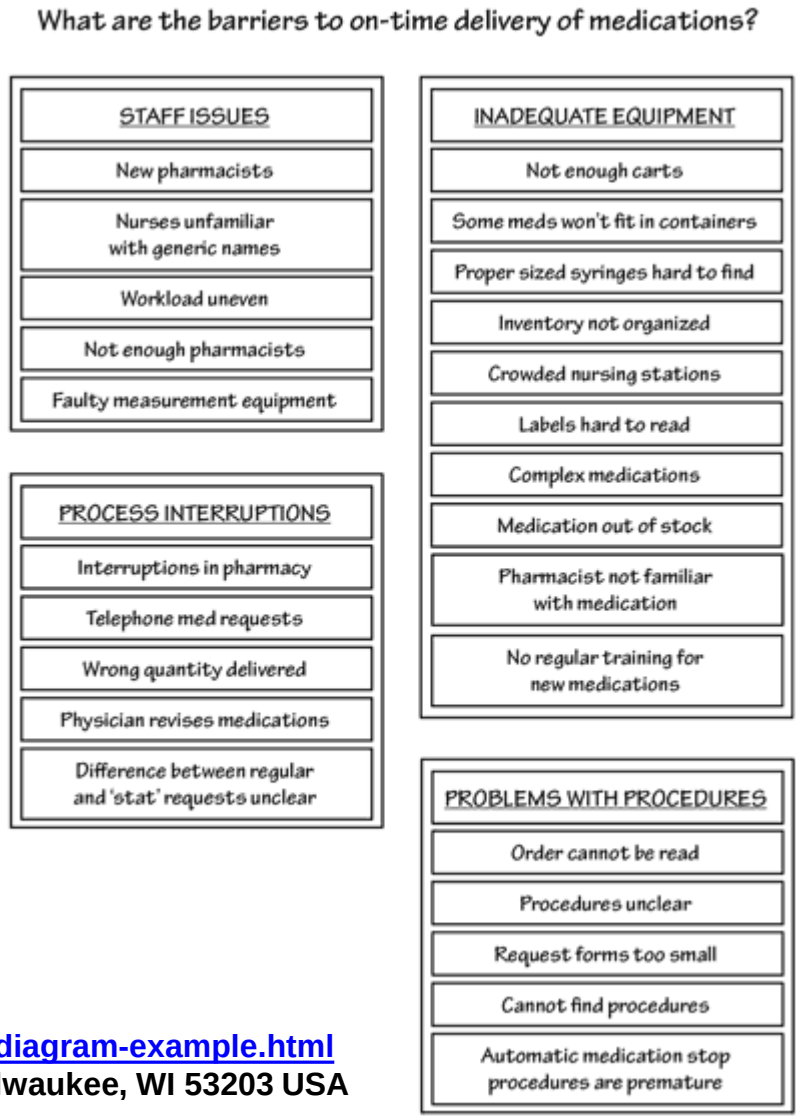
- Record each idea on cards or notes
- Look for ideas that seem to be related
- Sort cards into groups until all cards have been used.
- Once the cards have been sorted into groups the team may sort large clusters into subgroups for easier management and analysis

For Requirements, Affinity diagrams help:

- Fill gaps
- Ensure completeness & consistency
- Stakeholder involvement
- Group ownership of the list

Affinity Diagram Example

- Example is from health care field
- Items are first identified, then grouped.
- Group titles often come last
- There is no unique way of grouping
- Whole purpose is to develop an understanding that is reasonably complete and consistent, and that represents an agreement of multiple points of view



Source: http://asq.org/healthcare-use/links-resources/affinity-diagram-example.html
American Society of Quality, 600 North Plankinton Avenue, Milwaukee, WI 53203 USA

Optional Exercise: Affinity Diagrams

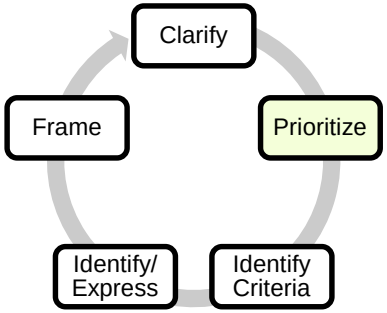


Some project groups might want to do this exercise for their projects, even if we do not do it in class

Other Clarification Techniques

- **Modeling**
- **Storytelling**
- **Scenarios**

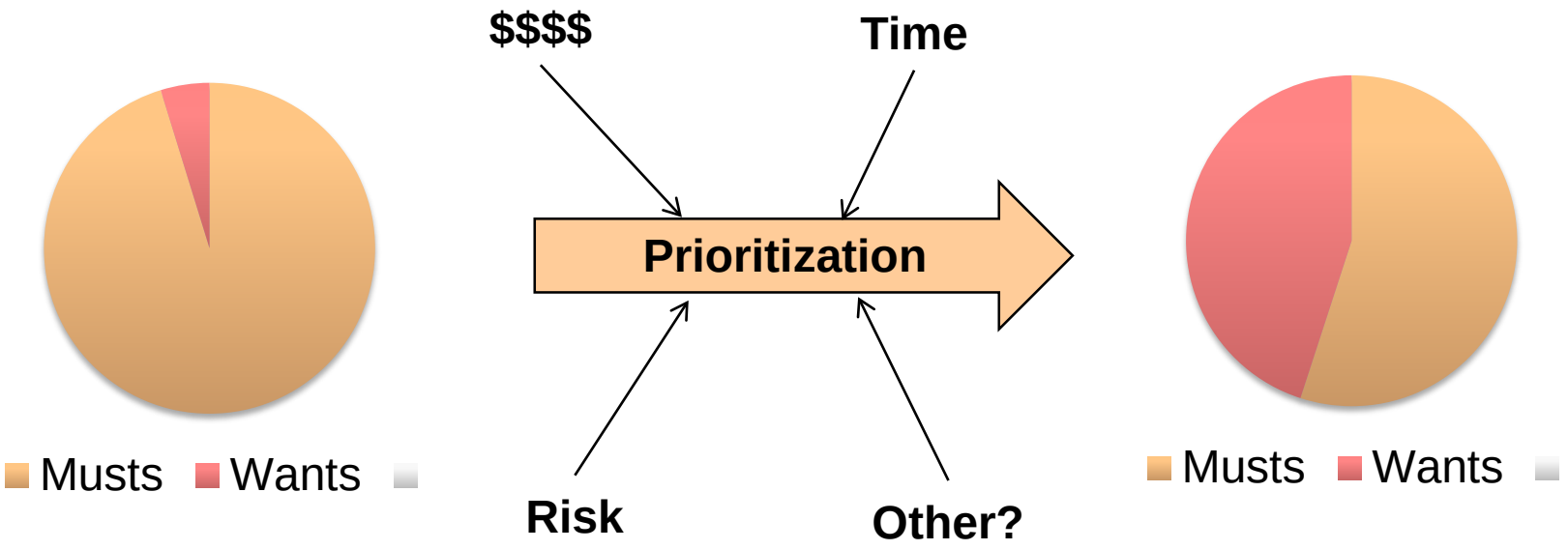
Prioritization



- **Why should we prioritize requirements?**
- **What are the implications of prioritizing requirements?**
- **How can requirements be prioritized?**
- **What does Alexander mean by “Input Priorities” and “Output Priorities”?**



Prioritization

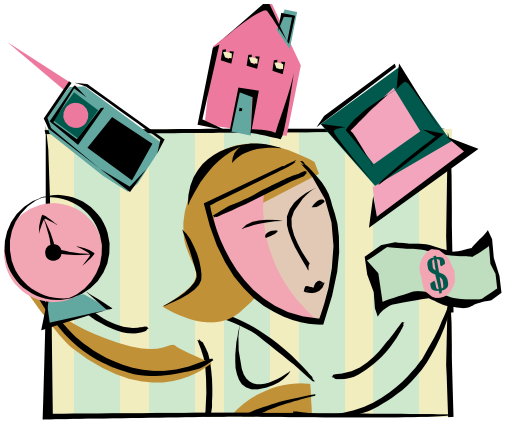


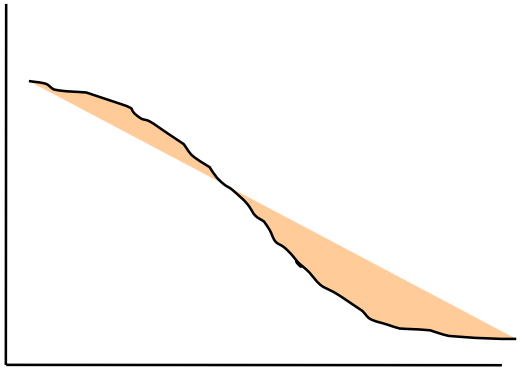
Everything is a must…..

Until the associated cost, schedule, and risk are considered

Prioritization

- Identify the “must haves”
  - Treat them as mandatory
  - Everything else is a “want”
- Rank or weight the remaining “wants”
- Try to balance the musts and wants:
  - To many musts limits the possibilities (precludes solution options)
- Prioritization should reflect the needs of the stakeholders in terms of their (stakeholder) values
- Value or utility diagrams are useful





Use "Cost" to Help Separate Musts and Wants

Every candidate requirement is a "Must" until we can understand the tradeoffs associated with achieving that candidate requirement

If the candidate requirement becomes an approved requirement:

- **How much will it cost?**
- **How long will it take?**
- **How much risk will we have to accept?**
- **What will we give up if we do not include this requirement**

An informed stakeholder might be willing to trade-off some requirements in favor of reduced cost, schedule, or risk

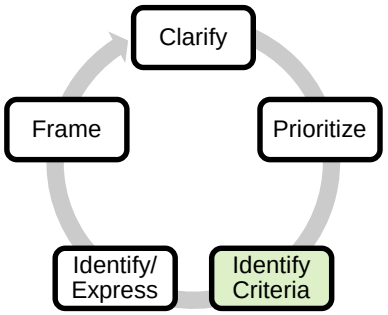
Can we "cost" a solution-independent requirement?

Can we "schedule" a solution-independent requirement?

Can we "assess risk" for a solution-independent requirement?

Identify Validation Criteria

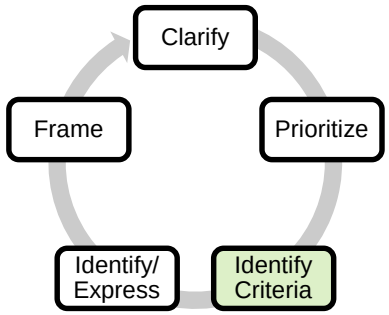
- Requirements are not complete or correct until criteria have been defined for:
  - Validating the solution-independent requirements
  - Verifying solution-dependent requirements
- Often the validation or verification criteria completes the requirement definition
- At-completion validation is best done by real users in the operational environment
- In-process validation is done throughout development cycle to increase probability of successful at-completion validation
- Validation is planned as requirements are defined
- Techniques include:
  - RVTM, RTVM, Traceability of various forms
  - “House of Quality” (Quality Function Deployment – QFD)
  - Modeling



Can you identify how modeling might contribute to validation?

Identify Validation Criteria

- In-process validation is a continuing process
- Buede describes this nicely in Chapter 11.2
  - Conceptual validity
  - Requirements validity
  - Design validity
  - Operational validity (culminates in at-completion validation)
- This is covered in, and is the subject of an important homework assignment in Sys510, System Architecture & Design, and won't be further discussed here





Session 2 – Stakeholder Needs

Lessons

1. Program Overview Update
2. Introduction to Requirements
3. Stakeholders and their Expectations
4. **Stakeholder Requirements**
5. Special Topics (if time permits)

Lesson Content

- Stakeholder Requirements Definition
- **The CONOPS**
- Solution Selection Criteria



Appendices*

E. Session 1 Summary
F. A Modeling Exercise
G. Other Appendices TBD

*Appendices A, B, C, and D were contained in Session 1 of the course; remember that Appendix D has a complete course bibliography. The course will reference, but will not cover, the appendices

Concept of Operations – Definitions (1/3)

NASA Systems Engineering Handbook:

The CONOPS describes how the system will be operated during the life-cycle phases to meet stakeholder expectations.

It describes the system characteristics from an operational perspective and helps facilitate an understanding of the system goals.

It stimulates the development of the requirements and architecture related to the user elements of the system.

It serves as the basis for subsequent definition documents and provides the foundation for the long-range operational planning activities.

Concept of Operations – Definitions (2/3)

INCOSE Systems Engineering Handbook:

The ConOps document, sometimes called the Operational Concept Document (OCD), defines the way the system will be used and must involve the input from a broad range of stakeholders, such as operations, maintenance, and management personnel.

The document also defines any critical, top-level performance requirements or objectives (stated either qualitatively or quantitatively) and system rationale, and contains a preliminary functional block diagram of the system with only the top-level functional "threads" specified, as well as the roles and responsibilities and the set of skills needed for operations and maintenance of the system.

The INCOSE SE Handbook described, but did not specifically *define* the CONOPS

Concept of Operations – Definitions (3/3)

Buede: Operational Concept:

Vision for what the system is (in general terms), a statement of the mission requirements, and a description of how the system will be used.

The shared vision is based on the perspective of the system's stakeholders of how the system will be developed, produced, deployed, trained, operated and maintained, refined and retired to overcome some operational problem and achieve the stakeholders operational needs and objectives.

The mission requirements are stated in terms of measures of effectiveness. The operational concept includes a collection of scenarios (one or more for each group of stakeholders in each relevant phase of the systems life cycle)

Buede did not define the CONOPS, but instead defined the Operational Concept

User CONOPS vs. System CONOPS

***User Concept of Operations*:**

- **This solution-independent artifact describes the operational and support scenarios and target operational and support environments for the system being defined.**
- **Describes how *any solution* would be used, prior to (independent of) solution selection decisions**

***System Concept of Operations*:**

- **This solution-dependent artifact that describes how the *selected solution* will be used to solve the problem to be solved in the target environment**

If your project identifies only one *CONOPS*, a best practice is to develop that *CONOPS* initially during the definition of Stakeholder Requirements as a User CONOPS, and then evolve that document to a System CONOPS as part of the Architecture Design Process.

In this lesson we are focused on the solution-independent USER CONOPS

Discussion:

- Who prepares the CONOPS?
  - Government or contractor
  - Buyer or seller
  - Needer or responder?



- What other artifacts could be included in the CONOPS?
- Is the CONOPS part of the system baseline?
  - Should it be under “change control”?

CONOPS Typical Content

The *CONOPS* should consider all aspects of operations including integration, test, and launch through disposal. Typical information contained in the *CONOPS* includes

- **a description of the major phases;**
- **operation timelines;**
- **operational scenarios;**
- **end-to-end communications strategy;**
- **command and data architecture;**
- **operational facilities;**
- **integrated logistic support (resupply, maintenance, and assembly);**
- **and critical events.**

The operational scenarios describe the dynamic view of the systems' operations and include how the system is perceived to function throughout the various modes and mode transitions, including interactions with external interfaces.

Tips on Developing a CONOPS

1. Include, but differentiate between, the *User CONOPS* (solution-independent) and the *System CONOPS* (solution-dependent for each candidate solution).
2. Include all operational and support scenarios.
3. Include all the operational and support environments.
4. The INCOSE SE Handbook suggests writing five concept documents:
   - Concept of Production,
   - Deployment,
   - Operations,
   - Support,
   - and Disposal.
5. Start with your organization template, and tailor appropriately to your project and system of interest
6. The *System CONOPS* is developed as a part of the Architecture Design Process *for each candidate solution*, and then further defined for the selected solution.
7. A CONOPS is required at every level in the system hierarchy; if your system is part of a larger system, then your CONOPS must consider your super system as part of your environment

CONOPS Guidance

- **For more information about developing the *CONOPS*, see ANSI/AIAA G-043-1992, *Guide for the Preparation of Operational Concept Documents***
- **The Department of Defense Architecture Framework (DoDAF) views can also be used to describe the system operational concept.**

Architecture Framework	Focus	Reference
Department of Defense (DODAF)	DOD Enterprise Architecture	http://www.defenselink.mil/cio-nii/policy/eas.shtml[4]
Ministry of Defense (MODAF)	Ministry of Defense (UK) enterprise architecture	www.modaf.org.uk[5]
Zachman	Early, general purpose	http://www.zifa.com//[6]
The Open Group (TOGAF)	Information technology enterprise architectures	http://www.opengroup.org/public/arch[7]
Federal Enterprise (FEA)	Shared development of common processes and information among Federal Agencies and other government agencies	http://www.feacinstitute.org//[8]

CONOPS Summary

- See the course texts and references for approaches to preparing a CONOPS
  - The NASA and INCOSE SE Handbooks are good sources
  - Identify your organizational standards. If any
- Remember that there are many different definitions and approaches for the CONOPS
  - Choose and tailor the one that makes most sense for your system of interest and your development methodologies
- Don’t underestimate the value of the research you did for Assignment #1 in this

The User CONOPS should describe how any system will be used* to solve the “problem to be solved” in the target context. It answers the “who, what, where, when, why, and how” questions. It describes the “context of implementation” of your system of interest. It is solution-independent within agreed constraints.

Optional Exercise: Scenarios or Use Cases



Instructor will provide directions for this exercise if time permits

Session 2 – Stakeholder Needs

Lessons

1. Program Overview Update
2. Introduction to Requirements
3. Stakeholders and their Expectations
4. **Stakeholder Requirements**
5. Special Topics (if time permits)

Lesson Content

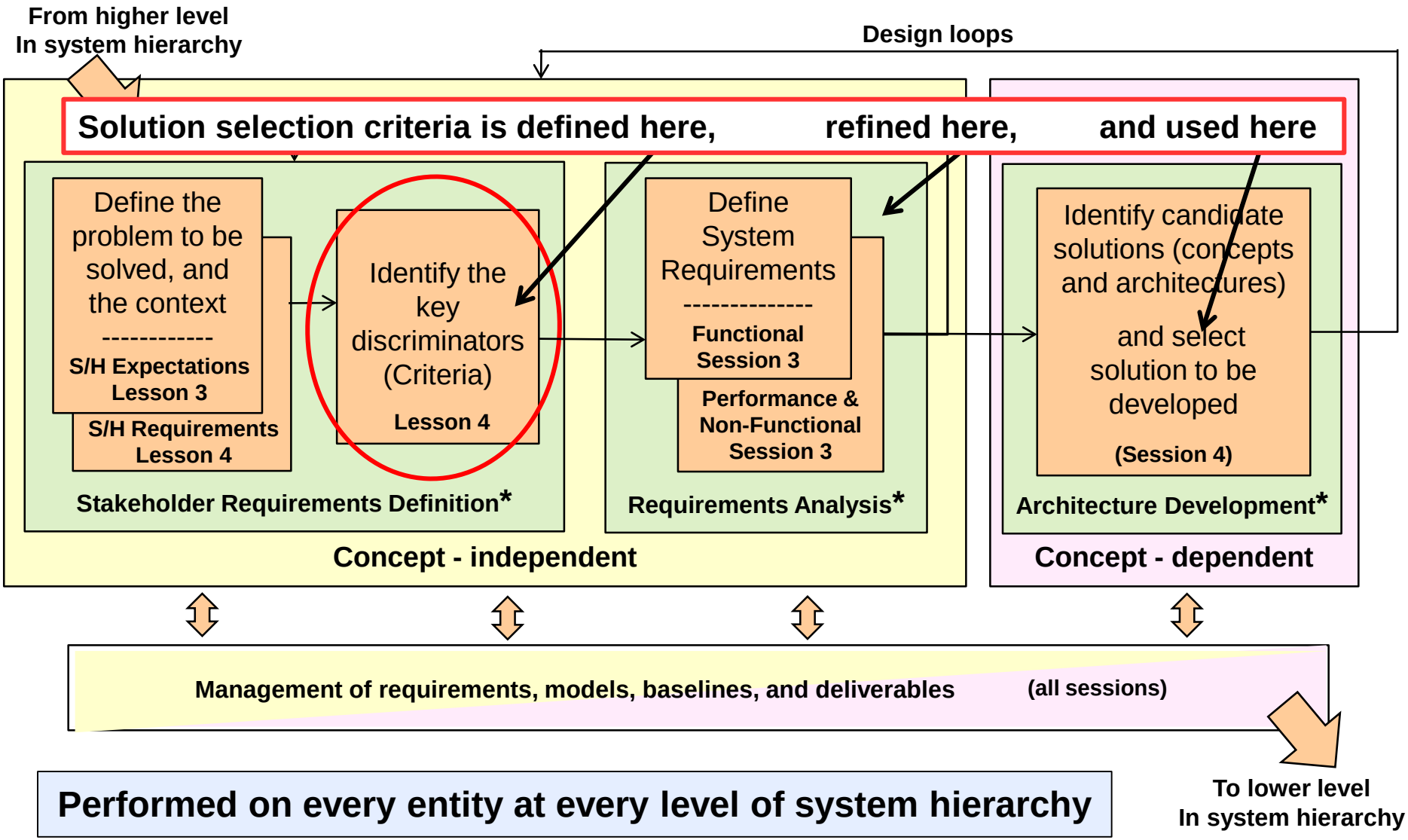
- Stakeholder Requirements Definition
- The CONOPS
- **Solution Selection Criteria**

Appendices*

E. Session 1 Summary
F. A Modeling Exercise
G. Other Appendices TBD

*Appendices A, B, C, and D were contained in Session 1 of the course; remember that Appendix D has a complete course bibliography. The course will reference, but will not cover, the appendices

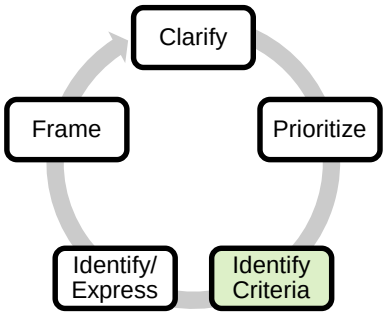
The Requirements Engineering Process Context for this Course (Left Side of Vee)



Performed on every entity at every level of system hierarchy

* Indicates INCOSE SE Handbook process name

Identify Solution Selection Criteria



- **Requirements are not complete or correct until criteria have been defined for:**
  - **Selecting the best solution (the best concept and the best architecture that solve the set of requirements)**
- **Techniques include:**
  - **MOEs/MOPs/KPPs**
  - **Objectives hierarchy**
  - **"House of Quality" (Quality Function Deployment – QFD)**
  - **RVTM, RTVM, various forms of "traceability matrix"**
  - **Cost as an Independent variable**
  - **Modeling techniques**
- **Developing solution selection criteria always starts with defining Musts and Weighted Wants, as discussed elsewhere in this course**



Solution Selection Criteria: Considerations (1 of 2)

Forsberg and Mooz* advocate a solution development and selection process that focuses:

- **first on the solution concept (the general idea of the solution), and**
- **then on the solution architecture (the entities making up the solution and their interfaces and relationships**

Their reasoning is that the selection criteria for each of these are different.

- **Solution concept selection criteria should be based on requirements that are the most critical to the success of the solution, and that are the hardest to achieve,**
- **Once the concept is selected, solution architecture, selection criteria should be based on ease of solution management (i.e.; ease of integration, ease of production, ease of upgrade, etc.)**

***Forsberg and Mooz, *Visualizing Project Management*, Third edition, John Wiley and Sons, 2005**

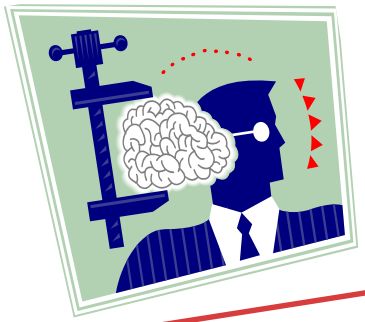
Solution Selection Criteria Considerations (2 of 2)

- How many requirements (number of shall statements) are typical for your projects/systems of interest?
- Is it practical to trade-off candidate solutions using hundreds of requirements?
- Best Practices:
  - Identify those critical requirements that are hardest to achieve, and use them to "discriminate" between the candidate solution concepts*
  - Identify those requirements that relate to ease of solution management and use them to discriminate between the candidate architectures (entities and their relationships)
- Discriminators always relate to one or more of the following: cost, schedule, technical, quality, or risk & opportunity

Where in the process model that we are using to represent solution development (left-side of the Vee) do we develop solution selection criteria?

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?

Systems Engineering Technical Short Course

(Session 2: Stakeholder needs)

Appendix E:
Session 1 Review

Robert B. Wheeler, Instructor

Session 2 Content - Stakeholder Needs

Lessons

1. **Program Overview Update**
2. **Introduction to Requirements**
3. **Stakeholders and their Expectations**
4. **Stakeholder Requirements Analysis**
5. **Special Topics (if time permits)**

Appendices*

- **E. Session 1 Summary**
- **F. A Modeling Exercise**
- **G. Other Appendices TBD**

*Appendices A, B, C, and D were contained in Session 1 of the course; remember that Appendix D has a complete course bibliography. The course will reference, but will not cover, the appendices

What is Systems Engineering?

Per INCOSE

Systems Engineering is an interdisciplinary approach and means to enable the realization of successful systems

It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem e.g.,

- **Operations**
- **Cost & Schedule**
- **Performance**
- **Training & Support**
- **Test**
- **Disposal**
- **Manufacturing**

Systems Engineering integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation. Systems Engineering considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs

From the INCOSE web site – www.incose.org – underlining and red font added

What is a System?

- A system is a construct or collection of different elements that together produce results not obtainable by the elements alone
- The elements, or parts, can include people, hardware, software, facilities, policies, and documents, that is, all things required to produce systems-level results
- The results include system level qualities, properties, characteristics, functions, behavior and performance
- The value added by the system as a whole, beyond that contributed independently by the parts, is primarily created by the relationship among the parts, that is, how they are interconnected

What Is a System?







System Hierarchy

- **System:** An entity which performs a significant useful service
- **Subsystems:** A major interacting elements of a system, each of which perform a subset of closely related system functions
- **Components:** The middle level of system elements which comprise a subsystem (Configuration Items)
- **Subcomponents:** Perform elementary functions and are composed of several parts
- **Parts:** The smallest primitive elements of a subcomponent

Many conventions are used; you should use the ones common to your environment

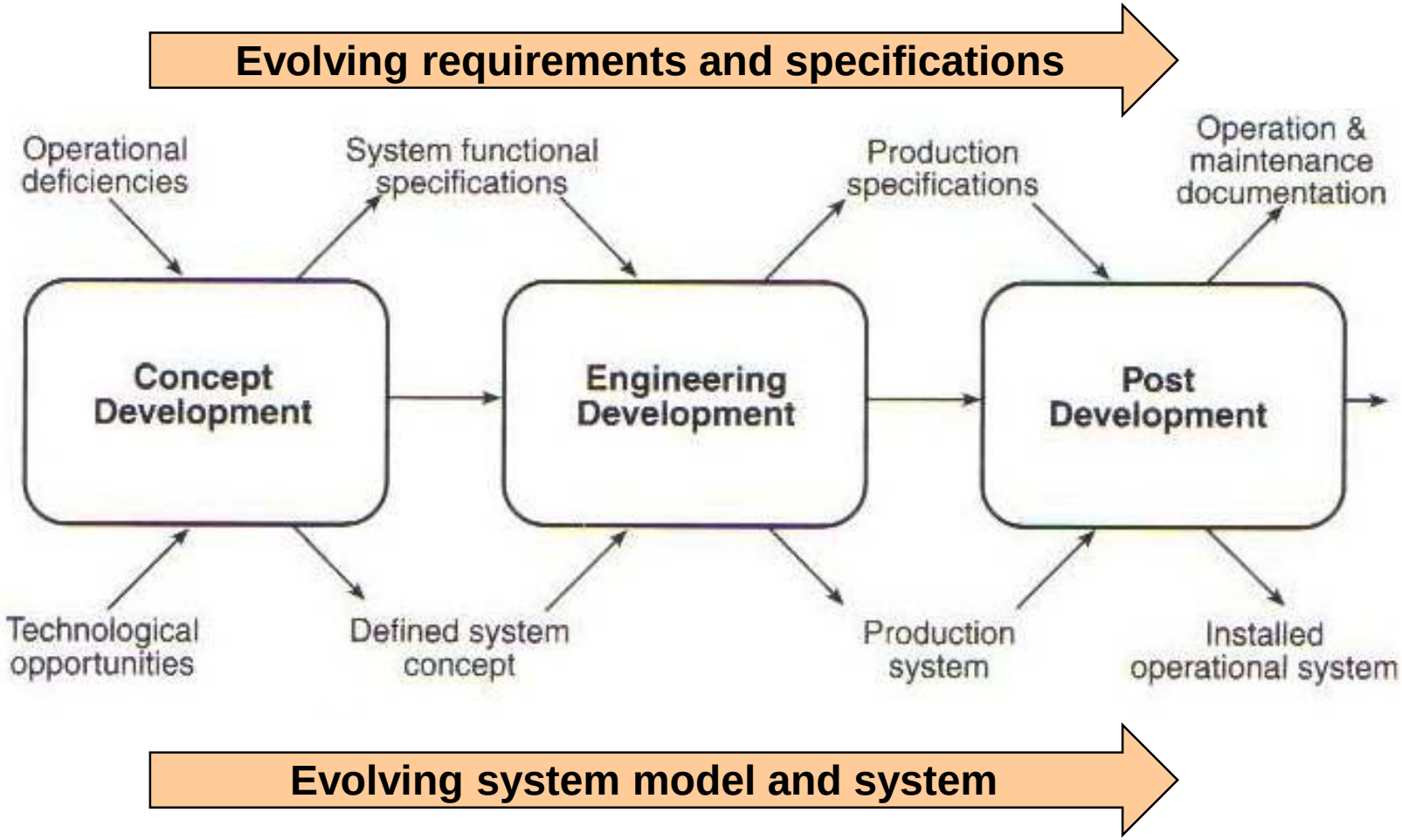
In this course, the term “entity” is used to refer to any level in the system hierarchy

Motivation for Systems Engineering

- There is tremendous potential for wasted effort on large projects, since their development requires that many subsystems be developed in parallel
- Without a clear understanding of what must be done for each subsystem the development team runs the risk of inconsistent designs, conflicting interfaces, or duplication of effort
- Systems engineering provides a systematic, disciplined approach to defining, for each member of the development team, what must be done for success

What does this have to do with Requirements Engineering?

System Life Cycle (per Kossiakoff)



Life Cycle Model Qs & As

3. When do we define what we build, test, and deploy?

2. When do we build and test what we deploy?

1. What do we do here?

Produce, transition, operate, maintain, support, secure, ……. , dispose

DEFINE	BUILD/TEST	DEPLOY
Pre-development* Phase Concept definition, Requirements engineering, and Architecture	Development Phase Design and development, System integration, Test and evaluation	Post-development Phase System implementation, Operations & Maintenance, and Transition

*"Concept" development

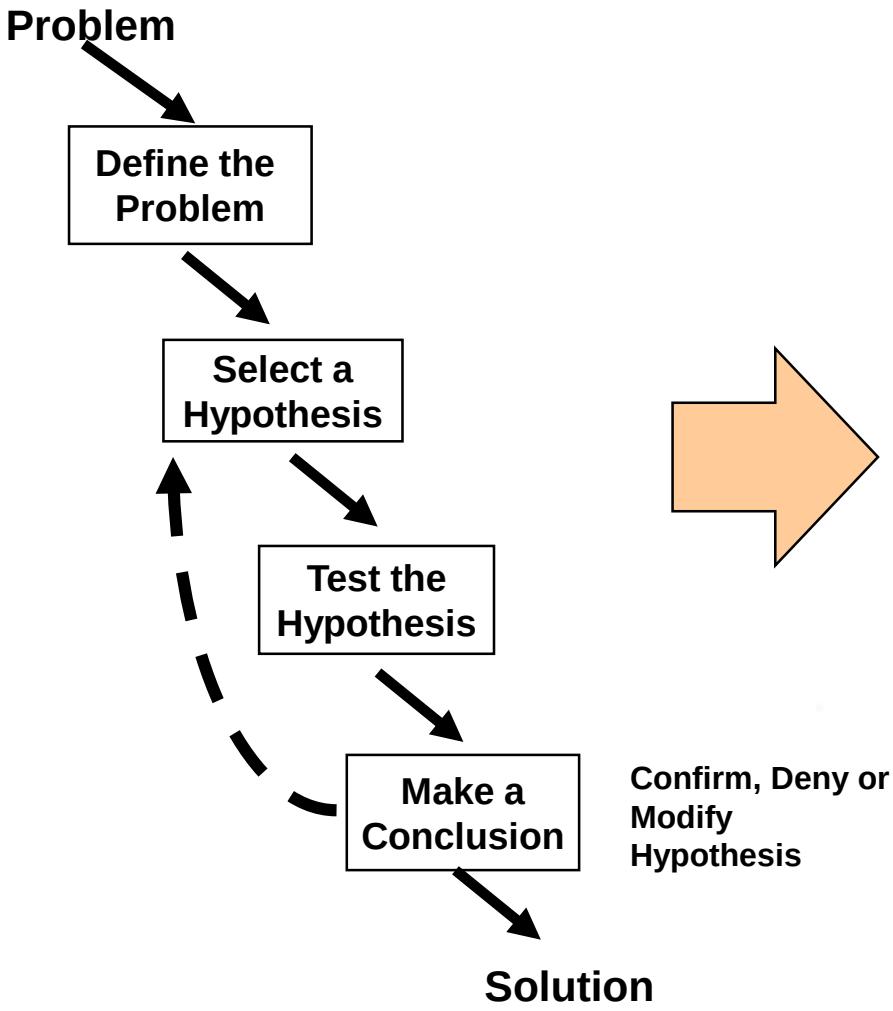
4. When do we:
- spend the most money and time?
- have the most leverage?

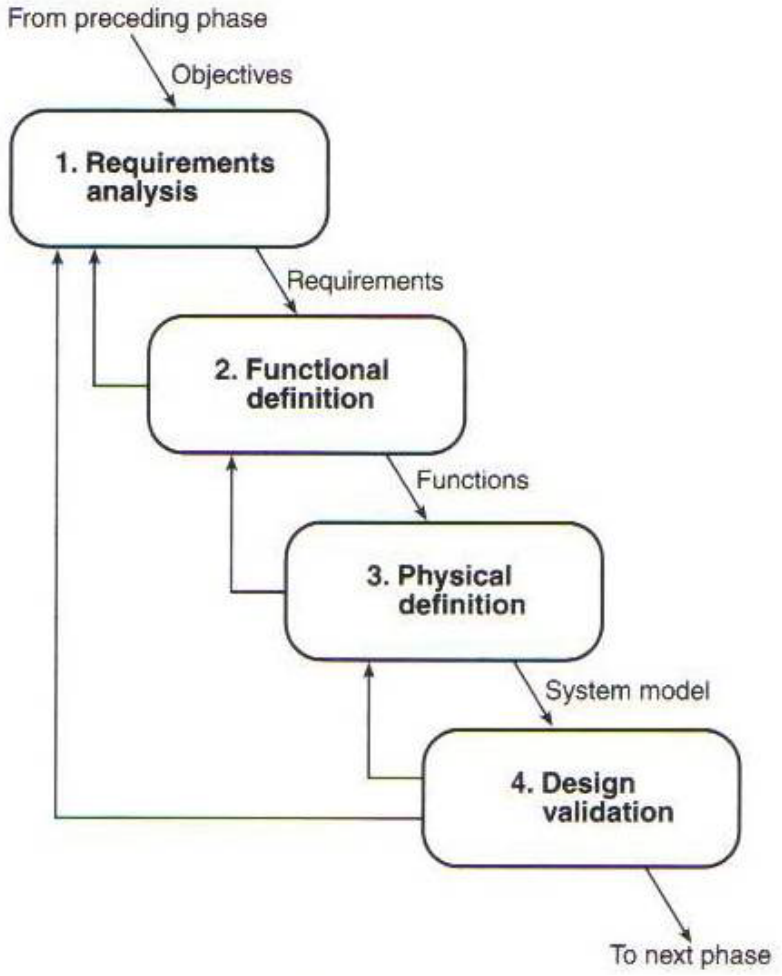
5. What is the focus of:
- Your organization's projects?
- Your current project(s)?

What does this suggest about our requirements engineering efforts?

The Systems Engineering Process

The scientific method…. informs the systems engineering process





The Systems Engineering Process is iterative and recursive

Some ideas from Session 1 (1/3)

- Complex systems can be represented as a hierarchical structure composed of things like subsystems, components, subcomponents, and parts
- System building blocks are components which are
  - Characterized by both physical and functional attributes
  - Distinct and perform a significant function
  - Specialized within the scope of a single engineering discipline
  - Common and re-usable across a variety of systems
- Functional elements are the functional equivalent of components and can be categorized into four classes of operating media:
  - Signal Elements, which sense and communicate information
  - Data Elements, which interpret, organize and manipulate information
  - Material Elements, which provide structure and process material
  - Energy Elements, which provide energy and power

Some ideas from Session 1 (2/3)

- **Components are physical realizations of functional elements**

- **The system environment is everything outside the system which interacts with it, including:**
  - **System operators**
  - **Maintenance and support systems**
  - **Shipping, storage and handling**
  - **Weather and other physical environments**
  - **The super system of which your system is a component**

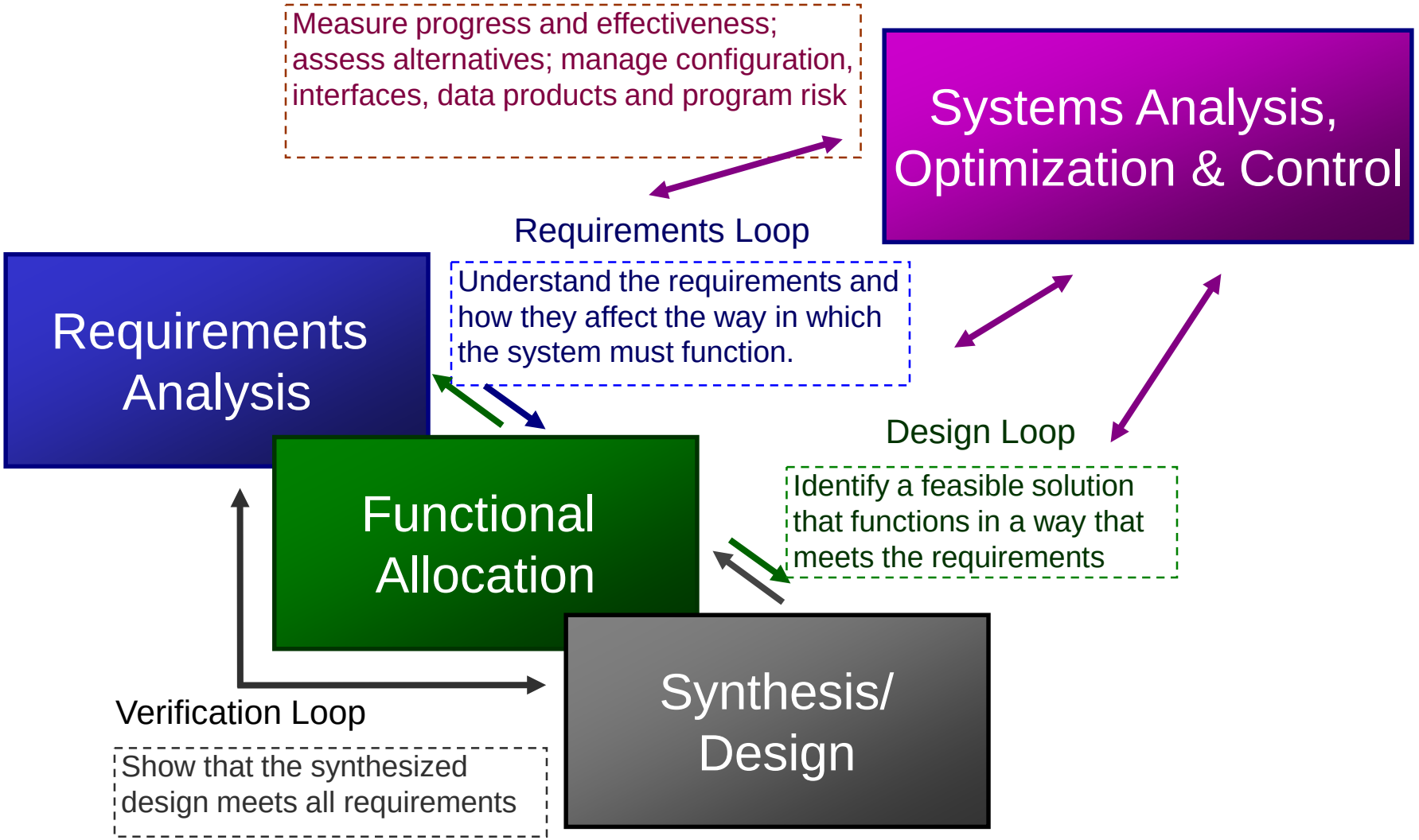
Some ideas from Session 1 (3/3)

- **Interfaces are critical systems engineering considerations, which**
  - **Include both external interfaces (interfaces with systems external to the system of interest) and internal intrafaces (interfaces between entities of the system of interest)**
  - **Effect interactions between and among components**
  - **Must be identified, specified coordinated and controlled**
  - **Must be tested to ensure smooth integration and ease of maintenance**

External interface requirements are defined as part of the solution-independent requirements engineering process

Intraface (internal interface) requirements are defined as part of the solution-dependent system design process

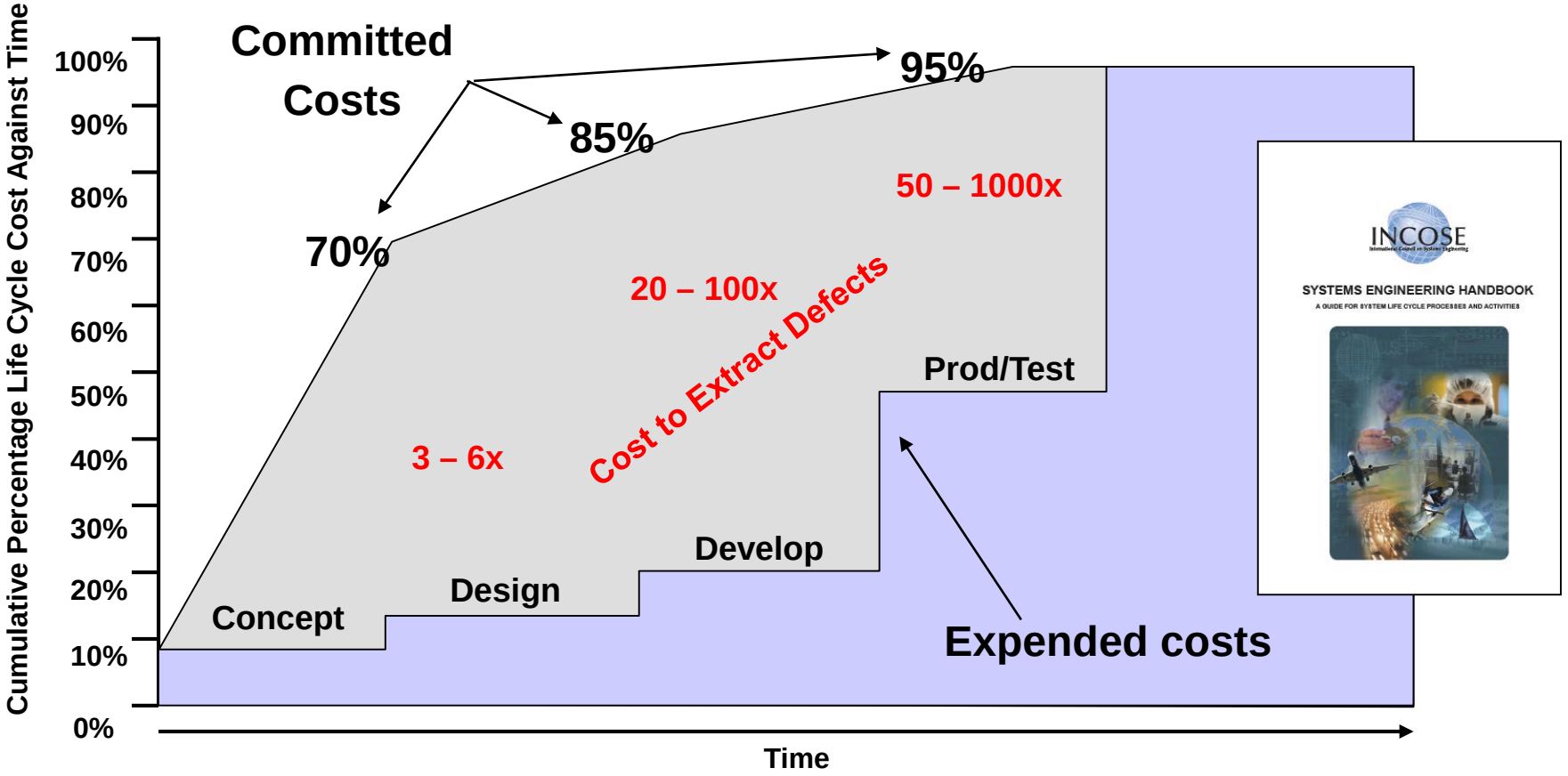
A Traditional View of the Systems Engineering Process Begins with Requirements Analysis



Why Are We Here? (1/3)

- Your expectations
  - See our exercise results from Session 1
- Add to these the instructor's expectations:
  - To have you all learn the material
  - To have you all succeed in the class
  - To meet your expectations. . . . . .
  - To have you be able to take the course concepts every week and make a positive difference to the performance of your jobs and the success of your projects.
- And, we are here because of the state of requirements engineering in the World today……..

Why Are We Here? (2/3)



Work and decisions made early in the project set the stage for project success or failure. In which of these stages are requirements developed?

Why Are We Here (3/3) – What Constitutes System Success?

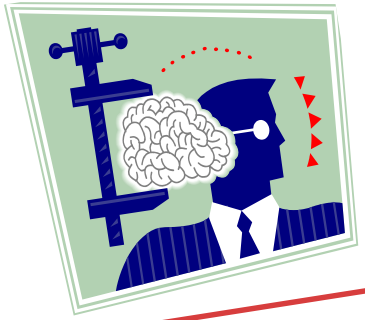
- ✓Delights customers and stakeholders
- ✓Satisfies specified requirements
- ✓Solves the "problem-to-be-solved" in the target context
- ✓Does what it is supposed to do efficiently and effectively
- ✓Meets budget, schedule, and technical requirements
- ✓Is producible, testable, sustainable, supportable, maintainable, secure, operable, safe, legal, ……..

These are typical results of the Session 1 discussion exercise

Are any of these related to requirements quality

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?

Systems Engineering Technical Short Course

(Session 2: Stakeholder Needs)

Appendix F
(Supplement to Lesson 5)
A Modeling Exercise

Robert B Wheeler, Instructor

Sessions 2 through 5 Agenda

- **Day 1**
  - Review project team assignments from previous Session
  - Presentation and discussion of Session topic
  - Project team working sessions on the application of the Session topics to their projects
  - Day 1 wrap-up and Q&A

- **Day 2**
  - Review Day 1 topic and project working sessions
  - Presentation and discussion of Session topic (concluded)
  - Project team working sessions (concluded)
  - Check-in with class on the application of the Session topic to their projects
  - Session wrap-up: Q&A and homework due next Session

Timing will be similar to session 1, and will be announced at beginning of each session

Session 2 – Stakeholder Needs

Lessons

1. Program Overview Update
2. Introduction to Requirements
3. Stakeholders and their Expectations
4. Stakeholder Requirements
5. Special Topics (if time permits)

Appendices*

E. Session 1 Summary

F. **A Modeling Exercise**

G. Other Appendices TBD

*Appendices A, B, C, and D were contained in Session 1 of the course; remember that Appendix D has a complete course bibliography. The course will reference, but will not cover, the appendices



SYS 579R:
Requirements Engineering

A Modeling Exercise (Optional)

Instructor: Robert B. Wheeler, ESEP, PMP

Date: March 01, 2014

Exercise: Modeling Applicable to Requirements Engineering

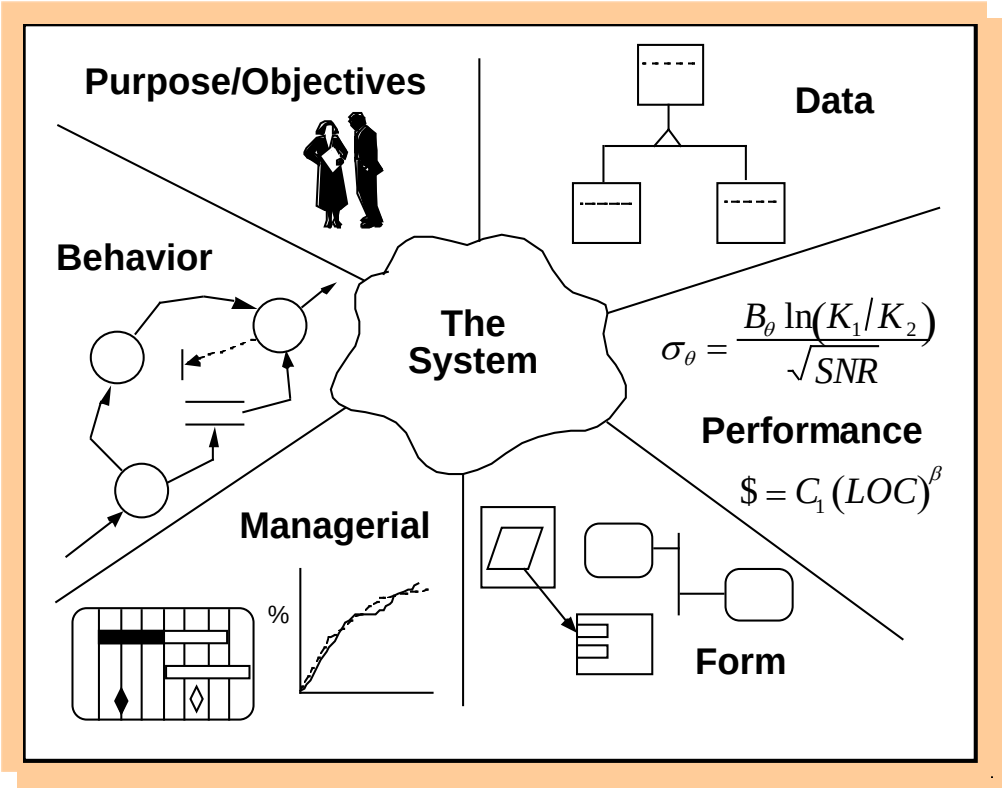
- **A group exercise to identify modeling applications to requirements engineering**
- **Based on the next seven slides that were taken from Rechtin and Maier, "*The Art of Systems Architecting*"**
  - ***First slide shows Rectin & Maier's "6 view" modeling framework.***
  - ***Second slide identifies several modeling techniques applicable to the "Purpose & Objectives" view.***
  - ***The remaining 5 slides are the subject of this exercise***
- ***The purpose of the exercise is to complete slides 3 through 7 by identifying modeling techniques for each view that are applicable to requirements engineering.***
- ***Exercise deliverable is a list of modeling techniques for each "view" that are applicable to requirements engineering***

The instructor will provide detailed instructions and timing for the exercise;

If time permits, the "Knowledge café" framework will be used

Some Modeling Techniques Applicable to Requirements Engineering and Discovery (1/7)

Let's use the "Six Views" framework introduced earlier:

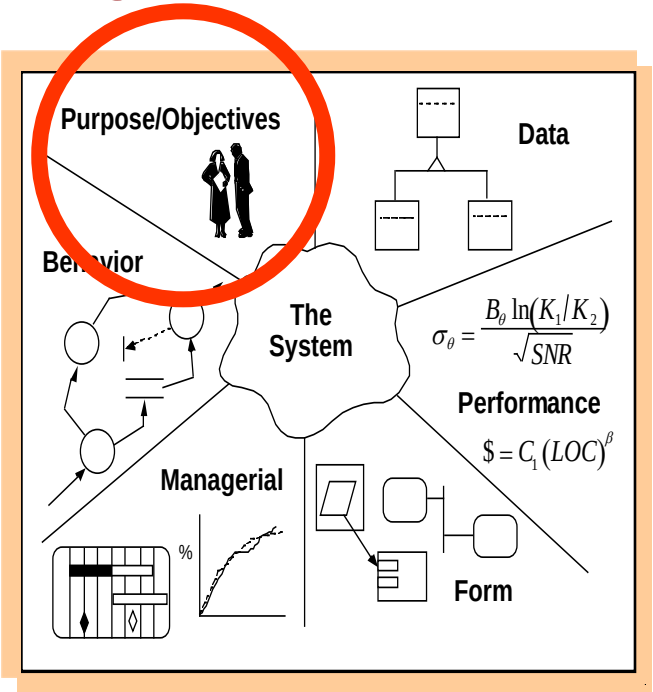


SOURCE: Rechtin and Maier, "*The Art of Systems Architecting*"

- **Which of these views pertain to requirements engineering?**
- **Can you give examples of techniques used in your environment that apply to each view?**

Some Modeling Techniques Applicable to Requirements Engineering and Discovery (2/7): Purpose & Objectives View

- Fundamental objectives (vs means objectives)
- Value tree
- Decision tree
- Inference networks
- Objectives hierarchy
- Mission task hierarchy
- Capabilities and effects analysis
- Capabilities tree
- MOEs vs. MOPs
- Operational concept

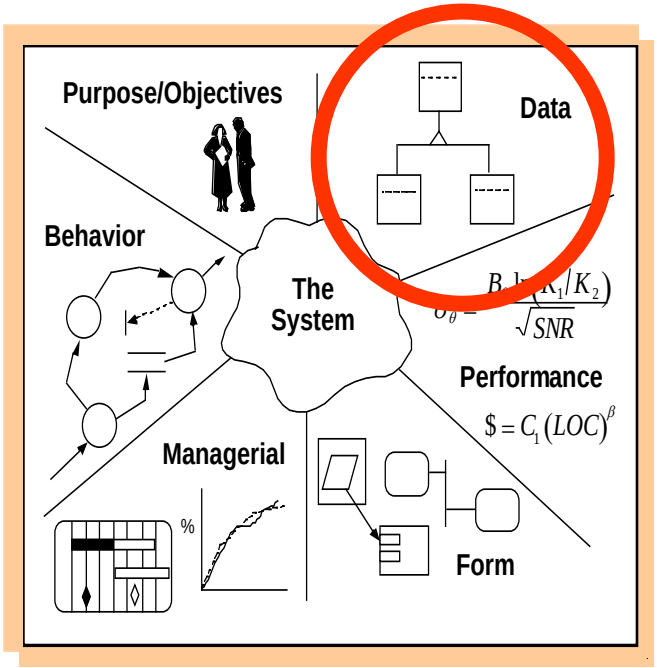


Exercise: Use the next 5 slides to identify modeling techniques applicable to requirements engineering

SOURCE: Rechtin and Maier, "*The Art of Systems Architecting*"

Modeling Techniques Applicable to Requirements Engineering and Discovery (3/7)

- **TBD**

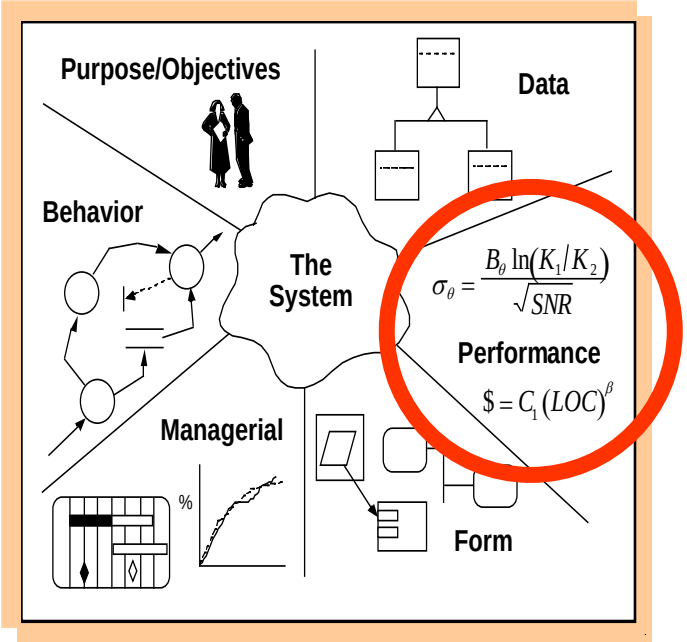


Exercise: Identify data modeling techniques applicable to requirements engineering

SOURCE: Rechtin and Maier, "*The Art of Systems Architecting*"

Modeling Techniques Applicable to Requirements Engineering and Discovery (4/7)

- **TBD**

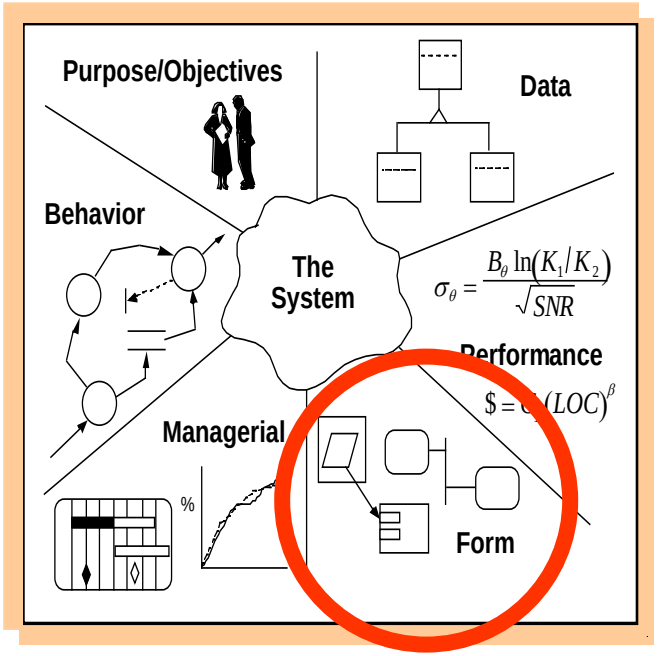


Exercise: identify performance modeling techniques applicable to requirements engineering

SOURCE: Rechtin and Maier, "*The Art of Systems Architecting*"

Modeling Techniques Applicable to Requirements Engineering and Discovery (5/7)

- **TBD**

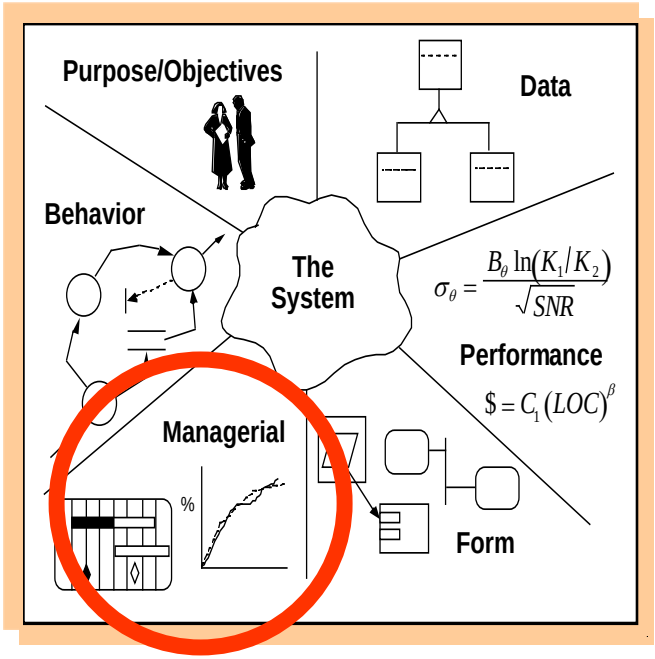


Exercise: identify Form modeling techniques applicable to requirements engineering

SOURCE: Rechtin and Maier, "*The Art of Systems Architecting*"

Modeling Techniques Applicable to Requirements Engineering and Discovery (6/7)

- **TBD**

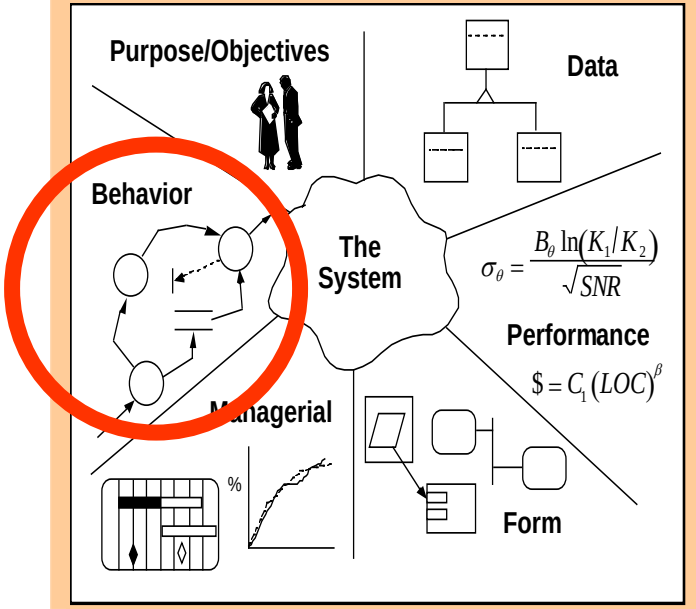


Exercise: identify Managerial modeling techniques applicable to requirements engineering

SOURCE: Rechtin and Maier, "*The Art of Systems Architecting*"

Modeling Techniques Applicable to Requirements Engineering and Discovery (7/7)

- **TBD**

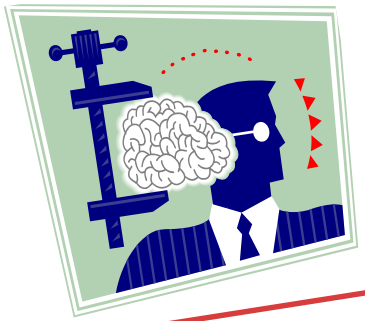


Exercise: identify Behavior modeling techniques applicable to requirements engineering

SOURCE: Rechtin and Maier, "*The Art of Systems Architecting*"

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		



All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?

Systems Engineering Technical Short Course for Waters Corporation

This course planned to be conducted at the Waters facilities as follows:

Session 1:	**January 27 – 29, 2015:**	**Overview and Concepts of Systems Engineering**
Session 2:	**February 16 – 17, 2015:**	**Stakeholder Needs**
Session 3:	**March 16 – 17, 2015**	**System Requirements**
Session 4:	**April 13 – 14, 2015**	**System Architecture and Design**
Session 5:	**May 13 – 14, 2015**	**System Integration and Test**
Session 6:	**June 10, 2015**	**Collaboration Training with WLI. (not part of WPI material)**
	June 11, 2015	**Capstone presentations**

Robert B. Wheeler, ESEP, PMP
Thomas F. Gannon, Ph.D.

This course is developed by Worcester Polytechnic Institute (WPI) for Waters Corporation. Course materials remain the property of WPI. WPI and the course developer (Robert Wheeler of Robert B Wheeler & Associates for sessions 1, 2, and 3) retain the right to use, reuse, and modify this content in future course deliveries.



Systems Engineering Technical Short Course

Session 3: System Requirements

March 16 – 17, 2015

Robert B. Wheeler, ESEP, PMP
Thomas F. Gannon, Ph.D.



Systems Engineering Technical Short Course

(Session 3: System Requirements)

Lesson 1:
Program Review Update

Welcome and Opening Remarks……

Logistics:

- **Facility**
- **Safety & emergencies**
- **Schedule & breaks**
- **Parking lot**
- **Other?**







Code of Conduct:

- On time
- Mutual respect
- Machines off & closed unless we are using them
- Participation!
- Other?

Remember from Session 1 – During this course you will: (1/2)

All Sessions: Apply the Systems Engineering principles and practices to your assigned projects

Sessions 1 through 3 (Bob Wheeler, instructor):

- **Develop and use a common vocabulary**
- **Apply systems thinking and holism to your projects**
- **Articulate the Problem to be Solved by your project**
- **Develop a Technical Risk Management Plan**
- **Elicit and express stakeholders expectations and requirements**
- **Develop User Concept of Operations (CONOPS), solution selection criteria, and acceptance criteria**
- **Conduct system functional and performance analyses**
- **Define and express system requirements**
- **Manage requirements through the system lifecycle**

Session 1	This Session 2	Session 3

Remember from Session 1 – During this course you will: (2/2)

Sessions 4 & 5 (Tom Gannon, instructor):

- **Develop creative alternative system architectures**
- **Conduct analyses of your architectural alternatives to meet stakeholder and system requirements**
- **Define a candidate architecture and design for your solution**
- **Develop strategies and plans for system integration, validation and verification testing**
- **Manage requirements through the system lifecycle (continued)**
- **Document the results of your work in appropriate artifacts**

Session 6 (Tom Gannon, instructor):

- **Present a status report on your project to Waters senior management**

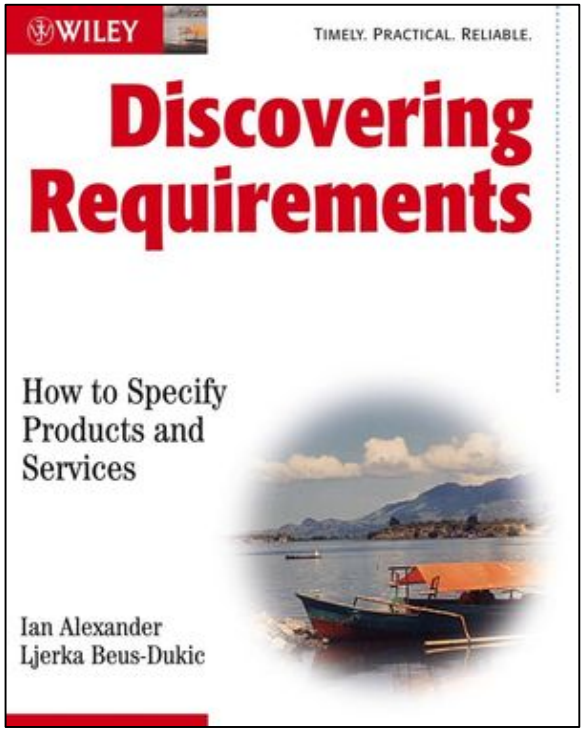
Schedule

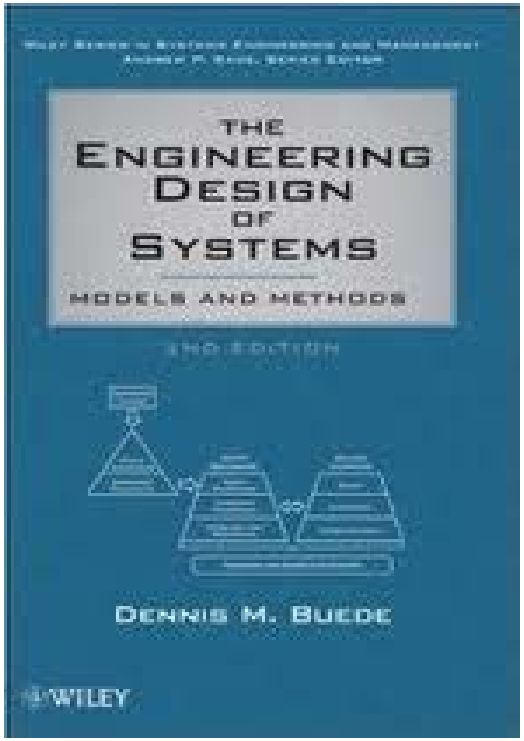
Session	Dates	Topic
1.	January 27 – 29	Introduction – Overview and Concepts of SE (US)
2.	February 16 – 17	Stakeholder Needs*
3.	March 16 – 17	Systems Requirements*
4.	April 13 – 14	Systems Architecture and Design (UK)*
5.	May 13 – 14	Systems Integration and Test*
6.	June 10	Collaboration Training with WLI
	June 11	Capstone Presentations (US)

Notes:

- Denotes sessions held for some participants via video-teleconference.
- Class times and room locations are shown later

Primary Requirements Course Texts





Ian Alexander and Ljerka Beus-Dukic, *Discovering Requirements*, Wiley, 2009

Dennis M. Buede, The Engineering Design of Systems, 2nd Edition, New Jersey: Wiley, 2009

Please see Session 1 Appendix D for the other three primary course texts, and many other applicable references

What Happens if I Miss a Class?

It is important to your own learning and to the group process to attend all the class sessions.

However, it is understood that some absences cannot be avoided.

If you miss a class, please take the responsibility to:

- **Review the missed lectures**
- **Contact the instructor with any questions you may have as a result of reviewing the lecture materials**
- **Provide a brief outline of the missed material, identifying what you believe to be the key learning points, to the instructor.**
- **Contact your team project leader to ensure that you are aware of any project work completed, or action items assigned, during your absence.**

All lecture material will be posted on the WPI course site:

- **You will be told how to use the WPI course site, and how to access the lecture material, early in session 1**

Video Tutorial for myWPI

Room Assignments

To Be Updated

Milford Rooms

Date	Time	Primary Location	Break-out Rooms			
27-Jan	8:00 - 4:00 pm	C4		N/A	N/A	N/A
28-Jan	8:00 - 4:00 pm	C4		N/A	N/A	N/A
29-Jan	8:00 - 4:00 pm	C4		N/A	N/A	N/A
16-Feb	7:00 - 4:00 pm	Atriums 2 & 3	1:00 - 4:00 pm	mfg conf	Atlantis	Global Ops
17-Feb	7:00 - 4:00 pm	C4	1:00 - 4:00 pm	mfg conf	Atlantis	Acquity
16-Mar	7:00 - 4:00 pm	Atriums 2 & 3	1:00 - 4:00 pm	mfg conf	Atlantis	Global Ops
17-Mar	7:00 - 4:00 pm	C4	1:00 - 4:00 pm	mfg conf	Atlantis	Acquity
13-Apr	7:00 - 4:00 pm	Atriums 2 & 3	1:00 - 4:00 pm	mfg conf	Atlantis	Global Ops
14-Apr	7:00 - 4:00 pm	C4	1:00 - 4:00 pm	mfg conf	Acquity	Atlantis
13-May	7:00 - 4:00 pm	C4	1:00 - 4:00 pm	mfg conf	Acquity	Atlantis
14-May	7:00 - 4:00 pm	C4	1:00 - 4:00 pm	mfg conf	Acquity	Atlantis
10-Jun	8:00 - 4:00 pm	WTC 2&3		N/A	N/A	N/A
11-Jun	8:00 - 1:00 pm	Director's Room		N/A	N/A	N/A

Wilmslow Rooms

Date	Time	Primary Location	Break-out Rooms			
16-Feb	12:00 - 18:00	Hydrogen	14:00 to 17:00	Helium	Neon	Silicon
17-Feb	12:00 - 18:00	Hydrogen	12:30 to 16:30	Helium	Oxygen	Silicon
16-Mar	12:00 –18:00	Hydrogen	14:00 to 17:00	Helium	Oxygen	Silicon
17-Mar	12:00 - 18:00	Hydrogen	12:30 to 16:30	Helium	Oxygen	Silicon
13-Apr	12:00 - 18:00	Hydrogen	14:00 to 17:00	Helium	Oxygen	Silicon
14-Apr	12:00 - 18:00	Hydrogen	12:30 to 16:30	Helium	Oxygen	Silicon
13-May	12:00 - 18:00	Hydrogen	14:00 to 17:00	Helium	Oxygen	Silicon
14-May	12:00 - 18:00	Hydrogen	12:30 to 16:30	Helium	Oxygen	Silicon

What are your Questions before We Look at our Homework Assignments?



Sessions 2 through 5 Agenda (Revised)

- **Day 1**
  - **Review project team assignments from previous Session**
  - **Presentation and discussion of Session topic**
  - ~~**Project team working sessions on the application of the Session topics to their projects**~~
  - **Day 1 wrap-up and Q&A**

- **Day 2**
  - ~~**Review Day 1 topic and project working sessions**~~
  - **Presentation and discussion of Session topic (concluded)**
  - **Project team working sessions**
  - **Check-in with class on the application of the Session topic to their projects**
  - **Session wrap-up: Q&A and homework due next Session**

Timing will be similar to session 1, and will be announced at beginning of each session

Homework Guidance

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

Focus on purpose of the deliverables and the relationships within the team(s) and organization. Then see how they support the Problem Statement.

Plan for approximately 2 days of work between class sessions.

Notify your manager and Facilitator of priority conflicts.

©2015 Waters Corporation

Session 2 Stakeholder Needs: Homework Assignment due at beginning of Session 3*

Develop the following deliverables:

- **Statement of the Problem to be Solved for your project**
  (A summarizing sentence, plus an explanatory paragraph that communicates clearly the problem you are trying to solve. Do this for the work project, and then for the course project; You may include a diagram)
- **Stakeholder Analysis** **(A table listing key stakeholders and their type, perceived expectations, and priority. Also an explanation of your assumptions, process used, and what you think are your key results)**
- **User CONOPS** **(Context diagrams, examples of operational and support scenarios, description of operational and support environments)**
- **Stakeholder Expectations** **(list of initially identified S/H expectations)**
- **Stakeholder Requirements and an initial cut at Traceability**
  (List of S/H requirements, trace to their source, and explanation of your requirements rationale and process used)

Update the previously developed deliverables

* Each project team must decide which of, and the extent to which, each of these deliverables will be done

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

PROJECT GROUPING

SYSTEMS CONTROLLER
Sponsor: Allen Caswell

Cindy Miller (US)
Richard Moulds (UK)
Richard Chapman (UK)
Steve Binney (US)
Justin Deary (US)
Phillip Young (UK)
David Gordon (UK)

NEXUS
Sponsors: Joe Luongo/John Chan

Abhijit Tarafder (US)
Wade Leveille (US)
David Lamborghini (US)
Ian Trivett (UK)
Daniel Kenney (UK)
Jim Usowicz (US)
John Heden (US)

SMART CONSUMABLES
Sponsor: Tom Walter

Jacob Fairchild (US)
Jason Hill (US)
Mathew DeLano (US)
Jan Driesen (GERMANY)
Alex Hooper (UK)
Mark Pickering (UK)
Senthil Bala (US)

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

PROJECT GROUPING

SYSTEMS CONTROLLER
Sponsor: Allen Caswell

Cindy Miller (US)
Richard Moulds (UK)
Richard Chapman (UK)
Steve Binney (US)
Justin Deary (US)
Phillip Young (UK)
David Gordon (UK)

NEXUS
Sponsors: Joe Luongo/John Chan

Abhijit Tarafder (US)
Wade Leveille (US)
David Lamborghini (US)
Ian Trivett (UK)
Daniel Kenney (UK)
Jim Usowicz (US)
John Heden (US)

SMART CONSUMABLES
Sponsor: Tom Walter

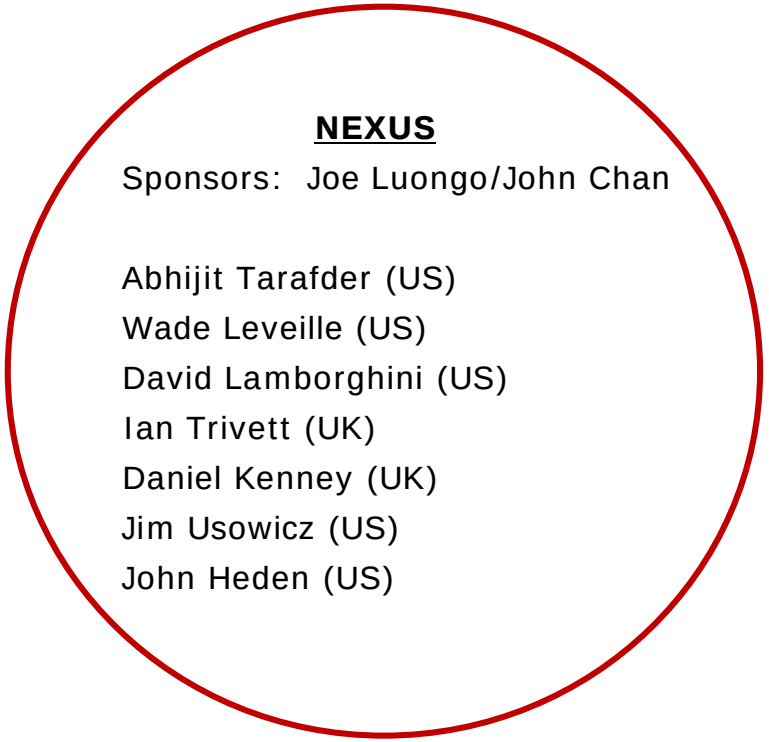
Jacob Fairchild (US)
Jason Hill (US)
Mathew DeLano (US)
Jan Driesen (GERMANY)
Alex Hooper (UK)
Mark Pickering (UK)
Senthil Bala (US)

©2015 Waters Corporation

WPI SYSTEMS ENGINEERING

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

PROJECT GROUPING



SYSTEMS CONTROLLER

Sponsor: Allen Caswell

Cindy Miller (US)
Richard Moulds (UK)
Richard Chapman (UK)
Steve Binney (US)
Justin Deary (US)
Phillip Young (UK)
David Gordon (UK)

NEXUS

Sponsors: Joe Luongo/John Chan

Abhijit Tarafder (US)
Wade Leveille (US)
David Lamborghini (US)
Ian Trivett (UK)
Daniel Kenney (UK)
Jim Usowicz (US)
John Heden (US)

SMART CONSUMABLES

Sponsor: Tom Walter

Jacob Fairchild (US)
Jason Hill (US)
Mathew DeLano (US)
Jan Driesen (GERMANY)
Alex Hooper (UK)
Mark Pickering (UK)
Senthil Bala (US)

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

PROJECT GROUPING

SYSTEMS CONTROLLER
Sponsor: Allen Caswell

Cindy Miller (US)
Richard Moulds (UK)
Richard Chapman (UK)
Steve Binney (US)
Justin Deary (US)
Phillip Young (UK)
David Gordon (UK)

NEXUS
Sponsors: Joe Luongo/John Chan

Abhijit Tarafder (US)
Wade Leveille (US)
David Lamborghini (US)
Ian Trivett (UK)
Daniel Kenney (UK)
Jim Usowicz (US)
John Heden (US)

SMART CONSUMABLES
Sponsor: Tom Walter

Jacob Fairchild (US)
Jason Hill (US)
Mathew DeLano (US)
Jan Driesen (GERMANY)
Alex Hooper (UK)
Mark Pickering (UK)
Senthil Bala (US)

©2015 Waters Corporation

Session 3 - System Requirements: Learning Objectives

- Conduct system functional and performance analysis
- Analyze and express system non-functional requirements
- Flow-down system level requirements to lower level entities
- Manage requirements through the system lifecycle
- Express system requirements in appropriate artifacts

Session 3 - System Requirements: Homework Assignment due at beginning of Session 4*

Develop the following deliverables:

- **Functional Architecture (see Buede text):** (Explain and illustrate your approach to develop a solution-independent functional architecture. Use any of the approaches from the course material)
- **Analysis of Non-Functional Requirements (NFRs):** (Explain your approach toward NFRs, and show examples of how you used that approach to develop NFRs. Include NFRs in your SRD or equivalent)
- **System Requirements Document (SRD) or equivalent expression of system requirements:** (Document your system requirements. Find a suitable template, tailor the template to your system, and record your system requirements)
- **Draft Requirements Traceability Matrix, or equivalent:** (Should trace to source, and point to validation/verification approach. Use templates and examples shown in the course materials, and include all requirements that you were able to identify and define)

Update the previously developed deliverables

- Each project team must decide which of, and the extent to which, each of these deliverables will be done
- If you cannot complete these deliverables for your system, identify what is not completed, and how you would complete the work
- The course text (Buede, The Engineering design of Systems) has artifact outlines

Sessions 2 through 5 Agenda (Revised)

- **Day 1**
  - **Review project team assignments from previous Session**
  - **Presentation and discussion of Session topic**
  - ~~**Project team working sessions on the application of the Session topics to their projects**~~
  - **Day 1 wrap-up and Q&A**

- **Day 2**
  - ~~**Review Day 1 topic and project working sessions**~~
  - **Presentation and discussion of Session topic (concluded)**
  - **Project team working sessions**
  - **Check-in with class on the application of the Session topic to their projects**
  - **Session wrap-up: Q&A and homework due next Session**

Timing will be similar to session 1, and will be announced at beginning of each session

Session 3 – System Requirements

Lessons

1. **Program Overview Update**
2. **Functional Requirements**
3. **Performance and Non-Functional Requirements**
4. **Requirements Management and Traceability**
5. **Special Topics (if time permits)**

- A traditional approach
- Functional architecture (Buede)
- Graphical modeling Techniques

- Performance requirements
- What are non-functional requirements
- Discovering non-functional requirements
- Requirements flow down

- Requirements management
- Traceability
- Baseline management

Appendices*

- **G. Alarm System Exercise**
- **H. Models and Methods**
- **I. Other Appendices TBD**

- Agile methods
- "House of Quality"
- Thoughts about Tools

*Appendices A through F were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		

All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?



Systems Engineering Technical Short Course

(Session 3: System Requirements)

Lesson 2: Functional Requirements

Session 3 – System Requirements

Lessons

1. **Program Overview Update**
2. **Functional Requirements**
3. **Performance and Non-Functional Requirements**
4. **Requirements Management and Traceability**
5. **Special Topics (if time permits)**

- Lesson context
- A traditional approach
- Functional architecture (Buede)
- Graphical modeling Techniques

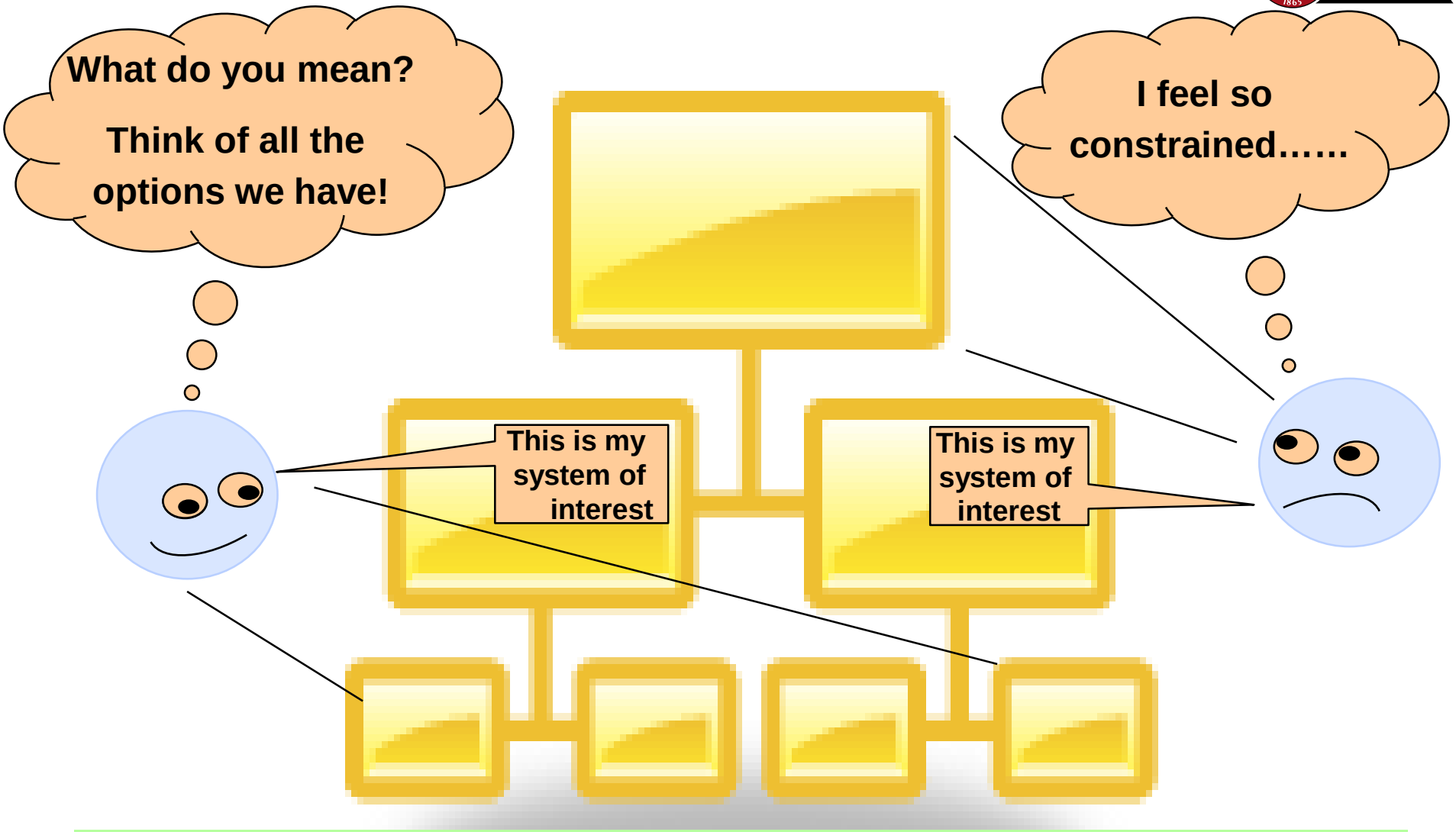
Appendices*

G. **Alarm System Exercise**

H. **Models and Methods**

I. **Other Appendices** ← **(None identified as of date of printing)**

*Appendices A through F were contained in earlier sessions of the course; Appendix D is a complete course bibliography. Appendices will not be formally covered

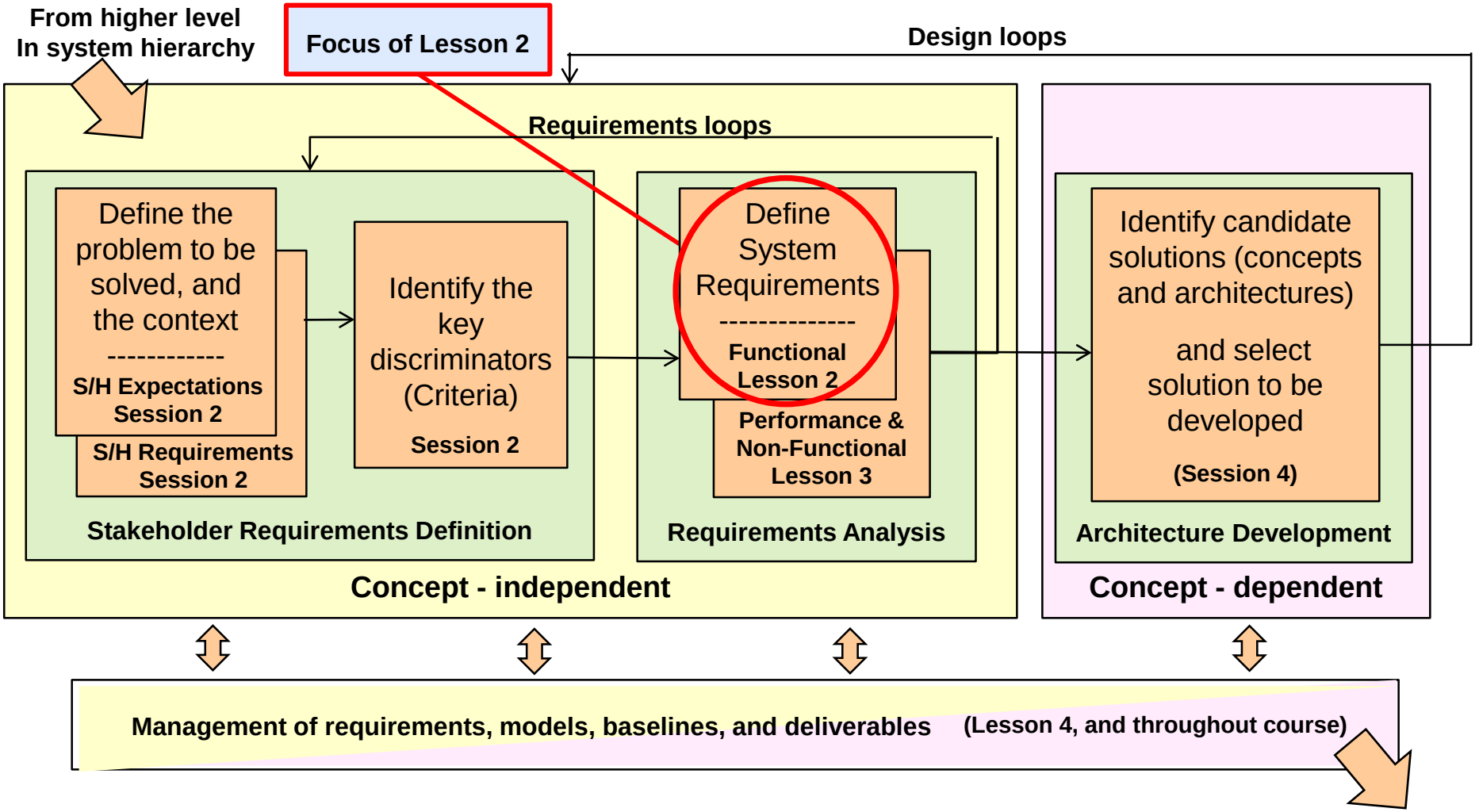


The view looking upward is usually solution-dependent, but the view looking downward is, at least initially, solution-independent

Session 3 Context

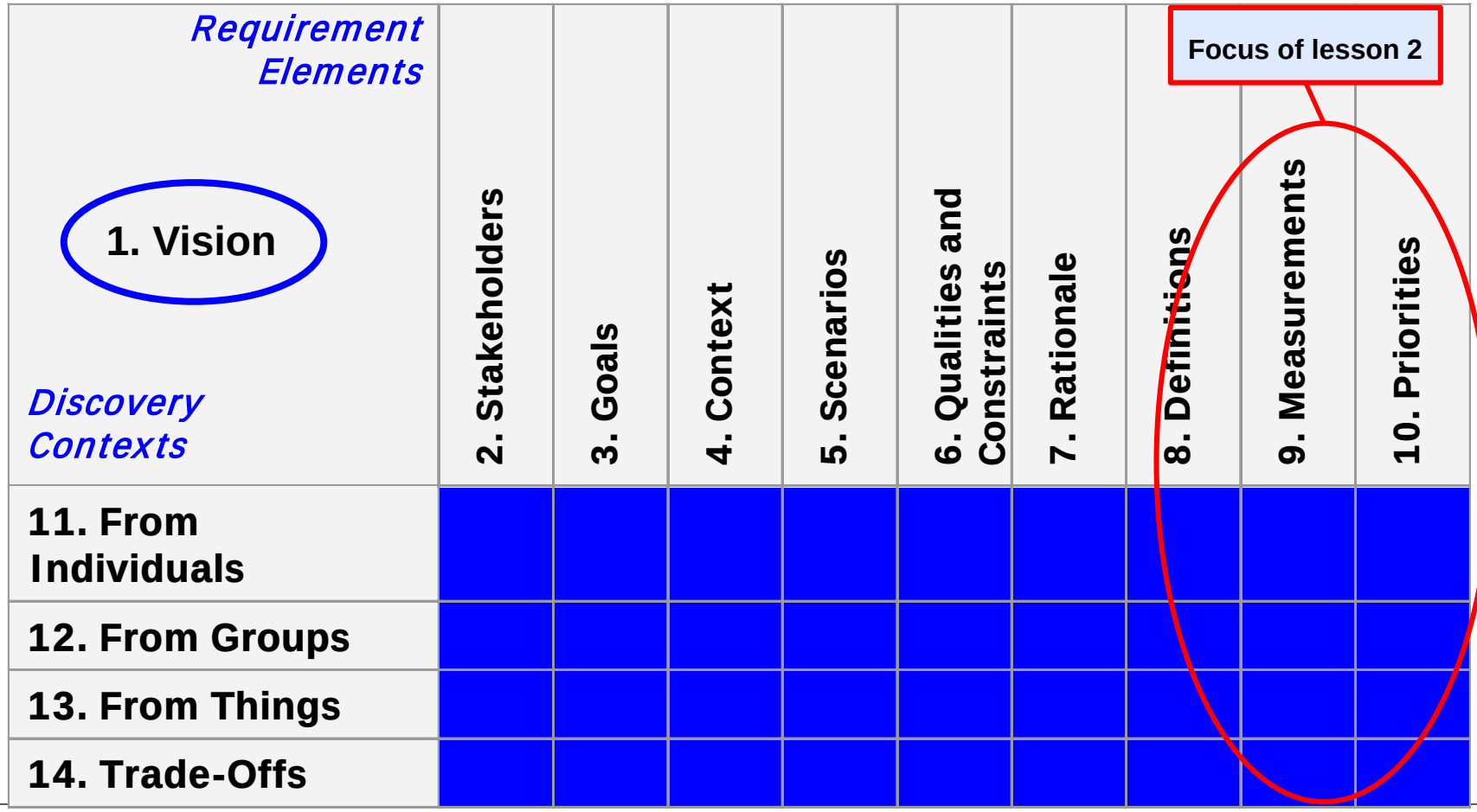
1. **Session 2 discussed the activities required to develop a set of *"solution-independent" (or "concept-independent")* *stakeholder requirements* in the language of users and other stakeholders. INCOSE calls this the "Stakeholder Requirements Definition Process". This was the focus of your Assignment 2.**
2. **Session 3 discusses the activities required to develop a set of *solution-independent system requirements* in engineering language. This includes much of what INCOSE calls the "Requirements Analysis Process". This will be the focus of your Assignment 3.**
3. **This lesson (2) discusses the functional analyses needed to evolve these stakeholder requirements into system functional requirements in engineering language.**
4. **The next lesson (3) will cover analyses to develop quantified performance requirements and non-functional requirements**
5. **Functional and performance analyses covered in these two lectures yield what Buede calls the solution-independent "Functional Architecture."**
6. **While this session focusses on the solution-independent requirements, it is important to note that the activities covered in session 3 are also performed after solution concept selection, to define the concept-dependent specification requirements; this is addresses in Lesson 3 (Requirements flow down)**

The Requirements Engineering Process Context for this Course (Left Side of Vee)



Performed on every entity at every level of system hierarchy

Using "Discovery" to Model the Requirements Engineering Process



What are your Questions before we start this lecture?



Session 3 – System Requirements

Lessons

1. **Program Overview Update**
2. **Functional Requirements**
3. **Performance and Non-Functional Requirements**
4. **Requirements Management and Traceability**
5. **Special Topics (if time permits)**

- Lesson context
- A traditional approach
- Functional architecture (Buede)
- Graphical modeling Techniques

Appendices*

G. **Alarm System Exercise**

H. **Models and Methods**

I. **Other Appendices TBD**

*Appendices A through F were contained in earlier sessions of the course; Appendix D is a complete course bibliography. Appendices will not be formally covered

Stakeholder Requirement Artifact Templates

- The results of doing what is discussed in this chapter are typically recorded in the System Requirements Document
- There are many artifacts that can be used to develop, record, and communicate functional and performance requirements
  - These will be discussed as we go along



Functional Analysis and Determination

- Functions are verbs – what the system shall do
- Functional Analysis: The translation of operational and support requirements into qualitative and quantitative design-to requirements
- Functional Analysis includes:
  - Translation of requirements to the functions necessary to achieve them
  - The sequence in which the functions will be preformed
  - Identification of selectable options to yield equivalent results
- Function interactions can be physical, logical and/or data relationships
- Functions can be performed by hardware, software, people, facilities, other system entities, or some combination

Function Analysis Sequence

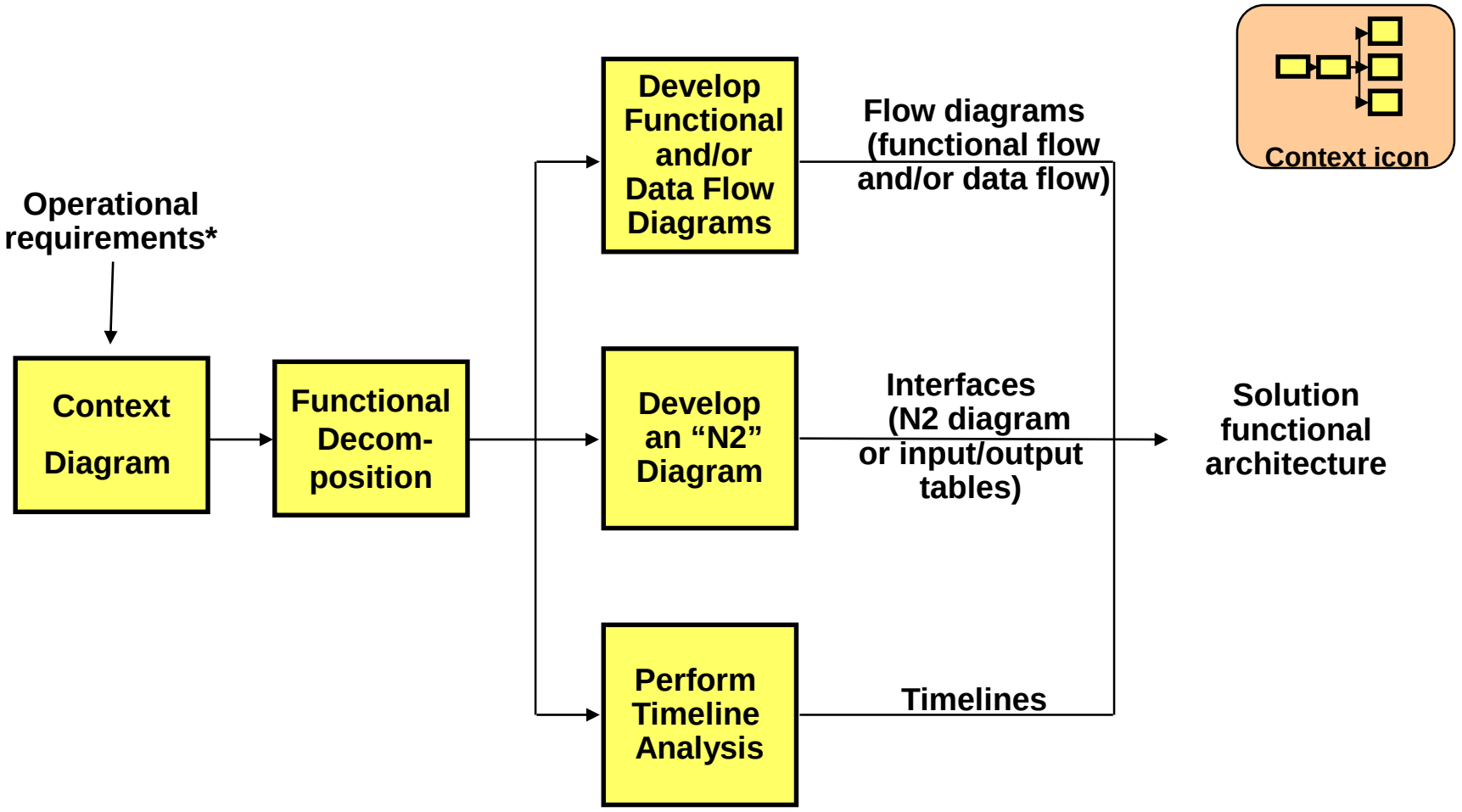
- Determine required primary functions
- Determine the logical function sequence and timing
- Determine alternatives and options
- Elaborate primary functions to determine required secondary functions

There are many methods and tools that can be used to do functional analysis. This lesson will illustrate a simple method. Other methods and tools can be found in:

- The supplementary information
- The INCOSE database at www.incose.org
- The literature

Select methods and models that you, your team, and your stakeholders understand and know how to use to accomplish your objectives.

A Functional Analysis Process



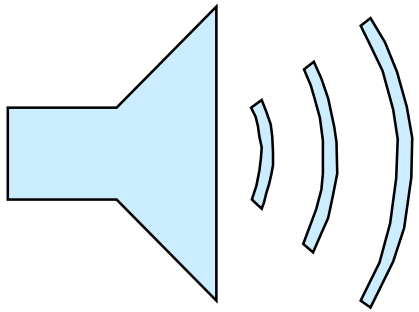
*For simple systems, we may be able to do this analysis on the entire system, starting with the operational requirements; For more complex systems we may want to first identify system states (such as "operating", "standby", "maintenance", or "shutdown"), and do the analysis for each states separately. A "state diagram" or a "state transition diagram" can be used to illustrate the states and their transitions.

An example: "Perform Alarm Function"

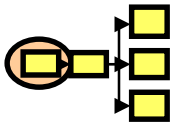
- The next several charts will:
  - Discuss a set of models (identified on the previous chart) that can be used in functional analysis
  - Illustrate them with an example.
  - Provide an exercise similar to the example

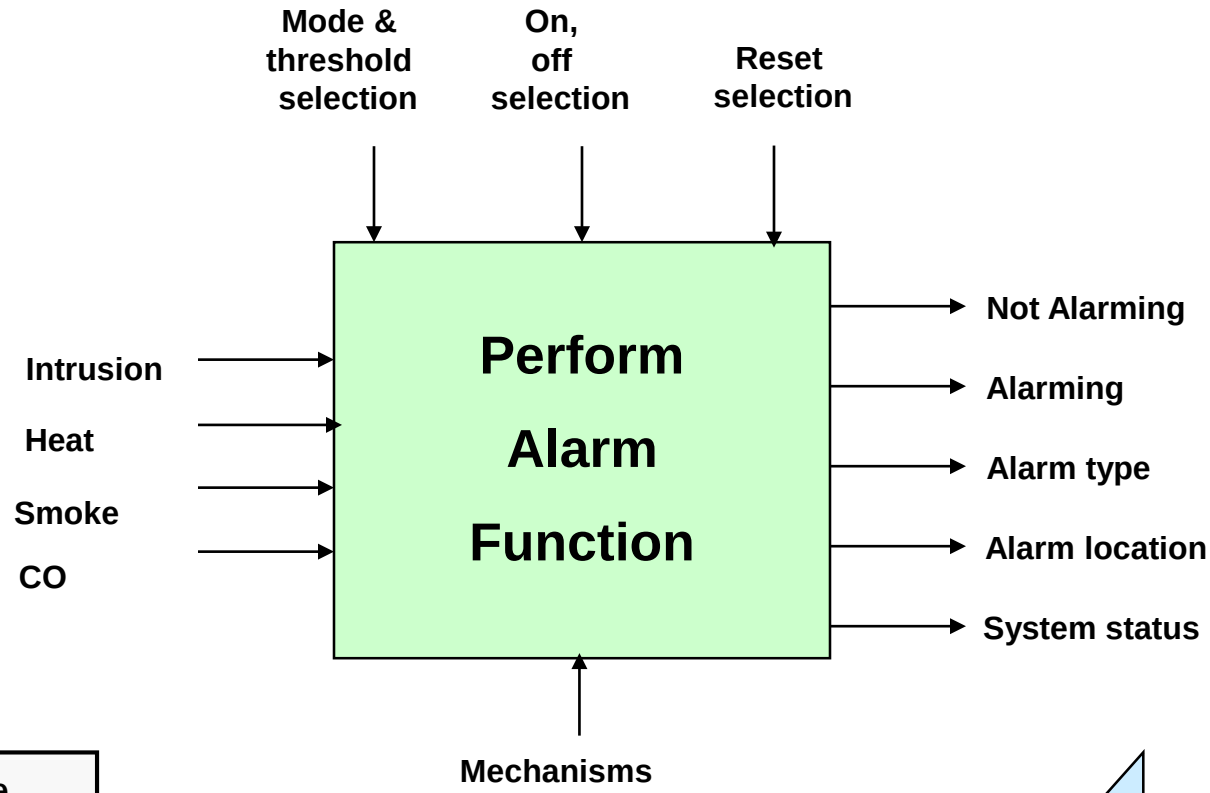
Example: "Perform Alarm Function"

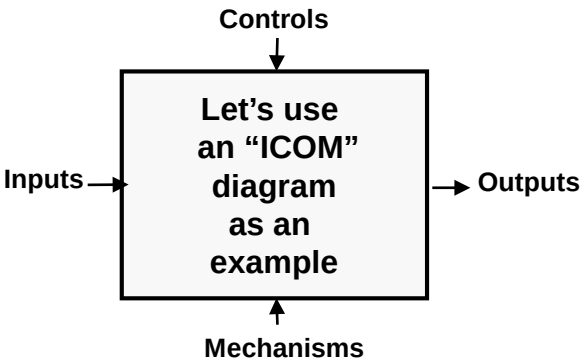
- Requirement: perform an alarm function to protect facilities, property, and people
- This is concept-independent. No concept or architecture has yet been assumed or selected.

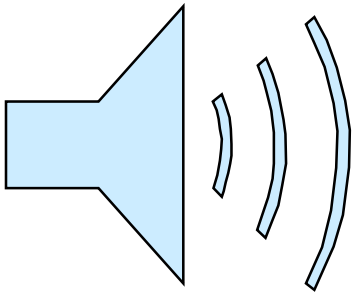


System Context Diagram: Example

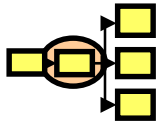




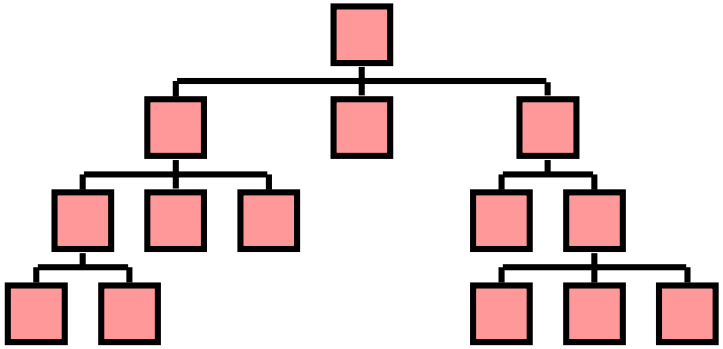




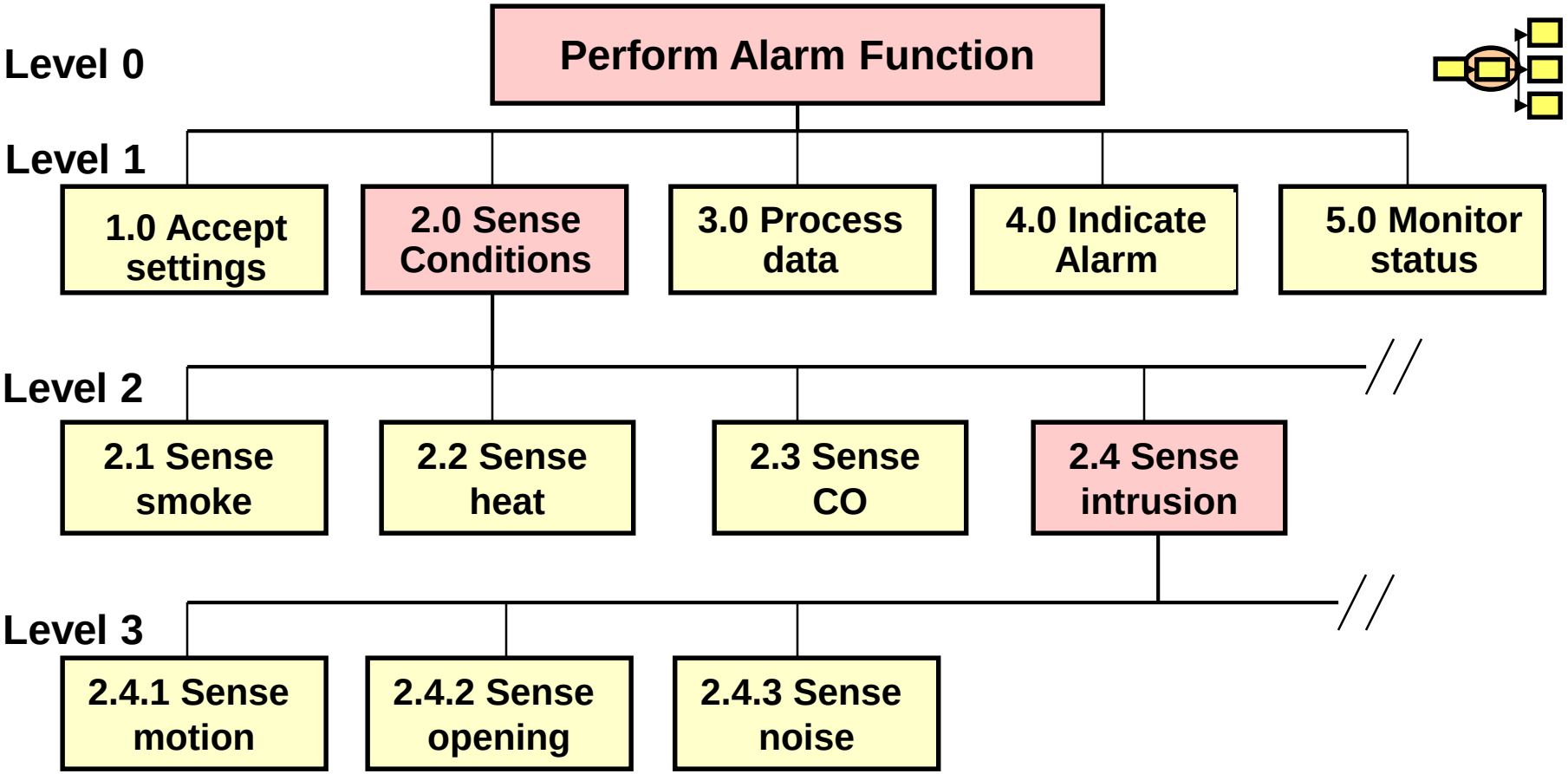
Functional Decomposition



- A hierarchical structured view of system functions that illustrates the totality of the system functions required
  - Illustrates functional decomposition
  - Decomposes functions from top to lower levels
  - Continues until all functional requirements are understood and documented
- Process is not unique; functions can be grouped in different ways
  - By common functions
  - To simplify interfaces
  - By technical discipline
- Also called:
  - Functional Structure Chart
  - Functional Breakdown Structure



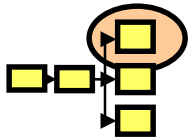
Functional Structure Chart for "Perform Alarm Function"

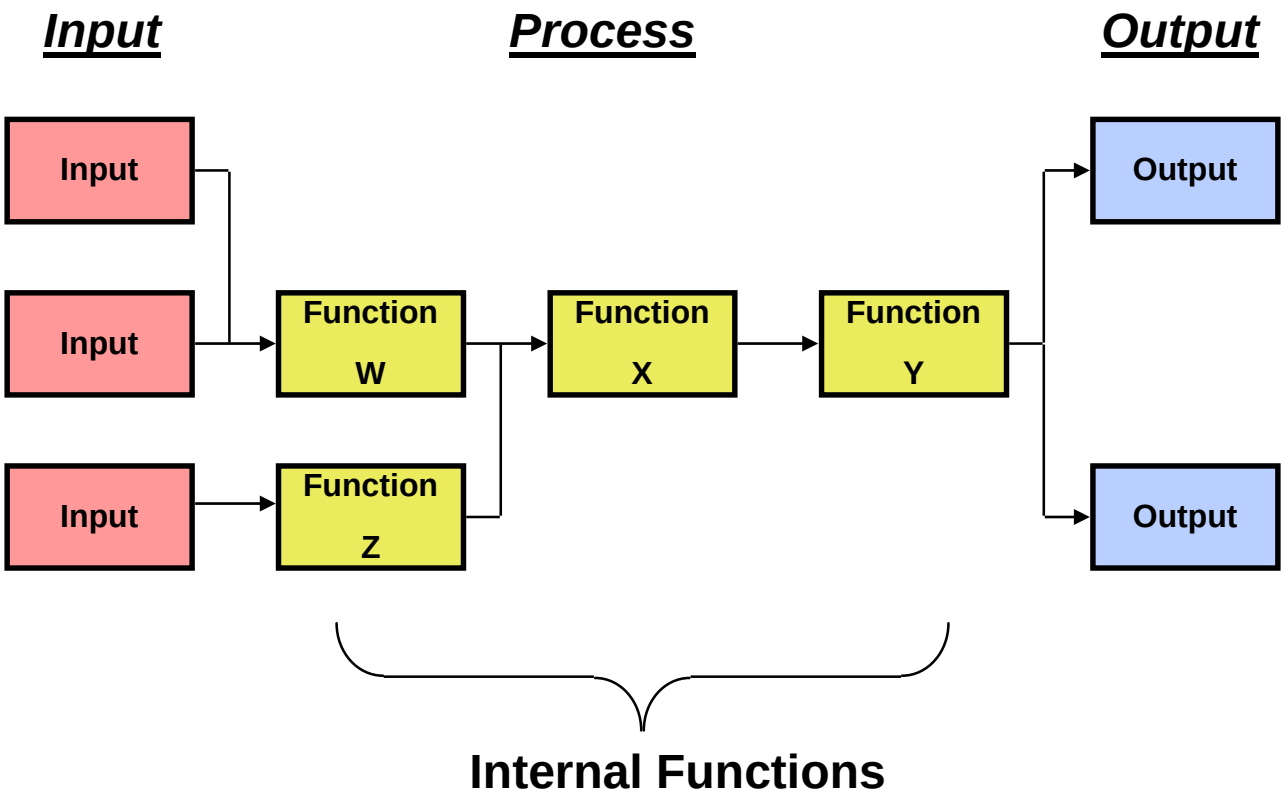


Although implied, there is no sequence information in this model.

The level of detail is a function of the information you need to make required decisions, the criticality of the decisions, and the risk associated with those decisions.

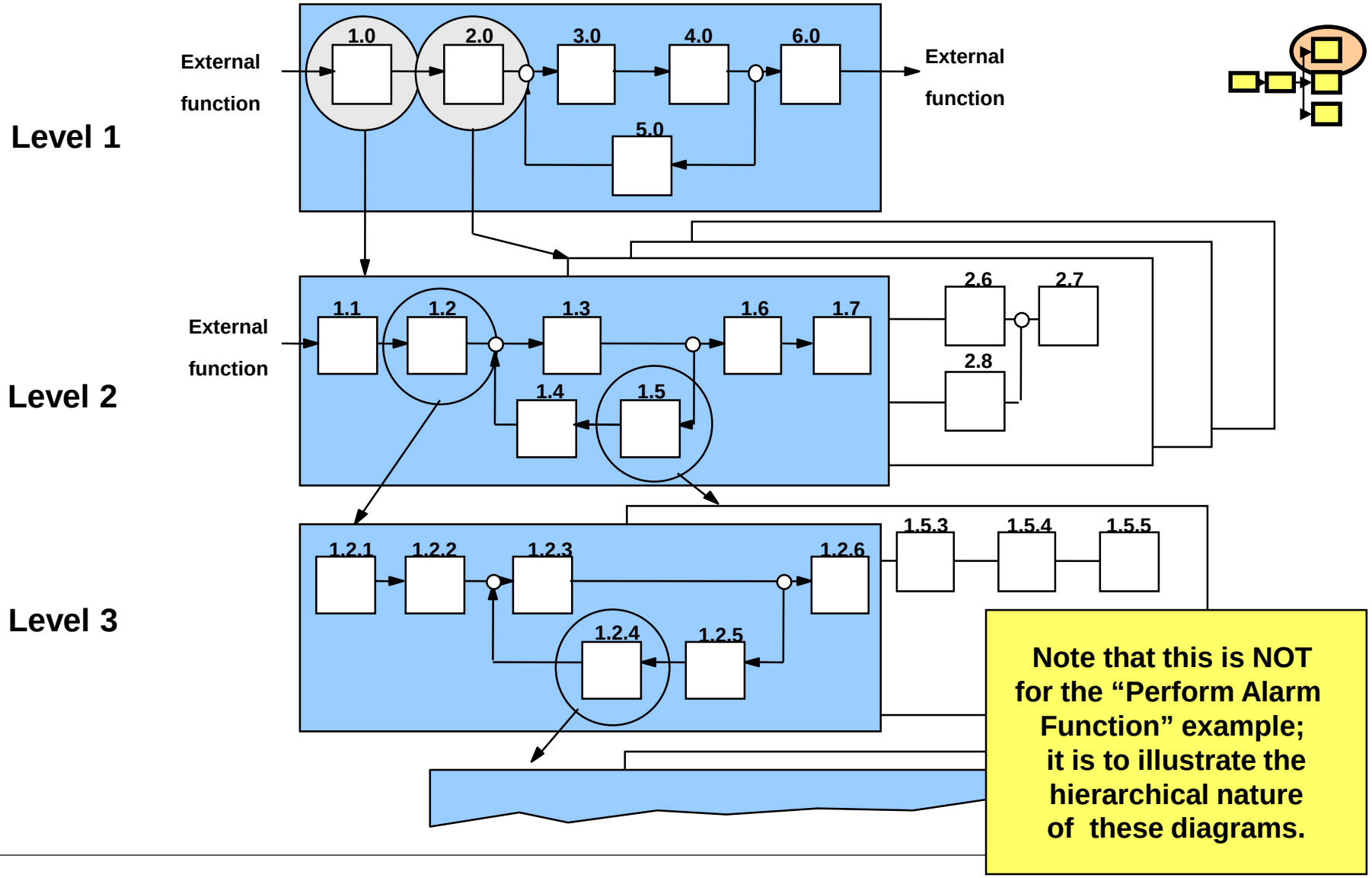
Functional Flow Block Diagram – Schematic





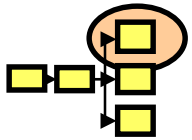
Note that we have switched from the "ICOM" (or IDEF0) format here, and are now using the more traditional FFBD format; Therefore, the ICOM inputs and controls become FFBD inputs.

Functional Flow Block Diagram – Indenture Levels



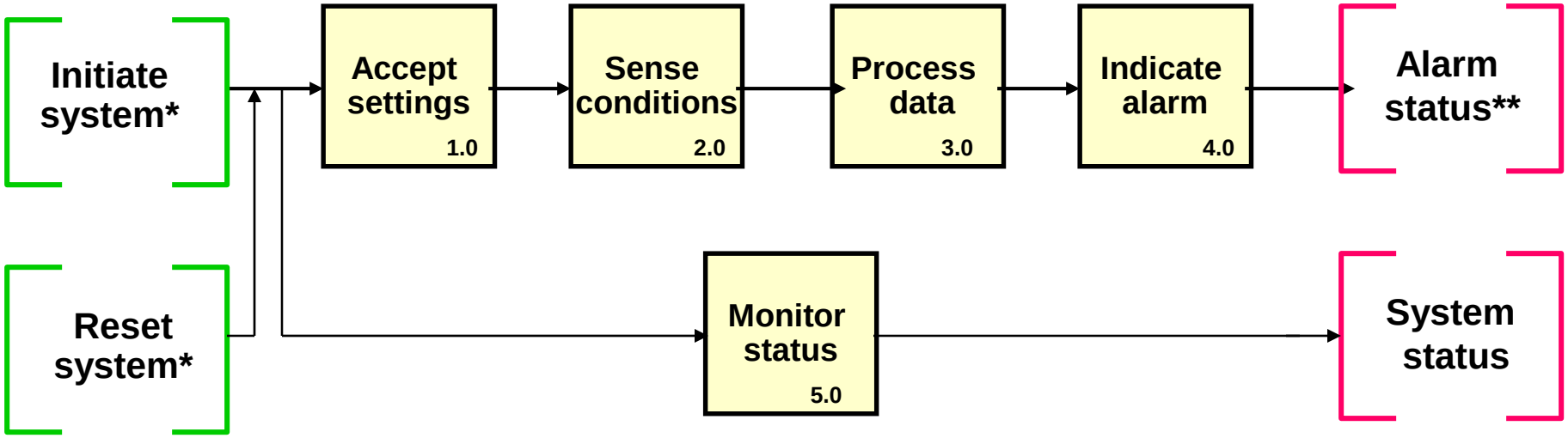
Note that this is NOT for the "Perform Alarm Function" example; it is to illustrate the hierarchical nature of these diagrams.

Functional Flow Block Diagram for "Perform Alarm Function"



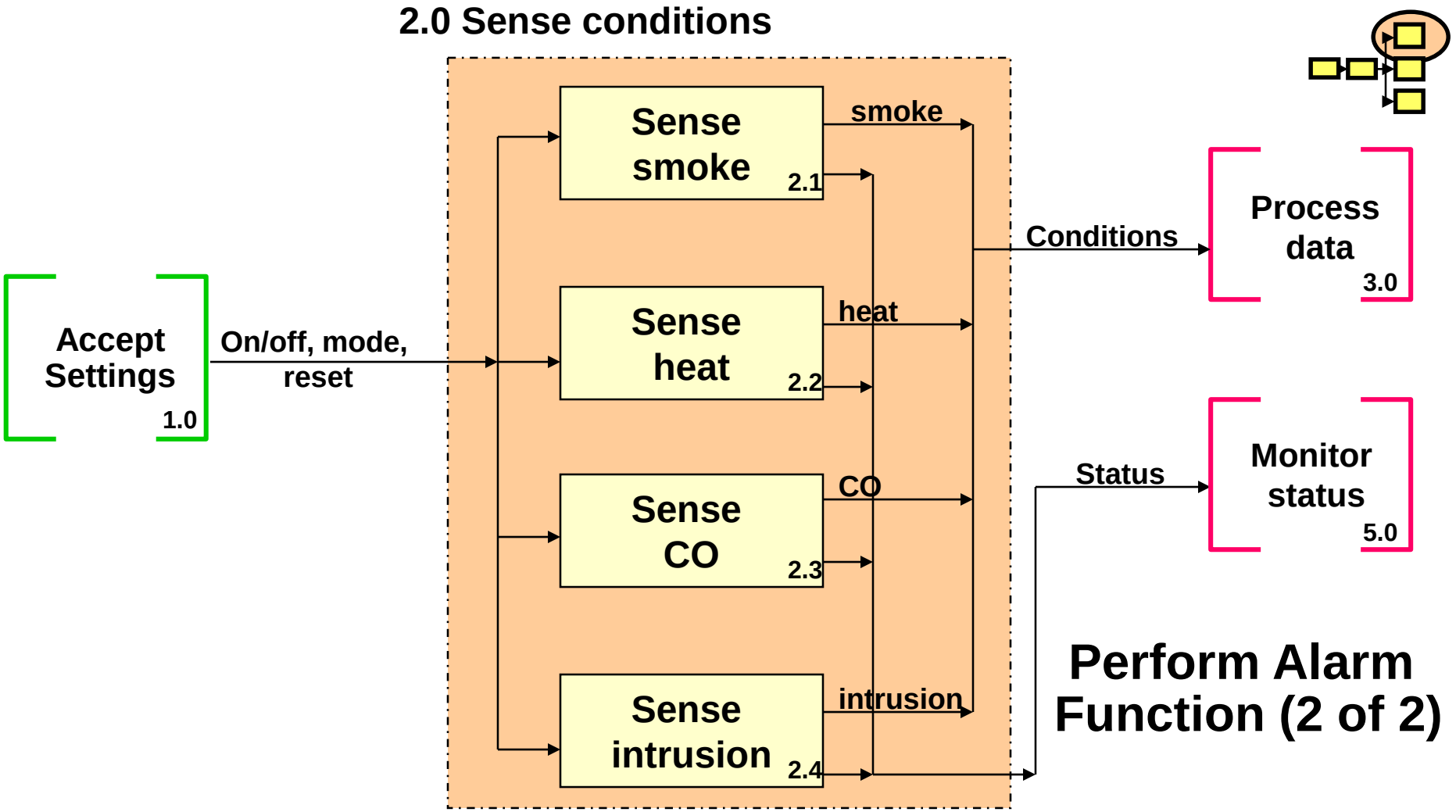
*Initiate or reset system includes:
- mode and threshold selection
- On/off selection

** Alarm status includes:
- Alarming or not alarming
- Alarm type
- Alarm location



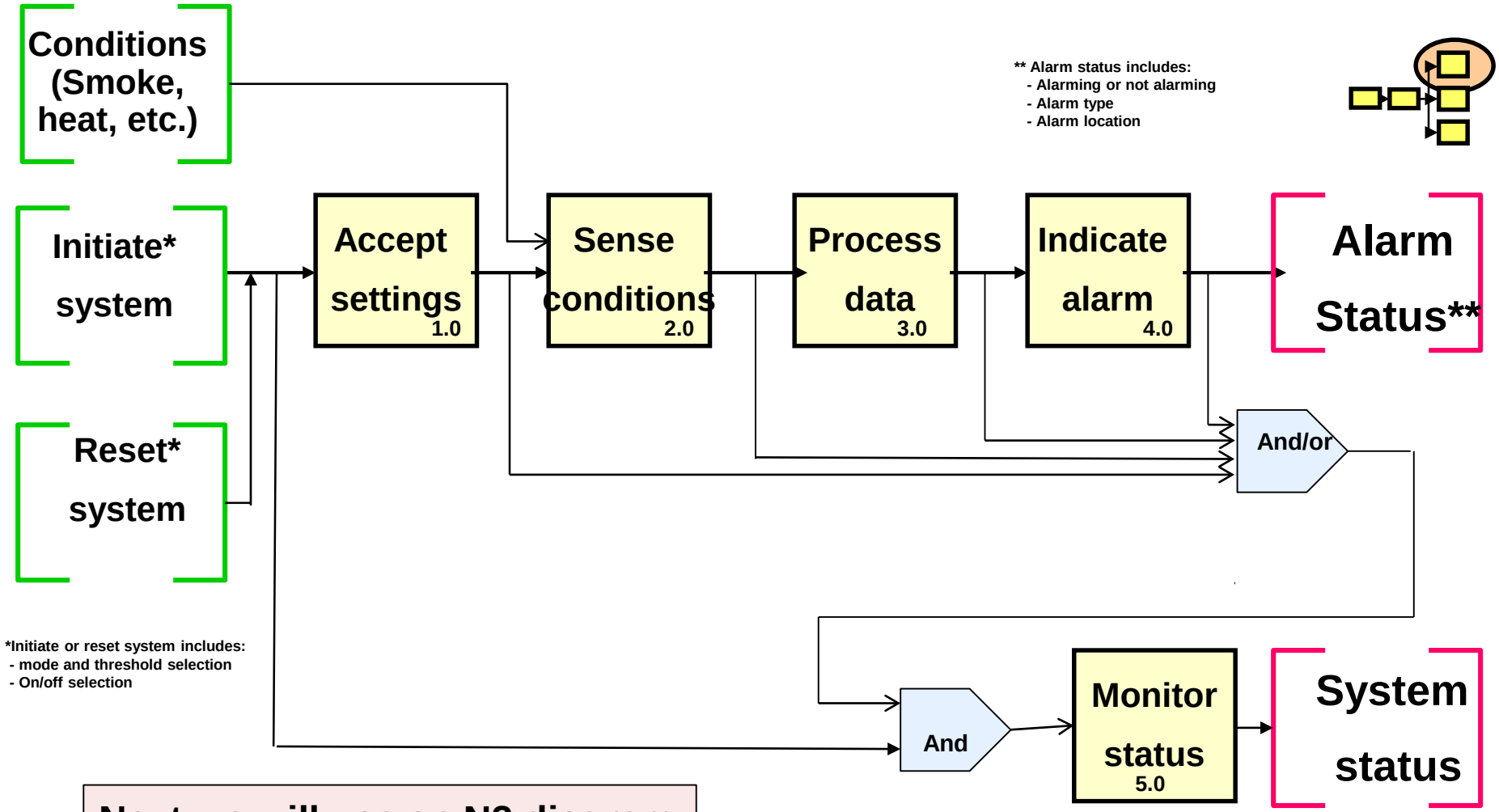
Perform Alarm Function (1 of 2)

Functional Flow Block Diagram: "Perform Alarm Function" Level 2



What inconsistency is identified by level 2 of this model?

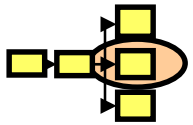
Functional Flow Block Diagram for "Perform Alarm Function" with Ideas from Previous Slide

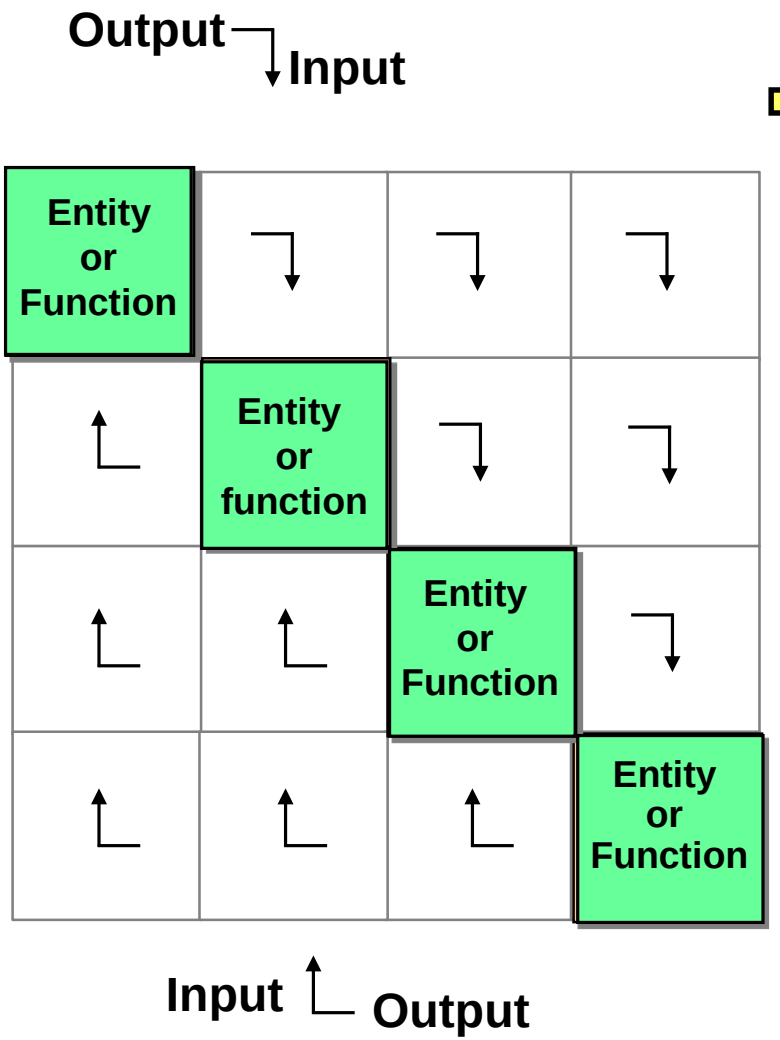


Next we will use an N2 diagram to develop the "interfaces"

Perform Alarm Function (1 of 2)

Generic N2 Diagram for Interface Analysis



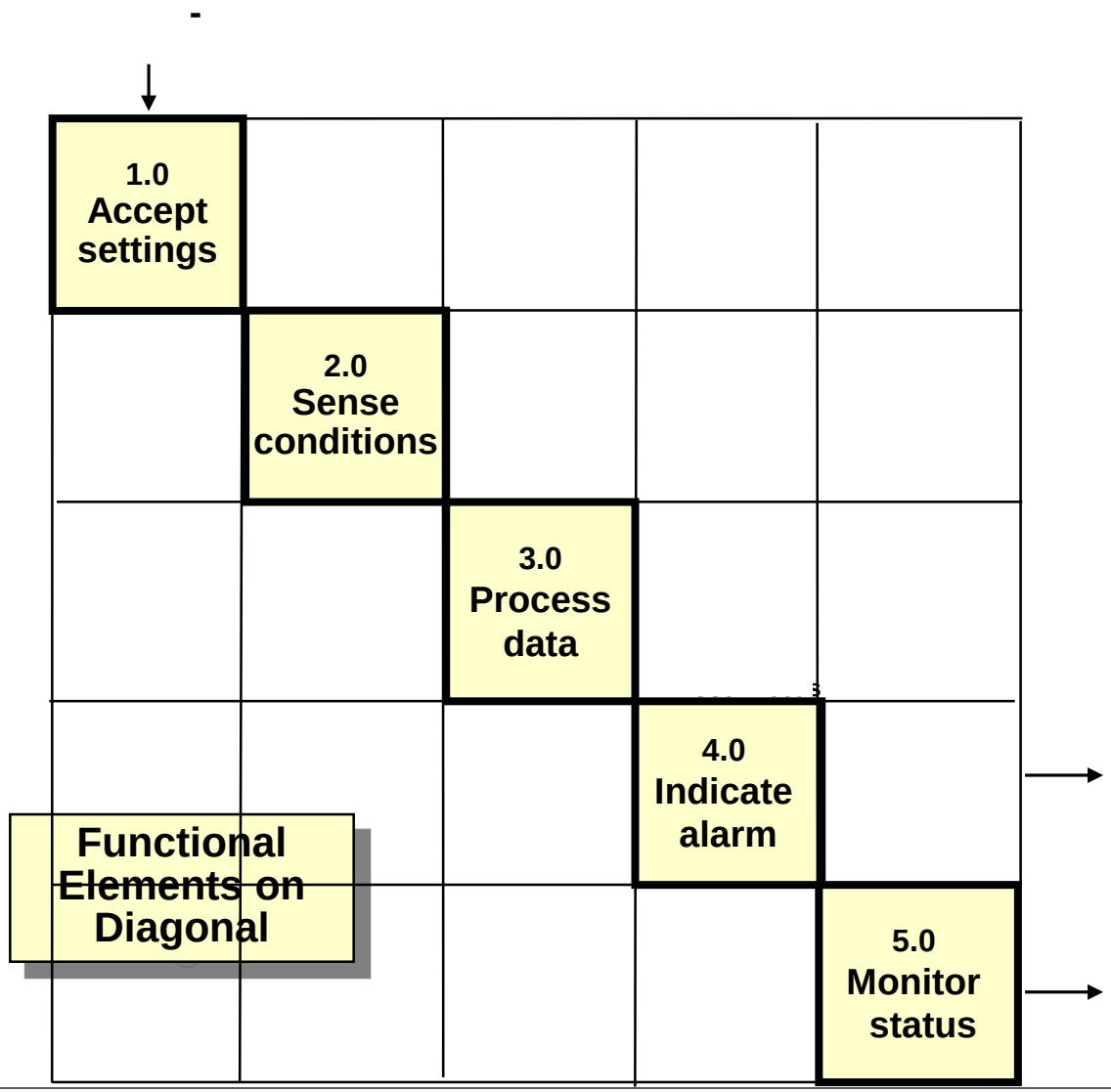


How to read N² Diagram

- Entities or functions are on the Diagonal.
- Entities have functions as inputs and outputs.
- Functions have Entities as inputs and outputs.
- Inputs are horizontal Outputs are Vertical

External inputs come in from the top; external outputs go out to the right

Alarm System Exercise – Part 5: Develop N2 Diagram for "Perform Alarm Function"



Instructions:

Create an N2 diagram for the Alarm System problem we have been using in the course.

Use the following as inputs to the exercise:

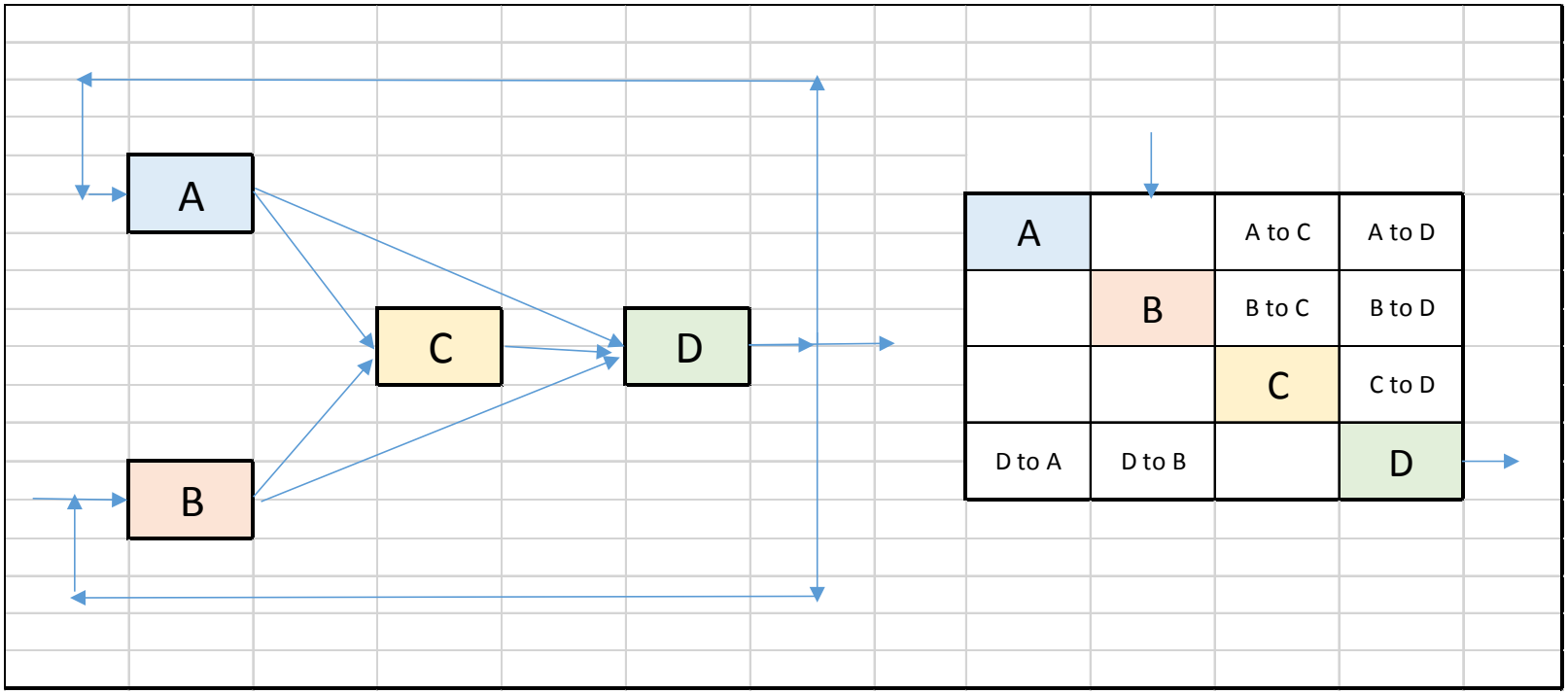
- **The template to the left**
- **The functional definition developed in this lesson**
- **Any other information you believe is relevant to the problem.**

Be prepared to brief the rest of the class on your result, and what you learned from the exercise

N2 Diagram for “Perform Alarm Function”

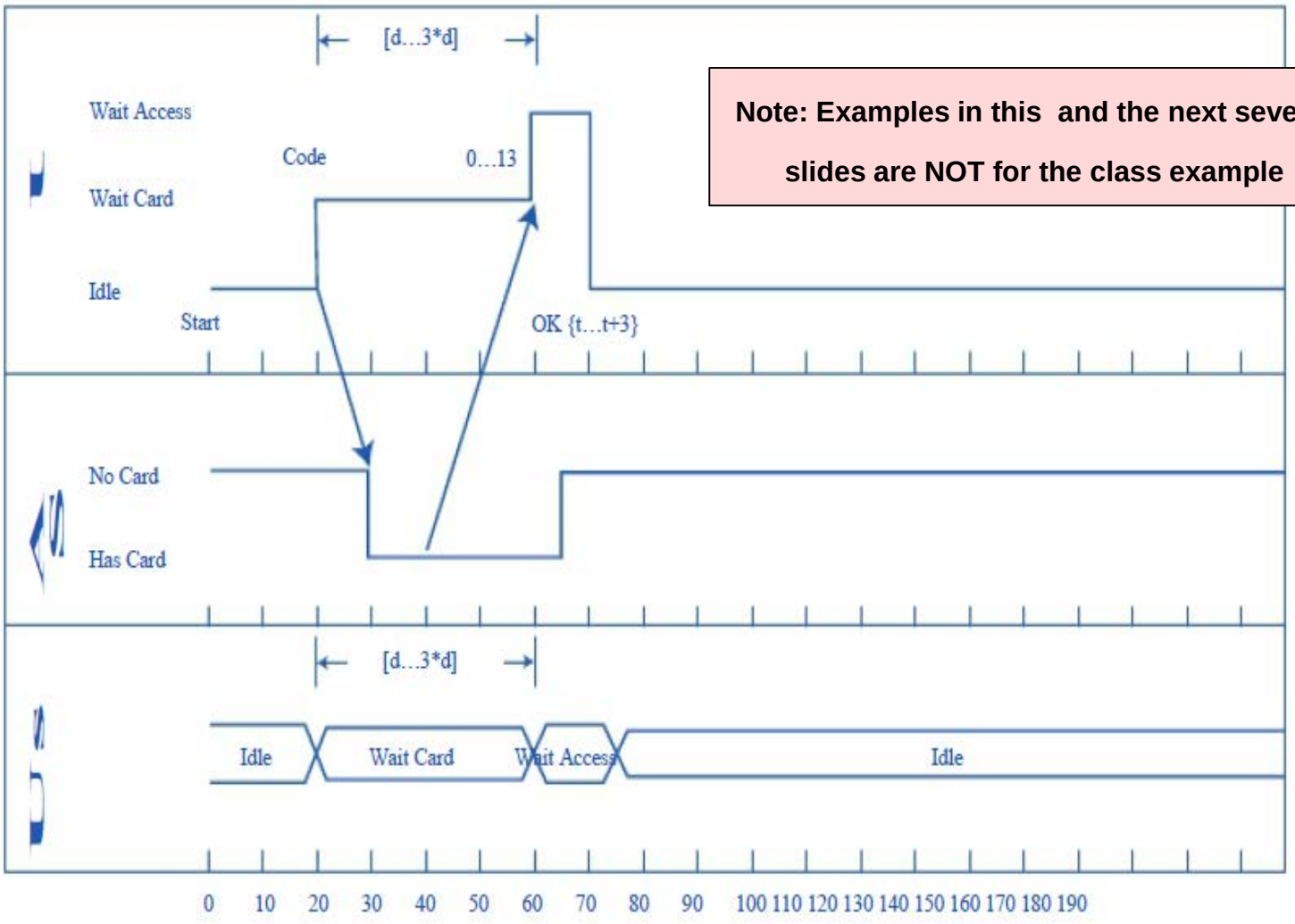
Slide will be uncovered during the lecture

Relationship between N^2 and FFBD



The N2 diagram can be used to label the FFBD interfaces

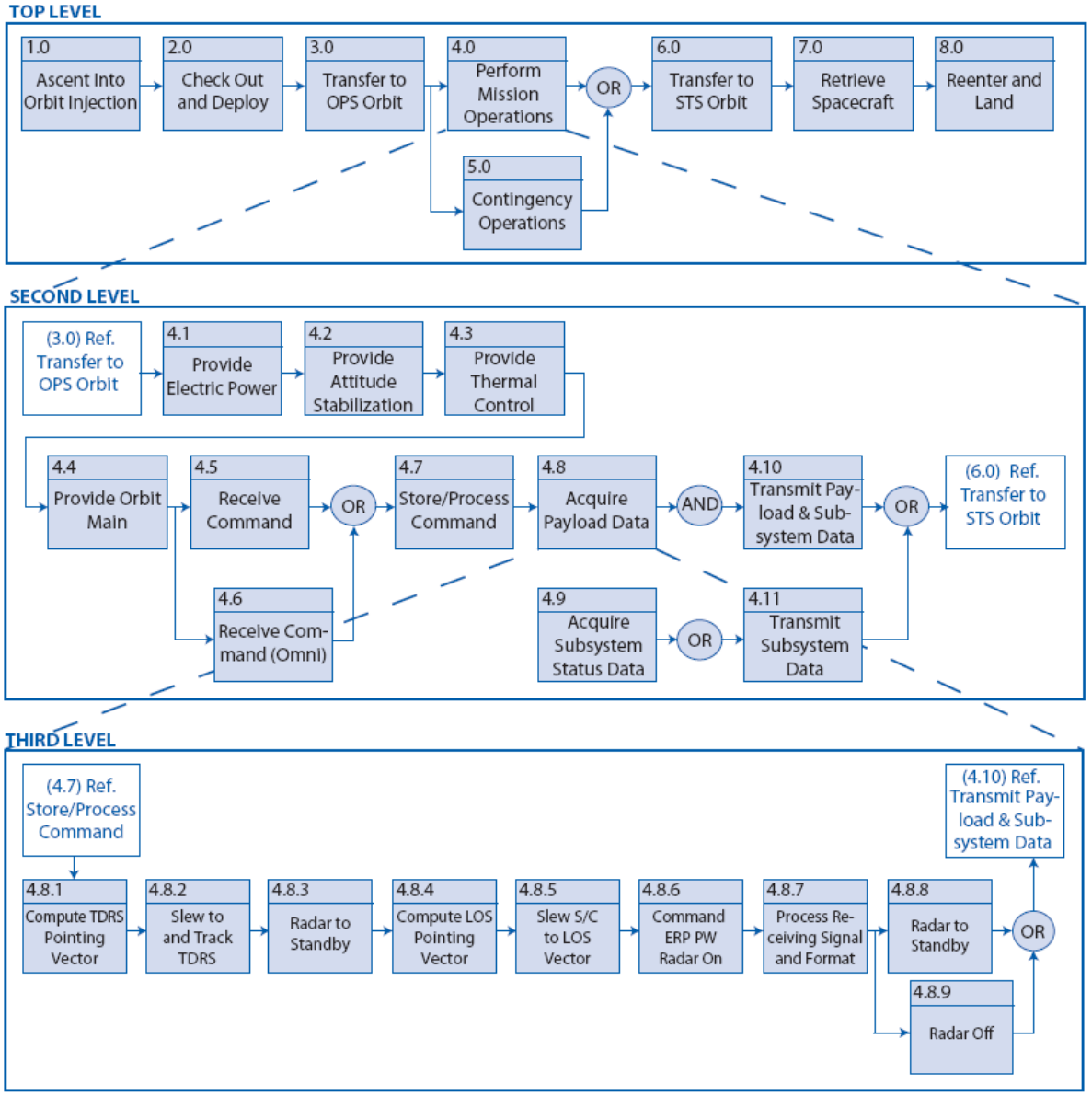
Timing Diagram - Example



Note: Examples in this and the next several slides are NOT for the class example

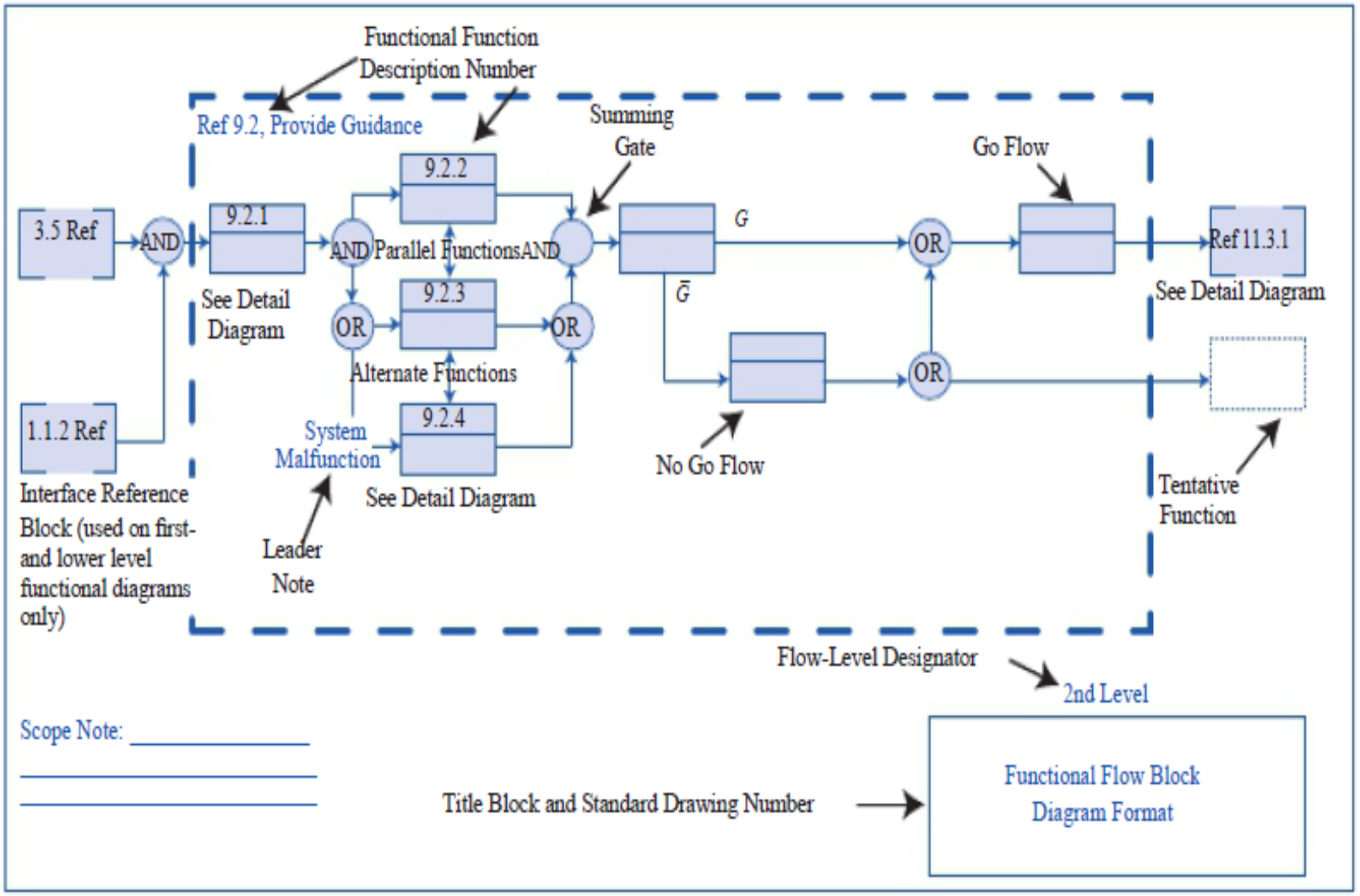
Source: NASA Systems Engineering Handbook, NASA/SP-2007-6105 Rev. 1. Washington, DC, 2007.

Functional Flow Block Diagram - Example



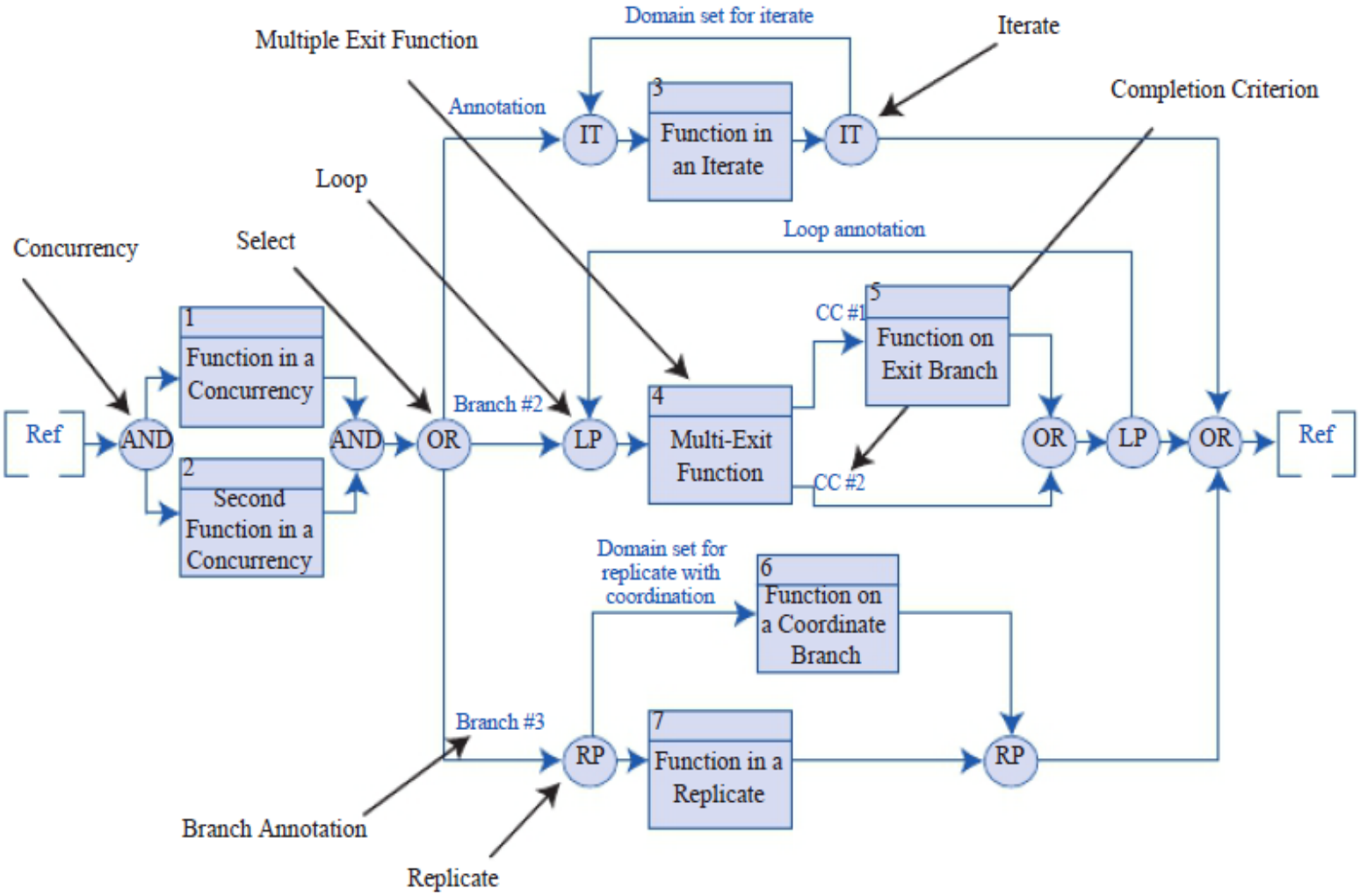
Source: NASA Systems Engineering handbook, NASA/SP-2007-6105 Rev. 1. Washington, DC, 2007.

Functional Flow Block Diagram - Example



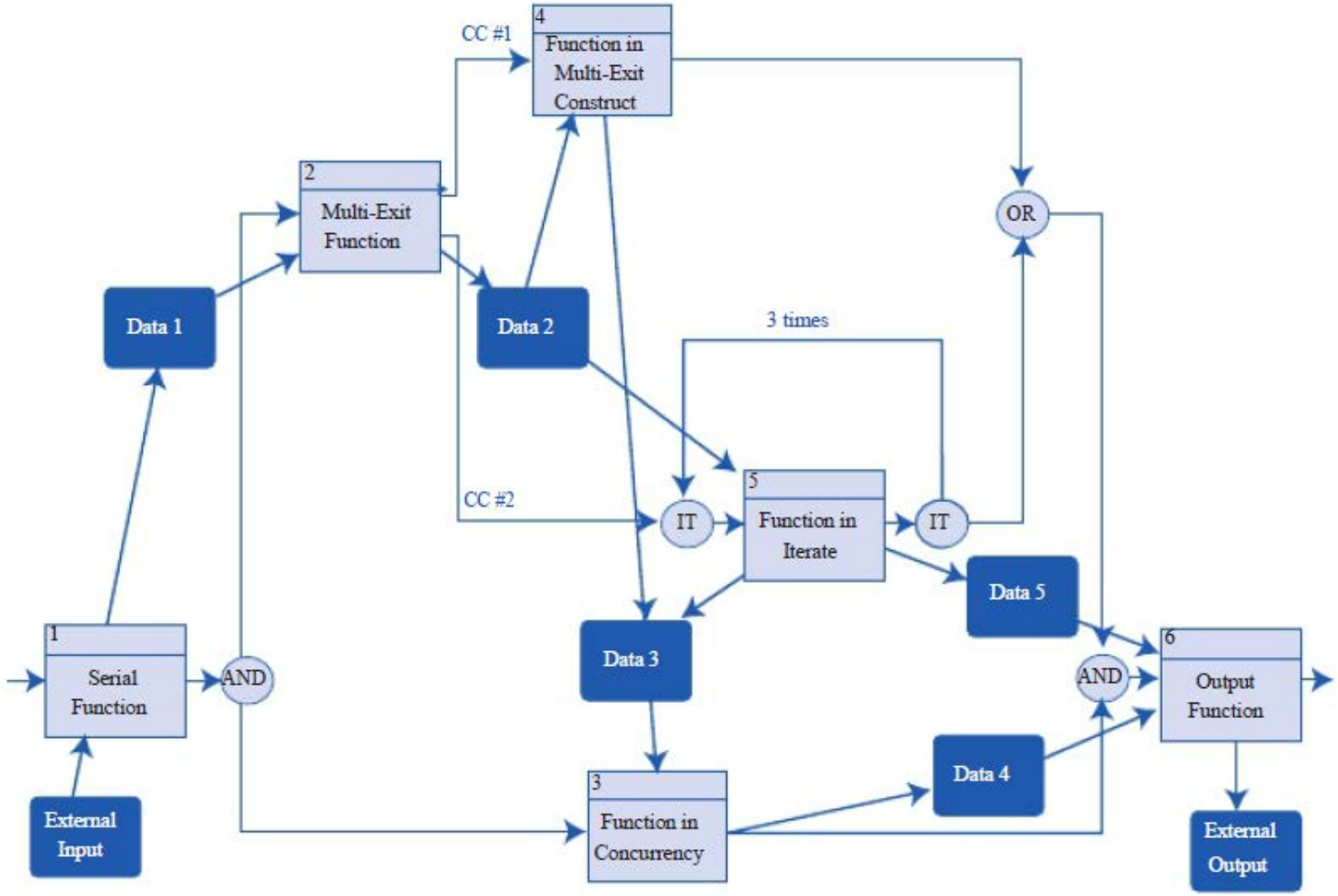
Source: NASA Systems Engineering handbook, NASA/SP-2007-6105 Rev. 1. Washington, DC, 2007.

Functional Flow Block Diagram – Example with Additional Control Constructs



Source: NASA Systems Engineering handbook, NASA/SP-2007-6105 Rev. 1. Washington, DC, 2007.

Extended Functional Flow Block Diagram Example



Source: NASA Systems Engineering handbook, NASA/SP-2007-6105 Rev. 1. Washington, DC, 2007.

Session 3 – System Requirements

Lessons

1. **Program Overview Update**
2. **Functional Requirements**
3. **Performance and Non-Functional Requirements**
4. **Requirements Management and Traceability**
5. **Special Topics (if time permits)**

- Lesson context
- A traditional approach
- Functional architecture (Buede)
- Graphical modeling Techniques

Appendices*

G. **Alarm System Exercise**

H. **Models and Methods**

I. **Other Appendices TBD**

*Appendices A through F were contained in earlier sessions of the course; Appendix D is a complete course bibliography. Appendices will not be formally covered



The slides in this section are adapted from those provided for instructors in conjunction with the course text:

The Engineering Design of Systems: Models and Methods, 2nd Edition

By Dennis Buede
Innovative Decisions, Inc.
dbuede@innovativedecisions.com

Chapter 7 – Functional Architecture Development

Terminology

- ***Mode*** **is a distinct operating capability of the system during which some or all of the system's functions may be performed to a full or limited degree**
- ***State of the system*** **is a static snapshot of the set of metrics or variables needed to describe fully the system's capabilities to perform the system's functions**
- ***Function*** **is a process that takes inputs in and transforms these inputs into outputs**
- ***Functionality*** **is a set of functions that is required to produce a particular output**
  - ***Simple functionality*****: an *ordered* sequence of functional processes that operates on a single input to produce a specific output**
  - ***Complete functionality:*** **a complete set of coordinated processes that operate on all of the necessary inputs for producing a specific output**

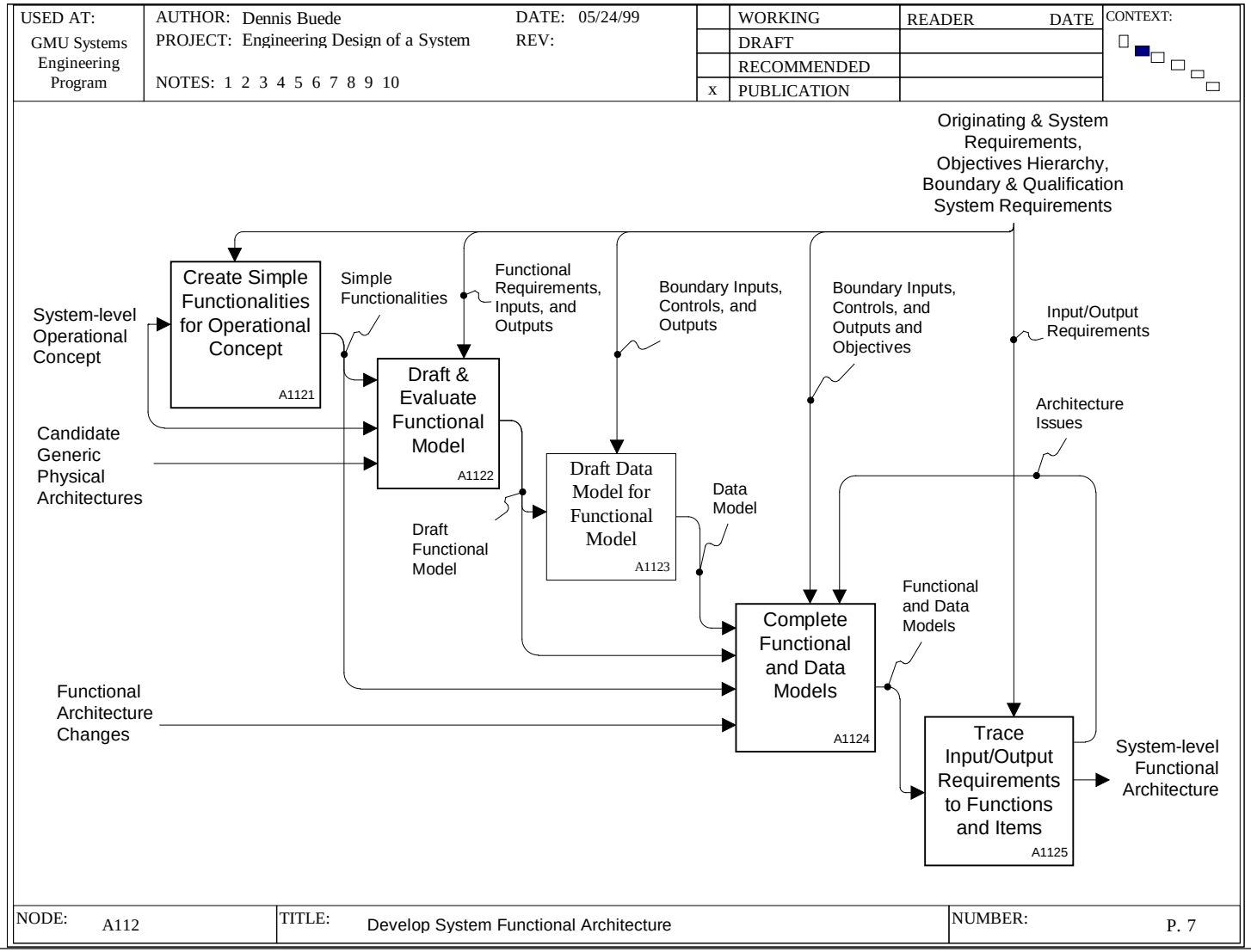
Functional Architecture

- A logical architecture that defines what the system must do, a decomposition of the system's top-level function. This very limited definition of the functional architecture is the most common and is represented as a directed tree.

- A logical model that captures the transformation of inputs into outputs using control information. This definition adds the flow of inputs and outputs throughout the functional decomposition; these items that comprise the inputs and outputs are commonly modeled via a data model (see Chapter 12). An IDEF0 model without any mechanisms is used as the modeling technique in this chapter to represent the functional architecture at this level of detail.

- A logical model of a functional decomposition plus the flow of inputs and outputs, to which input/output requirements have been traced to specific functions and items (inputs, outputs, and controls)

Functional Architecture Process



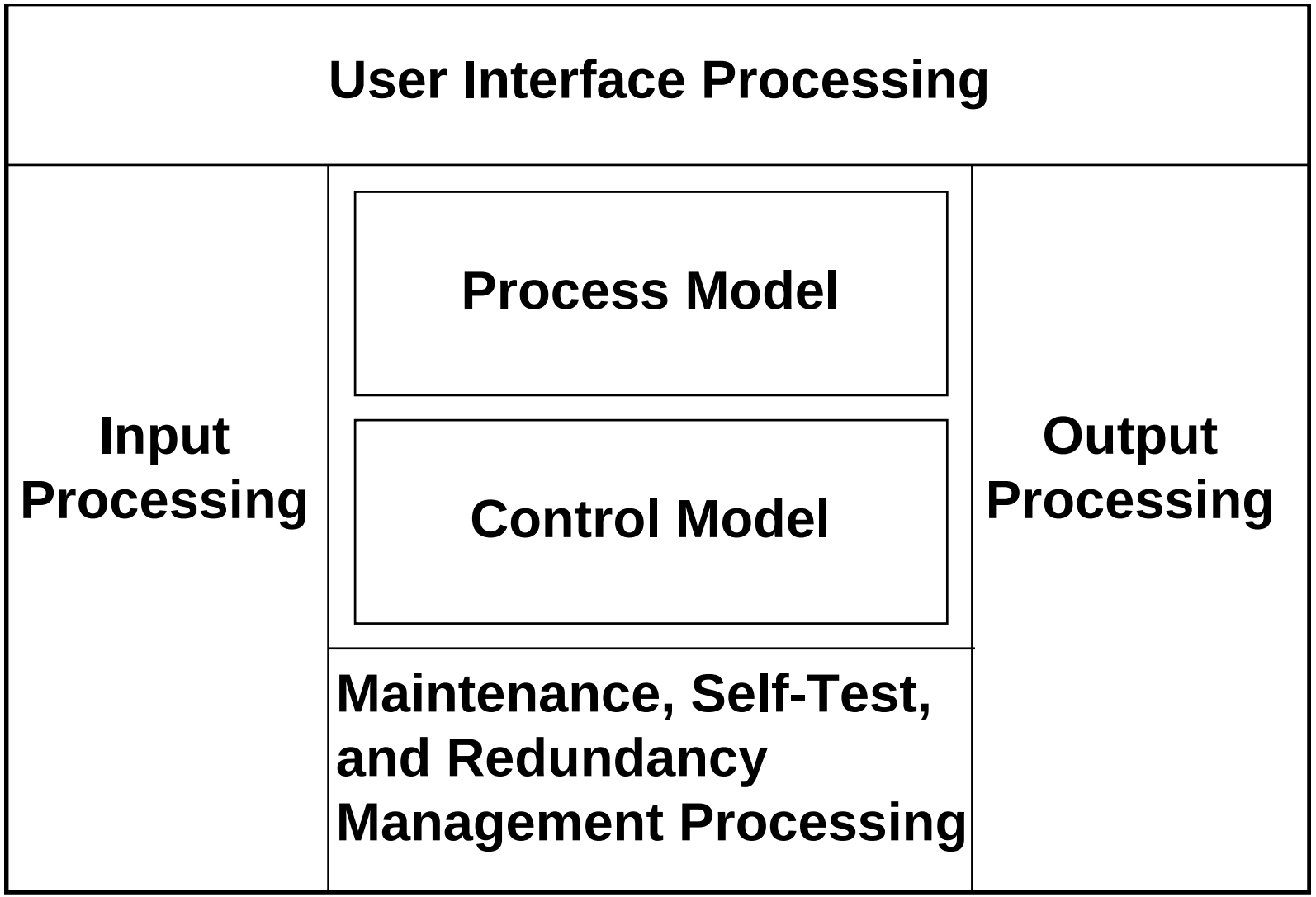
Decomposition vs. Composition

- **Decomposition (top-down)**
  - **Partition system function a level at a time**
  - **Need sound definition of all inputs & outputs**
- **Composition (bottom-up)**
  - **Define many functionalities (bottom-level functions)**
  - **Synthesize functional hierarchy from many bottom-level functions**
- **Use both is the best solution**

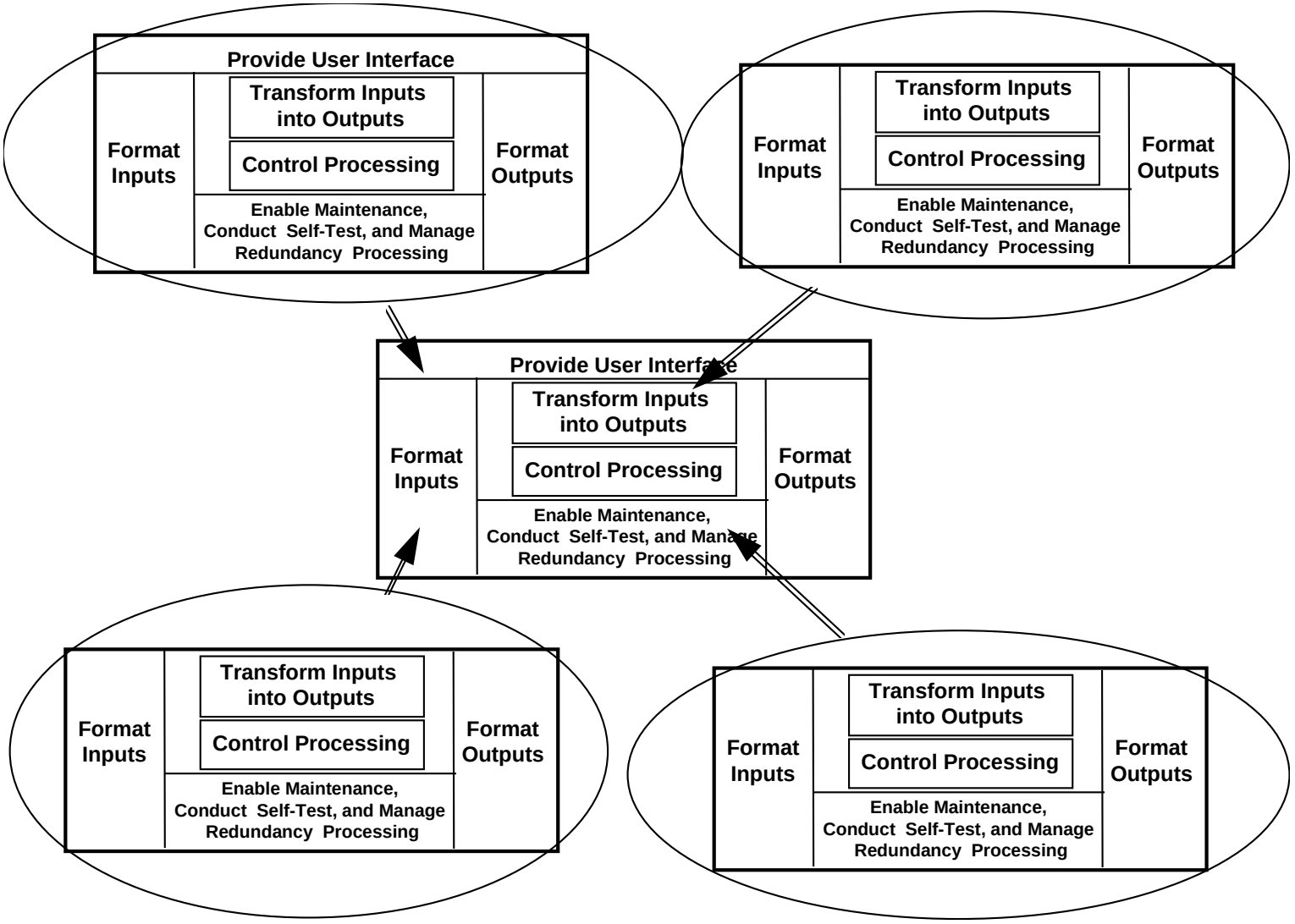
Defining Functional Partitions

- Use operating modes
- Use outputs
- Use inputs & controls
- Use stimulus-response threads
- Use Hatley-Pirbhai template
- Use “living systems” template
- Segment into fundamental and custodial

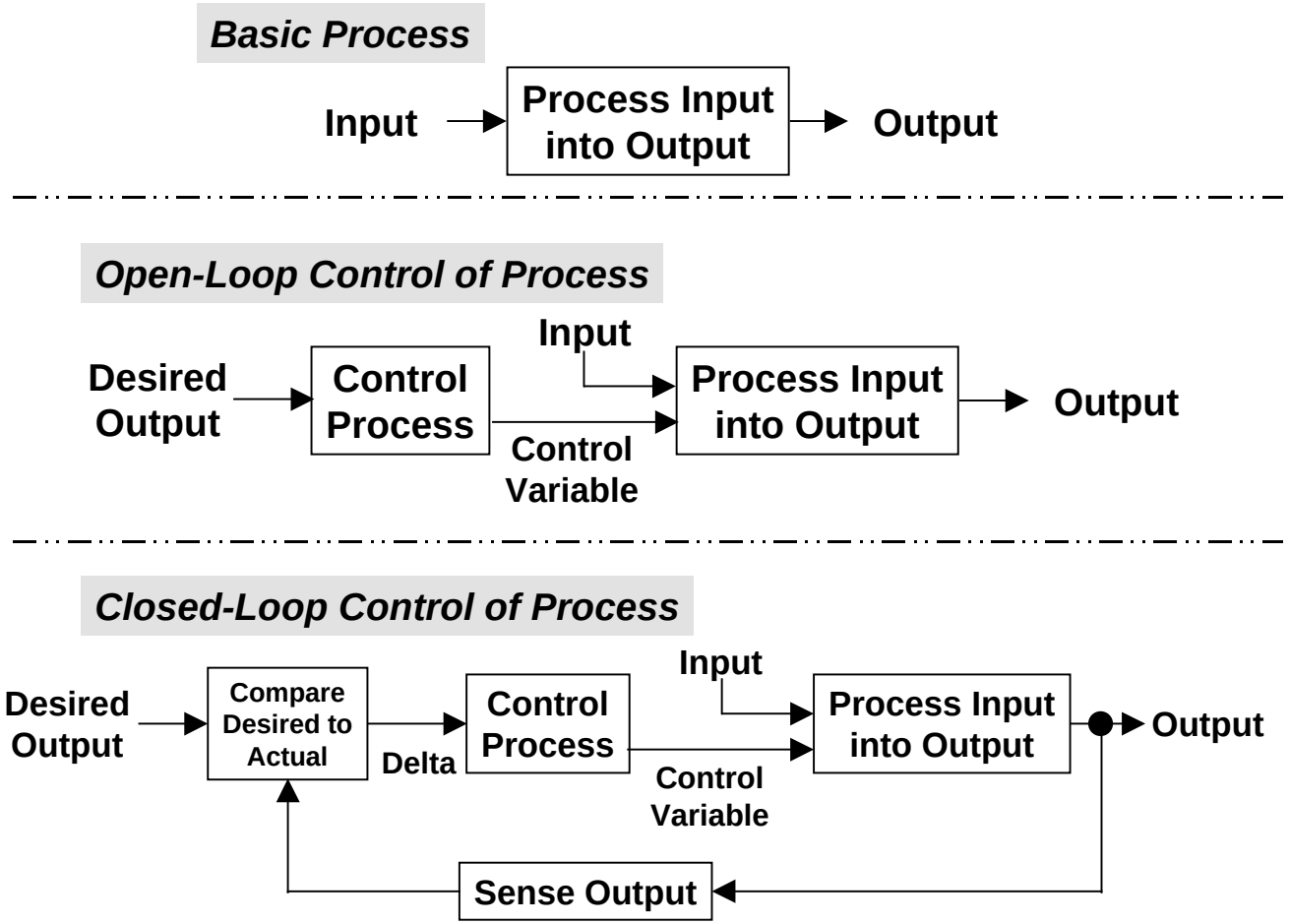
Hatley-Pirbhai Template



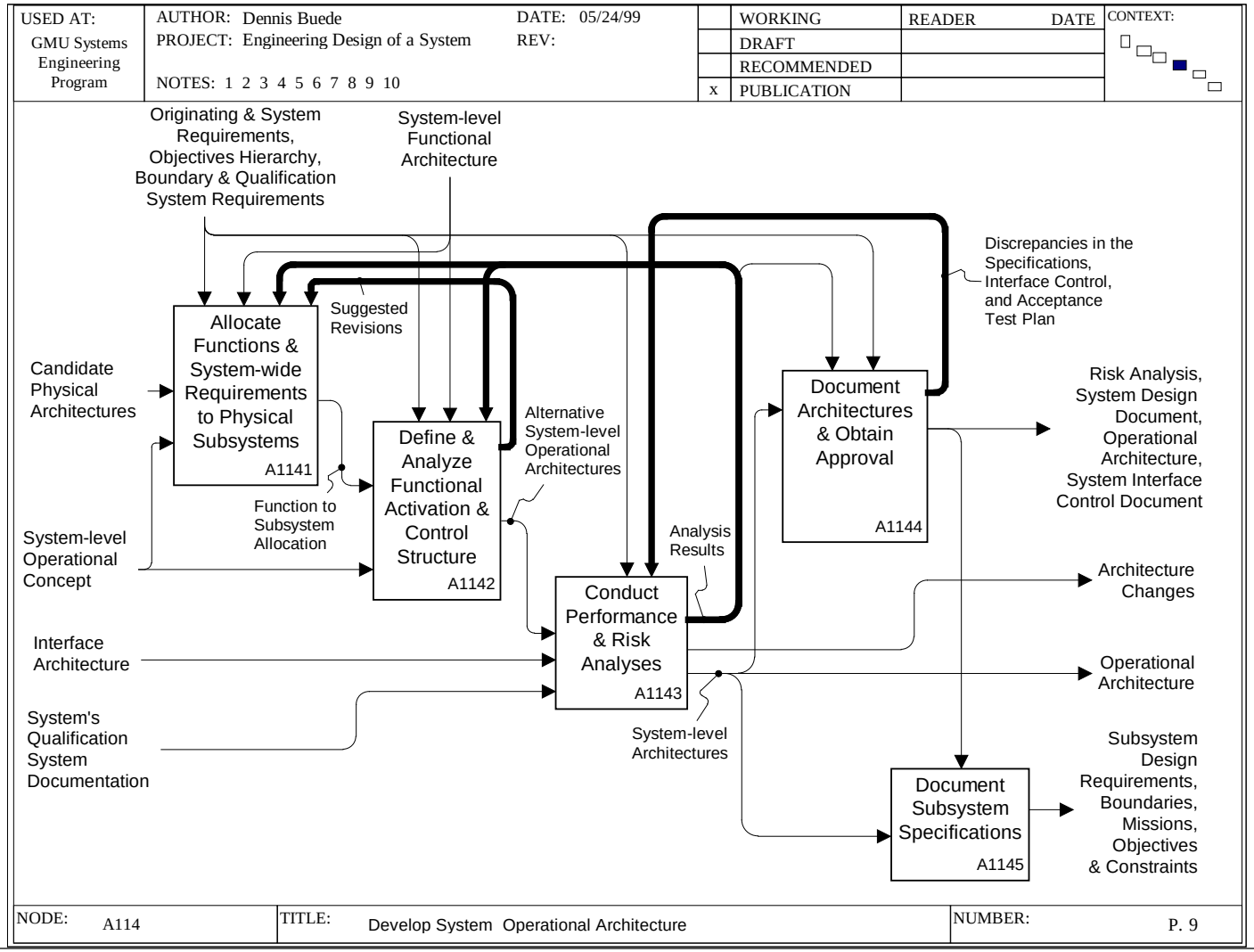
Hatley-Pirbhai Decomposition



Feedback & Control in Functional Design



Feedback Control in Operational Architecture Development



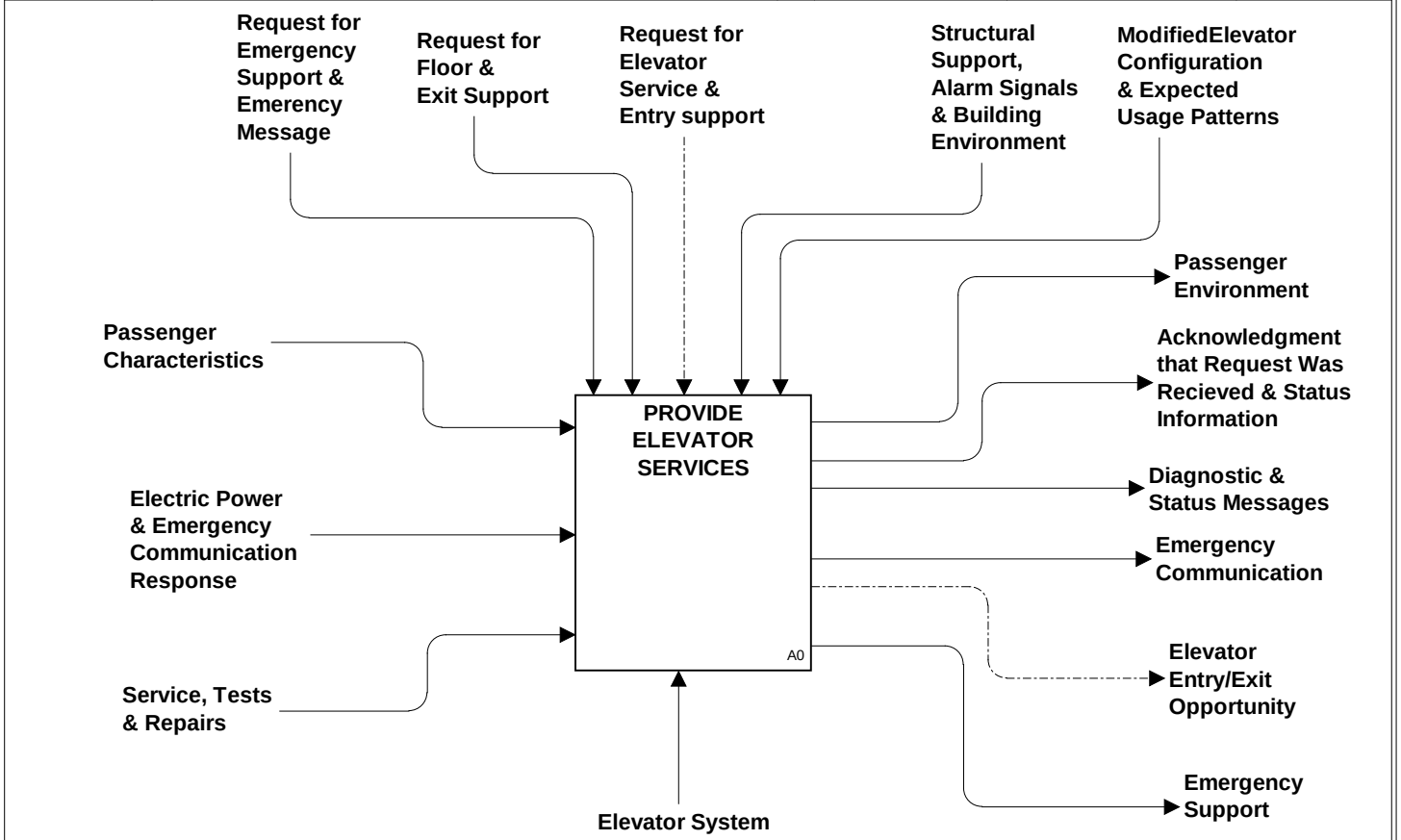
The Engineering Design of Systems: Models and Methods, 2nd Edition, by Dennis Buede

Evaluation of Functional Hierarchy

- **Shortfalls – absence of functionality**
  - **Absence of functionalities for input set**
  - **Inability to produce desired output**
  - **Insufficient feedback/control to produce desired output**
- **Overlaps – redundancy in functionality**
- **Evaluation technique – scenario tracing**

Scenario Tracing, #1

USED AT:	AUTHOR: Dennis Buede	DATE: 05/24/99		WORKING	READER DATE	CONTEXT:
George Mason Univ.	PROJECT: Elevator Case Study	REV:	x	DRAFT		Top
	NOTES: 1 2 3 4 5 6 7 8 9 10			RECOMMENDED		
				PUBLICATION		



NODE: A-0	TITLE: Provide Elevator Services	NUMBER: P. 2

The course text elaborates this illustration of scenario tracing using IDEF0 modeling

Tracing Requirements

Functions	Input/Output Requirements (A Sample)					
	Input Requirements		Output Requirements		Functional Requirement	External Interface Requirement
	The elevator system shall receive calls for up and down service from all floors of the building.	The elevator system shall receive passenger activated fire alarms in each elevator car.	The elevator system shall provide adequate illumination.	The elevator system shall open and close automatically upon arrival at each selected floor.	The elevator system shall control elevator cars efficiently.	The elevator system shall use a phone line from the building for emergency calls.
0 Provide Elevator Services	X	X	X	X	X	X
1 Accept Passenger Requests + Provide Feedback	X	X				X
1.1 Support Waiting Passengers	X					
1.2 Support Riding Passengers						
1.3 Support Passengers in Emergency		X				X
2 Control Elevator Cars						
3 Move Passengers between Floors			X	X		
3.1 Receive + Discharge Passengers				X		
3.2 Travel to Next Stop						
3.3 Provide Comfortable Atmosphere			X			
4 Enable Effective Maintenance and Servicing						

Common Mistakes (1/3)

1. **Including the external systems and their functions. The functional architecture only addresses the top-level function of the system in question. The external system diagram establishes the inputs, controls, and outputs for this function. A boundary has been drawn around the system to exclude the external systems and their functions.**

2. **Choosing the wrong name for a function. The function name should start with an action verb and include an object of that action. The verb should not contain an objective or performance goal such as maximize, but should describe an action or activity that is to be performed.**

Common Mistakes (2/3)

3. **Creating a decomposition of a function that is not a partition of that function. For example, a student once decomposed "A0: Provide Elevator Services" into "A1: Transport Users," "A2: Evaluate System Status," and "A3: Perform Security & Maintenance Operations." "A1: Transport Users" was then decomposed as follows: "A11: Provide Access to Elevator," "A12: Transport Users," and "A13: Provide Emergency Operations." A12 cannot be a child of itself. The subfunctions of a function should all be at the same level of abstraction [Chapman et al., 1992].**

4. **Including a verb phrase as part of the inputs, controls, or outputs of a function. Verb phrases are reserved for functions.**

Common Mistakes (3/3)

5. Violating the law of conservation of inputs, controls, and outputs. That is, every input, control, and output of a particular function must appear on the decomposition of that function, and there can be no new ones.

6. Trivializing the richness of interaction between the functions that decompose their parent. Consider many possible simple functionalities that comprise the children of a parent function and then develop the inputs, controls, and outputs that enable these simple functionalities to exist, including the necessary feedback and control.

7. Creating outputs from thin air. The most common mistake is to define a function that monitors the system's status but that does not receive inputs about the functioning or lack of functioning of other parts of the system.

Alarm System Exercise – Part 6: Scenario Tracing



Instructor will provide directions for this exercise

Session 3 – System Requirements

Lessons

1. **Program Overview Update**
2. **Functional Requirements**
3. **Performance and Non-Functional Requirements**
4. **Requirements Management and Traceability**
5. **Special Topics (if time permits)**

- Lesson context
- A traditional approach
- Functional architecture (Buede)
- Graphical Modeling Techniques

Appendices*

G. **Alarm System Exercise**

H. **Models and Methods**

I. **Other Appendices TBD**

*Appendices A through F were contained in earlier sessions of the course; Appendix D is a complete course bibliography. Appendices will not be formally covered



The slides in this section are adapted from those provided in Conjunction with our course text:

The Engineering Design of Systems: Models and Methods, 2nd Edition

By Dennis Buede
Innovative Decisions, Inc.
dbuede@innovativedecisions.com

Chapter 12 – Graphical Modeling Techniques

Buede Ch 12 is discussed in System Architecture & design, and is shown here also due to its importance to Requirements Development

Modeling Approaches & Methods

- **Data Modeling** (relationships between inputs and outputs)
  - Entity-relationship diagrams (ERDs)
  - Higraphs
- **Process Modeling** (functional decomposition and flow of inputs and outputs for those functions)
  - Data flow diagrams (DFDs)
  - IDEF0
  - N^2 charts
- **Object-oriented Modeling***
  - Object modeling technique (OMT)
  - Real-time object-oriented modeling (ROOM)

- **Behavior Modeling** (control, activation, and termination of system functions needed to meet performance requirements)
  - Function flow block diagrams (FFBDs)
  - Behavior diagrams (BDs)
  - State-transition diagrams (STDs)
  - Statecharts
  - Control flow diagrams (CFDs)
  - Petri nets (PNs)

Many examples of these are shown in the course text: *The Engineering Design of Systems*, by Dennis Buede

*Buede states that OOD is increasingly used in SE, and uses the other three domains identified on this chart

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		

All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



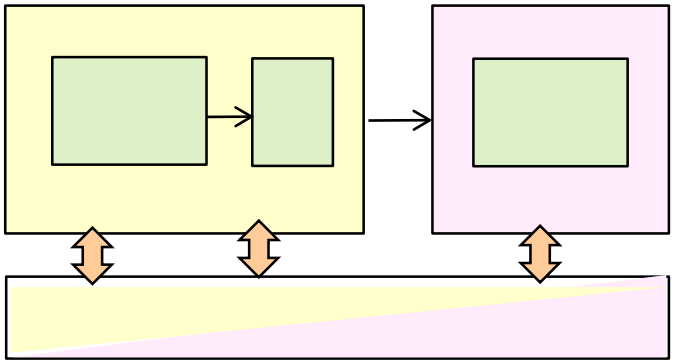
What are your Questions before We Continue?

Back-up slides

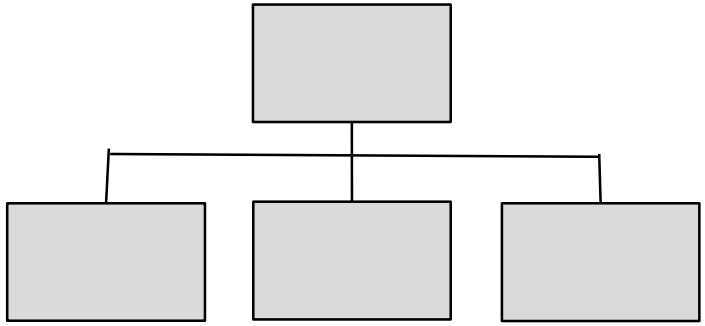
- The following slides are available for use to help context the material in this lesson.
- These slides are best used right after slide 6

The Requirements Engineering Process Context for this Course (Left Side of Vee) (1/6)

Process: Left side of the Vee
For all Architecture Levels



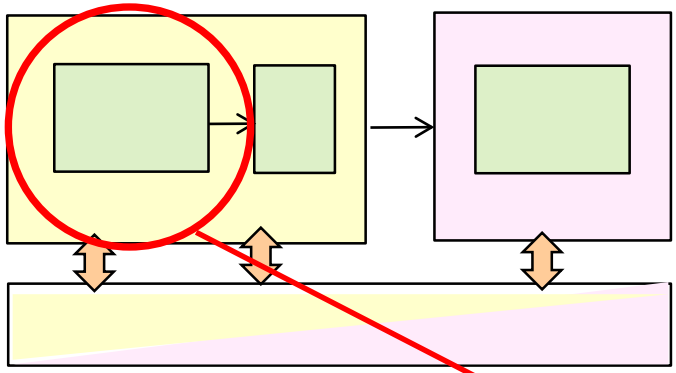
Product Breakdown Structure
For Your System of Interest



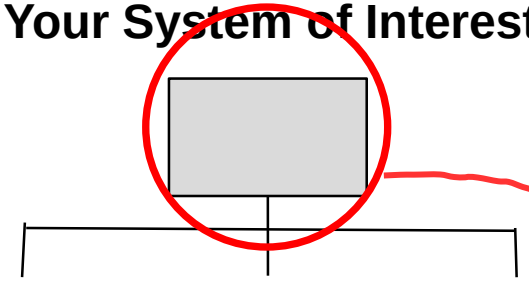
The slide sequence starting with this slide is best viewed after slides 6

The Requirements Engineering Process Context for this Course (Left Side of Vee) (2/6)

Process: Left side of the Vee
For all Architecture Levels



Product Breakdown Structure
For Your System of Interest

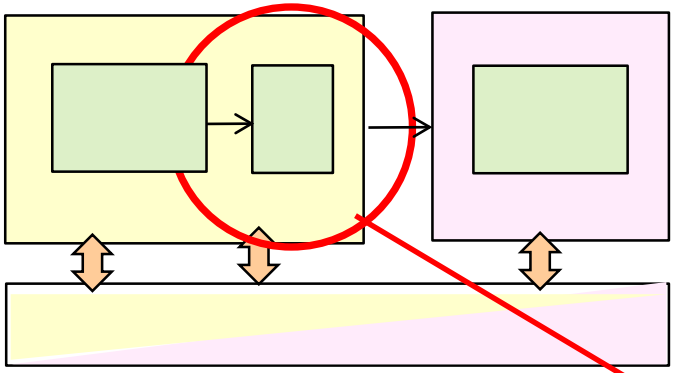


We start by Determining the Stakeholder Requirements for the top level of your system of interest.
This is a solution-independent problem;

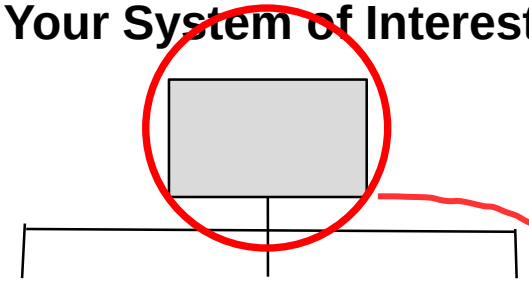
This is what we have done through the Session 2

The Requirements Engineering Process Context for this Course (Left Side of Vee) (3/6)

Process: Left side of the Vee
For all Architecture Levels



Product Breakdown Structure
For Your System of Interest

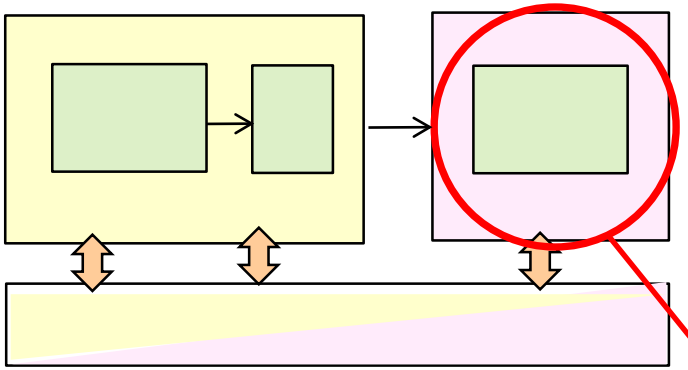


Then we develop the system requirements for the top level of your system of interest by performing requirements analyses (Functional, performance and non-functional). This is a solution-independent problem;

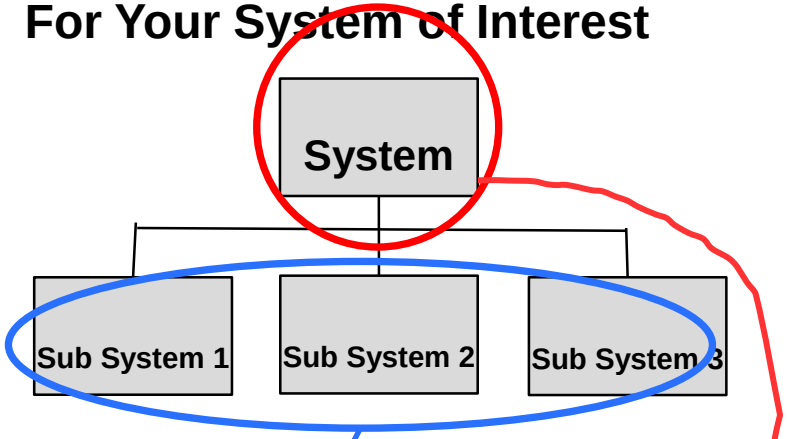
This is what we will do next

The Requirements Engineering Process Context for this Course (Left Side of Vee) (4/6)

Process: Left side of the Vee
For all Architecture Levels



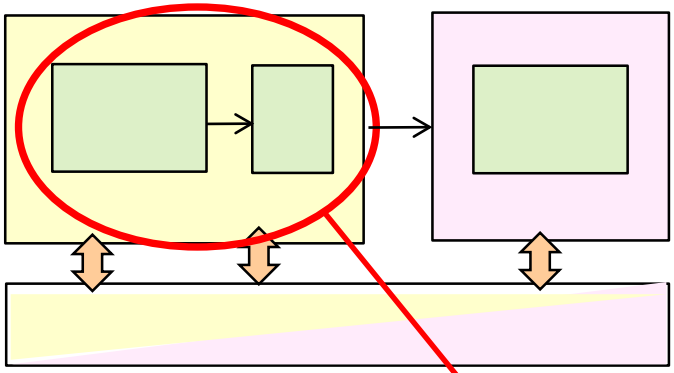
Product Breakdown Structure
For Your System of Interest



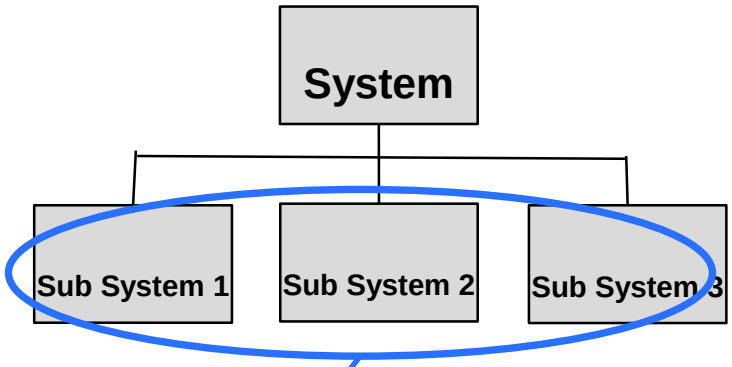
Then we identify and select the solution
for the top level of your system of interest.
This includes identifying the entities
at the next level of the PBS.
This begins the solution-dependent process

The Requirements Engineering Process Context for this Course (Left Side of Vee) (5/6)

Process: Left side of the Vee
For all Architecture Levels



Product Breakdown Structure
For Your System of Interest

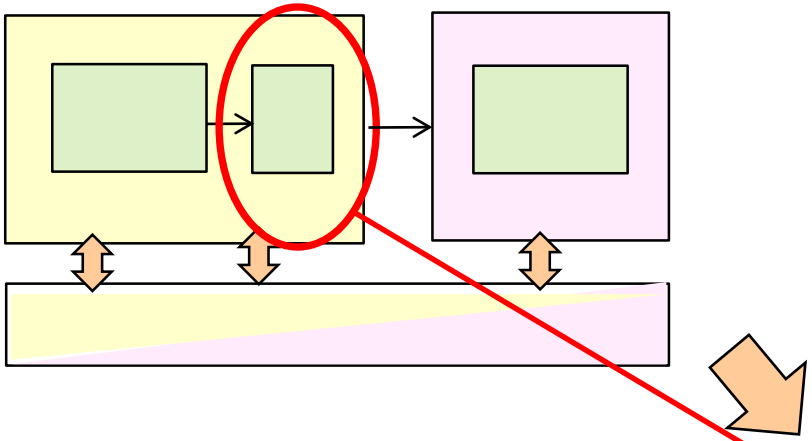


Then we repeat the process for each of the subsystems
But now we start with the system level requirements,
allocate them to the subsystems,
and perform the S/H requirements and system requirements analyses for each subsystem
This is solution-dependent from the System perspective,
but initially solution-independent at subsystem level

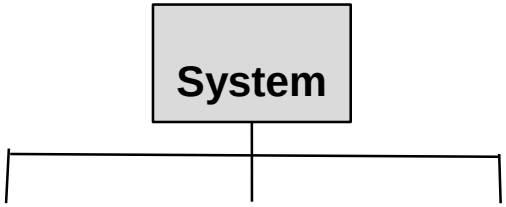
This involves the principles and practices from session 2, 3, and 4

The Requirements Engineering Process Context for this Course (Left Side of Vee) (6/6)

Process: Left side of the Vee
For all Architecture Levels



Product Breakdown Structure
For Your System of Interest



Note: when we do the systems functional and performance analyses at the systems level, we decompose the problem functionally, but not yet physically:
Therefore, we identify functional and performance requirements that will ultimately be performed by the subsystems;
We just haven't yet allocated to the solution space yet
This complete understanding of the problem in problem space is what Buede calls the (solution-independent) Functional Architecture



Systems Engineering Technical Short Course

(Session 3: System Requirements)

Lesson 3: Performance and Non-Functional Requirements

Session 3 – System Requirements

Lessons

1. **Program Overview Update**
2. **Functional Requirements**
3. **Performance and Non-Functional Requirements**
4. **Requirements Management and Traceability**
5. **Special Topics (if time permits)**

- Performance requirements
- What are non-functional requirements
- Discovering non-functional requirements
- Requirements flow down

Appendices*

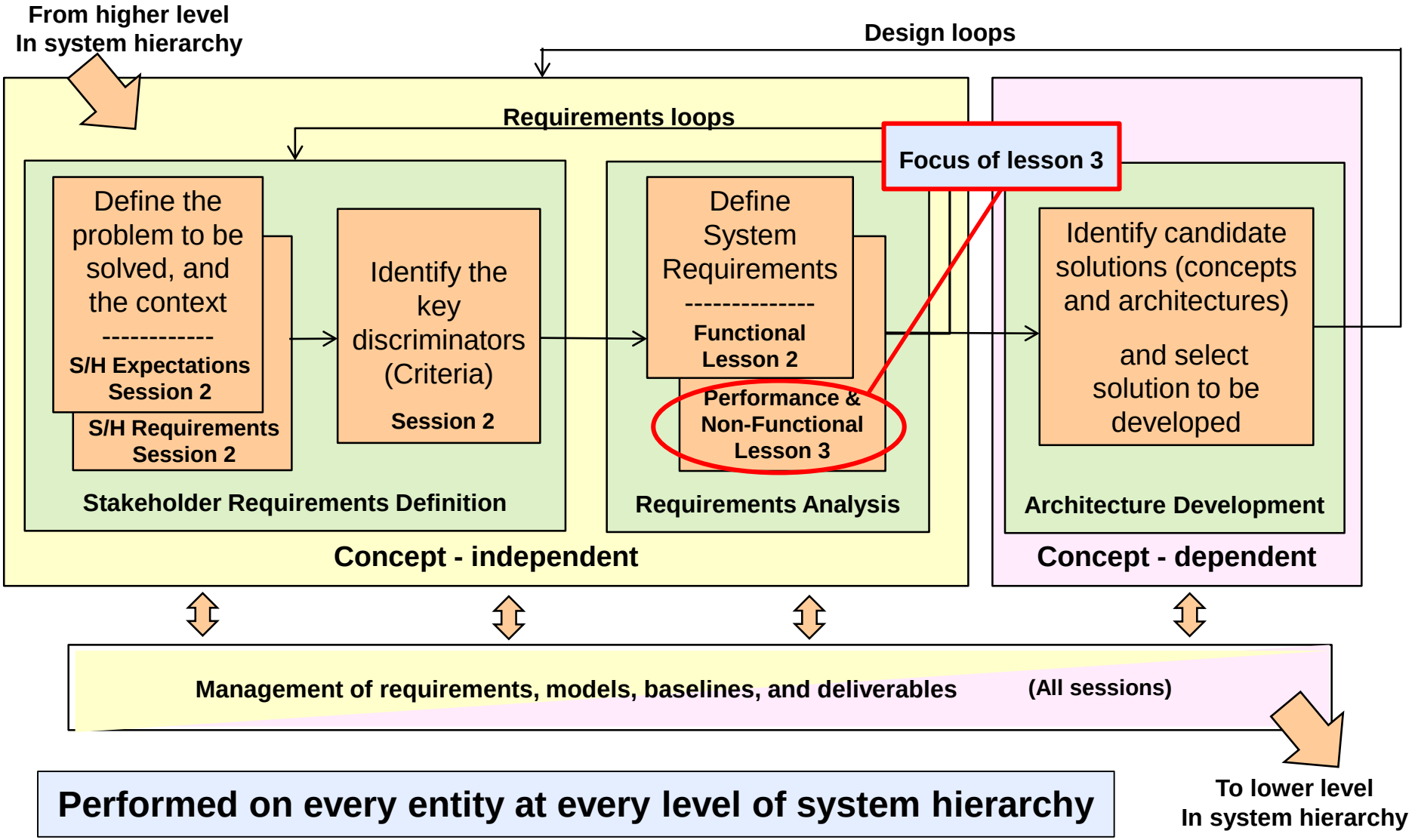
G. **Alarm System Exercise**

H. **Models and Methods**

I. **Other Appendices TBD**

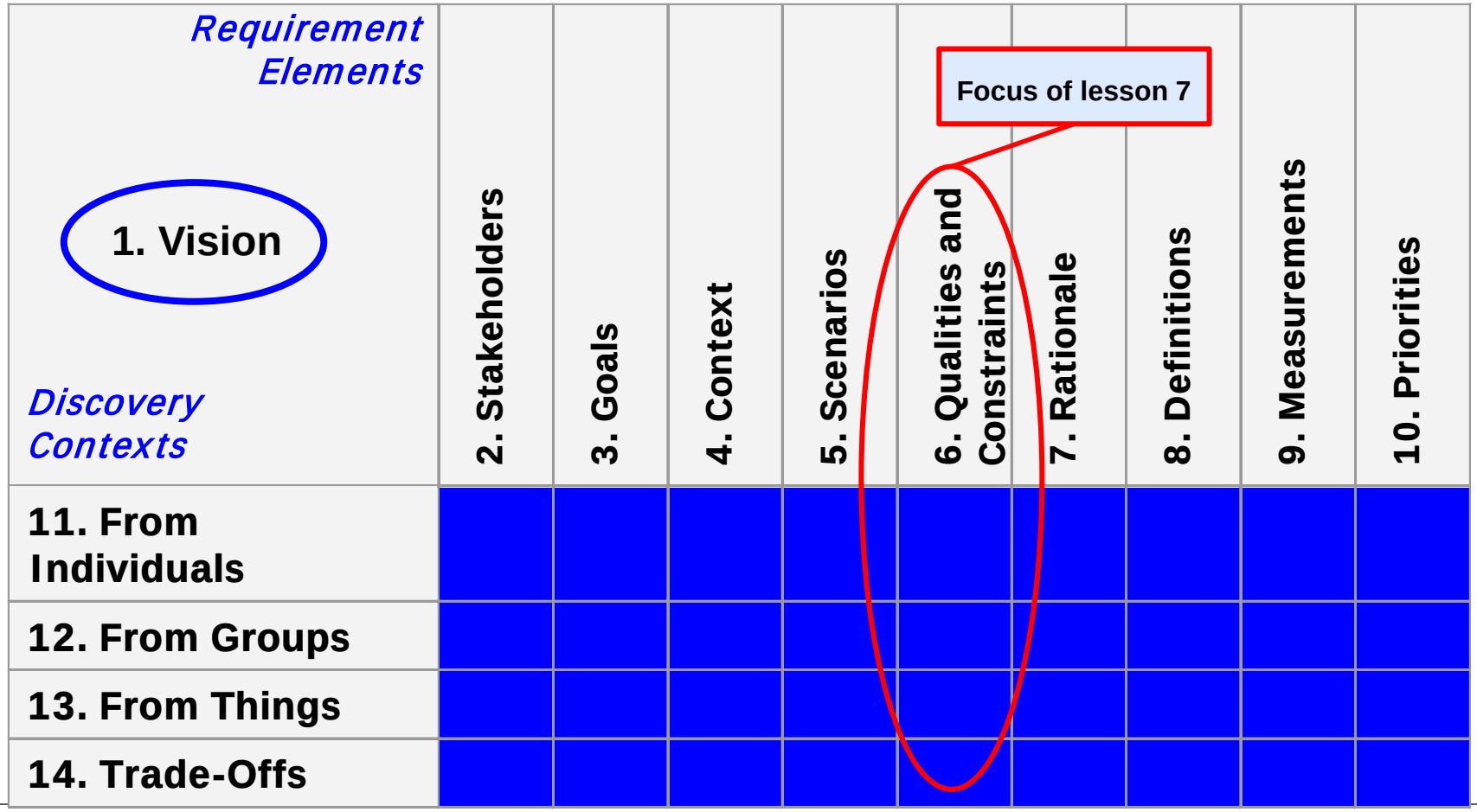
*Appendices A through F were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

The Requirements Engineering Process Context for this Course (Left Side of Vee)



Performed on every entity at every level of system hierarchy

Using "Discovery" to Model the Requirements Engineering Process



Quantifying Behavior

Performance requirements are developed through:

- **Quantification of stakeholder requirements,**
- **Allocation: Applying heuristics (rules of thumb), and**
- **Derivation: Analysis using measurements and laws of physics**

Performance requirements answer the following types of questions:

- **When?**
- **How accurate?**
- **How often?**
- **How much?**
- **Until when?**
- **MTBF?**
- **Repair time?**
- **Utilization rate?**
- **Can you think of others?**

Allocation

- Rule of thumb or engineering standards method based on:
  - Hard evidence
  - Informed judgment
  - Reliable historical data
  - “Heuristics”
- Best done by people with expertise in the technical area
- Applicable to familiar territory
- Usually doesn’t scale well or work with new technology

Always document your assumptions and rationale.

Typical Allocation Subjects

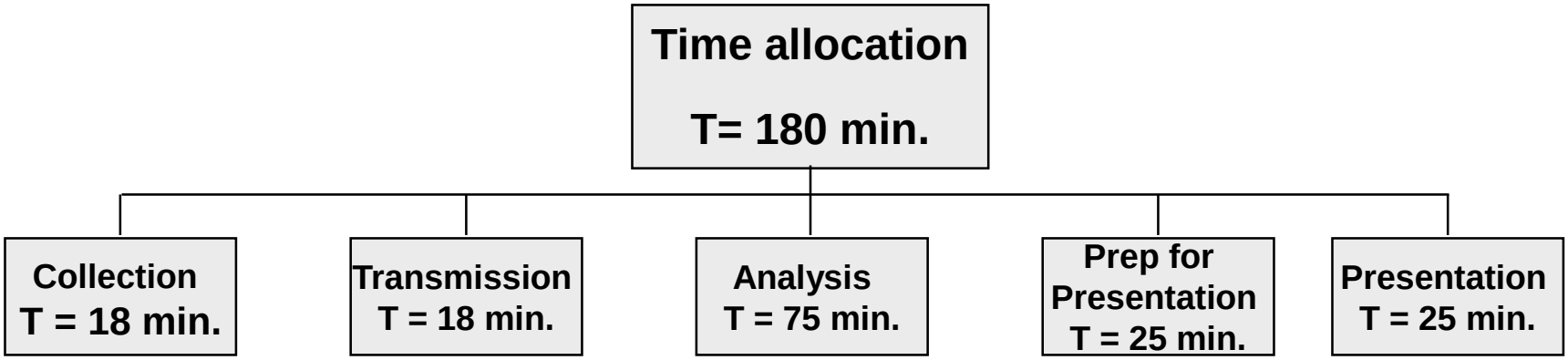
- Size
- Weight
- Power
- Reliability budget
- Antenna gain
- Link margins
- Memory margin
- Processing speed
- Pointing accuracy

- Pressure vessels
- Concrete mix
- Rebar size
- Conductor separation
- Fastener sizes
- Bonding capability
- Qualification margins
- Stopping distance
- Cost budget

Allocation applies when there is sufficient experience in similar environments

Example – Allocation by Heuristics

- An intelligence system shall answer a request for information in 3 hours.
- System experts, based on previous experience and knowledge of technology, allocate time to functions.



Always leave a reserve or contingency.

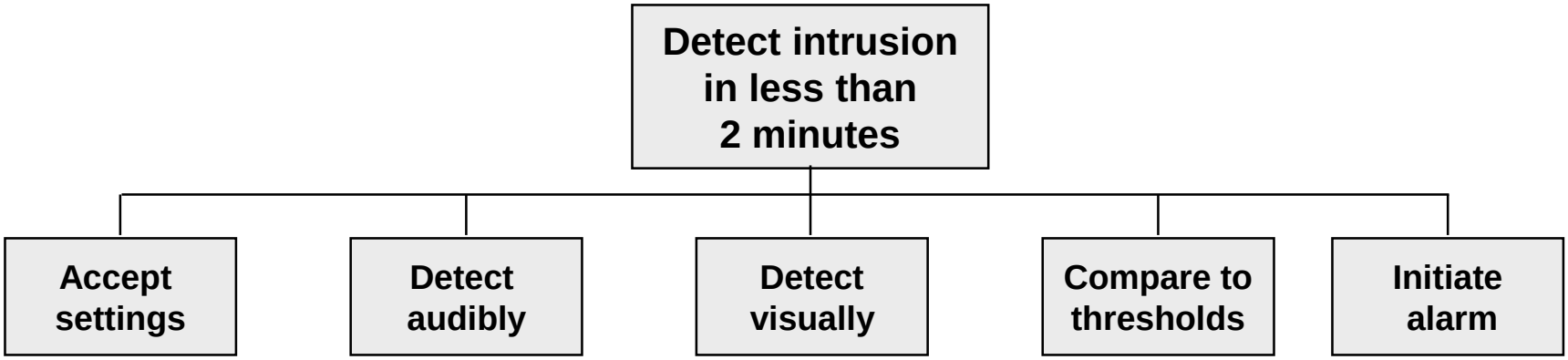
Derivation

- The calculation of required performance
  - Usually based on laws of physics or empirical tests
  - May require modeling of many types and level of robustness
  - Often an “off-core of the Vee” activity
  - Required when allocation via heuristics and experience will not work
- Experimental models and simulations used to:
  - Come to consensus of requirements
  - Prove technical feasibility
  - Investigate promising opportunities
  - Mitigate risk
  - Resolve uncertainties
  - Trade-off performance parameters

These types of studies and analyses can offer significant value

Example – Derivation

- An alarm system must detect intruders in less than 2 minutes after entry.
- When there is insufficient experience or rules of thumb, technical analyses must be done to calculate the performance values for each sub function



- For this example, mathematical analysis would have to be done to determine the performance requirements associated with these sub functions

Derivation: Simple Example

- The operational cost of a new vehicle intended for heavy use is a function of annual maintenance cost, fuel use, and insurance.
- If the following data is assumed:
  - Annual maintenance = $2000.00
  - Fuel economy = 10 miles/gallon
  - Fuel cost = $4.00/gallon
  - Annual Insurance costs are = $2000.00
  - Miles driven per year = 24000
- The annual operational cost, without depreciation, is:

  = $2000.00 +($4.00 x 24000 miles / 10) + $2000

  = $13,600.00



Derivation: Detailed Example (1 of 2)

- A submarine must maintain an acoustic advantage over other submarines. Assuming passive sonar is used, the following equation might be used:

$$DT = SL - DI_s - TL - (NL - DI)$$

- Where*:
  - DT = Detection threshold
  - SL = Source signal level
  - DI_s = Source directivity index (function of source)
  - TL = Transmission Loss (function of range and sea state)
  - NL = Noise level (function of own ship and environment)
  - DI = Directivity Index (function of ownship equipment)

*All variables are in db, and are logarithmic

Can this be a solution-independent analysis?

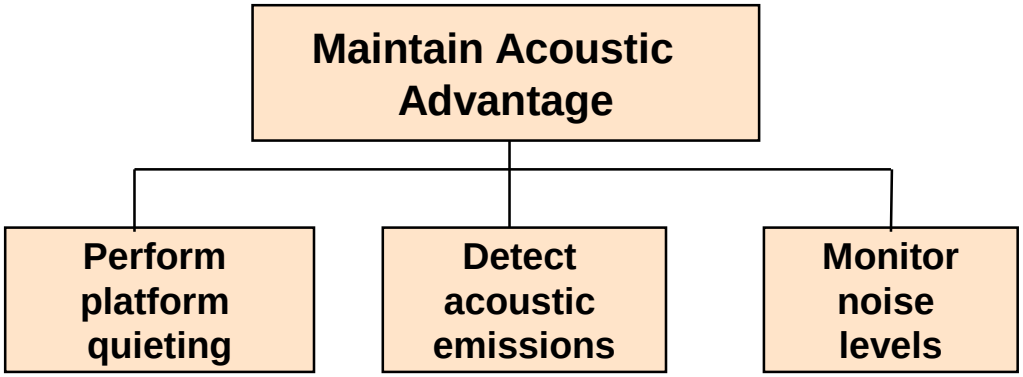
Derivation: Detailed Example (2 of 2)

- Assuming this is a solution-independent calculation, we might want to know the required detection threshold, as a function of own-ship parameters, for specific target characteristics (e.g. range, TL, and DI_s).
- We can use the equation from the last page to compute such values and draw graphs of detection threshold vs. the quantity (NL - DI).
- We can also model the parameters in the equation, and compute DI for various combinations of these parameters.
  - This modeling can be high level, or
  - Can actually compute each parameter as a function of all the variables on which they are dependent.

How does this help us prior to selection of the solution?

Solution - Independent Requirements Tradeoffs

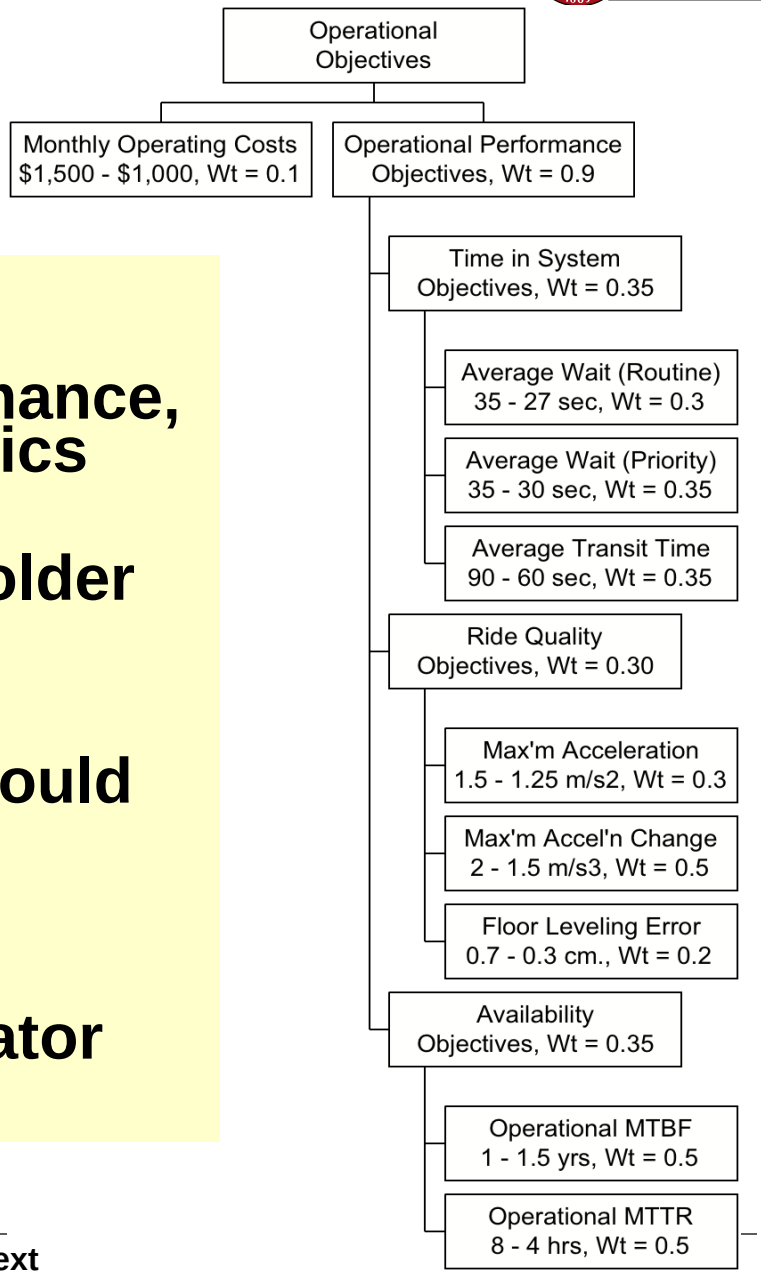
- Consider the submarine acoustic advantage requirement again. Prior to selecting a concept, we might want to perform trade studies between different sets of derived requirements.
- For example, consider the trade off between candidate performance allocations to the three functions shown below.



- Using inputs from our concurrent engineering team, analyses and modeling can be performed to compute the requirement values and allocation of the resulting requirements to the system functions.

This helps us answer the question: What do we emphasize – detection capability or quieter equipment?

Objectives Hierarchy



Definition: Hierarchical representation of major performance, cost, and schedule characteristics

- **Will be major factor in stakeholder satisfaction (validation)**
- **Solution – independent**
- **Performance requirements should trace back to these**
- **Should make up the primary solution selection criteria**
- **Example shown is for an elevator**

Source: Chapter 2 of the Buede text

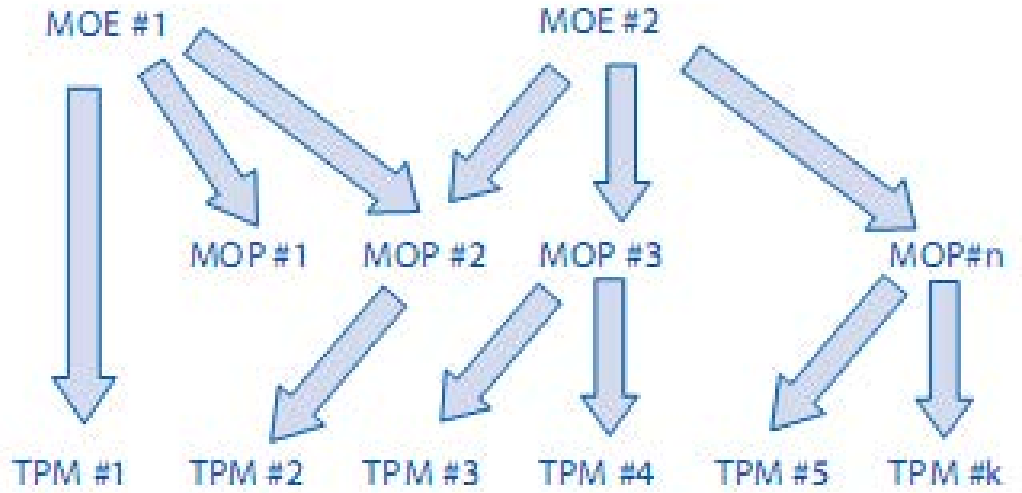
Measures of Effectiveness Reflect What is Most Important to the Customer

- Are operational measures of success closely related to achievement of operational or mission objectives
- provide insight into the accomplishment of mission needs
- Are solution-independent
- Focus on how well (not how) mission or operational objectives are achieved
- Are used to:
  - Define high-level operational requirements from customer viewpoint
  - As standards against which alternative solutions can be compared
  - Measure/project operational success

MOEs are typically supported by a hierarchy of measures:

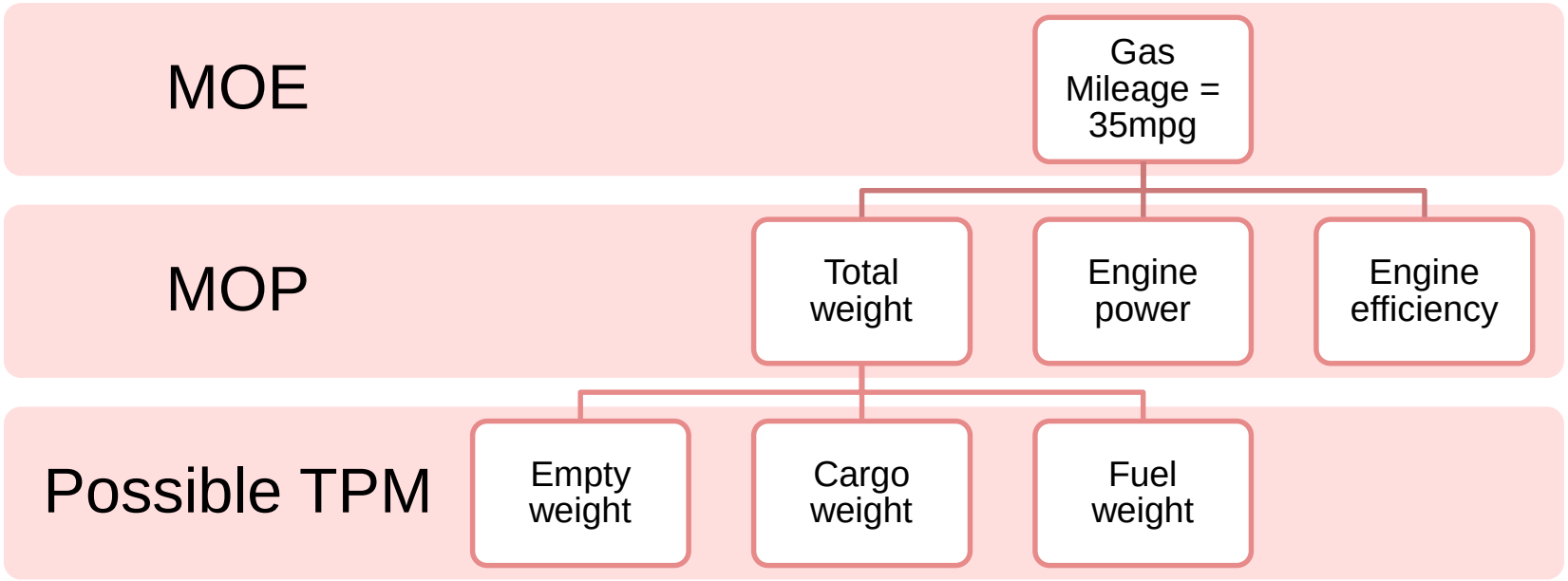
- Key Performance Parameters (KPPs)
- Measures of Performance (MOPs)
- Technical Performance Measures (TPMs)

MOE, MOP & TPM Relationships



MOEs	Derived from stakeholder expectation statements; deemed critical to mission or operational success of the system
MOPs	Broad physical and performance parameters; means of ensuring meeting the associated MOEs
TPMs	Critical mission success or performance attributes; measurable; progress profile established, controlled, and monitored

Technical Measures Hierarchy Example



Requirements Format - Functional/Performance Requirements

- **Subject - the system/sub-system/product/component clearly stated**
- **Verb - shall/shall not**
- **Requirement - the function/constraint**
- **Conditions (if any) - the circumstances in which the requirement functions or constrains**

e.g. The heating system shall maintain a temperature of 70 ± 1 degrees (F), when the outside ambient temperature is at greater than -25 degrees(F)

Subject	Verb	Requirement	Condition(s)
The heating system	shall	maintain a temperature of 70 ± 1 degrees (F)	When the outside ambient temperature is greater than -25 degrees (F)

Alarm System Exercise – Parts 7 & 8: MOEs, MOPs, & Performance Requirements



Instructor will provide directions for the exercise

What are your Questions?



Session 3 – System Requirements

Lessons

1. **Program Overview Update**
2. **Functional Requirements**
3. **Performance and Non-Functional Requirements**
4. **Requirements Management and Traceability**
5. **Special Topics (if time permits)**

- Performance requirements
- What are non-functional requirements
- Discovering non-functional requirements
- Requirements flow down

Appendices*

G. **Alarm System Exercise**

H. **Models and Methods**

I. **Other Appendices TBD**

*Appendices A through F were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

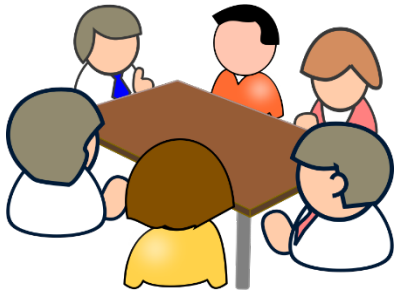
What are Non-Functional Requirements?

- **Any requirement that is not a functional *requirement* . Includes qualities and constraints**
  (Alexander, with italics by Wheeler)
- ***Requirements that are not directly related to the primary capability provided by the system of interest, such as availability, supportability, security, and training***
  (INCOSE SE Handbook, v3.2)
- **The character, or the way the functions behave. The qualities your "product" must have, or how well it does the things it does.**
  (Robertson and Robertson. *Mastering the Requirements Process*, 2nd edition, Addison Wesley, 2005)

In this course, non-functional requirements will be defined as (1) qualities (requirements) imposed by the engineering specialties and (2) other constraints

Discussion: Questions about Non-Functional Requirements

- What are examples of non functional requirements from your experiences?
- What are more important: Functional or non-functional requirements?
- In which category are “performance requirements”?
- When are non-functional requirements defined?
- What do we mean by the “engineering specialties”?



Non-Functional Requirements Considerations

- Evolve from stakeholders expectations
- Are subject to prioritization and trade-offs
- Usually modify, and can sometimes be the source of, functional requirements
- Normally can't be defined independently of the selected concept and architecture
- Require involvement of the stakeholders and the engineering specialists on the team
- Require use of the engineering specialty analytical processes
- Are usually levied across major project entities; Design analysis is required to ensure these are satisfied

Typical Non-Functional System Requirements – Alexander (1/2)

- **Constraints**
  - **Design constraints**
    - **Use of COTS**
    - **Forbidden materials (e.g., toxic)**
    - **Construction (e.g., materials, methods, processes)**
    - **Disposal (e.g. recycling, reprocessing, reuse)**
  - **Regulations and standards**
  - **Human factors, human system integration**
  - **Environmental (only a few examples shown below)**
    - **Temperature range**
    - **Humidity, water-proofing, corrosion resistant**
    - **Vibration, shock-proofing**
    - **Electromagnetic compatibility (allowed emissions and required immunity)**
  - **Physical:**
    - **Size, shape, weight, power consumption, finish, color, labeling**

Adapted from Alexander Fig 6.3

Typical Non-Functional System Requirements – Alexander (2/2)

- **Development Qualities**
  - **Producibility (manufacturability)**
  - **Flexibility (includes upgradability, scalability, modifiability)**
  - **Testability (verifiability)**
- **Usage Qualities**
  - **Dependability (includes availability, reliability, maintainability, safety, security, survivability)**
  - **Usability**
  - **Interoperability**
- **Program Requirements**
  - **Development requirements**
  - **Test requirements (includes approach, test equipment, simulators, scenarios, trials, parallel operations)**
  - **Cost**
  - **Schedule**

Note: Alexander also includes:

"performance (many measures often best handled as attributes of functions)"

Maintainability for bicycle?

"Comfortability" of the airline seat?

Efficiency or Performance (cleanness) of washing machine?

"Thrillingability"of the roller-coaster

Adapted from Alexander Fig 6.3

Non-Functional System Requirements: NASA Viewpoint

- **NASA identifies the following requirements types:**
  - **Functional**
  - **Performance**
  - **Interface**
  - **Environmental:**
    - **Radiation, thermal, acoustic, mechanical loads, contamination, radio frequency, and others**
  - **Reliability**
  - **Producibility**
  - **Maintainability**
  - **Availability**
  - **Upgradability**
  - **Safety**
  - **Human factors**
  - **Others. . . . .**

Lesson 6

Non-Functional

This list adapted from the NASA SE Handbook; Designations as “non-functional“ are by your instructor

Non-Functional System Requirements: Another Viewpoint:

- **Jeffrey O. Grady, "*System requirements Analysis*", Ch 3.7, Academic Press, Burlington, MA, 2006**
  - **Discusses specialty engineering requirements analysis, including generic process, and several specific specialties**
  - **Includes constraints**
  - **Includes the engineering specialties of:**
    - **Reliability**
    - **Parts, materials, and process (PMP)**
    - **Maintainability**
    - **Availability**
    - **Producibility**
    - **Human factors**
    - **System safety**
    - **Design to cost/Life-Cycle Cost**
    - **Electromagnetic capability**
    - **System security**
    - **Mass properties**
    - **Environmental Impact**

Grady emphasizes the need to involve the specialists in the SE process

Non-Functional System Requirements: INCOSE Viewpoint (1/2):

- The INCOSE Systems Engineering Handbook, v3.2, Chapter 9
  - Discusses specialty engineering activities in terms of the analyses required
  - Discusses each of the following analyses

- Acquisition logistics – Integrated logistics support
- Cost effectiveness
- Electromagnetic compatibility
- Environmental impact
- Interoperability
- Life-Cycle Cost
- Manufacturing and producibility
- Mass properties
- Safety and health hazard
- Sustainment
- Training needs
- Usability/human systems integration
- Value engineering

INCOSE also emphasizes involving the specialists in the SE process

What are Non-Functional Requirements - Summary

- Includes many categories of important engineering specialties that must be integrated into the SE process
- Each organization should identify their applicable engineering specialties
- Each project must identify specifically what specialties to consider, and what analyses are required for each
- Except for the most simple projects, experts are required to ensure the system requirements and design adequately consider the engineering specialties

For the purposes of this course, "performance requirements" are the quantification of functional requirements , and are not considered "non-functional requirements". . . . . .

. . . . . But the analyses required for each requirement, and the involvement of the proper experts, are more important than the name we give to each requirement type . . . . . .

. . . . . And identifying and properly defining ALL requirements, regardless of what they are called, is most important

Session 3 – System Requirements

Lessons

1. **Program Overview Update**
2. **Functional Requirements**
3. **Performance and Non-Functional Requirements**
4. **Requirements Management and Traceability**
5. **Special Topics (if time permits)**

- Performance requirements
- What are non-functional requirements
- Discovering non-functional requirements
- Requirements flow down



Appendices*

G. **Alarm System Exercise**

H. **Models and Methods**

I. **Other Appendices TBD**

*Appendices A through F were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

Discovering Non-Functional Requirements

- Analyze stakeholder expectations (goals) *
- Revisit your stakeholder analysis
  - Many may be NFRs
  - Quantify them
  - Are any MOEs identified?
- Use checklists *
- Use standards

* Discussed in this section

Using Goals to Discover Non-Functional Requirements (1/3)

- Alexander defines a "goal" as something a stakeholder wants, but that it may not be expressed in measureable (and hence testable) form.
- Results from a goals workshop can be useful to understand non-functional requirements
- The next two slides show:
  - Top-level results from a goals workshop for a multi-function device to support leisure activities (one of the ongoing problems covered in the Alexander text)
  - A decomposition of one of the goals (dependability)
- As the higher level goals are decomposed, two general rules can be observed:
  - Quality goals (i.e., non-functional requirements) often break down into functional requirements:
    - Dependability => longer operation without external power
    - Security => virus detection
  - Decomposing higher level goals yields more specific requirements

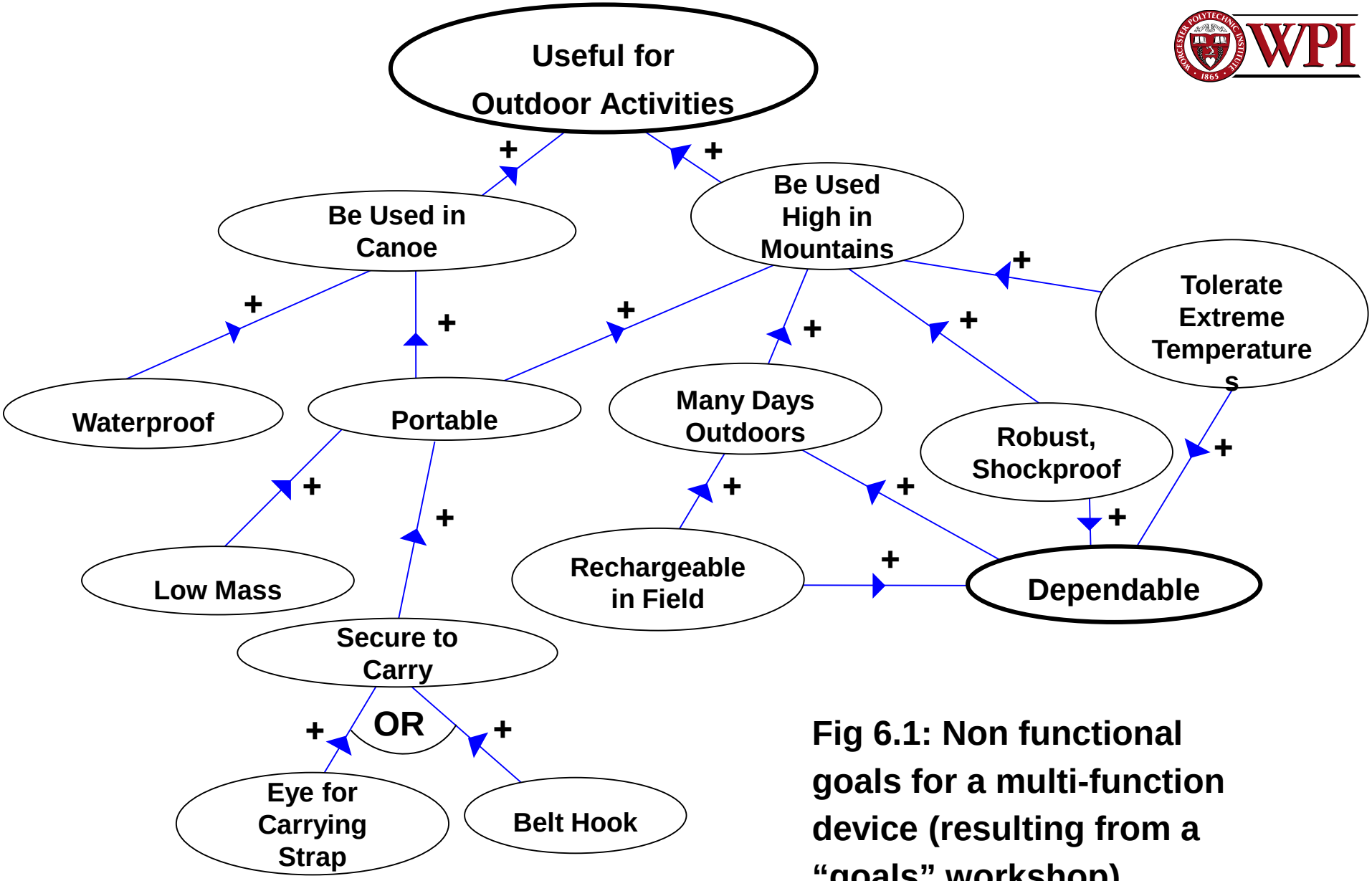
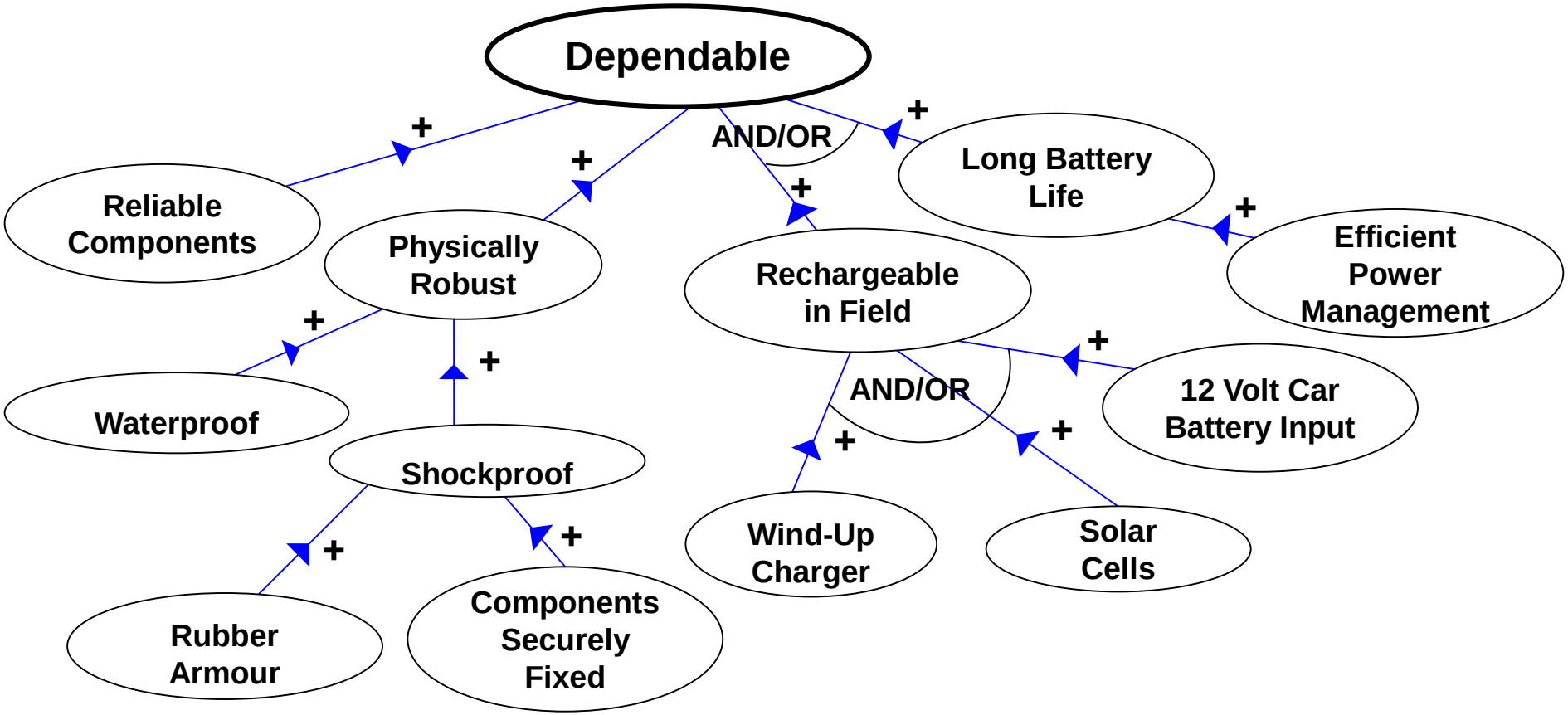


Fig 6.1: Non functional goals for a multi-function device (resulting from a "goals" workshop)

Source: Alexander Fig 6-1

Using Goals to Discover Non-Functional Requirements (3/3)



Are these all solution-independent?

Could they be written as solution-independent?

Can non-functional requirements be defined ?

Source: Alexander Fig 6-2

Using Checklists to Discover Non-Functional Requirements

- Projects are all unique
- Robust checklists are useful to identify non-functional requirements that *might* be applicable to your project
- See the slides earlier in this lesson for examples of checklists
- See http://www.scenarioplus.org.uk for a more detailed checklist
- See Alexander Fig 6.4 for software projects

Alarm System Exercise – Part 9: Non-Functional Requirements



Instructor will provide directions for the exercise

Session 3 – System Requirements

Lessons

1. **Program Overview Update**
2. **Functional Requirements**
3. **Performance and Non-Functional Requirements**
4. **Requirements Management and Traceability**
5. **Special Topics (if time permits)**

- Performance requirements
- What are non-functional requirements
- Discovering non-functional requirements
- Requirements flow down



Appendices*

G. **Alarm System Exercise**

H. **Models and Methods**

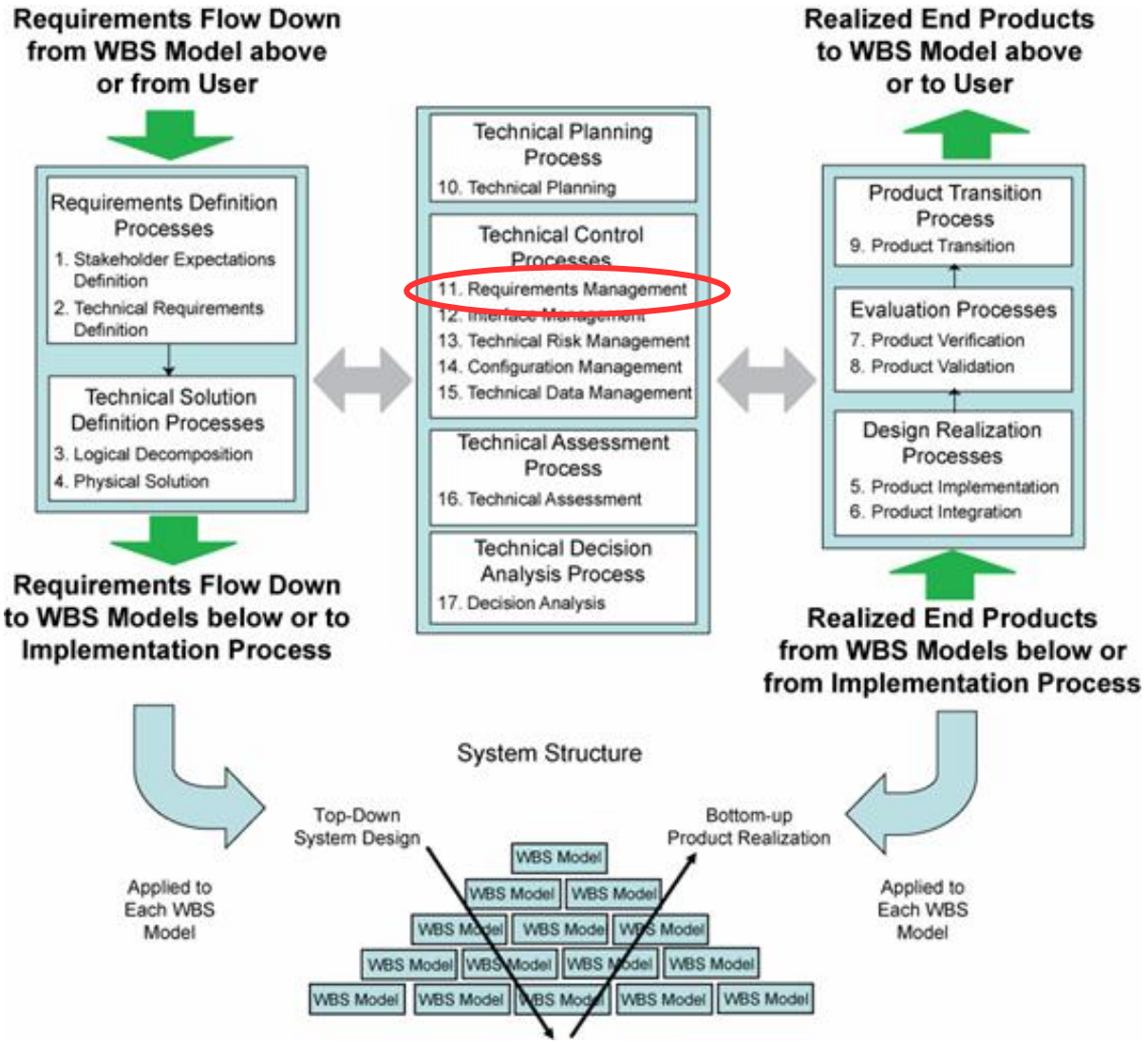
I. **Other Appendices TBD**

*Appendices A through F were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

Lesson Context

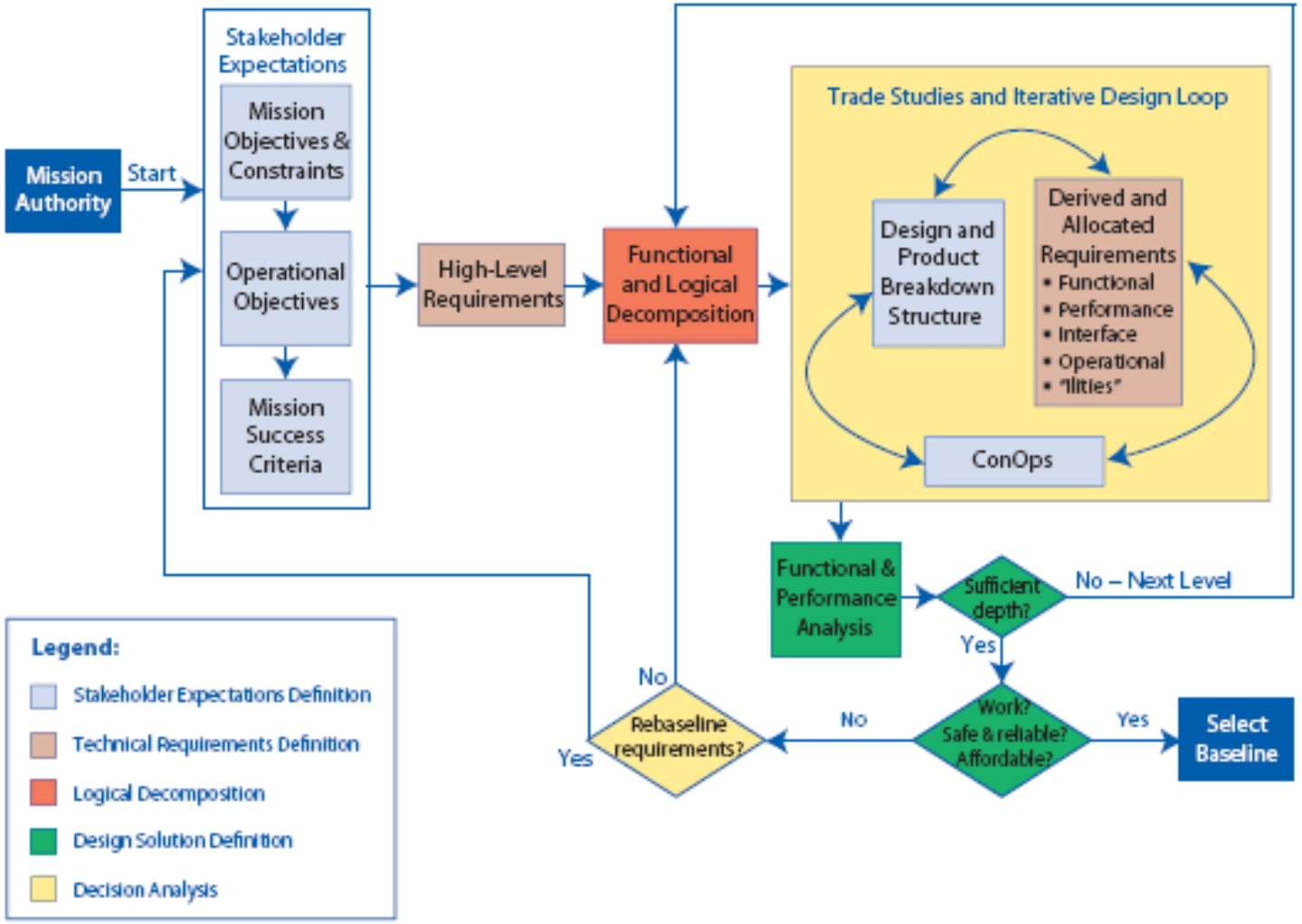
- So far this course has focused on the solution-independent requirements
- The concept has been that if we understand how to define requirements, then we can define them at any level, and we can define solution-dependent requirements also.
- We just discussed decomposition of functional requirements, allocation of performance requirements to functions, and development of non-functional requirements
- Now we consider how we can flow down requirements through the physical entities of the solution, and how we can record and maintain traceability
- Lesson #4 looks at Requirements Management; how we can manage the requirements through the system life cycle

NASA Requirements Management Process within their "SE Engine" Model



NASA Procedural Requirements for Systems Engineering (NPR) 7123.1A

NASA SE Handbook: Relationship among the System Design Processes



NASA Systems Engineering handbook, Revision 1, 2007

Requirements Flow down and Traceability (1 of 3)

- **Use functional and performance analysis techniques to:**
  - **Develop a functional architecture (concept-independent)**
  - **Allocate performance requirements to functions**
- **Allocate all system requirements to system entities including hardware, software, manual operations, interfaces, services, and all others**
- **Ensure all functional and performance requirements and design constraints (derived or directly allocated) are allocated to a system architecture entity**
- **Ensure bi-directional traceability is recorded between all "parent" and "child" requirements across all system hierarchical levels**
- **Ensure the history of each requirement is maintained in a traceability vehicle, preferable a database**

Requirements Flow down and Traceability (2 of 3)

- **Trade-off analyses will be required to make flow-down decisions:**
  - **For allocation to functional architecture**
  - **For allocation to solution entities**
  - **Similar to the trade offs to be discussed in Session 4**
- **Techniques include:**
  - **Heuristics/rules of thumb/best practices**
  - **SME involvement and processes**
  - **Mathematical and other analyses**
  - **Models/simulations/prototypes**
- **Rationale for these decisions must be:**
  - **Developed and recorded**
  - **Validated with appropriate stakeholders**
  - **Maintained, archived, and traced to the supported decisions**

Requirements Flow down and Traceability (3 of 3)

Use a traceability database tool to generate:

- **Requirements statements with Project Unique Identifier (PUID)**
- **Requirements traceability matrices showing all requirements and their traces**
- **Requirements traceability matrices showing requirements and their associated validation and verification requirements**
- **Lists of TBDs, TBRs, and TBSs**
- **Specification trees**
- **Requirements metrics**
- **Include the rationale for all requirements decisions**

Use the traceability database to ensure requirements accountability – keeping track of the satisfaction of every requirement

	X			X
		X		
	X		X	

NASA SE Handbook: Requirements Flow Down

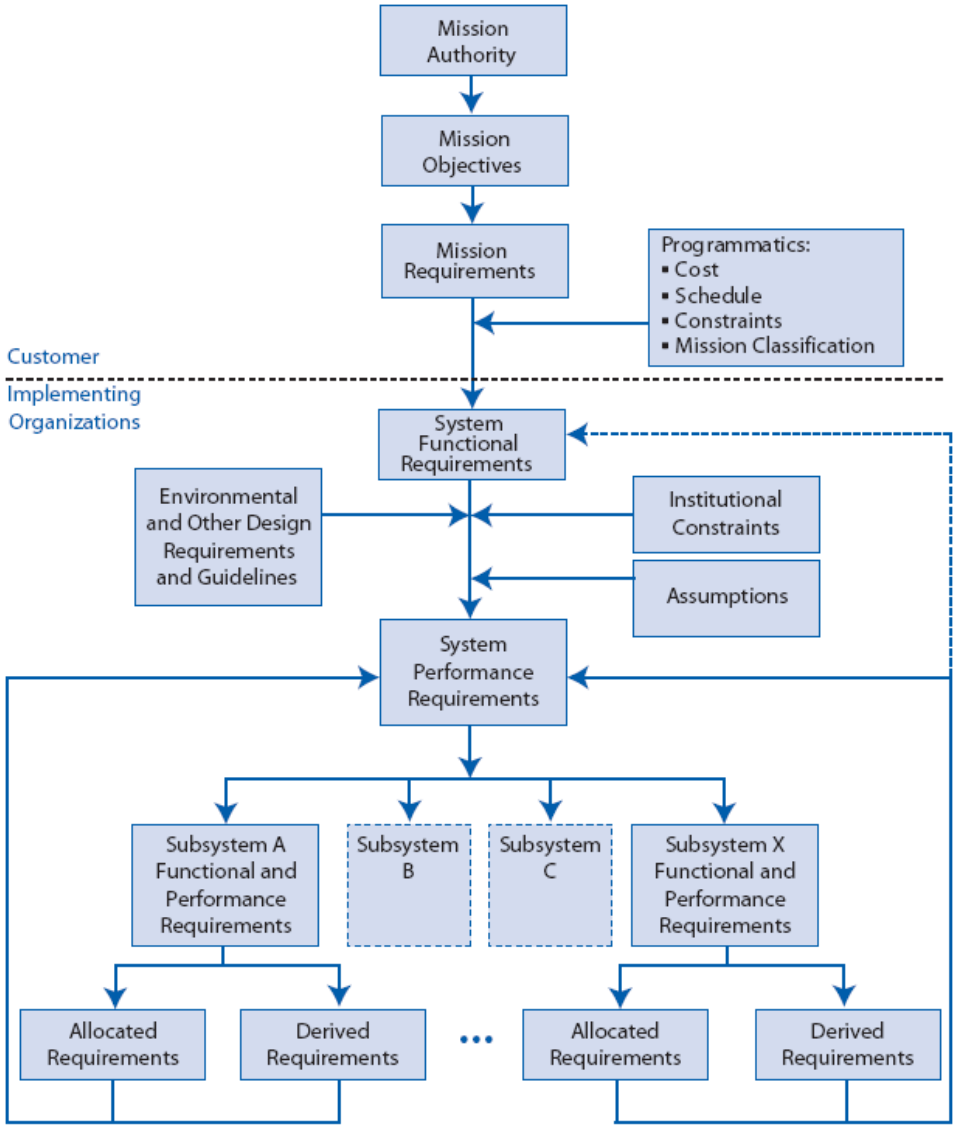


Figure 4.2-3 The flowdown of requirements

NASA Systems Engineering handbook, Revision 1, 2007

NASA SE Handbook: Technical Baseline Evolution

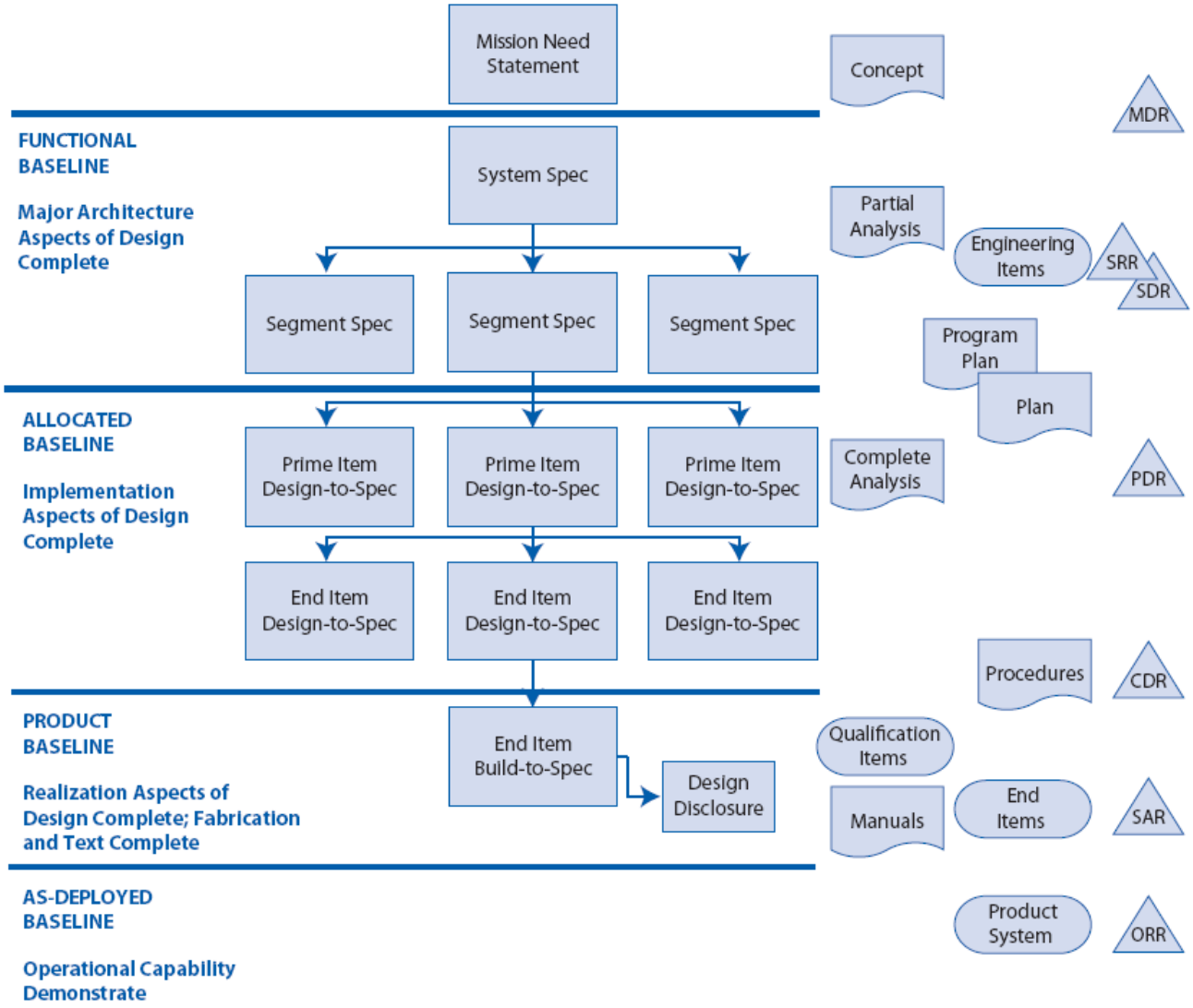


Figure 6.5-3 Evolution of technical baseline

NASA SE Handbook: PBS Example

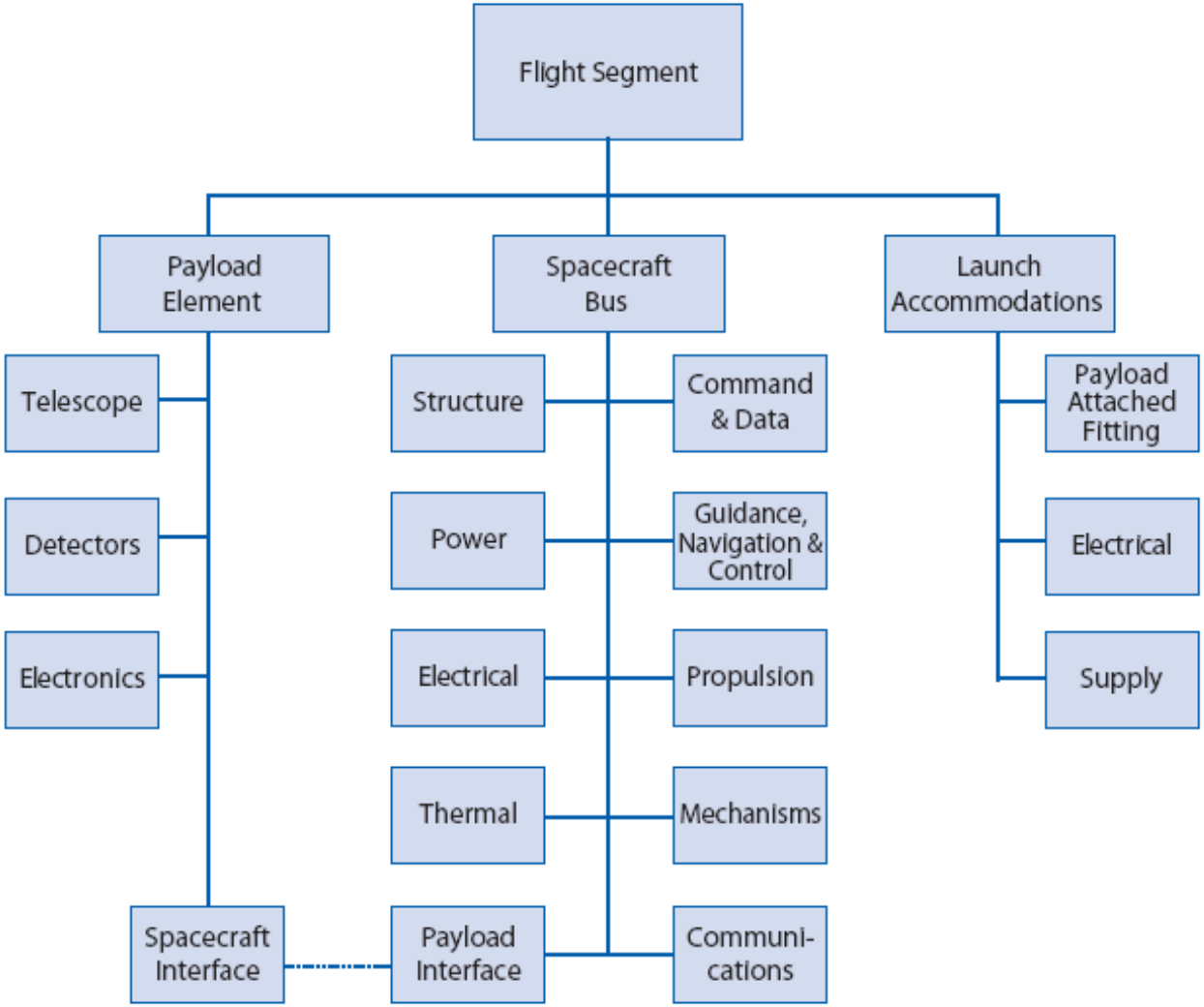
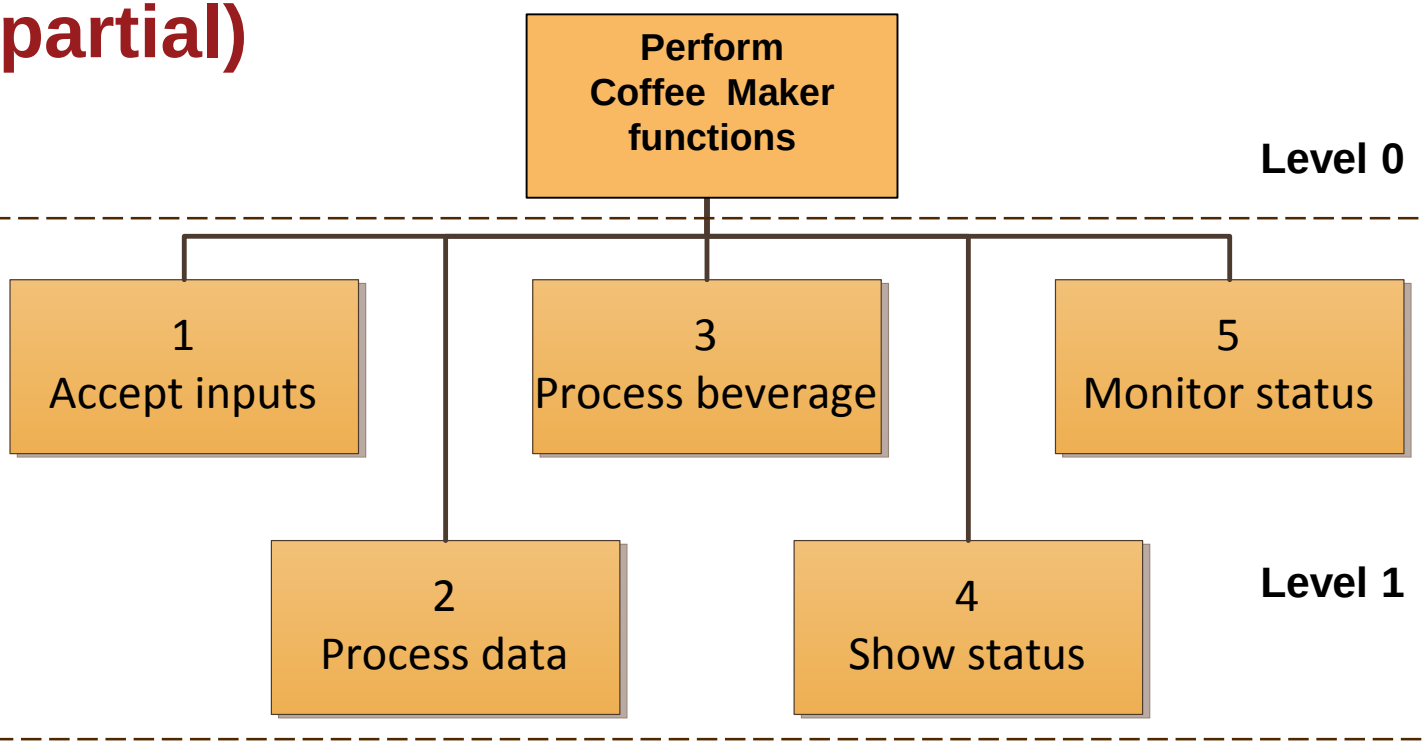


Figure 4.3-2 Example of a PBS

Allocation Example: Performance to Function (partial)

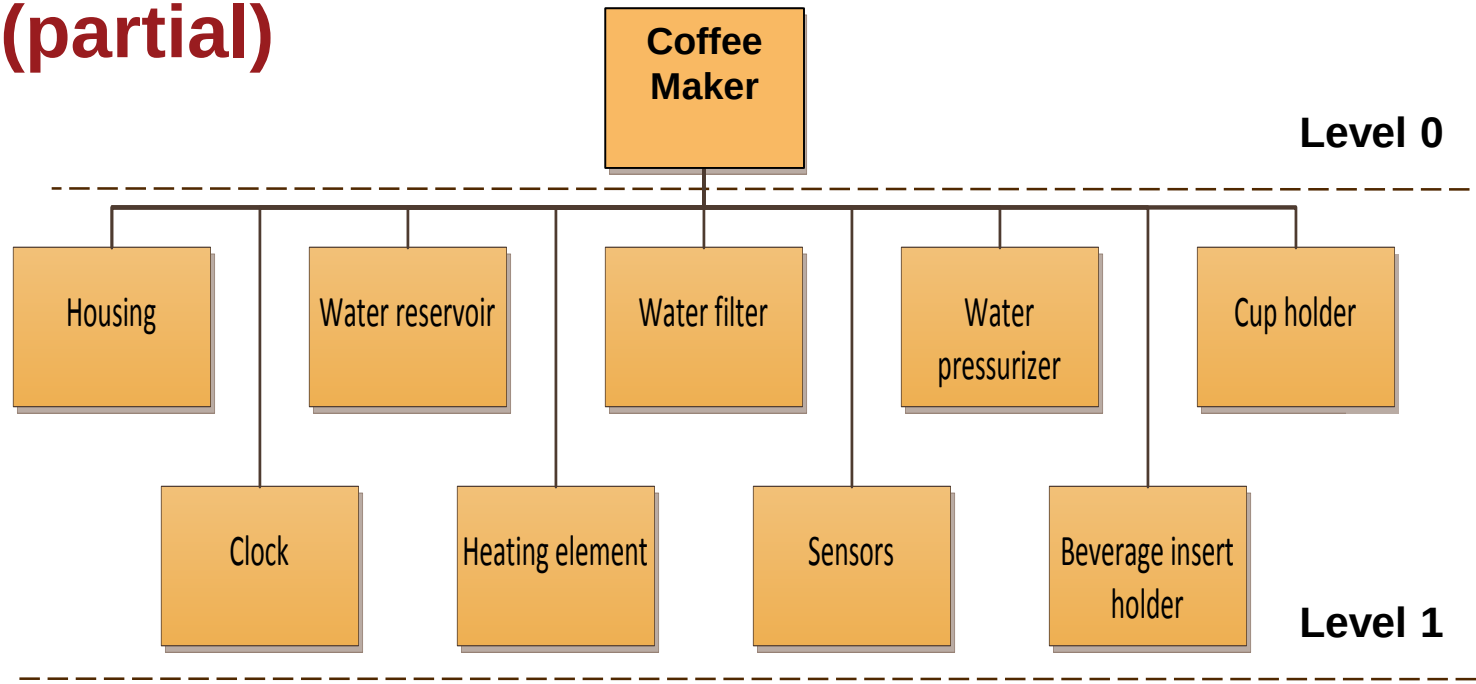
- **Do this only if it informs decisions**
- **Might not be meaningful at level 1**
- **Analysis might be required**
- **Optioneering might be required**
- **Saying "allocate" is not enough; the actual allocated requirement is needed, with rationale**
- **Allocate might require derivation**



Perf Req	1	2	3	4	5
Brew in 1 min	allocate	allocate	allocate		
Last 3 years	←==========All must meet this==========→				
Abort if exception	allocate	allocate	allocate		allocate
Universal power*	allocate				
Less than 10 #	←====== If needed, could allocate or derive=========→				
Serve at set temp		allocate	allocate		allocate

Allocation Example: Performance to Physical (partial)

- **Analysis might be required**
- **Could be by allocation or derivation**
- **"X" is not enough; the actual allocated requirement is needed, with rationale**
- **You might discover that PBS is not right or not complete**



Perf Req	1	2	3	4	5	6	7	8	9
Brew in 1 min		x		x		x		x	x
Last 3 years	x	x	x	x	x	x	x	x	X
Abort if exception		x							
Universal power*	x								
Less than 10 #	x	x	x	x	x	x	x	x	x
Serve at set temp			x	x		x			

NASA SE Handbook: Requirements Allocation Sheet

ID	DESCRIPTION	REQUIREMENT	TRACED FROM	PERFORMANCE	MARGIN	COMMENTS	REF
M2	Launch Vehicle	Pegasus XL with HAPS	P2, P4	Complies	NA		F.2.c
M3	Observatory Mass	The observatory total mass shall not exceed 241 kg	M1, M2	192.5 kg	25.20%		F.5.b
M4	Data Acquisition Quality	The mission shall deliver 95% data with better than 1 in 100,000 BER	P1	Complies	NA	Standard margins and systems baselined; formal system analysis to be completed by PDR	F.7
M5	Communication Band	The mission shall use S-band SQPSK at 5 Mbps for spacecraft downlink and 2 kbps uplink	S12, P4	Complies	NA	See SC27, SC28, and G1, G2	F.3.f, F.7
M7	Tracking	MOC shall use NORAD two-line elements for observatory tracking	P4	Complies	NA		F.7
M8	Data Latency	Data latency shall be less than 72 hours	P12	Complies	NA		F.7
M9	Daily Data Volume	Accommodate average daily raw science data volume of 10.8 Gbits	P1, S12	Complies	12%	Margin based on funded ground contacts	F.3.e, F.7
M10	Ground Station	The mission shall be compatible with the Rutherford Appleton Laboratory Ground Station and the Poker Flat Ground Station	P1	Complies	NA		F.7
M11	Orbital Debris (Casualty Area)	Design observatory for demise upon reentry with <1/10,000 probability of injury	P3	1/51,000	400%	See Orbital Debris Analysis in Appendix M-6	F.2.e, App.6
M12	Orbital Debris (Lifetime)	Design observatory for reentry <25 years after end of mission	P3	<10 years	15 years	See Orbital Debris Analysis in Appendix M-6	F.2.e, App.6

Figure F- 5 Requirements allocation sheet

Alarm System Exercise – Part 10: Requirements Allocation and Flow Down

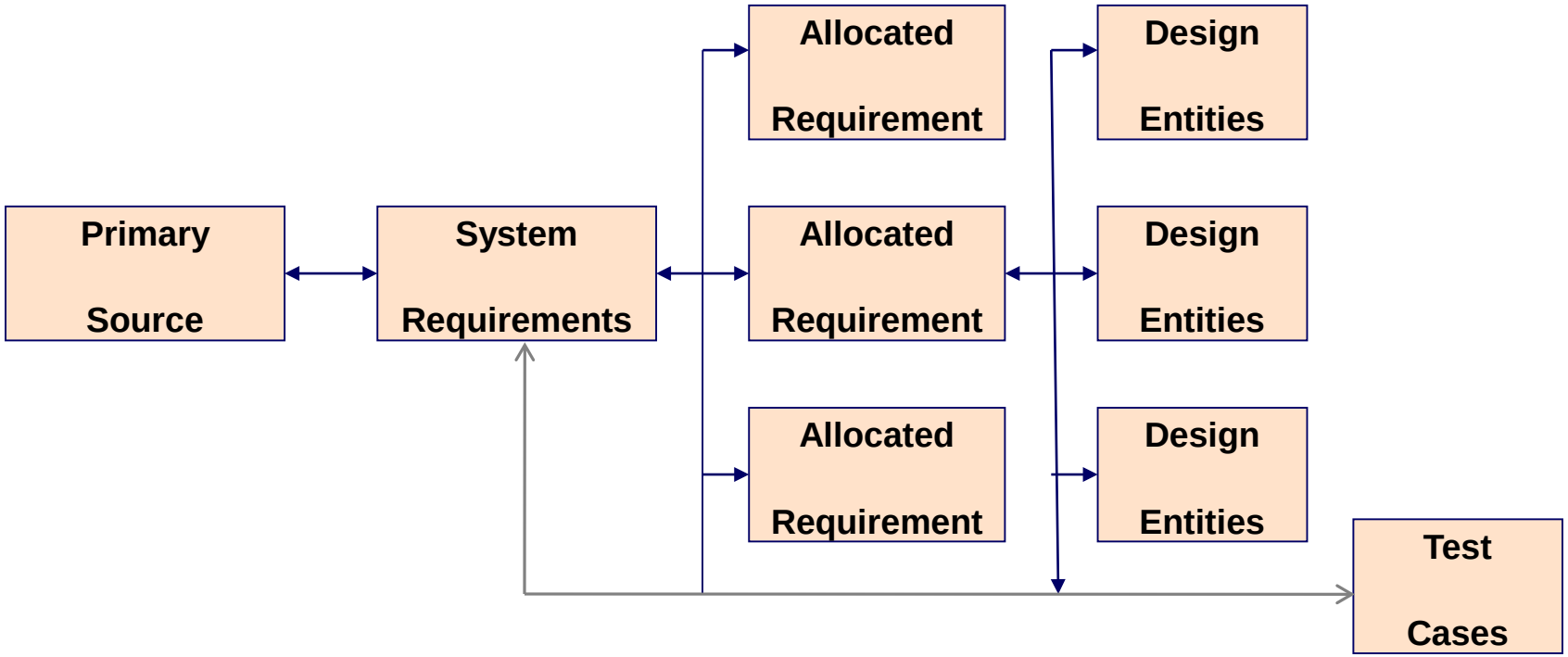
- If you have not already done so, create a level one PBS for the alarm system
- Pick a set of your top level requirements, and show how you would allocate those requirements To the level 1 system entities
- Discuss how you would demonstrate (illustrate) traceability of the allocated requirements back up to their parents
- Prepare to share with the class a brief summary of your results and your insights from the exercise. Include how you would decide what the lower level solution dependent requirements would be



* How can you be sure you have addressed ALL solution entities with the 7±2 heuristic?

Requirements Traceability

The following figure depicts the bidirectional traceability of requirements



Requirements Traceability Matrix – Generic Example

Primary Source	System Specification	Software Requirements	Design Document	Test Cases	Test Report
CONOPS Pg 13, Pa 4	3.4.5	3.2.1	5.3.1	2.2.1	2.2.1
		3.2.3	5.3.4	2.2.5	2.2.5
Marketing Requirements Document Pg 6, Pa 2	3.4.6	3.2.7	5.6.2	2.4.1	2.4.1
		3.4.1			
Telcon Minutes 2/15/2015	3.5.4	3.3.1	5.4.1	2.3.2	2.3.2
		3.2.2	5.5.1	2.5.2	2.5.2
Memo 1/15/2015	3.6.4	3.7.2	5.5.2	2.4.5	2.4.5

The table depicts a Requirements Traceability Matrix (RTM); Each referenced artifact should be identified with document number, revision, and date

Some Thoughts on Traceability

- Called by many names
  - Requirements Traceability Matrix (RTM)
  - When combined with V&V planning: RTVM, RVTM, etc.
- Highly valuable SE Tool:
  - Managing requirements through life cycle
  - Planning and tracking V&V
- Traceability should be bi-directional
  - Trace information allows parent-to-child or child-to-parent tracing
- Looks like a spreadsheet, but should be implemented via a database
  - Example: DOORS

The course text (Visualizing Project Management) contains a nice example of a traceability matrix

Lesson 4 further discusses traceability, including how to manage requirements using traceability

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		

All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?



Systems Engineering Technical Short Course

(Session 3: System Requirements)

Lesson 5:
Special Topics
(as time permits)

Session 3 – System Requirements

Lessons

1. **Program Overview Update**
2. **Functional Requirements**
3. **Performance and Non-Functional Requirements**
4. **Requirements Management and Traceability**
5. **Special Topics (if time permits)**

- Agile methods
- "House of Quality"
- Tools

Appendices*

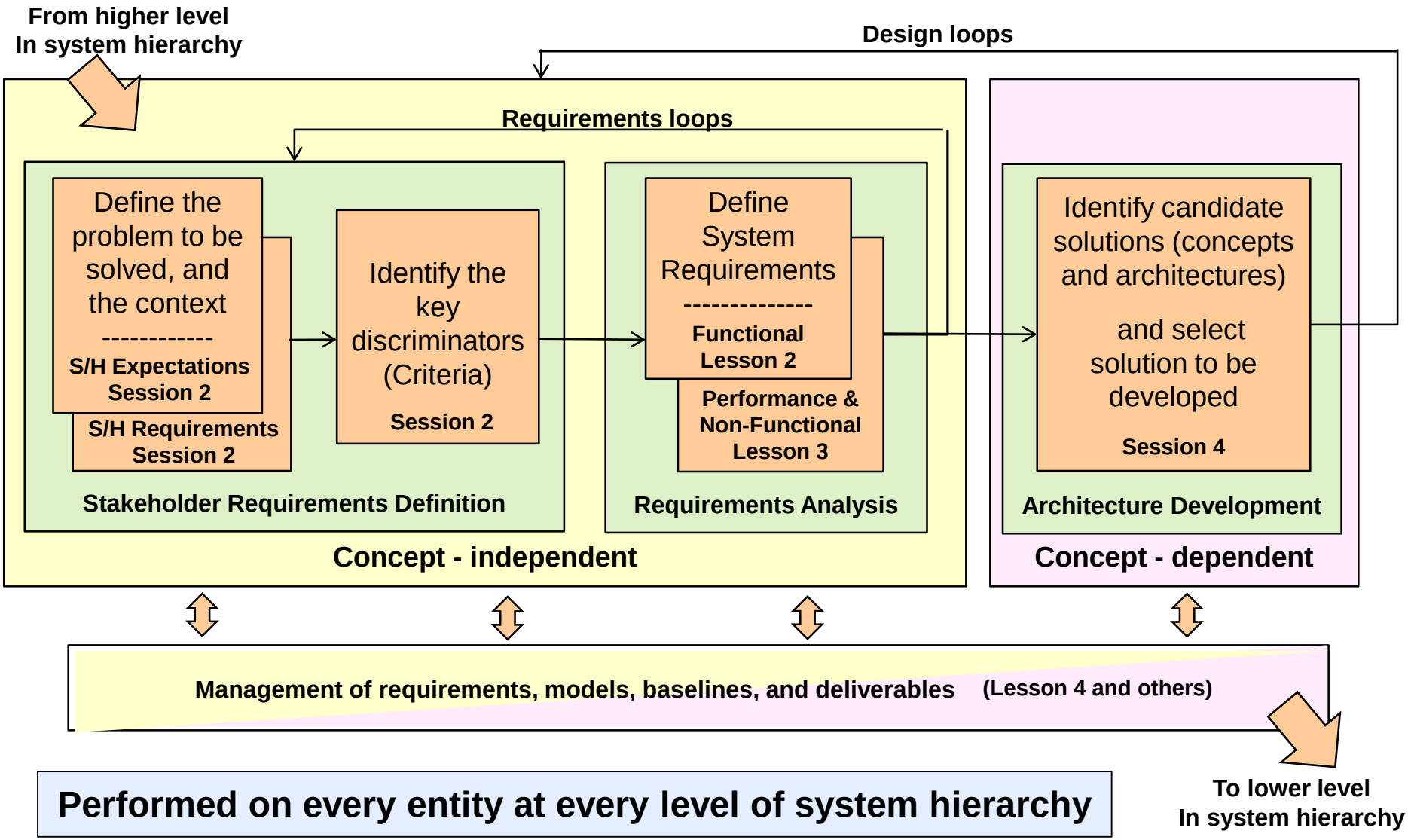
G. **Alarm System Exercise**

H. **Models and Methods**

I. **Other Appendices TBD**

*Appendices A through F were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

The Requirements Engineering Process Context for this Course (Left Side of Vee)



Performed on every entity at every level of system hierarchy

Using "Discovery" to Model the Requirements Engineering Process

Most of these techniques apply to agile...

Discovery Contexts \ *Requirement Elements*	2. Stakeholders	3. Goals	4. Context	5. Scenarios	6. Qualities and Constraints	7. Rationale	8. Definitions	9. Measurements	10. Priorities
1. Vision									
11. From Individuals									
12. From Groups									
13. From Things									
14. Trade-Offs									

Agility in Development

- **Agile - Characterized by quickness, lightness, and ease of movement; nimble**
- **Agile Systems Engineering – quick, nimble response to changes and development of solutions to achieve maximum velocity to delivery**
- **Sometimes called "QRC – Quick Reaction Capability"**
- **Used where it makes sense**

Remember the questions:
"Is the benefit of tailoring worth the associated risk?", and
"Is the benefit of the traditional process worth the cost"

Characteristics of Agile Development

- Respect for process
- Focus on customer satisfaction
- Customer involvement
- Co-located, dedicated teams
- “Make a promise, keep a promise” discipline
- The documentation is the code
- Results and responsibility for solving issues is owned by the team
- Change is ok
- Technical excellence
- Short delivery cycles
- “Done” means “done”

Agile development is in-process validation in action

The Agile Alliance

- **The Agile Alliance is a nonprofit organization with global membership, committed to advancing Agile development principles and practices. Agile Alliance supports those who explore and apply Agile principles and practices in order to make the software profession more productive, humane and sustainable. We share our passion to deliver software better everyday.(www.agilealliance.com).**

Agile ≠ Ad hoc



Extreme Programing (XP)

Extreme Programming is a discipline of software development based on values of simplicity, communication, feedback, courage, and respect. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation.

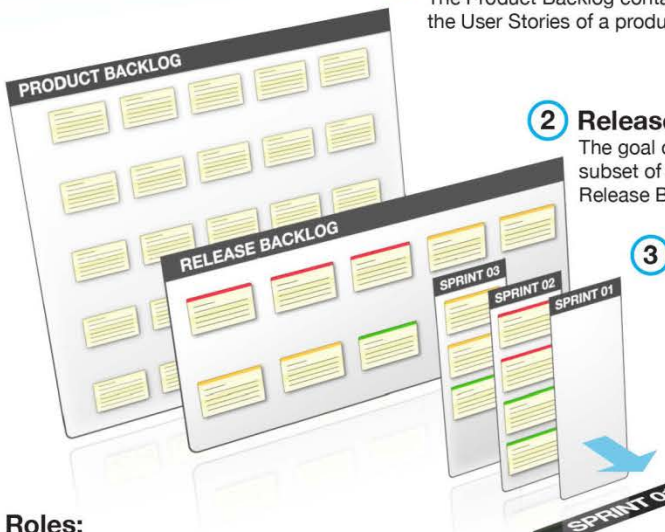
http://www.xprogramming.com

axosoft

Scrum Diagram:

1 Product Backlog

The Product Backlog contains a wish list of all the User Stories of a product.



2 Release Backlog

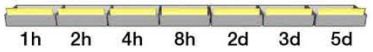
The goal of a given release is to deliver a subset of the Product Backlog, known as the Release Backlog.

3 Sprint Backlogs

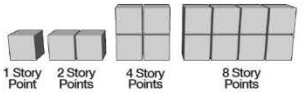
Each Sprint (or short duration milestone) takes a small chunk of the Release Backlog and gets it Ship-Ready!

Estimation Techniques:

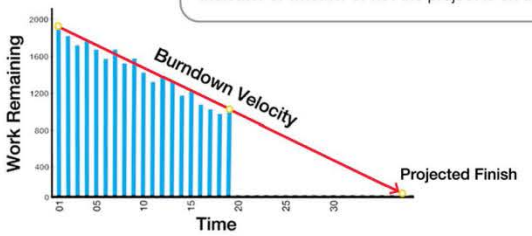
Hours: When estimating work, it's important to have the entire team use some standards. Use 1h, 2h, 4h, 8h, etc. No estimations in-between.



Story Points: You can also estimate work in comparison to the complexity of a well known but simple component.



Remember that in Scrum, the estimates are only part of the story. The Burndown Velocity is the true indicator of whether or not the project is on track.



4 Burndown Chart

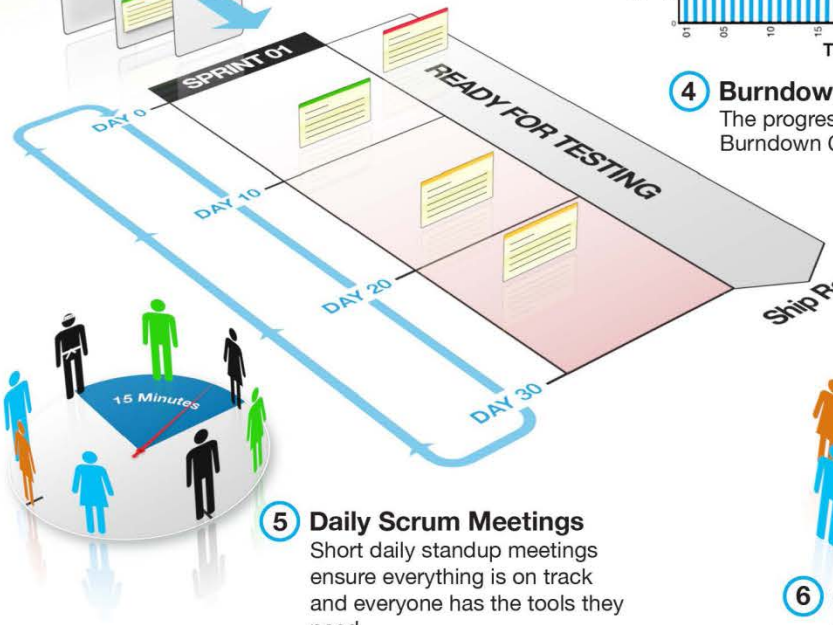
The progress of the team is monitored using a Burndown Chart.

Team Roles:

Product Owner: Is responsible for what goes into the product backlog and prioritizes it. Would probably make a good dictator if given the chance.

Scrum Master: A team facilitator. Ensures teams have what they need to get the job done. Also, sets up meetings and monitors everything. Also, kicks ass when necessary.

Developers & Testers: They write code and make sure it does what it's suppose to do. Duh!



5 Daily Scrum Meetings

Short daily standup meetings ensure everything is on track and everyone has the tools they need.

6 Sprint Retrospective

After each sprint, a longer retrospective meeting helps fine-tune the process.

For more info:

Intro to Scrum Video: www.axosoft.com/scrumvideo

This work is licensed under a Creative Commons Attribution-NoDerivs 3.0 Unported License. © 2012 Axosoft, LLC.

http://www.ontimenow.com/downloads/axosoft-scrum-diagram.pdf

Additional Sources

- Agile Manifesto - http://agilemanifesto.org/
- The Agile Alliance - http://www.agilealliance.org
- Scrum Info - http://www.controlchaos.com/
- XP info - http://www.xprogramming.com

Agile Development Summary

- Project teams must be intentional about the development processes they use
- Some project environments lend themselves to Agile methods
- Whatever process is used, it must be selected, understood, tailored, properly implemented, and measured/continuously improved.
- Teams must be trained in the methodology selected
- Not everybody is comfortable with Agile Methods
- The process activities covered in this course are performed in agile methods; Only the order, and manner in which they are performed differs

Remember

Agile ≠ Ad hoc

Session 3 – System Requirements

Lessons

1. **Program Overview Update**
2. **Functional Requirements**
3. **Performance and Non-Functional Requirements**
4. **Requirements Management and Traceability**
5. **Special Topics (if time permits)**

- Agile methods
- "House of Quality"
- Tools



Appendices*

- G. **Alarm System Exercise**
- H. **Methods and Models**
- I. **Other Appendices TBD**

*Appendices A through F were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

Quality Function Deployment (QFD)

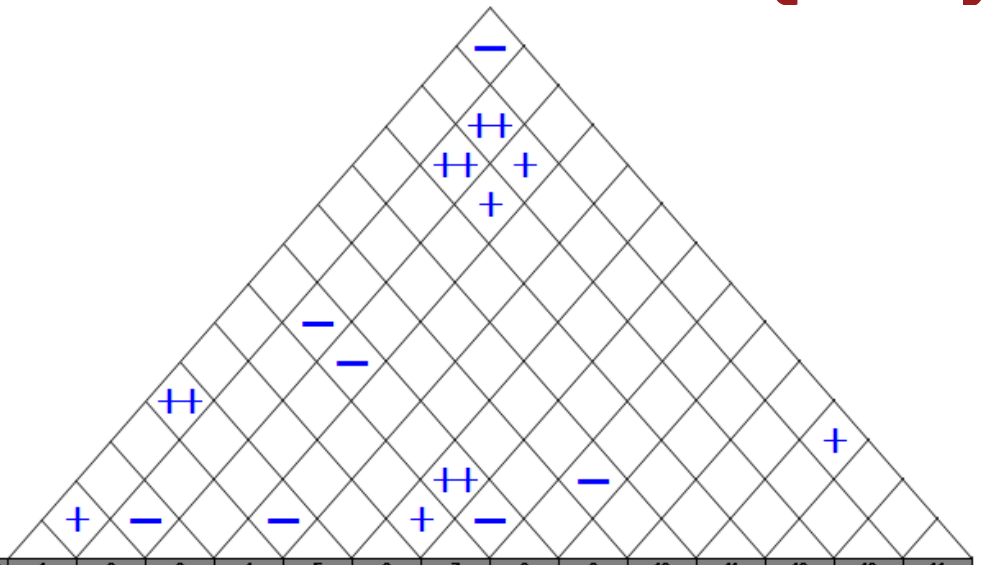
- A trade-off method initially devised by Yoji Akao to relate design elements to stakeholder requirements
- Quality Function Deployment: Integrating Customer Requirements into Product Design (ISBN: 0-915299-41-0) and QFD: The Customer-Driven Approach to Quality Planning & Deployment (ISBN 9283311221).
- Published in USA by John R. Hauser & Don Clausing (1988) The house of quality. *Harvard Business Review*, May–June, 63-73
- QFD compares alternative design features, relates them to requirements, and helps select the best requirements set

http://en.wikipedia.org/wiki/Quality_function_deployment

QFD Highlights

- QFD often referred to (perhaps incorrectly) as "House of Quality"
- A good example of what Alexander calls "optioneering"
- QFD compares alternative design features, relates them to requirements, and helps select the best requirements set
- Can be augmented with other powerful techniques

QFD Matrix with "House of Quality"



Column #	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Quality Characteristics / Demanded Quality (a.k.a. "Customer Requirements" or "Whats")	GPS navigation	Wind-up Charger	Solar Power	Barometric Altimeter	GPS Altimeter	Music Player	Solid State Memory	Hard Disk Drive	Magnetic Compass	Fire-lighter	Polycarbonate Casing	Lamp	Radio	Distress Beacon
Useful for Outdoor Pursuits	⊙			⊙	⊙	▲			⊙	○		⊙	▲	○
Light to carry							⊙				⊙			
Long battery lifetime		⊙	⊙				⊙		⊙					
Low Cost								⊙	⊙			⊙	○	
Reliable					○		⊙		⊙		⊙	○	○	
Waterproof							⊙				⊙			
Robust							⊙		⊙		⊙			
Fashionable	⊙	⊙	○		⊙	⊙	⊙			○	⊙		▲	⊙

- Maps how concepts and architectures satisfy prioritized requirements
- Illustrates the qualitative value of the requirements satisfaction
- Highlights areas where solution features are most effective and counter effective

Alexander Figure 14.3

Session 3 – System Requirements

Lessons

1. **Program Overview Update**
2. **Functional Requirements**
3. **Performance and Non-Functional Requirements**
4. **Requirements Management and Traceability**
5. **Special Topics (if time permits)**

- Agile methods
- "House of Quality"
- A few thoughts on Tools

Appendices*

G. **Alarm System Exercise**

H. **Models and Methods**

I. **Other Appendices TBD**

*Appendices A through F were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

Tools for Requirements Engineering (1/2)

- Many tools exist to aide in performing requirements engineering
- INCOSE maintains a good tool database, searchable by category, that is maintained by the Tools Database Working Group (TDWG). You can find it on the INCOSE web site at www.incose.org
- While many sophisticated tools exist, experience indicates that these tools:
  - have ben slow to be adopted
  - sometimes try to drive the process rather than vice-versa
  - are beginning to be more accepted

Tools for Requirements Engineering (2/2)

- Tools have many benefits, but they are not free
- Selection of tools should be based on a solid business case that provides adequate return on investment
- We must recognize that new tools require a learning curve before they achieve their full potential
- The entire team needs to be trained in the use and benefits and risks associated with any tools that are used on a project

Optional Exercise: *Identifying tools used by students for Requirements Engineering*

Instructions: Meet in small groups and make a list of all tools that you use or have used for requirements engineering. Include all tools, whether they are automated or not.

Indicate for what purpose you used each tool



What is your perception of the value of the tool, and the ease of use? Use the format shown to the right

Be prepared to participate in a class discussion of the results

Value to Project	Ease of Use		
	Easy	Moderate	Difficult
High			
Medium			
Low			

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		

All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?



Systems Engineering Technical Short Course

(Session 3: System Requirements)

Lesson 4:
Requirements Management

Session 3 – System Requirements

Lessons

1. **Program Overview Update**
2. **Functional Requirements**
3. **Performance and Non-Functional Requirements**
4. **Requirements Management and Traceability**
5. **Special Topics (if time permits)**

- Requirements management
- Traceability
- Baseline management

Appendices*

G. **Alarm System Exercise**

H. **Models and Methods**

I. **Other Appendices TBD**

*Appendices A through F were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

References include The INCOSE Systems Engineering Handbook, v3.2.2; The NASA Systems Engineering Handbook, Rev 1, and the Course texts

Requirements Management Definitions

Formal definition (Buede credits Stevens and Martin. 1995, p. 11):

- **The identification, derivation, allocation and control**
- **in a consistent, traceable, correlatable, verifiable manner**
- **of all the system functions, attributes, interfaces, and verification methods that a system must meet**
- **including customer, derived (internal), and specialty engineering needs.**

More "limited, and perhaps more common," definition (Buede credits Gotel and Finkelstein. 994, p. 95)

- **The care and feeding of the requirements, sometimes called requirements traceability**

Buede, The Engineering Design of Systems, 2nd Edition, Pg 194

Requirements Management Scope is throughout the System Lifecycle

- Selecting the requirements to be developed
- Ensuring common understanding of requirements
- Obtaining and maintaining provider and supplier commitment to requirements
- Ensuring alignment of project plans, work products, and requirements
- Establishing and managing bi-directional traceability
- Ensuring requirements can be, and are, validated and verified
- Managing requirements changes until system deactivation

Includes the overall management of requirements development, elaboration, negotiation, approval, validation, verification, achievement, and change throughout the system lifecycle

Is enabled by effective bi-directional traceability, and by robust requirements development and management processes

Enterprise Requirements Management

Many organizations have an Enterprise-level process to:

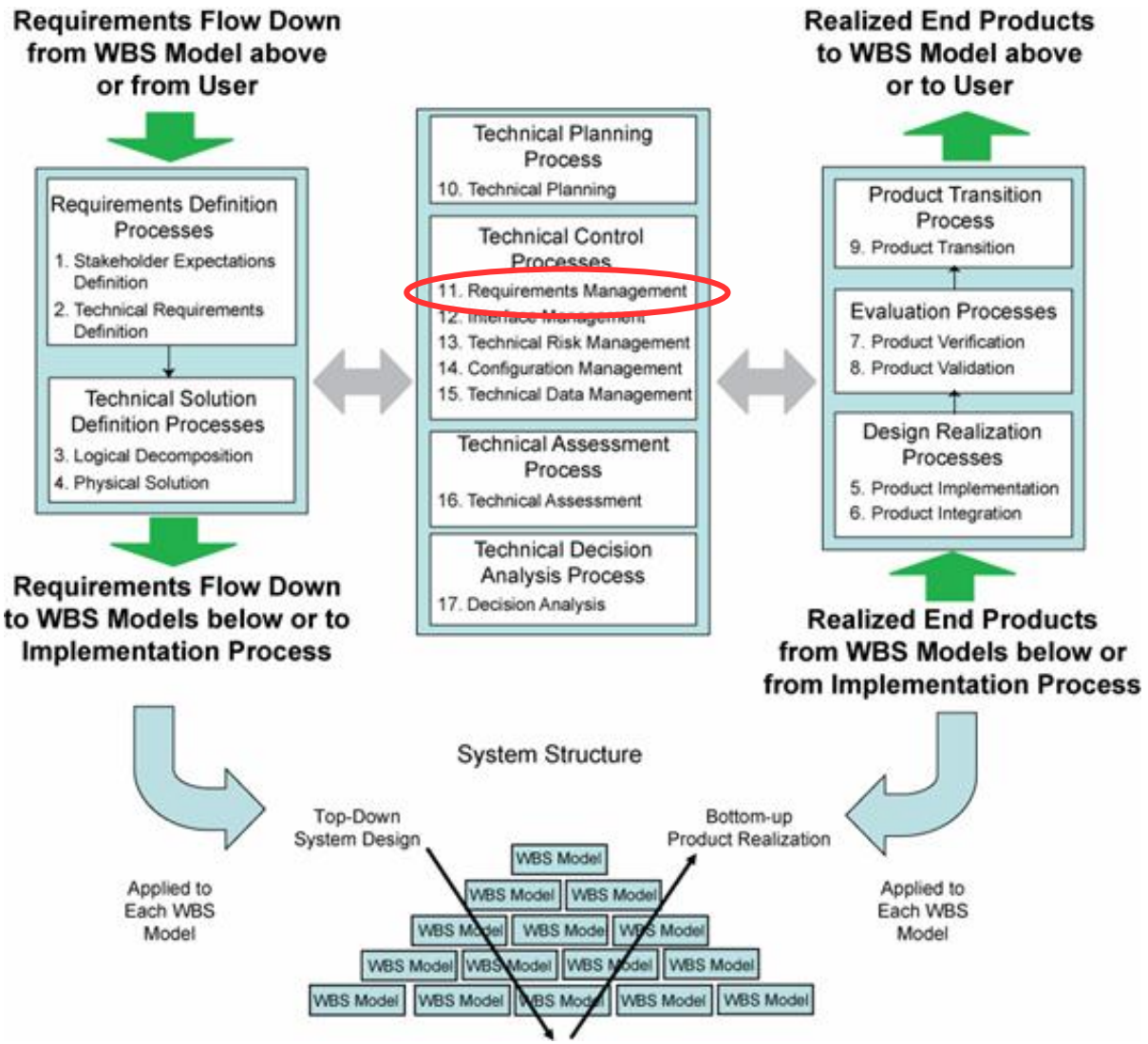
- **Select requirements to which the enterprise will respond**
- **Manage the stakeholder expectations**
- **Manage achievement of the requirements including V&V**
- **Manage requirements changes**

Example:

- **NASA (next slide)**

The INCOSE SE Handbook, based on ISO 15288 SE International Standard, does not call for a separate requirements management process, but includes all of the activities discussed herein in other processes

NASA Requirements Management Process within their "SE Engine" Model



NASA Procedural Requirements for Systems Engineering (NPR) 7123.1A

An Enterprise Must Select Requirements (Opportunities) to Pursue

- Selecting requirements
- Ensuring understanding
- Obtaining commitments
- Ensuring alignment
- Managing traceability
- Ensuring V&V
- Managing changes

There are many "Problems to be Solved"; Which ones are worthy of our resources?

- **Existing systems: performance shortfalls, obsolescence, No longer effective or efficient, unmitigated risk is too high**
- **New or changing: mission, threat , intelligence need, operational or support environment**
- **Opportunity: emerging technology, opportunity for innovation**

Enterprises must decide which "problems to solve" based on:

- **Criticality of the problem**
- **Predicted ability to solve**
- **Potential value to the enterprise**
- **Resources required vs. resources available**

Enterprises typically develop business cases or "mission cases" to aid in the decision of what problems to solve, and which solutions best solve the problem

Analyses of Alternatives (AOA) at the highest level of the enterprise are often used to make these decisions

Ensuring Mutual Understanding and Commitment to the Requirements

- Selecting requirements
- Ensuring understanding
- Obtaining commitments
- Ensuring alignment
- Managing traceability
- Ensuring V&V
- Managing changes

The Systems Engineering process emphasizes Communication and mutual understanding:

- **Requirements development techniques**
- **Modeling**
- **Process audits**
- **In-process validation**
- **Interdisciplinary teams and Stakeholder involvement**

Commitments, agreements, and alignment are obtained through the above and:

- **Proper contract structure**
- **Specifications and statements of work**
- **Lifecycle reviews and control (decision) gates**
- **Integrated planning**

What do we mean by integrated planning? Can you think of examples?

Integrated Planning to Ensure Project and Requirements Alignment

- Selecting requirements
- Ensuring understanding
- Obtaining commitments
- **Ensuring alignment**
- Managing traceability
- Ensuring V&V
- Managing changes

Integrated planning involves the stakeholders in the planning"

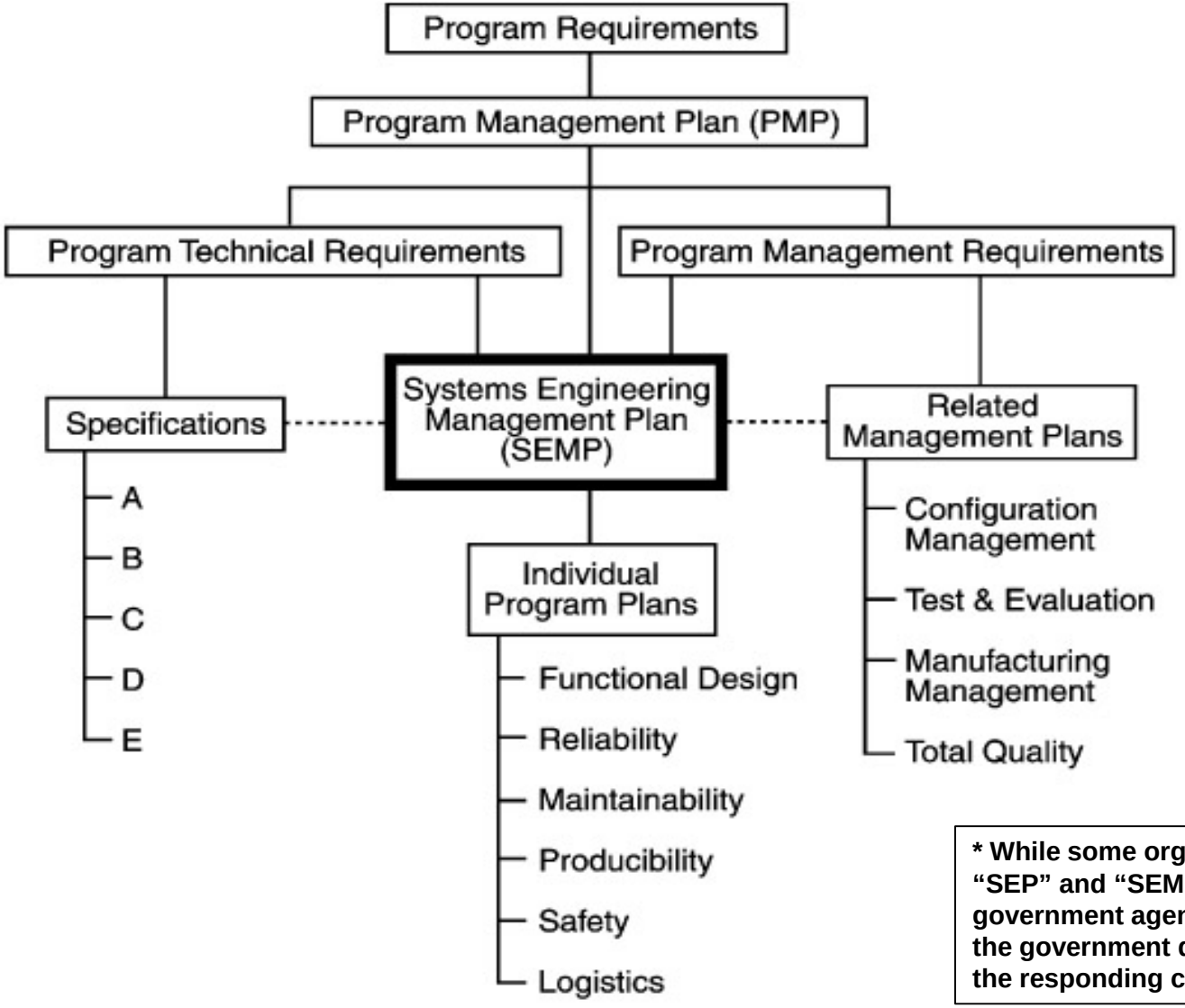
- **"Concurrent Engineering"**
- **Integrated Project teams (IPTs)**
- **Integrated Product and Process Development (IPPD)**
- **Results in integrated plans and schedules (below)**

An overall program or project plan Integrates all the other plans, including the plans for the technical work:

- **Hierarchy of plans with project or program plan at the top**
- **Includes Systems Engineering Plan (SEP)**
  - **Includes the plans and processes for the technical work**
  - **Integrates the engineering specialties and their plans**
  - **Subservient to program or project plan**

How do we integrate our plans and our requirements?

SEP* is Key Part of Program Planning



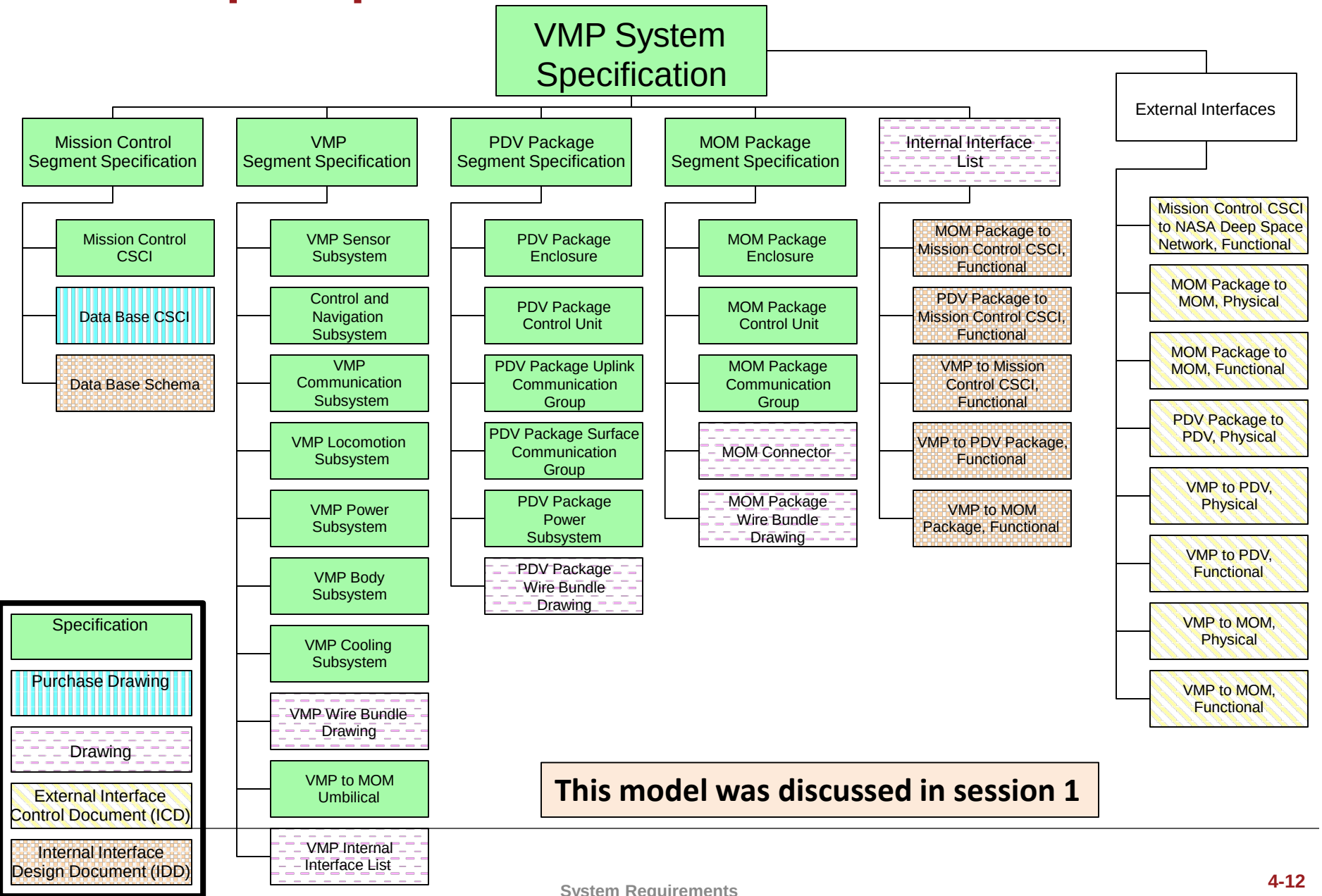
*** While some organizations use the terms "SEP" and "SEMP" synonymously, some government agencies now view the SEP as the government document, and the SEMP as the responding contractor document**

Product Breakdown Structure (PBS) in the form of a Specification Tree

- The PBS reflects the system architecture hierarchy
- Each entity in the hierarchy (often called configuration items) has a specification
- Integration Specifications are added to the nodes
- The project WBS adds activities to the PBS, creating a "solution-oriented WBS (next slide)
- This planning approach:
  - helps achieve integrated plans and alignment of project plans and requirements
  - Facilitates requirements traceability

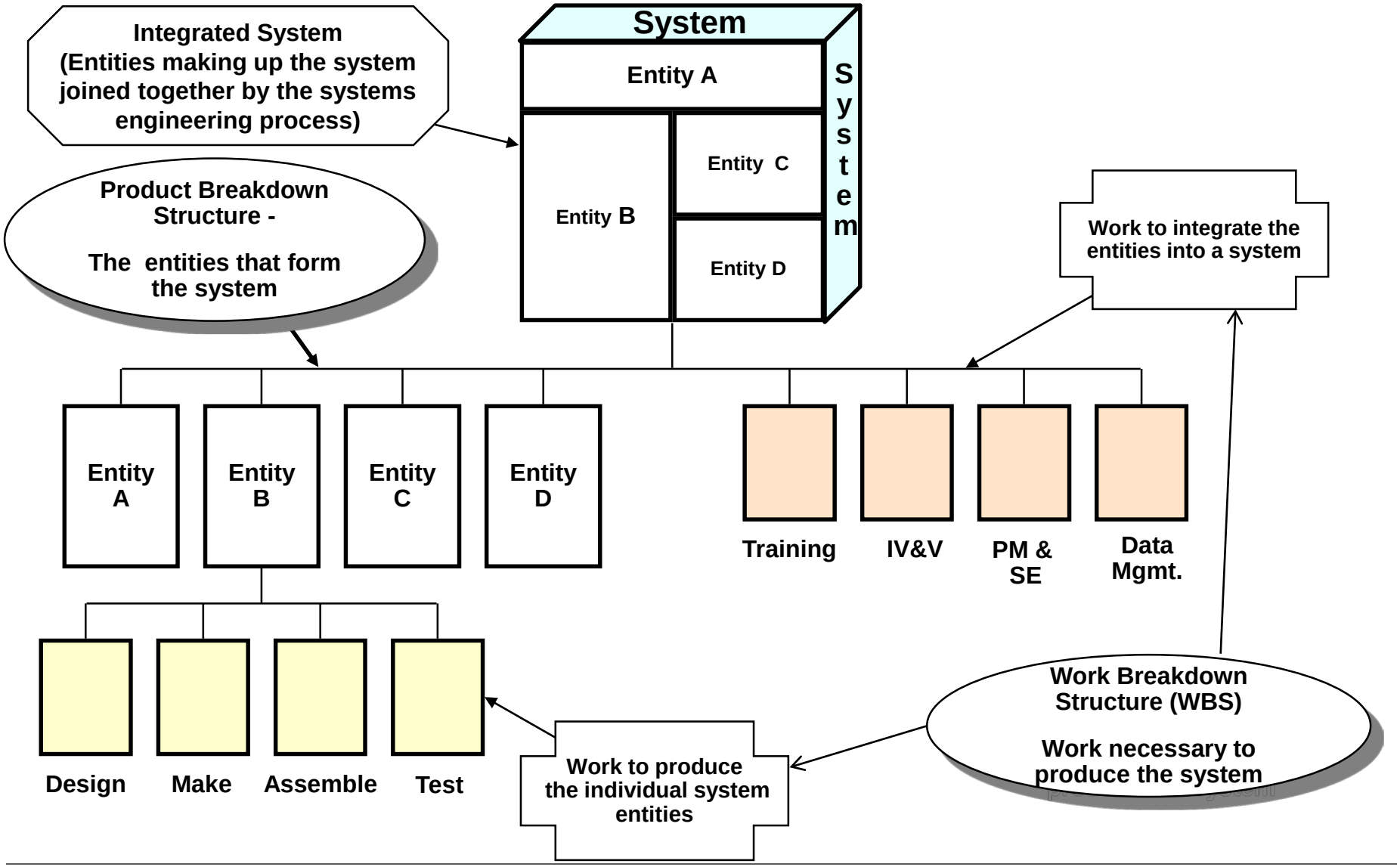
The INCOSE SE Handbook, v3.2.2, Figure 4-8, is an example of a Project Specification Tree, also known as a Product Breakdown Structure (PBS)

Sample Specification Tree



This model was discussed in session 1

The Solution-Oriented WBS Adds the Activities to the PBS



Adapted from : Forsberg, Mooz & Cotterman, *"Visualizing Project Management,"* Third edition

Session 2 - Stakeholder Needs

Lessons

1. **Program Overview Update**
2. **Functional Requirements**
3. **Performance and Non-Functional Requirements**
4. **Requirements Management and Traceability**
5. **Special Topics (if time permits)**

- Requirements management
- Traceability
- Baseline management



Appendices*

G. **Alarm System Exercise**

H. **Methods and Models**

I. **Other Appendices TBD**

*Appendices A through F were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

Managing Requirements with Requirements Traceability

- Selecting requirements
- Ensuring understanding
- Obtaining commitments
- Ensuring alignment
- **Managing traceability**
- **Ensuring V&V**
- Managing changes

Definitions (NASA SE Handbook, Rev 1, 2007)

- **Traceability – a discernible association between two or more logical entities (e.g., requirements, system elements, verifications, or tasks).**
- **Bidirectional traceability – an association between two or more logical entities that is discernible in either direction (i.e., to and from an entity)**

Requirements development inherently includes bidirectional traceability; each requirement should be traced to its source:

- **Higher level requirement or need**
- **Stakeholder**
- **Source artifact**
- **Rationale**

How are requirements traced to their rationale? Can you think of examples of how this applies to your projects?

Requirements traceability is discussed throughout this Technical Short Course; this section focuses on requirements management

Requirements Traceability and V&V

Definitions (a reminder)

- **Validation:**
  - **Confirmation, through the provision of objective evidence , that the requirements for a specific intended use or application have been fulfilled [ISO 9000: 2000 via INCOSE SE Handbook, v3.2.2]**
  - **The right system was (or is being) built**
  - **Validation applies to solution-independent requirements**
- **Verification:**
  - **Confirmation, through the provision of objective evidence, that specified requirements have been fulfilled [ISO 9000: 2000, via INCOSE SE Handbook, v 3.2.2]**
  - **The system was (or is being) built right**
  - **Verification applies to solution-dependent requirements**
- **Requirements traceability supports management of V&V**

Be careful with these definitions; the terms verification and validation are defined differently by different sources and authors; the above definitions are used throughout this course.

Requirements Traceability and V&V

PUID	Requirement	Source	Rationale	"Other"
1				
2				
3				

Basic traceability:

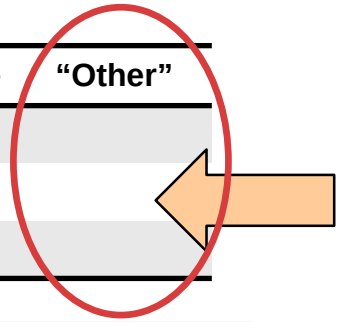
- **Is often represented in matrix* form**
- **Assigns a Project Unique identifier (PUID) to every expectation, need, or requirement when first identified**
- **Identifies the source (stakeholder; artifact; parent expectation, need or requirement; or other source)**
- **Points to the associated rationale . . . .**
- **. . . . and, can provide other important information, particularly with respect to V&V**

What are the "other" columns in the traceability matrix that might be useful to us?

*** Traceability best practices call for the use of a relational database, but the representation is often via a "spreadsheet-like matrix**

What "Other Columns" are Useful?

PUID	Req	Source	Rationale	"Other"
1				
2				
3				



Other Information

Validation Information

Verification Information

- **Plan**
  - **Method (T,D,I, A)**
  - **Brief description**
  - **Criteria**
  - **In-process or at-completion**
- **Responsibility**
- **Test plan paragraph ID**
- **Test procedure step**
- **Result:**
  - **Pass/fail**
  - **Quantified data**
  - **Anomaly number(s)**
- **Other……**

Class Discussion:

Beware the "T", "D", "I", and "A"

- **Typically we dutifully put one or more of these letters in our verification column, claim victory, and go on to our next task**
- **But what value do these 4 letters provide?**
- **What should we provide, and why?**

Validation Stages

Stage	Activities	
1. In Process	• Validation planning • Models • Stakeholder involvement	• Decision gate approval • IV&V witnessing • Is the project still worthwhile?
2. At - Completion	• Operational success • Formal validation • Anomaly management • Stakeholder satisfaction	• Does the developed solution solve the problem to be solved in the proper environment?

At-completion validation occurs in the target (operational and support) environments with real users

In-process validation occurs throughout the system lifecycle:

- **Every artifact is an opportunity for validation**
- **Best Requirements practices build validation into every step**
- **Primary purpose is to maximize probability of successful at-completion validation**

Session 2 - Stakeholder Needs

Lessons

1. **Program Overview Update**
2. **Functional Requirements**
3. **Performance and Non-Functional Requirements**
4. **Requirements Management and Traceability**
5. **Special Topics (if time permits)**

- Requirements management
- Traceability
- Baseline management



Appendices*

G. **Alarm System Exercise**

H. **Models and Methods**

I. **Other Appendices TBD**

*Appendices A through F were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

Alarm System Exercise – Part 11: Validation Planning



The instructor will provide directions for the exercise

Baseline Management Controls Requirements and Requirements Changes

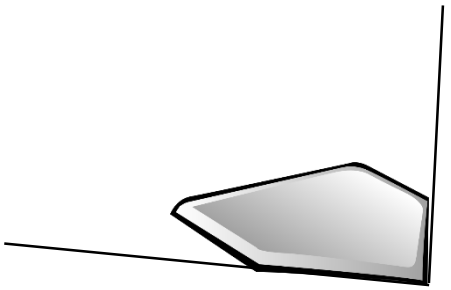
Baseline Management is:

- **The gated review and approval of the evolving baseline at prescribed decision gates, and**
- **The formal review and approval of changes to the previously approved baseline.**

Baselines are managed by controlling the artifacts by which they are represented

Baseline Management is enabled by configuration management systems and processes

- Selecting requirements
- Ensuring understanding
- Obtaining commitments
- Ensuring alignment
- Managing traceability
- Ensuring V&V
- Managing changes



Reference: Communicating Project Management Mooz, Forsberg, Cotterman John Wiley & Sons, 2002

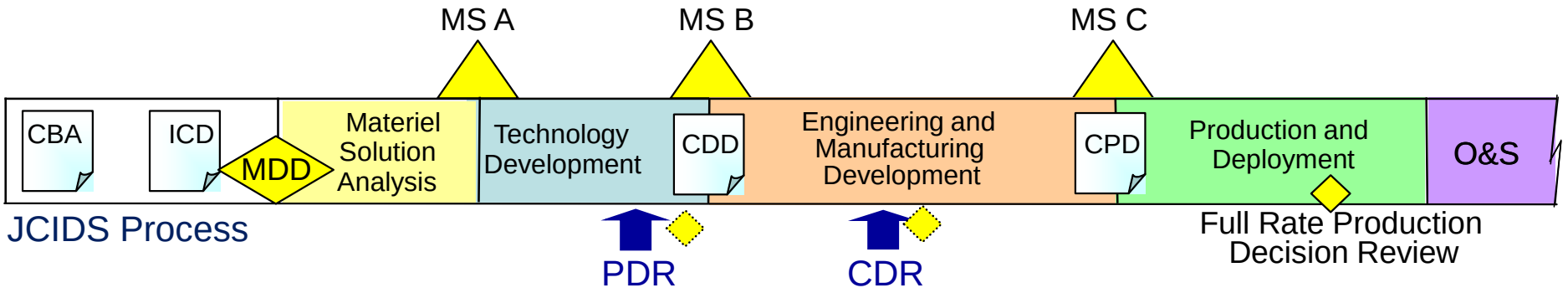
System Life Cycle Development Decision Gates Control the Evolving Baseline

The System Development Life Cycle is made up of:

- **Periods and phases (divide work into manageable pieces)**
- **Decision gates (Go/No-Go Decisions to proceed to the next phase or period)**
- **Artifacts (Work products representing the system that inform the decisions)**
- **Activities (Tasks to produce the needed artifacts)**

The evolving baseline reflects the system maturity

Defense Acquisition Management System



Some Terms

MDAP: Major Defense Acquisition Program
MS: Milestone
JCIDS: Joint Capability Integration Development System
CBA: Capabilities Based Assessment
ICD: Initial Capabilities Document
MDD: Materiel Development Decision
CDD: Capability Development Document
CPD: Capability Production Document
O&S: Operations and Support
PDR: Preliminary Design Review
CDR: Critical Design Review
MDA = Milestone Decision Authority

Each decision gate evolves the baseline, including the requirements

Objectives of Baseline Management

- Identify, define, and communicate the current baseline to stakeholders,
- Ensure baseline integrity (Consistent cost, schedule, and technical baseline aspects),
- Obtain stakeholder agreement with, approval of, and commitment to the baseline,
- Manage artifacts,
- Manage solution definition elaboration, and
- Focus the team creativity on the work to be done, rather than redoing what has been approved.

A key priority of baseline management is to manage the requirements being developed, and changes to the requirements already developed.

The Solution-Independent Baseline

- **Typical artifacts include:**
  - **Stakeholder Requirements Statement or Document**
  - **User CONOPS**
  - **System Requirements Document (SRD)**
  - **Interface Requirements Documents (if not part of SRD)**
  - **Evolving models***
  - **Traceability document**
- **Supporting artifacts not formally baselined, but archived to preserve the rationale for the baseline, include:**
  - **Reports, analyses, sketches, incidental models*, telecoms, memos, emails, and other artifacts supporting baseline decisions**

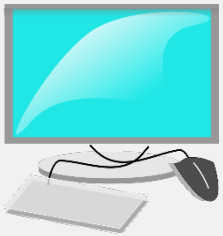
*** Incidental models support a specific decision, or a few specific decisions, and are archived; an evolving model is updated throughout the system lifecycle to support decisions to progressively mature the system baseline; these models should be part of the controlled baseline**

Baseline Management Best Practices

- The "Baseline Control Book"
  - Points to all baselined artifacts
  - Includes title, revision level, date, and owner of each artifact
  - If used, is part of the baseline



- The requirements database
  - A database containing all of the requirements information collected on the project
  - Includes both baselined and supporting data
  - Provides for traceability and change management
  - Can produce needed artifacts



- Configuration management systems and processes put in place at the beginning of the project and used to provide baseline management
- Model based Systems Engineering (MBSE) that evolves the system model, under configuration management, through the system life cycle

Baselining the Solution-Independent Requirements

- Implement requirements development and requirements processes and best practices
- Develop the requirements with stakeholder involvement
- Negotiate to obtain stakeholder agreement
- Progressively baseline your results
  - Obtain approval of artifacts at each life cycle process decision gate
  - Put approved requirements under formal change control
- Use the configuration management processes, systems, and tools appropriate to your project.

Distinguish between what has been previously baselined, and what is still “work in progress”; Control each properly.

Don’t throw away all your good requirements development efforts by being careless about requirements changes

Configuration Management

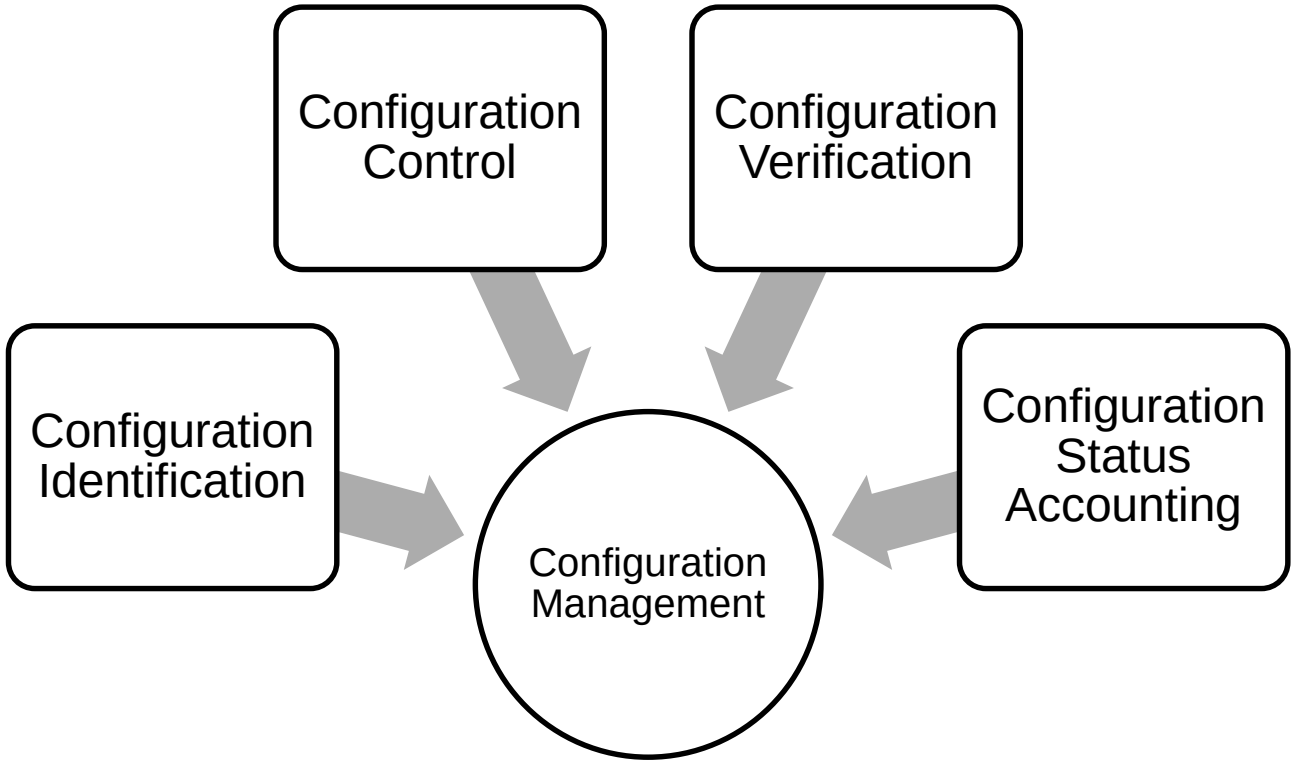
Definitions

- **The purpose of the Configuration Management Process is to establish and maintain the integrity of all identified outputs of a project or process and make them available to concerned parties (as stated in ISO/IEC 15288:2008)**

- **A management discipline applied over the product's lifecycle to provide visibility into, and to control changes to, performance and functional, and physical characteristics NASA SE Handbook, Rev1, 2007, p. 151 (commas by rbw)**

Configuration management includes the systems, processes, and tools by which requirements baselines are controlled

Configuration Management Scope

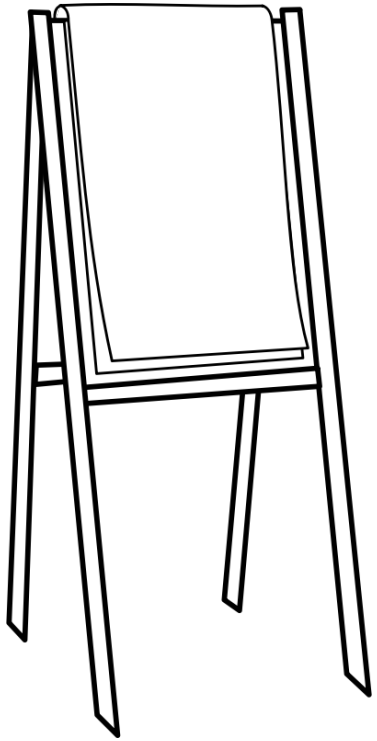


Configuration management enables baseline management

Why do We Need Configuration Management

- **Helps ensure:**
  - **Communication across the team**
  - **Mutual understanding of the current baseline**
  - **Team is working on the right work**
  - **Proper baseline management**
  - **Project accountability**
- **Forces intentional decisions about changes**
- **Requires robust impact analysis**
- **Focuses creativity on decisions still to be made**
- **Encourages concurrent engineering**

Discussion: What are Configuration Management Best Practices?



As SEs, What are your Baseline Management and Configuration Management Responsibilities?

- Recognize the need for baseline management and configuration management requirements for CM process, and insist that they be met.
- Know what CM system, processes, and tools are in place for you to use, and use it accordingly.
- Ensure that your requirements are managed during system development, and during system O&M
- Be the advocate for configuration management on your project.
- Remember,
  - Changes happen, and are often needed
  - Change control is not change prevention

There is no such thing as a small change

(Norm Augustine)

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		

All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?



Systems Engineering Technical Short Course

(Session 3: System Requirements)

Appendix G:
Alarm System Exercise

Exercise Content

Problem to be solved

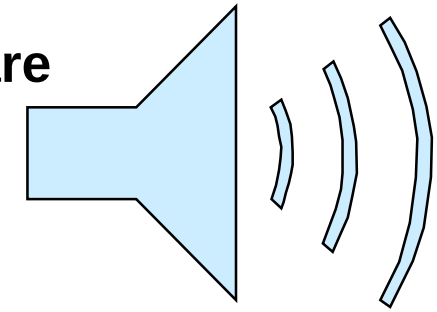
1. **Stakeholder list**
2. **Stakeholder analysis**
3. **Context diagram**
4. **CONOPS scenarios**
5. **N2 diagram**
6. **Scenario tracing**
7. **MOEs and MOPs**
8. **Performance requirements**
9. **Non-functional requirements**
10. **Requirements allocation and flow down**
11. **Verification and validation planning**

This exercise was started in session 2, and will be continued through session 3

Problem to be Solved and Context –

We will use evolve this exercise as we go through Sessions 2 & 3

- **Your team is responsible for developing requirements for an alarm system to protect facilities, property, and people**
- **This system will be used in homes and small businesses to detect intrusions, fire, smoke, and carbon monoxide.**
- **The system will be interconnected with a monitoring service and appropriate government agencies (fire, police, etc.).**
- **The system will be operated by the building owner or manager via a simple user interface**
- **The system must be secure against tampering or disablement**
- **The system must be able to operate over long intervals without external power**
- **You may add other requirements that you feel are necessary**



Alarm System Exercise - Part 1: Develop a List of Stakeholders for this problem

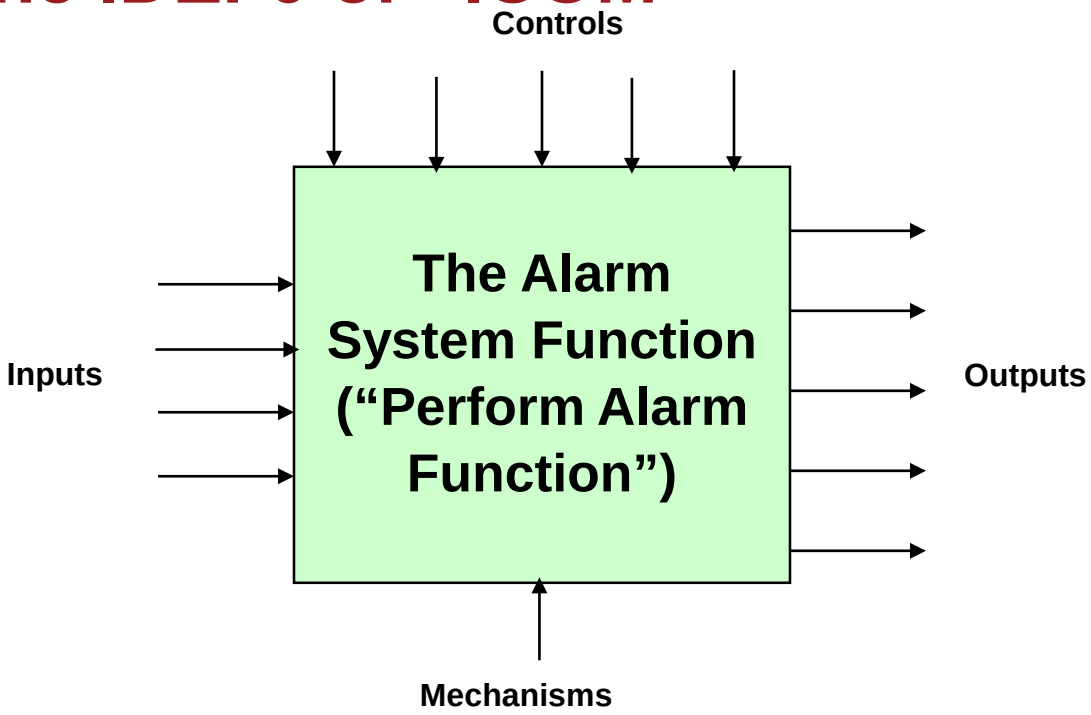
- **Based on the problem to be solved and context on the earlier slide, develop a list of stakeholders for the Alarm System Development Project**
- **You may use a traditional context diagram to help you develop the list of stakeholders (if time permits**
- **You will use this list later for subsequent exercises**

Alarm System Exercise - Part 2: Perform a Stakeholder Analysis for this Problem

- Start with your earlier list of stakeholders
- Use the format from the earlier slide (below)
- You may use this list later for subsequent exercises

Stakeholder	Type	Expectation	Priority

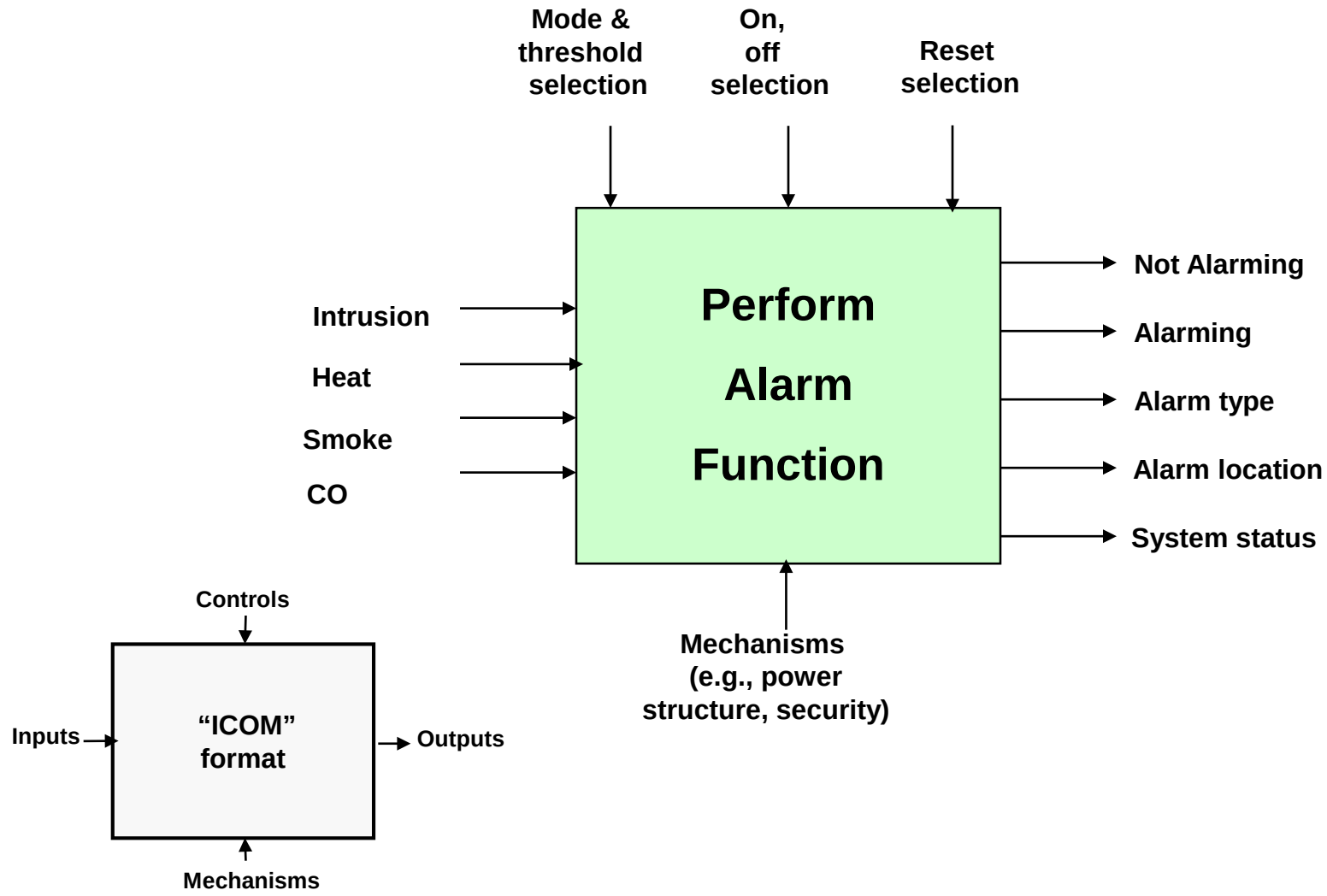
Alarm System Exercise - Part 3 (1/2): Develop a System Context Diagram using the IDEF0 or "ICOM"



- An "ICOM" is read as follows: Inputs are transformed by the system (or function, or process) into the outputs, subject to the controls, and supported by the mechanisms.
- This is a solution-independent problem; no solution has yet been assumed or selected.
- The ICOM or IDEF0 model is particularly useful as a starting point for functional analysis

For more information on the IDEF0 modeling approach, see Dennis M. Buede, The Engineering Design of Systems, 2nd Edition, New Jersey: Wiley, 2009

Alarm System Exercise - Part 3 (2/2): System Context Diagram Sample Solution



Slide will be uncovered during the lecture

Alarm System Exercise – Part 4 (1/3) : Background Information (Alexander, et al)

- Scenarios are a highly valuable means of understanding stakeholder expectations (and, ultimately, requirements)
- Three key means of discovering scenarios are:
  - Interviews (storytelling)
  - Observation
  - Workshops
- Standard patterns
  - Whole life scenarios e.g., (development, bringing into service, normal operations, maintenance, upgrade, disposal)
  - Day-in-the-life-of (e.g., starting up, daily health check, self test, cleaning, re-supply, daily operations such as transactions, closing down)
  - Transaction/operation/task
  - Data handling (e.g., Create, read, update, delete)
- Documenting scenarios
  - Unstructured stories (brief, informal, maybe examples, easy to document)
  - Storyboards (Time sequences of simple diagrams to tell stories graphically)
  - Operational scenarios (Relatively formal engineering stories)
  - Use cases (Highly structured stories of operational scenarios)

"Operational" must be interpreted as referring to support as well as operations

Alarm System Exercise – Part 4 (2/3) : Sample CONOPS Scenarios for an Elevator

- **Passengers request up service; receive feedback that their request was accepted; receive input that the elevator car is approaching and then an entry opportunity is available; enter the elevator car; request a floor; receive feedback that their request was accepted; receive feedback regarding door closing, floor levels transited, and floor at which the elevator is stopping; are notified of exit opportunity; and exit elevator**

- **A maintenance person is ordered to repair an individual elevator car; places the elevator in "partial maintenance" mode (allowing other cars to continue to operate normally); stops the car to be repaired; diagnoses the problem with the car; repairs and tests the car; and places the system in full operational mode**

Are these scenarios solution-independent or solution-dependent?

Although not shown here, these can be generated graphically with differing degrees of formality and tool usage

"Operational" must be interpreted as referring to support as well as operations

Alarm System Exercise – Part 4 (3/3) : Develop Operational Scenarios for Alarm System

Part A:

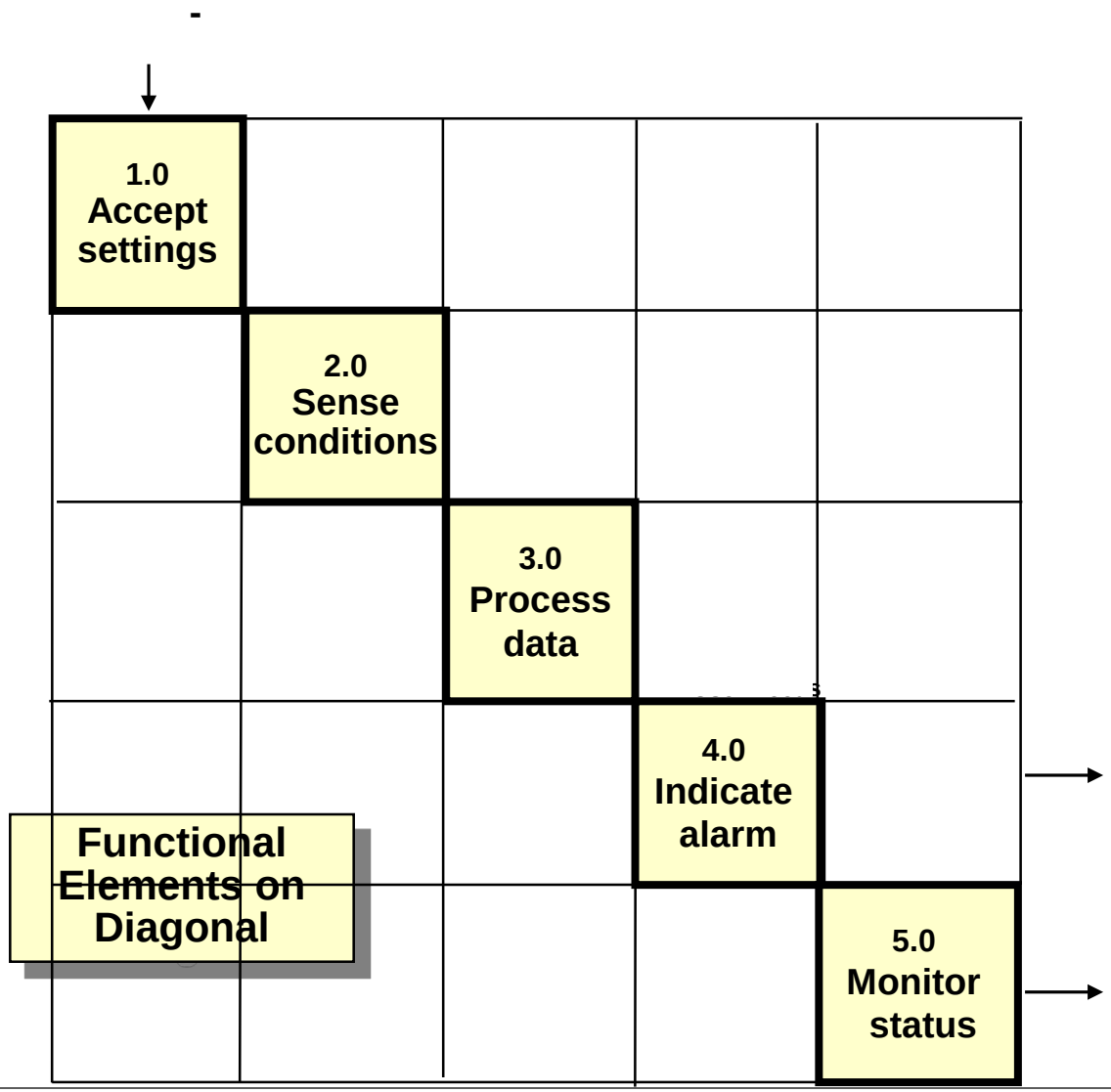
- **Keep them as solution-independent as possible**
- **Pattern them after the examples on the previous page**
- **Include at least one good operational scenario, and one good support scenario**
- **Be prepared to share your result with the class, including:**
  - **One of your scenarios, and**
  - **Your observations for the exercise**

Part B: (if time permits)

- **Develop two questions to ask another group regarding their scenarios, to clarify the intent (and to illustrate the power of questions used to clarify requirements)**

"Operational" must be interpreted as referring to support as well as operations

Alarm System Exercise – Part 5: Develop N2 Diagram for "Perform Alarm Function"



Instructions:

Create an N2 diagram for the Alarm System problem we have been using in the course.

Use the following as inputs to the exercise:

- **The template to the left**
- **The functional definition developed in this lesson**
- **Any other information you believe is relevant to the problem.**

Be prepared to brief the rest of the class on your result, and what you learned from the exercise

Alarm System Exercise – Part 6: Scenario Tracing

- Look over the scenarios that you created for the alarm system is session 2
- Review the functional architecture that has been completed so far for the alarm system. You can use the instructor's example plus whatever additional work your group has done
- Try to validate that your functional architecture supports the scenarios you created.
- Explain your results. If your scenario cannot be supported by your functional architecture, can you explain why?

Alarm System Exercise – Parts 7 & 8: MOEs, MOPs, & Performance Requirements

These two exercises are planned to be done together:

Directions for these exercise are provided on the next two slides

Alarm System Exercise - Part 7: Identify at least One MOE, and at least One Supporting MOP for each MOE

Measures of Effectiveness (MOE)

- Are operational measures of success closely related to achievement of operational or mission objectives
- provide insight into the accomplishment of mission needs
- Are solution-independent
- Focus on how well (not how) mission or operational objectives are achieved

Measure of Performance (MOP):

- Measures that characterize physical or functional attributes relating to the system operation, measured or estimated under specified conditions

MOEs are typically supported by a hierarchy of measures:

- Key Performance Parameters (KPPs)
- Measures of Performance (MOPs)
- Technical Performance Measures (TPMs)

Alarm System Exercise - Part 8: Write at least One Performance Requirement for the Alarm System in the Correct Format

- **Subject - the product/component clearly stated**
- **Verb - shall/shall not**
- **Requirement - the function/constraint**
- **Conditions (if any) - the circumstances in which the requirement functions or constrains**

- **e.g. The radar shall detect a crossing 1 meter square target at a distance of 100 km when there is a direct line of sight to the target.**

Subject	Verb	Requirement	Condition(s)
The radar	shall	Detect all crossing targets whose radar cross section is 1 meter square or greater, at a distance of up to 100 km	when there is a direct line of sight to the target.

For the purpose of this exercise you may have to "make-up" the numbers needed to quantify the requirement; in a real project, you would use requirements analysis to determine these quantities from the S/H requirements.

Alarm System Exercise – Part 9: Non-Functional Requirements

Based on your understanding of the problem so far, and on the course materials just completed, think about the non-functional requirements for the alarm system:

1. **Name 3 NFRs that you think are particularly important for the alarm system**
2. **Discuss How will you quantify those NFRs?**
3. **Be prepared to share your results with the class**

Alarm System Exercise – Part 10: Requirements Allocation and Flow Down

- If you have not already done so, create a level one PBS for the alarm system
- Pick a set of your top level requirements, and show how you would allocate those requirements To the level 1 system entities
- Discuss how you would demonstrate (illustrate) traceability of the allocated requirements back up to their parents
- Prepare to share with the class a brief summary of your results and your insights from the exercise. Include how you would decide what the lower level solution dependent requirements would be

Alarm System Exercise – Part 11: Verification and Validation Planning

- Select three important requirements from your earlier work on the alarm system problem.
  - Consider making these a mix of performance and Non-functional requirements
- Discuss how each of these requirements will be validated or verified
- Decide what you would enter for those requirements on the traceability matrix regarding the intended verification or validation
- Be prepared to share your results with the class.

End of Exercise

Systems Engineering Technical Short Course

(Session 3: System Requirements)

Appendix H: Methods and Models

The Methods and Models in this appendix are from the course text (Buede), and are used throughout the system life cycle to define requirements and solutions.

Session 3 – System Requirements

Lessons

1. Program Overview Update
2. Functional Requirements
3. Performance and Non-Functional Requirements
4. Requirements Management and Traceability
5. Special Topics (if time permits)

Appendices*

G. Alarm System Exercise

H. Models and Methods

*Appendices A through F were contained in earlier sessions of the course; Appendix D is a complete course bibliography. Appendices will not be formally covered

Modeling Approaches & Methods

- **Data Modeling** (relationships between inputs and outputs)
  - Entity-relationship diagrams (ERDs)
  - Higraphs
- **Process Modeling** (functional decomposition and flow of inputs and outputs for those functions)
  - Data flow diagrams (DFDs)
  - IDEF0
  - N^2 charts
- **Object-oriented Modeling***
  - Object modeling technique (OMT)
  - Real-time object-oriented modeling (ROOM)

- **Behavior Modeling** (control, activation, and termination of system functions needed to meet performance requirements)
  - Function flow block diagrams (FFBDs)
  - Behavior diagrams (BDs)
  - State-transition diagrams (STDs)
  - Statecharts
  - Control flow diagrams (CFDs)
  - Petri nets (PNs)

Many examples of these are shown in the back-up slides following this lesson

*Buede states that OOD is increasingly used in SE, and uses the other three domains identified on this chart



For more information on models and methods, see chapter 12 of the course text:

The Engineering Design of Systems: Models and Methods, 2nd Edition

By Dennis Buede
Innovative Decisions, Inc.
dbuede@innovativedecisions.com

Chapter 12 – Graphical Modeling Techniques

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		

All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?

Systems Engineering Technical Short Course for Waters Corporation

This course planned to be conducted at the Waters facilities as follows:

Session 1:	**January 27 – 29, 2015:**	**Overview and Concepts of Systems Engineering**
Session 2:	**February 16 – 17, 2015:**	**Stakeholder Needs**
Session 3:	**March 16 – 17, 2015**	**System Requirements**
Session 4:	**April 13 – 14, 2015**	**System Architecture and Design**
Session 5:	**May 13 – 14, 2015**	**System Integration and Test**
Session 6:	**June 10, 2015**	**Collaboration Training with WLI. (not part of WPI material)**
	June 11, 2015	**Capstone presentations**

Robert B. Wheeler, ESEP, PMP
Thomas F. Gannon, Ph.D.

This course is developed by Worcester Polytechnic Institute (WPI) for Waters Corporation. Course materials remain the property of WPI. WPI and the course developer (Robert Wheeler of Robert B Wheeler & Associates for sessions 1, 2, and 3) retain the right to use, reuse, and modify this content in future course deliveries.



Systems Engineering Technical Short Course

Session 4: Systems Architecture and Design

April 13 - 14, 2015

Robert B. Wheeler, ESEP, PMP
Thomas F. Gannon, Ph.D.



Systems Engineering Technical Short Course

(Session 4: Systems Architecture and Design)

Lesson 1:
Program Review Update

Welcome and Opening Remarks……

Logistics:

- **Facility**
- **Safety & emergencies**
- **Schedule & breaks**
- **Parking lot**
- **Other?**







Code of Conduct:

- On time
- Mutual respect
- Machines off & closed unless we are using them
- Participation!
- Other?

Remember from Session 1 – During this course you will: (1/2)

All Sessions: Apply the Systems Engineering principles and practices to your assigned projects

Sessions 1 through 3 (Bob Wheeler, instructor):

- **Develop and use a common vocabulary**
- **Apply systems thinking and holism to your projects**
- **Articulate the Problem to be Solved by your project**
- **Develop a Technical Risk Management Plan**
- **Elicit and express stakeholders expectations and requirements**
- **Develop User Concept of Operations (CONOPS), solution selection criteria, and acceptance criteria**
- **Conduct system functional and performance analyses**
- **Define and express system requirements**
- **Manage requirements through the system lifecycle**

Session 1 | Session 2 | Session 3

Remember from Session 1 – During this course you will: (2/2)

Sessions 4 & 5 (Tom Gannon, instructor):

- Develop creative alternative system architectures
- Conduct analyses of your architectural alternatives to meet stakeholder and system requirements
- Define a candidate architecture and design for your solution
- Develop strategies and plans for system integration, validation and verification testing
- Manage requirements through the system lifecycle (continued)
- Document the results of your work in appropriate artifacts

Session 6 (Tom Gannon, instructor):

- Present a status report on your project to Waters senior management

This Session 4 | Session 5

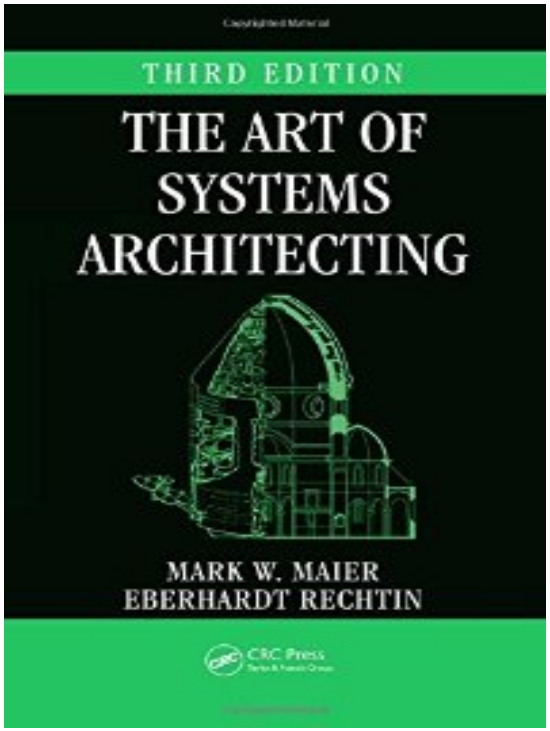
Schedule

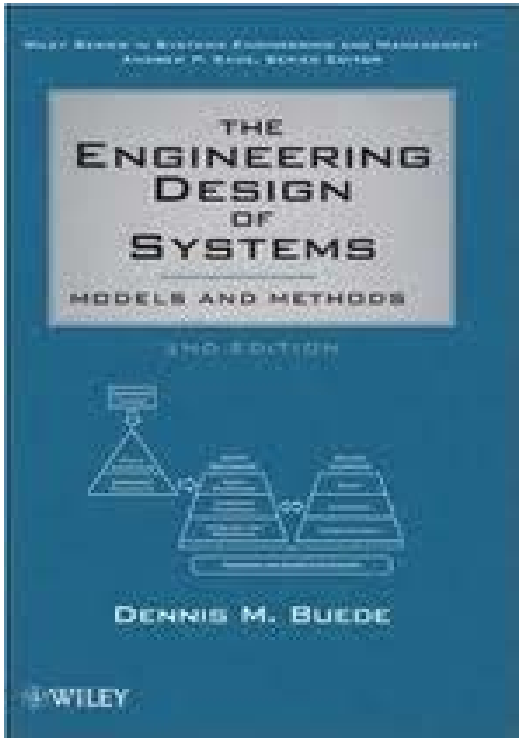
Session	Dates	Topic
1.	January 27 – 29	Introduction – Overview and Concepts of SE (US)
2.	February 16 – 17	Stakeholder Needs*
3.	March 16 – 17	Systems Requirements*
4.	April 13 – 14	Systems Architecture and Design (UK)*
5.	May 13 – 14	Systems Integration and Test*
6.	June 10	Collaboration Training with WLI
	June 11	Capstone Presentations (US)

Notes:

- Denotes sessions held for some participants via video-teleconference.
- Class times and room locations are shown later

Primary Requirements Course Texts





Mark W. Maier and Eberhardt Rechtin, *The Art of Systems Architecting*, 3rd Edition, CRC, 2009

Dennis M. Buede, The Engineering Design of Systems, 2nd Edition, New Jersey: Wiley, 2009

Please see Session 1 Appendix D for the other three primary course texts, and many other applicable references

What Happens if I Miss a Class?

It is important to your own learning and to the group process to attend all the class sessions.

However, it is understood that some absences cannot be avoided.

If you miss a class, please take the responsibility to:

- **Review the missed lectures**
- **Contact the instructor with any questions you may have as a result of reviewing the lecture materials**
- **Provide a brief outline of the missed material, identifying what you believe to be the key learning points, to the instructor.**
- **Contact your team project leader to ensure that you are aware of any project work completed, or action items assigned, during your absence.**

All lecture material will be posted on the WPI course site:

- **You will be told how to use the WPI course site, and how to access the lecture material, early in session 1**

Video Tutorial for myWPI

What are your Questions before We Look at our Homework Assignments?



Sessions 2 through 5 Agenda (Revised)

- **Day 1**
  - **Review project team assignments from previous Session**
  - **Presentation and discussion of Session topic**
  - ~~**Project team working sessions on the application of the Session topics to their projects**~~
  - **Day 1 wrap-up and Q&A**

- **Day 2**
  - ~~**Review Day 1 topic and project working sessions**~~
  - **Presentation and discussion of Session topic (concluded)**
  - **Project team working sessions**
  - **Check-in with class on the application of the Session topic to their projects**
  - **Session wrap-up: Q&A and homework due next Session**

Timing will be similar to session 1, and will be announced at beginning of each session

Homework Guidance

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

Focus on purpose of the deliverables and the relationships within the team(s) and organization. Then see how they support the Problem Statement.

Plan for approximately 2 days of work between class sessions.

Notify your manager and Facilitator of priority conflicts.

©2015 Waters Corporation

Session 3 - System Requirements: Homework Assignment due at beginning of Session 4*

Develop the following deliverables:

- **Functional Architecture (see Buede text):** (Explain and illustrate your approach to develop a solution-independent functional architecture. Use any of the approaches from the course material)
- **Analysis of Non-Functional Requirements (NFRs):** (Explain your approach toward NFRs, and show examples of how you used that approach to develop NFRs. Include NFRs in your SRD or equivalent)
- **System Requirements Document (SRD) or equivalent expression of system requirements:** (Document your system requirements. Find a suitable template, tailor the template to your system, and record your system requirements)
- **Draft Requirements Traceability Matrix, or equivalent:** (Should trace to source, and point to validation/verification approach. Use templates and examples shown in the course materials, and include all requirements that you were able to identify and define)

Update the previously developed deliverables

- Each project team must decide which of, and the extent to which, each of these deliverables will be done
- If you cannot complete these deliverables for your system, identify what is not completed, and how you would complete the work
- The course text (Buede, The Engineering design of Systems) has artifact outlines

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

PROJECT GROUPING

SYSTEMS CONTROLLER

Sponsor: Allen Caswell

Cindy Miller (US)
Richard Moulds (UK)
Richard Chapman (UK)
Steve Binney (US)
Justin Deary (US)
Phillip Young (UK)
David Gordon (UK)

NEXUS

Sponsors: Joe Luongo/John Chan

Abhijit Tarafder (US)
Wade Leveille (US)
David Lamborghini (US)
Ian Trivett (UK)
Daniel Kenney (UK)
Jim Usowicz (US)
John Heden (US)

SMART CONSUMABLES

Sponsor: Tom Walter

Jacob Fairchild (US)
Jason Hill (US)
Mathew DeLano (US)
Jan Driesen (GERMANY)
Alex Hooper (UK)
Mark Pickering (UK)
Senthil Bala (US)

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

PROJECT GROUPING

SYSTEMS CONTROLLER

Sponsor: Allen Caswell

Cindy Miller (US)
Richard Moulds (UK)
Richard Chapman (UK)
Steve Binney (US)
Justin Deary (US)
Phillip Young (UK)
David Gordon (UK)

NEXUS

Sponsors: Joe Luongo/John Chan

Abhijit Tarafder (US)
Wade Leveille (US)
David Lamborghini (US)
Ian Trivett (UK)
Daniel Kenney (UK)
Jim Usowicz (US)
John Heden (US)

SMART CONSUMABLES

Sponsor: Tom Walter

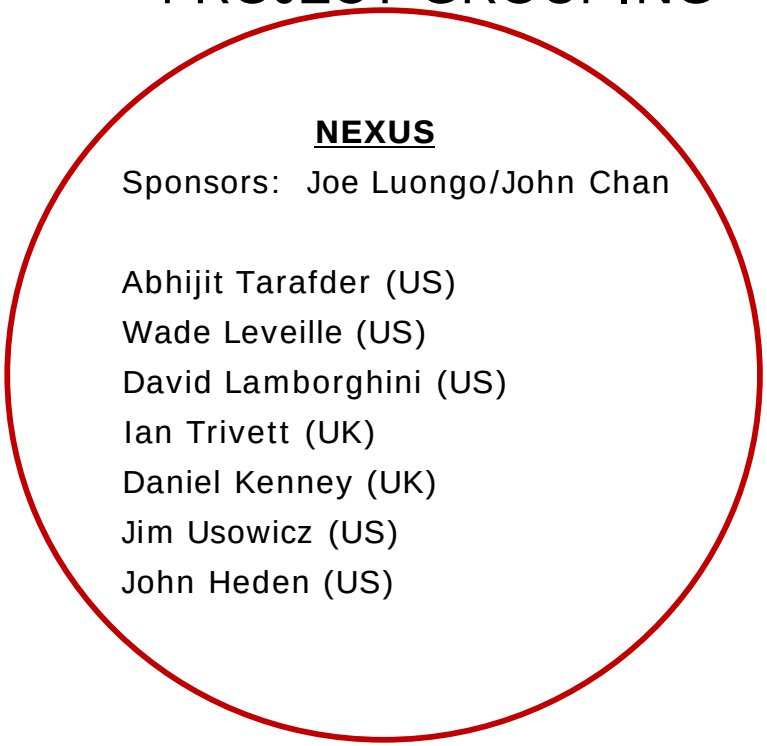
Jacob Fairchild (US)
Jason Hill (US)
Mathew DeLano (US)
Jan Driesen (GERMANY)
Alex Hooper (UK)
Mark Pickering (UK)
Senthil Bala (US)

©2015 Waters Corporation

WPI SYSTEMS ENGINEERING

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

PROJECT GROUPING



SYSTEMS CONTROLLER

Sponsor: Allen Caswell

Cindy Miller (US)
Richard Moulds (UK)
Richard Chapman (UK)
Steve Binney (US)
Justin Deary (US)
Phillip Young (UK)
David Gordon (UK)

NEXUS

Sponsors: Joe Luongo/John Chan

Abhijit Tarafder (US)
Wade Leveille (US)
David Lamborghini (US)
Ian Trivett (UK)
Daniel Kenney (UK)
Jim Usowicz (US)
John Heden (US)

SMART CONSUMABLES

Sponsor: Tom Walter

Jacob Fairchild (US)
Jason Hill (US)
Mathew DeLano (US)
Jan Driesen (GERMANY)
Alex Hooper (UK)
Mark Pickering (UK)
Senthil Bala (US)

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

PROJECT GROUPING

SYSTEMS CONTROLLER
Sponsor: Allen Caswell

Cindy Miller (US)
Richard Moulds (UK)
Richard Chapman (UK)
Steve Binney (US)
Justin Deary (US)
Phillip Young (UK)
David Gordon (UK)

NEXUS
Sponsors: Joe Luongo/John Chan

Abhijit Tarafder (US)
Wade Leveille (US)
David Lamborghini (US)
Ian Trivett (UK)
Daniel Kenney (UK)
Jim Usowicz (US)
John Heden (US)

SMART CONSUMABLES
Sponsor: Tom Walter

Jacob Fairchild (US)
Jason Hill (US)
Mathew DeLano (US)
Jan Driesen (GERMANY)
Alex Hooper (UK)
Mark Pickering (UK)
Senthil Bala (US)

©2015 Waters Corporation

Session 4 - System Architecture and Design: Learning Objectives

- Develop creative alternative system architectures which address system operational, functional and physical viewpoints
- Conduct analyses of architectural alternatives to meet stakeholder and system requirements
- Manage system architecture and design through the system lifecycle
- Express system architectures and designs in appropriate artifacts

Session 4 - System Architecture and Design:
Homework Assignment due at beginning of Session 4*

Develop the following deliverables:

- **ConOps and Operational View:** (Develop a high-level ConOps and Operational Architectural View for your system. Use any of the approaches from the course material)
- **Logical (Functional) View:** (Develop a Logical View for your system using your functional architecture from Session 3 decomposed to at least one additional level of detail. Use any of the approaches from the course material)
- **Alternative Design Concepts:** (Brainstorm and identify 2 or 3 alternative high-level design concepts which meet your functional requirements and system requirements. Use any of the approaches from the course material)
- **Evaluation Criteria:** (Develop the criteria that you would use to evaluate your alternative design concepts, and the method(s) that you would use to select the best overall concept for further development as your design solution. Use any of the approaches from the course material)

Update the previously developed deliverables

- Each project team must decide which of, and the extent to which, each of these deliverables will be done
- If you cannot complete these deliverables for your system, identify what is not completed, and how you would complete the work
- The course texts and appendices have artifact outlines

Sessions 2 through 5 Agenda (Revised)

- **Day 1**
  - **Review project team assignments from previous Session**
  - **Presentation and discussion of Session topic**
  - ~~**Project team working sessions on the application of the Session topics to their projects**~~
  - **Day 1 wrap-up and Q&A**

- **Day 2**
  - ~~**Review Day 1 topic and project working sessions**~~
  - **Presentation and discussion of Session topic (concluded)**
  - **Project team working sessions**
  - **Check-in with class on the application of the Session topic to their projects**
  - **Session wrap-up: Q&A and homework due next Session**

Timing will be similar to session 1, and will be announced at beginning of each session

Session 4 – System Architecture and Design

Lessons

1. **Program Overview Update**
2. **System Architecture Basics**
3. **Systems Design Process**
4. **Analysis of Alternatives**
5. **Model-Based Systems Engineering**
6. **Special Topics (if time permits)**

- Systems Architecture Context
- Architectural Levels and Views
- Architecture Frameworks

- NASA Design Process
- Generic Design Process

- Formal Decision Process
- Evaluation Methods

- Motivation
- SysML Overview
- Distilled Example

- IDEF0 In-depth
- Architecture Framework Examples

Appendices*

I. **SysML Diagrams and Language Concepts**

J. **Architecture Frameworks In-depth**

*Appendices A through H were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		

All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?



Systems Engineering Technical Short Course

(Session 4: System Architecture and Design)

Lesson 2:
System Architecture Basics

Session 4 – System Architecture and Design

- Systems Architecture Context
- Architectural Levels and Views
- Architecture Frameworks

Lessons

1. **Program Overview Update**
2. **System Architecture Basics**
3. **Systems Design Process**
4. **Analysis of Alternatives**
5. **Model-Based Systems Engineering**
6. **Special Topics (if time permits)**

Appendices*

I. **SysML Diagrams and Language Concepts**

J. **Architecture Frameworks In-depth**

*Appendices A through H were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

Architecture









- **What does an architect do?**
- **What are his/her tools?**
- **What are the products?**

- ***(The architecture?)***

What is a System?

Etymology: Late Latin *systemat-, systema,* from Greek *systEmat-, systEma,* from *synistanai* to combine, from *syn-* + *histanai* to cause to stand. Date: 1603

1: a regularly interacting or interdependent group of items forming a unified whole <a number *system*>: as a (1) : a group of interacting bodies under the influence of related forces <a gravitational *system*> (2) : an assemblage of substances that is in or tends to equilibrium <a thermodynamic *system*> b (1) : a group of body organs that together perform one or more vital functions <the digestive *system*> (2) : the body considered as a functional unit c : a group of related natural objects or forces <a river *system*> d : a group of devices or artificial objects or an organization forming a network especially for distributing something or serving a common purpose <a telephone *system*> <a heating *system*> <a highway *system*> <a data processing *system*> e : a major division of rocks usually larger than a series and including all formed during a period or era f : a form of social, economic, or political organization or practice <the capitalist *system*>

2-3

2: an organized set of doctrines, ideas, or principles usually intended to explain the arrangement or working of a systematic whole <the Newtonian *system* of mechanics>

3: an organized or established procedure <the touch *system* of typing> b : a manner of classifying, symbolizing, or schematizing <a taxonomic *system*> <the decimal *system*>

4: harmonious arrangement or pattern : ORDER <bring *system* out of confusion -- Ellen Glasgow>

What is an Architect?

- **Definition: *Architect***

Function: *noun*
Etymology: Middle French *architecte,* from Latin *architectus,* from Greek *architektOn* master builder, from *archi-* + *tektOn* builder

Date: 1563

1 : a person who designs buildings and advises in their construction

2 : a person who designs and guides a plan or undertaking <the *architect* of American foreign policy>

So What is a Systems Architect?

- **Definition: *System Architect***

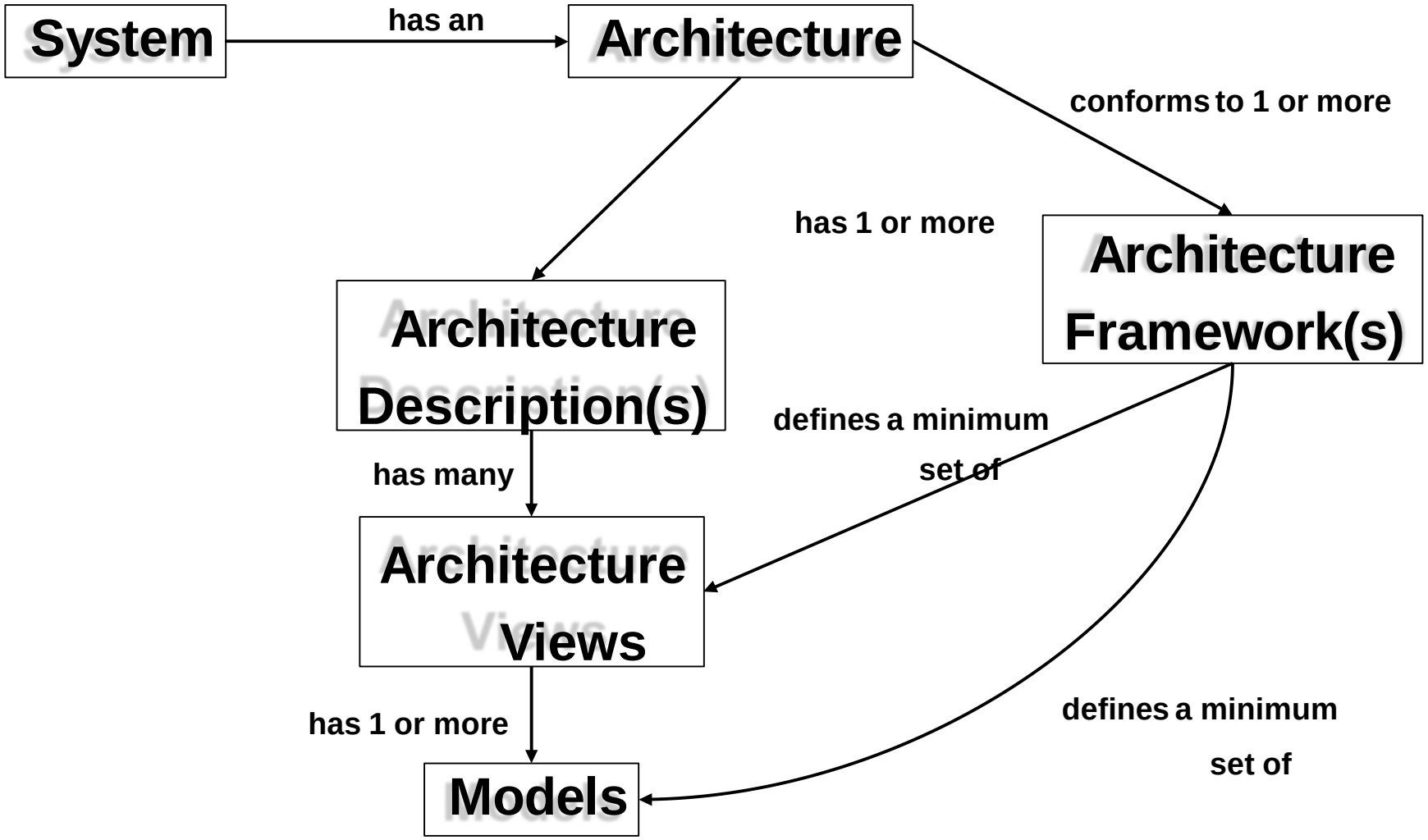
systems

A person who designs buildings and advises in their construction,

where a *system* is

A regularly interacting or interdependent group of items forming a unified whole , a group of devices or artificial objects or an organization forming a network especially for distributing something or serving a common purpose.

Architecture Terms



Definitions of Architecture

- **Webster's Dictionary**
  - The art or science of building; specifically the art or practice of designing and building structures…
  - Formation or construction as or as if the result of conscious act
  - A unifying or coherent form or structure
  - Architectural product or work
  - A method or style of building
  - The manner in which the components of a computer or computer system are organized and integrated

- **INCOSE**
  - System Architecture: "The fundamental and unifying system structure defined in terms of system elements, interfaces, processes, constraints, and behaviors."

- **M&R**
  - The structure – in terms of components, connections, and constraints – of a product, process, or element

Definitions of Architecture (Con't)

DoD Architecture Framework (DoDAF)

The structure of components,

their relationships,

and the principles and guidelines governing their design and evolution over time

IEEE 1471*

The fundamental organization of a system embodied in its components,

their relationships to each other, and to the environment,

and the principles guiding its design and evolution.

* IEEE 1471 – "Recommended Practice for Architectural Description of Software-Intensive Systems"

Definitions

- **Architecture provides:**
  - **Structure and mechanisms for knowledge transfer**
    - **Mutually consistent orthogonal views or models of the problem and solution space**
    - **Common definitions, data, and frame of reference**
    - **Predictive capability to support operational planning and investment decisions**
  - **Performance assessment, interoperability, and control logic validation through *executable architectures***
    - **Model-based Systems Engineering (MBSE)**
  - **A continuum for system of systems (SoS) design**

What Are Architectures Used For?

- **Resource planning and acquisition strategy**
  - **Support investment decisions to evolve legacy systems or develop revolutionary new systems**
- **System of systems definition**
  - **Operational concept**
  - **Interoperability analysis**
    - **Interface specification and control**
  - **Evaluation of behavior and performance**
  - **Identification of system duplications and gaps**
  - **Time-phased evolution**
  - **System development**
- **Knowledge repository of Artifacts**

Definitions

Artifact

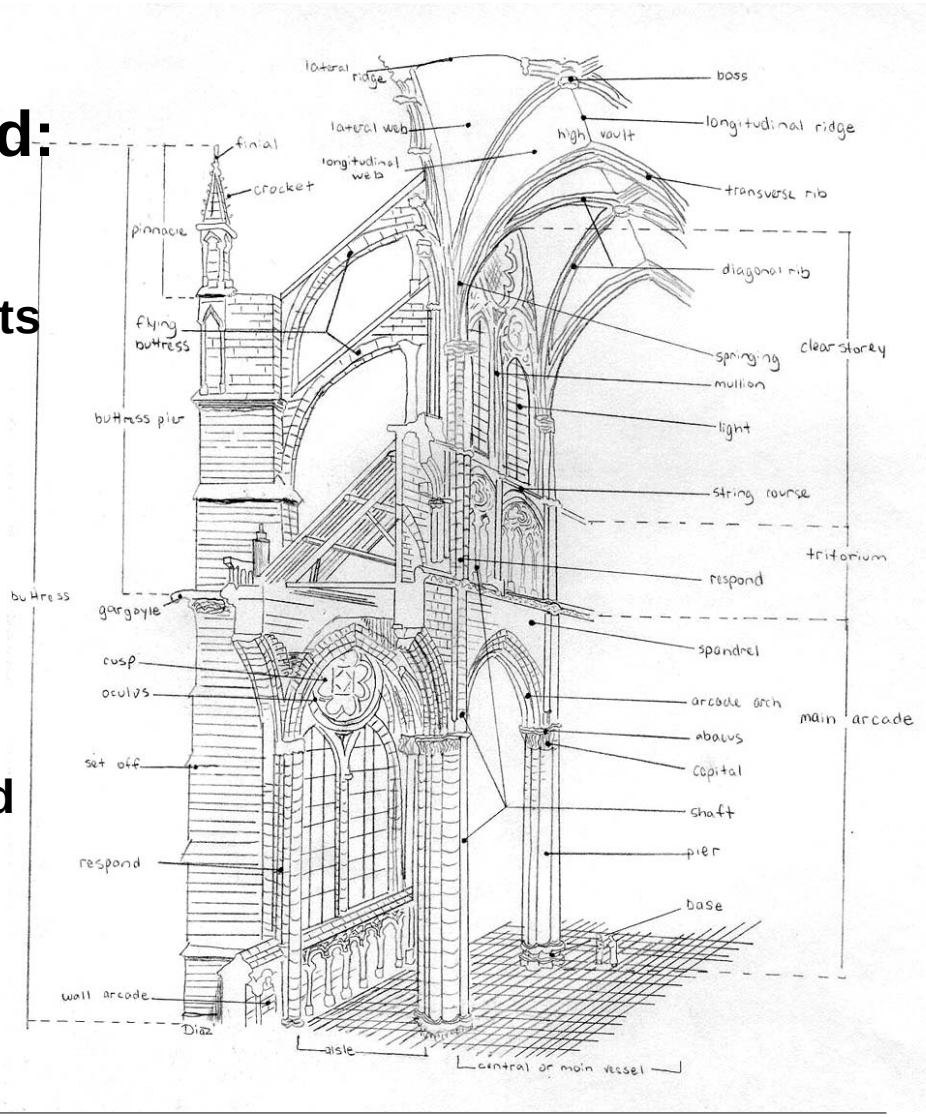
1. An abstract representation of some aspect of an existing or to-be-built system, component, or view. 2. A piece of information that is used or produced by a system development process, such as an external document or a work product.

Examples of individual artifacts are a graphical model, structured model, tabular data, and structured or unstructured narrative. Individual artifacts may be aggregated.

Systems Architecture vs. Design

- **Systems *Architecture* is used:**
  - **To make buy decisions**
  - **To discriminate between options**
  - **To “discover” the true requirements**
  - **To drive one or more systems to a common “use” or purpose**

- **Systems *Design* is used:**
  - **To develop system components**
  - **To build the system**
  - **To understand configuration changes as the system is modified**



Systems Architecting vs. Systems Engineering

- **System Architecting:**
  - **Inductive processes**
  - **Focuses on function**
    - **Deals with complex processes**
  - **Dealing with "unmeasurables"**
  - **Using non-quantitative tools and guidelines**
    - **Based on heuristics & lessons-learned**

- **Systems Engineering:**
  - **Deductive processes**
  - **Focuses on form**
    - **Derives structure by subdividing a system into progressively smaller elements which can be analyzed and built**
  - **Dealing with measurable quantities**
  - **Using analytical tools**
    - **Derived from mathematics and the hard sciences**

Heuristics (Inductive Reasoning)

- A "fuzzy" (imprecise) context-dependent technique well suited to "messy" (real) system architecture problems
  - A "rule of thumb"
- A generalization of specific experience to the general case for future applications
  - E.g., "Simplify, simplify, simplify" (within reason)
- It helps sort through the myriad of possibilities to identify best choices
- Coupled with quantitative decision methods for ranking alternatives forms the basic mental toolkit of a system architect

Rechtin and Maier's Thoughts

- **When you read Maier & Rechtin, experience is described as the number one teacher of how best to architect a system**

- **Their premise is the following:**

> **Success comes from wisdom**
> **Wisdom comes from experience**
> **Experience comes from mistakes**

- **The key is… *Heuristics***

"There are three kinds of mathematicians,

Those that know how to count and those that don't."

Heuristic Examples

- **Don’t assume that the original statement of the problem is necessarily the best, or even a right one**
- **In partitioning, choose the elements so that they are as *independent* as possible; that is, elements with low external complexity and high internal complexity**
- **Simplify. Simplify. Simplify. (“KISS”)**
- **Build in and maintain options *as long as possible* in the design and implementation of complex systems. You will need them later.**

Some Architecture Heuristics

- An insight is worth a thousand market surveys
- Success is defined by the customer, not the architect
- In architecting a new program, all the serious mistakes are made in the first day
- The most dangerous assumptions are the unstated ones
- The choice between products may well depend upon the set of drawbacks the users can handle best
- As time to delivery decreases, the threat to user utility increases
- If you think your design is perfect, it's only because you haven't shown it to someone else
- If you don't understand the existing system, you can't be sure you are building a better one

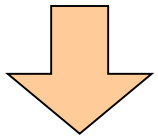
ARE HEURISTICS ALWAYS "TRUE"?

So in Conclusion…

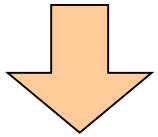
The art in architecting lies not in the wisdom of the heuristics, but in the wisdom of knowing which heuristics to apply, a priori, to the current project.

Architectural Context

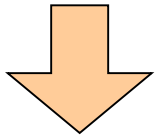
Needs



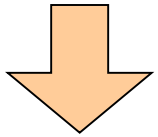
Operational Concept



Functional Capabilities



System Architecture



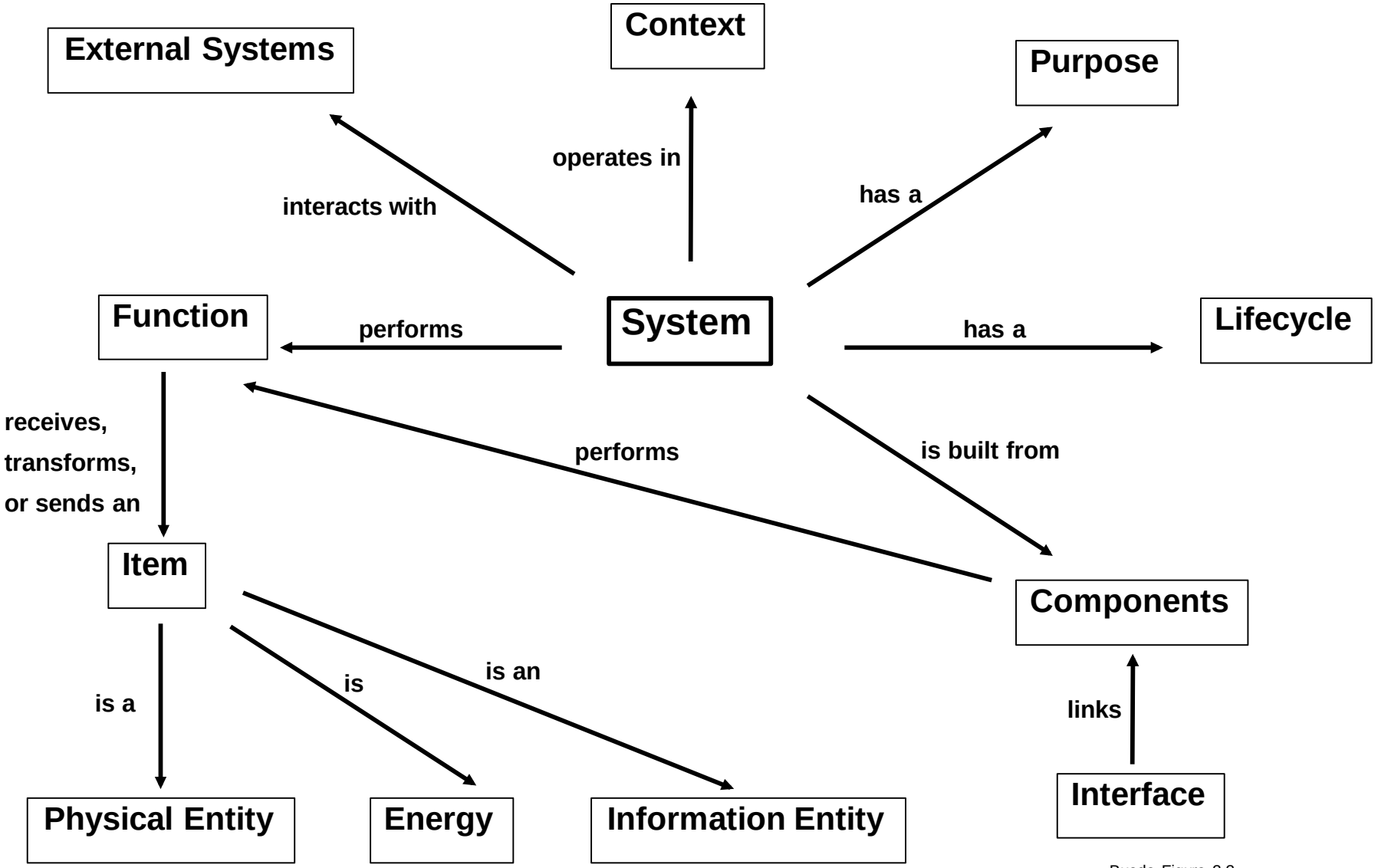
- What needs are we trying to fill?
- What is wrong with the current situation?
- Is the need clearly articulated?

- Who are the intended users?
- How will they use our products?
- How is this different from the present?

- What specific capability will we provide?
- To what level of detail?
- Are element interfaces well defined?

- What is the overall plan of attack?
- What elements make up the overall approach?
- Are these complete, logical, and consistent?

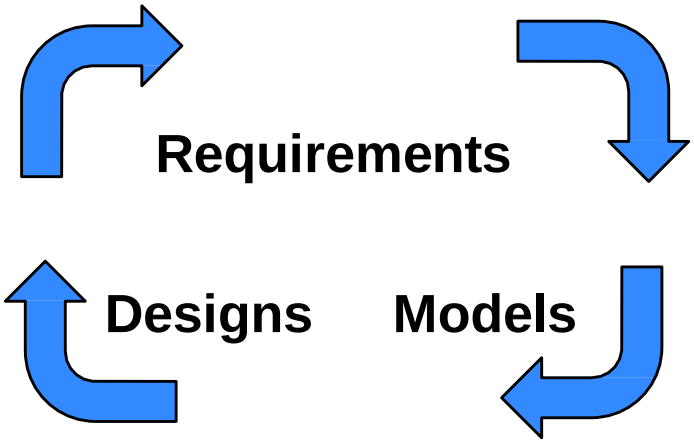
System Concepts



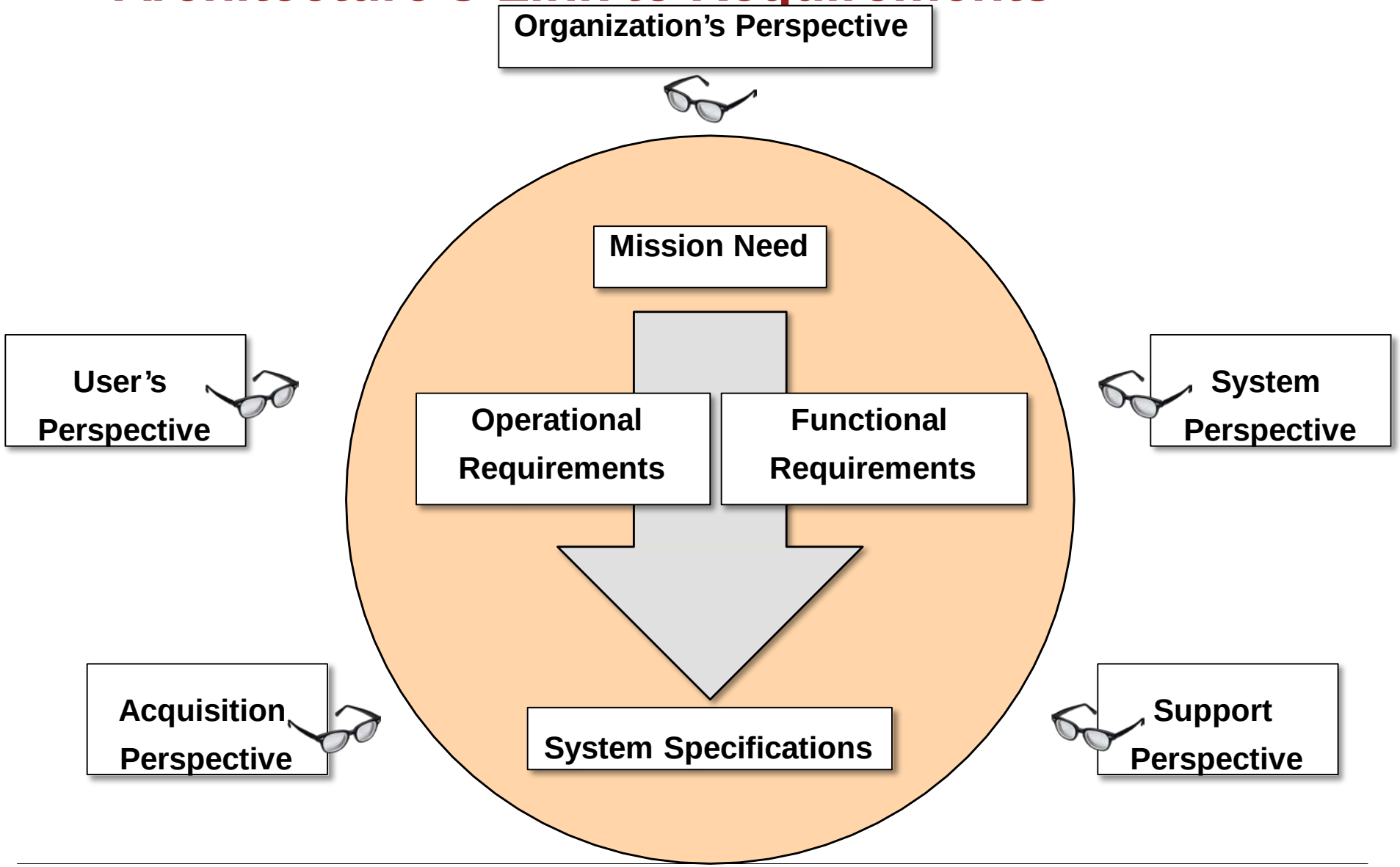
Buede Figure 2.2

System Development Elements

- **Requirements**
  - Used to specify essential functionality and performance of the system
  - Used to specify relevant constraints (such as cost, schedule, design, environment, production, etc.)
- **Models**
  - Used to analyze behavior, functional configuration, physical arrangement, and other attributes of the system
- **Designs**
  - Used to build or construct the system

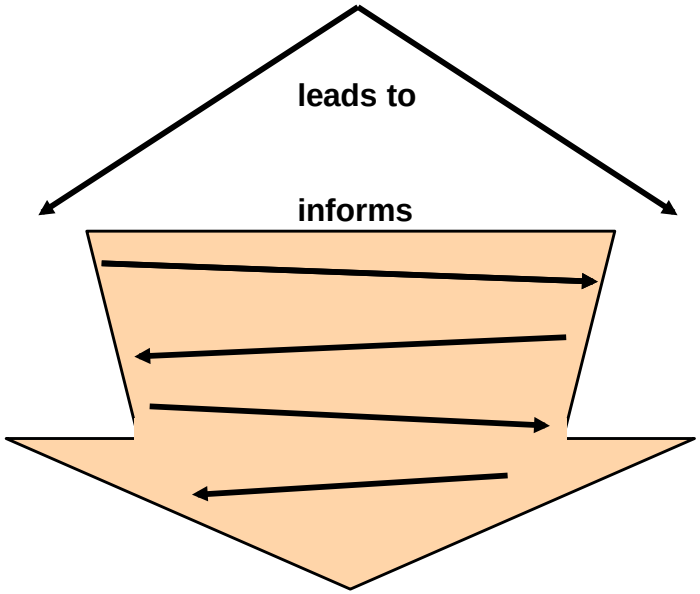


Architecture's Link to Requirements



Architecture and Design Process

Operational Concept



Functional (or logical) Architecture

Defines what the system must do (i.e. the system's functions and the data that flows between systems and subsystems).

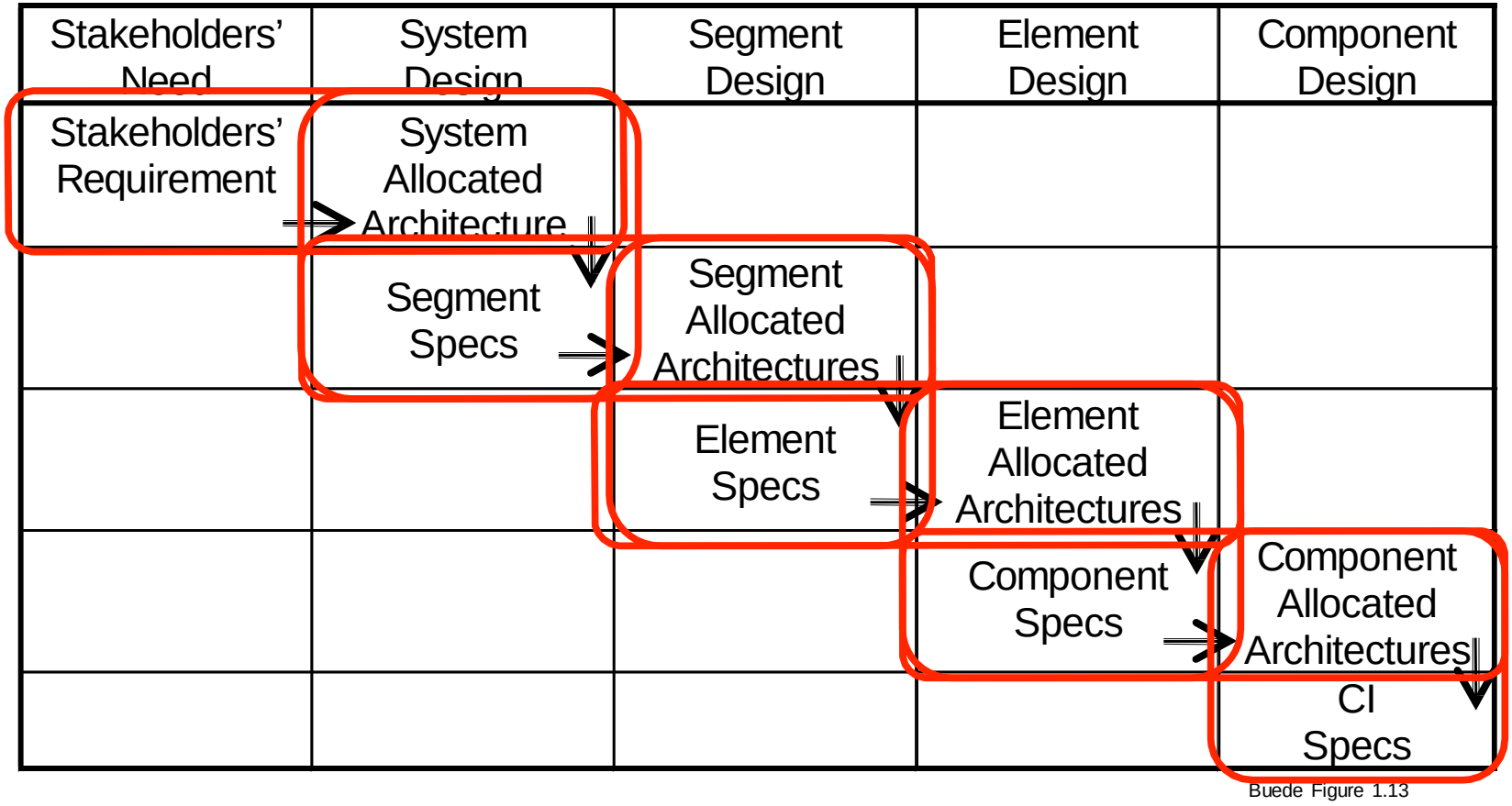
Allocated Architecture

Mapping of functions to resources to required system behavior

Physical Architecture

Represents the partitioning of physical resources available to perform the system's functions.

Architecture and Design Process



Buede Figure 1.13

Session 4 – System Architecture and Design

- Systems Architecture Context
- Architectural Levels and Views
- Architecture Frameworks

Lessons

1. **Program Overview Update**
2. **System Architecture Basics**
3. **Systems Design Process**
4. **Analysis of Alternatives**
5. **Model-Based Systems Engineering**
6. **Special Topics (if time permits)**

Appendices*

I. **SysML Diagrams and Language Concepts**

J. **Architecture Frameworks In-depth**

*Appendices A through H were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

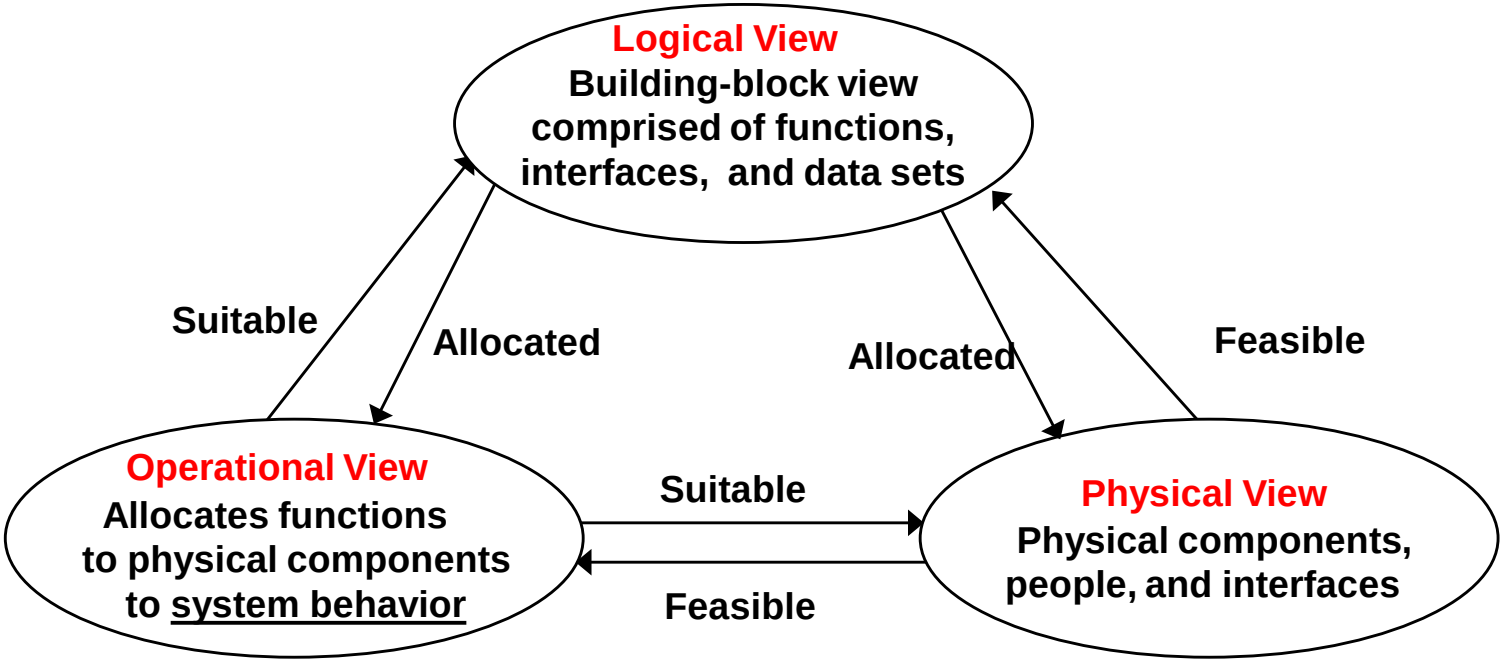
Architecture Levels

- **System Architectures**
  - **Identifies logical, physical and operational architecture views for System Components: segments, subsystems, HW items, SW items, users**
- **Hardware Architectures**
  - **Identifies logical, physical and operational architecture views for Hardware Components of a system, segment or subsystem (e.g. processor, bus, memory, storage devices, interface devices, communications interfaces, sensor devices, displays, etc.)**
- **Software Architectures**
  - **Identifies logical, physical and operational architecture views for Software Components(s) of a system, segment or subsystem (e.g. operating systems, networking, database services, common utilities and services, applications, user interfaces**

Levels versus Views

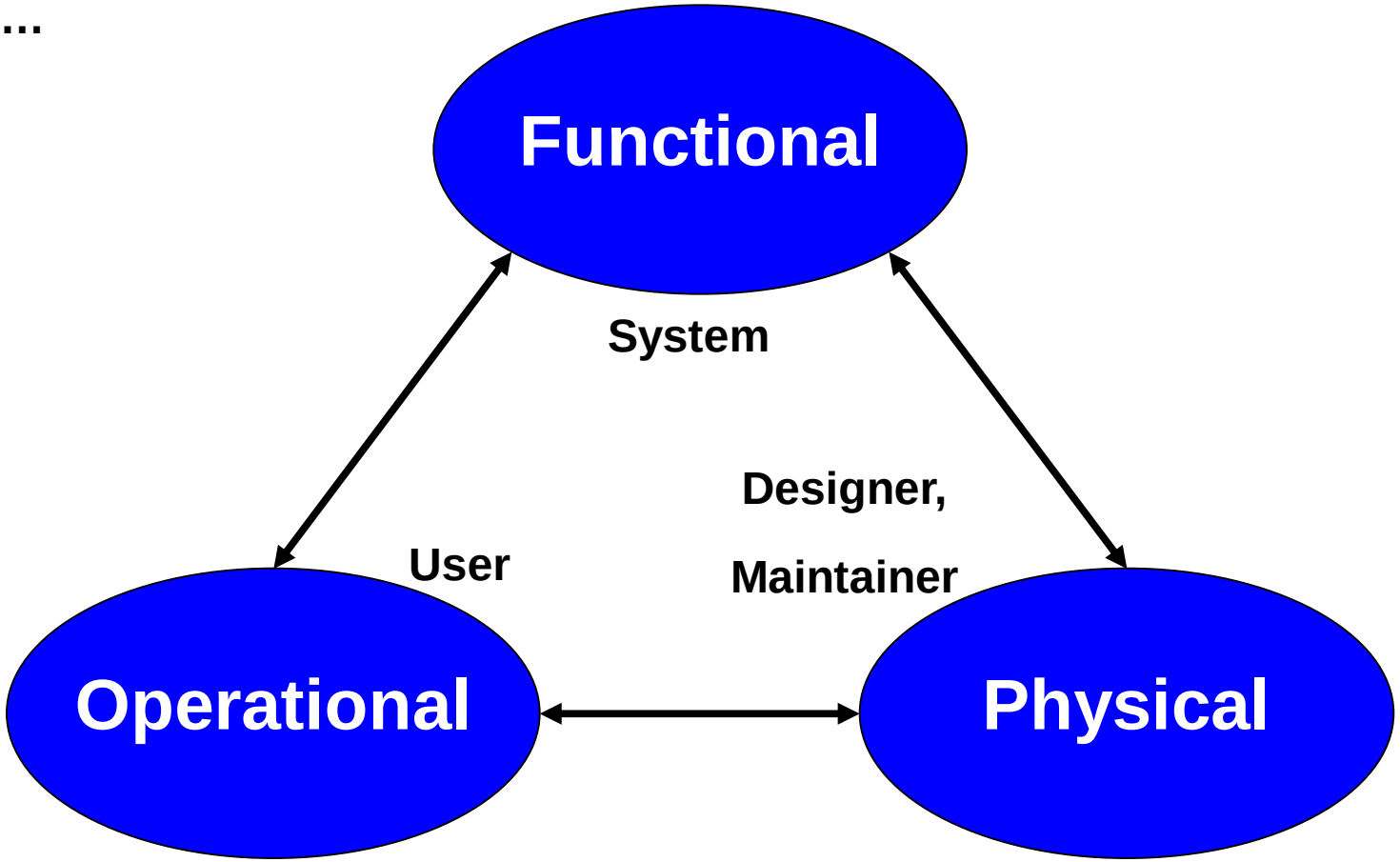
	Views		
Levels	**Logical**	**Physical**	**Operational**
System	▪ System Boundary ▪ System Functions ▪ System Interfaces ▪ System Data Sets	▪ Segment / Subsystem ▪ LAN, WAN Interface ▪ User Interface ▪ HW, SW, Database ▪ System Standards	▪ Operational Phases ▪ System Scenarios
Hardware	▪HW Product Boundary ▪Allocated HW Functions / Interfaces	▪ User Interface Device ▪ Computer, Processor, Bus ▪ Sensor, Memory, Storage ▪ Interface Cards, Cables ▪ Hardware Standards	▪ Detailed Scenarios, Task Flows (Use Cases)
Software	▪SW Product Boundary ▪Allocated SW Product Functions / Interfaces ▪ Logical Data Model	▪ User Interface ▪ Applications ▪ Common Utilities ▪ Data base Services ▪ Networking / Comm ▪ Operating System ▪ Software Standards	

Architectural Views



More Perspectives…

Buede…



Operational View

- **Operational Architecture View Identifies**
  - **Operational Phases, Scenarios, Task Flows (with Components, Allocated Functions, Interfaces, Data Flows)**

Phase A	Phase B	Phase C	Phase D	Phase E

Scenario:

Initial Conditions:

Stimuli:

Task Step A
Task Step B

Task Step C

.

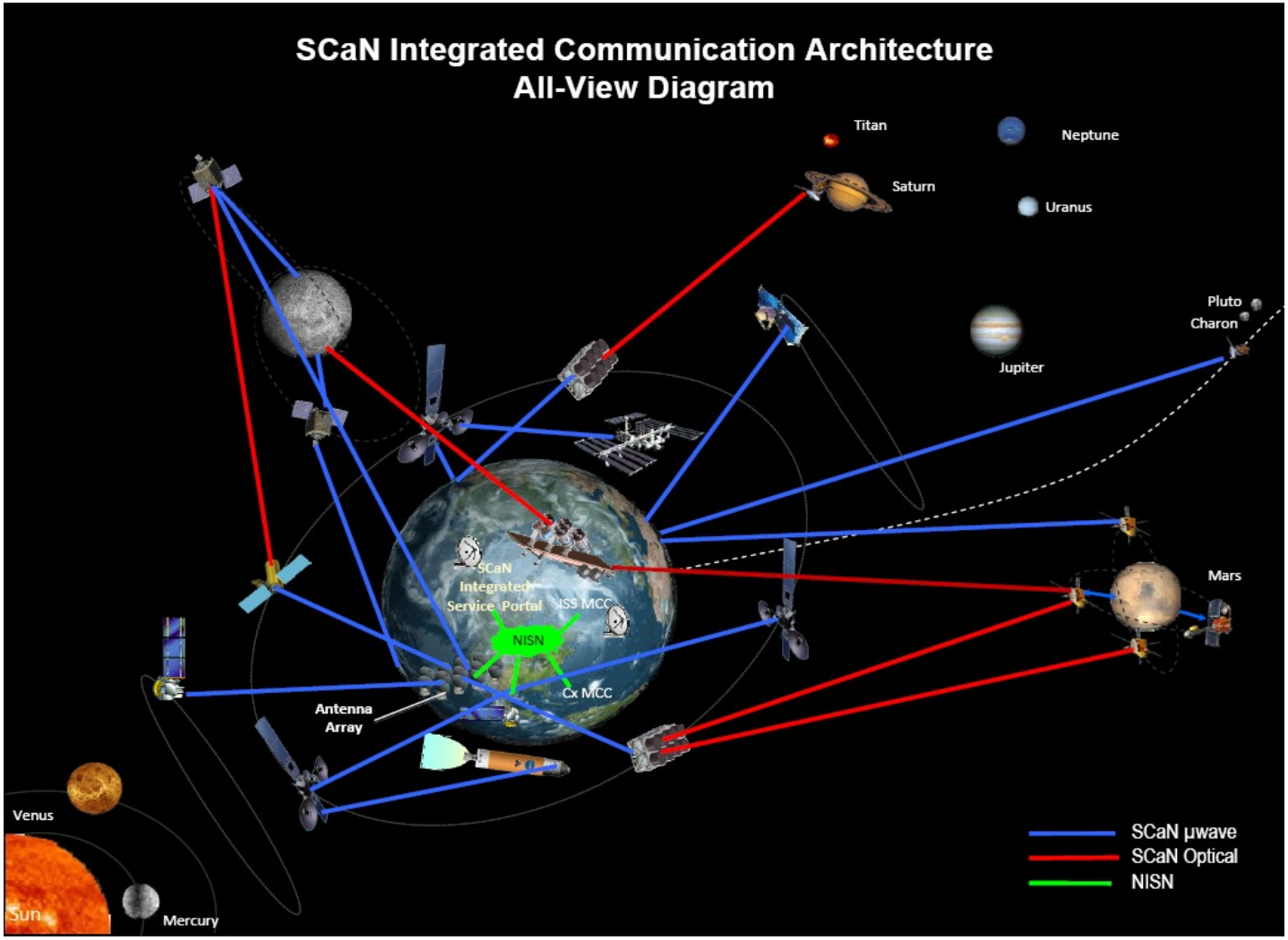
.

Response:

Operator Stimulus	Component 1	Component 2	Component 3	Component N
				Response

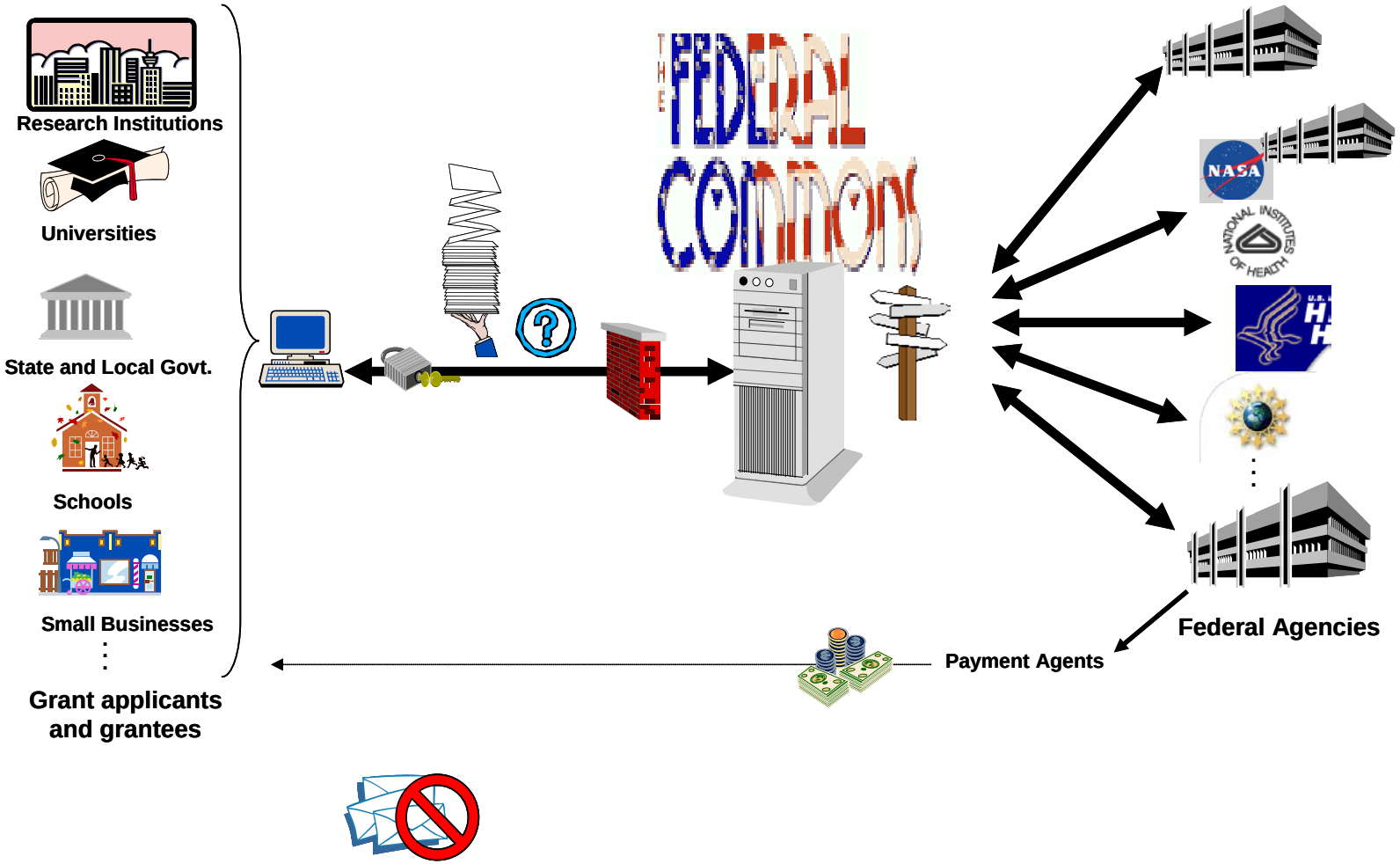
Operational View

Space Communications



Operational View

Operational Concept for Common E-Grants Management



High-level graphical description of the business concept described in the architecture.

Good Operational Architecture Attributes

- **Complete**
  - Addresses all operational system phases
  - Addresses all key operational scenarios
  - Defines task flows for each operational scenario
- **Traceable**
  - Components, functions, interfaces trace to scenario task flows
- **Clear, consistent, balanced**
  - Uses common phase, scenario naming throughout project
  - Uses function, component names from other architecture views
  - Follows 7 ± 2 phase, scenario rule of thumb at each level
  - Non-overlapping stimulus - response boundaries

Good Operational Architecture Attributes

Common Mistakes - Effects

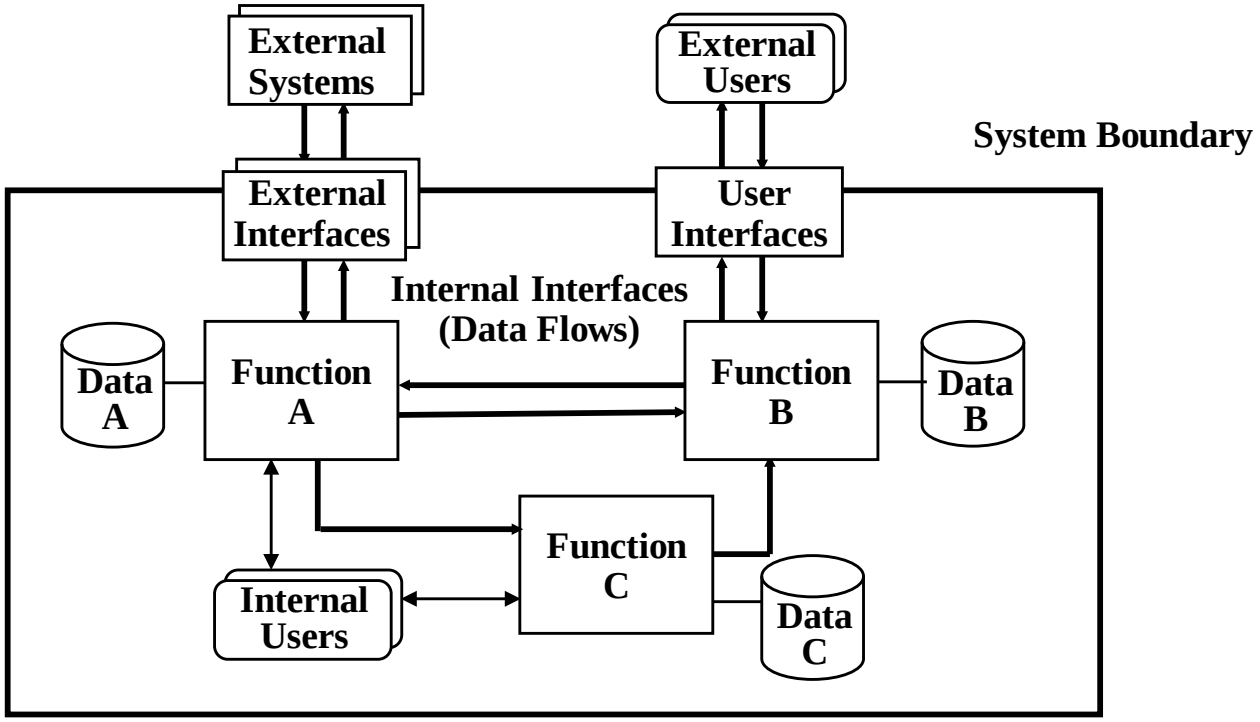
- **Incomplete - Costly changes late in development**
  - **BEWARE!!! - on some programs this view is not done at all!!!**
  - **Missing phases, scenarios (typically system administration, maintenance)**
  - **Task flows not consistent with standard customer practices**

- **Untraceable – Never ending change**
  - **Not clear how or if components and functions address scenarios**
  - **Test and integration scenarios difficult to write, many rewrites**

- **Unclear, inconsistent, unbalanced - Never ending change**
  - **Phase and scenario naming conventions same as functions**
  - **Mixture of contractor terminology and customer terminology**
  - **Poorly partitioned phases and scenarios (some very large, some small, overlapping or unclear boundaries)**

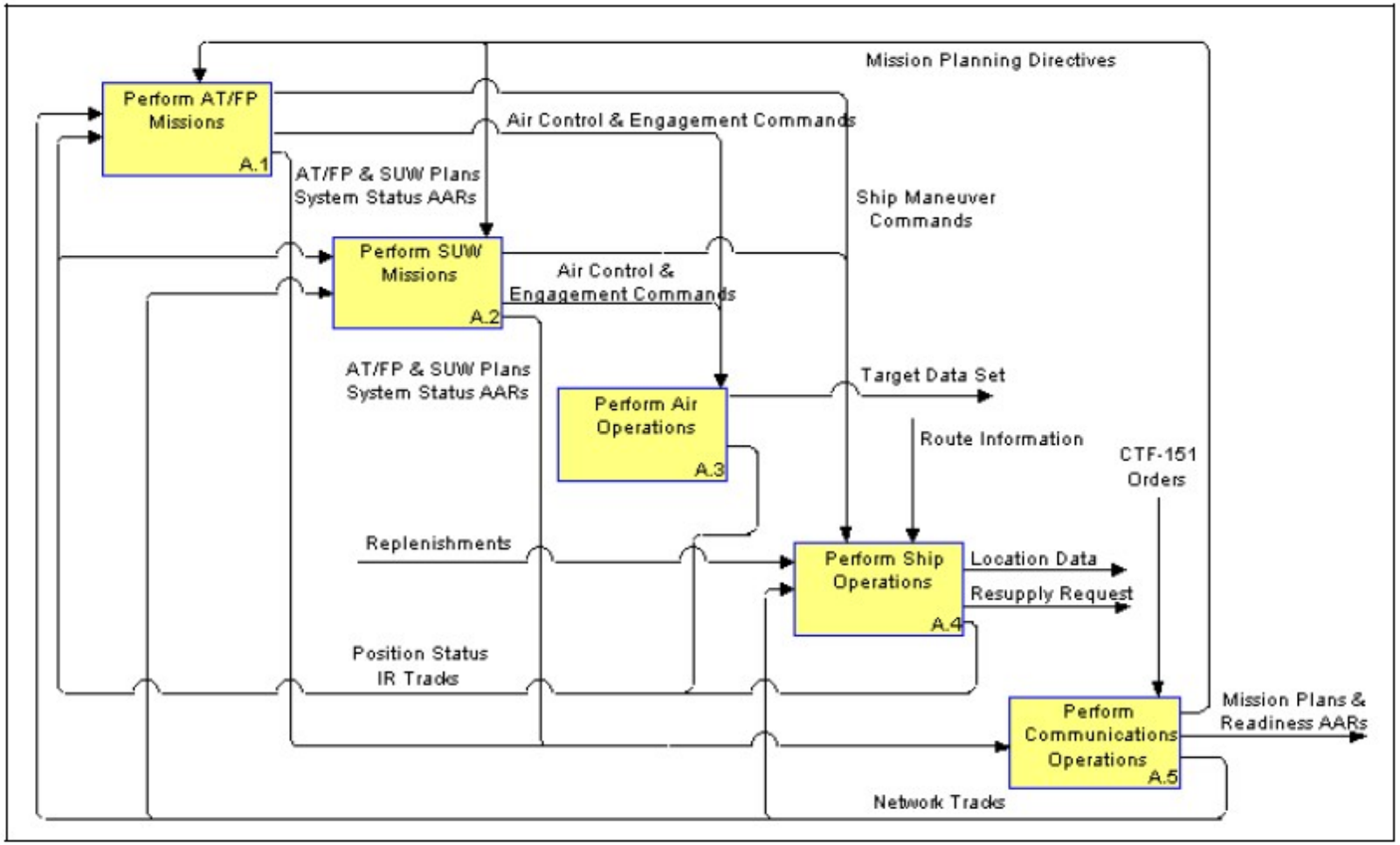
Logical (Functional) View

- **Logical Architecture View Identifies**
  - **System Boundary, External Systems, Interfaces and Users**
  - **Major Functions, Interfaces, Data Flows, Data Sets, Internal Users**



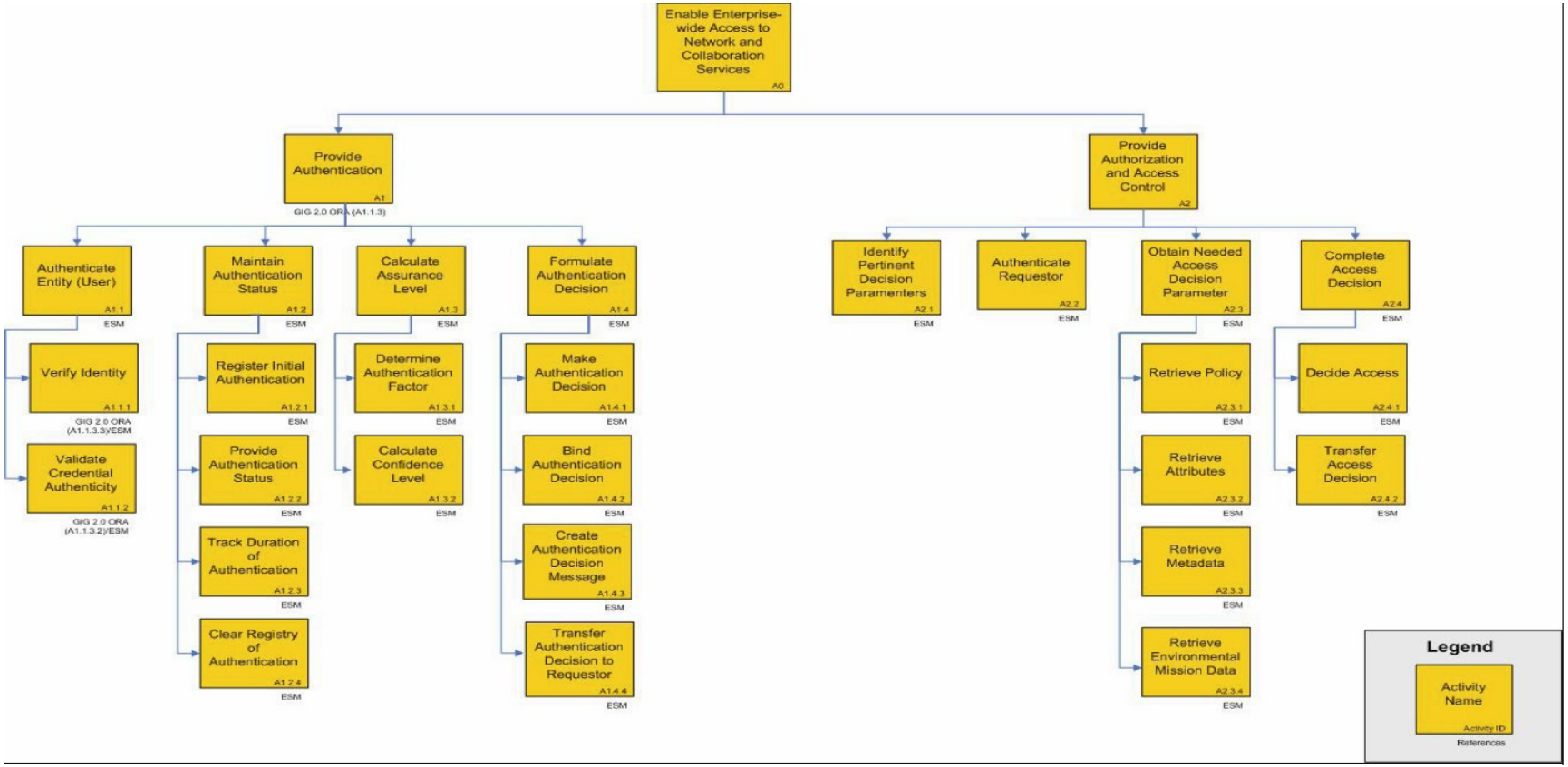
Logical (Functional) View

Navy Operations



Logical (Functional) View

Enterprise-wide Access to Network and Collaboration Services



Good Logical Architecture Attributes

- **Complete**
  - Addresses all driving system requirements
  - Identifies all relevant external systems, interfaces and users
  - Identifies all relevant internal functions, interfaces, data sets, users
- **Traceable**
  - Each function and interface traces to requirements
- **Clear, Consistent, Balanced**
  - Uses “action verb-object noun” function syntax
  - Functions have well defined inputs, outputs, sub-functions
  - Follows 7 ± 2 functions rule of thumb at each level
  - Uses consistent level of functional abstraction at each level

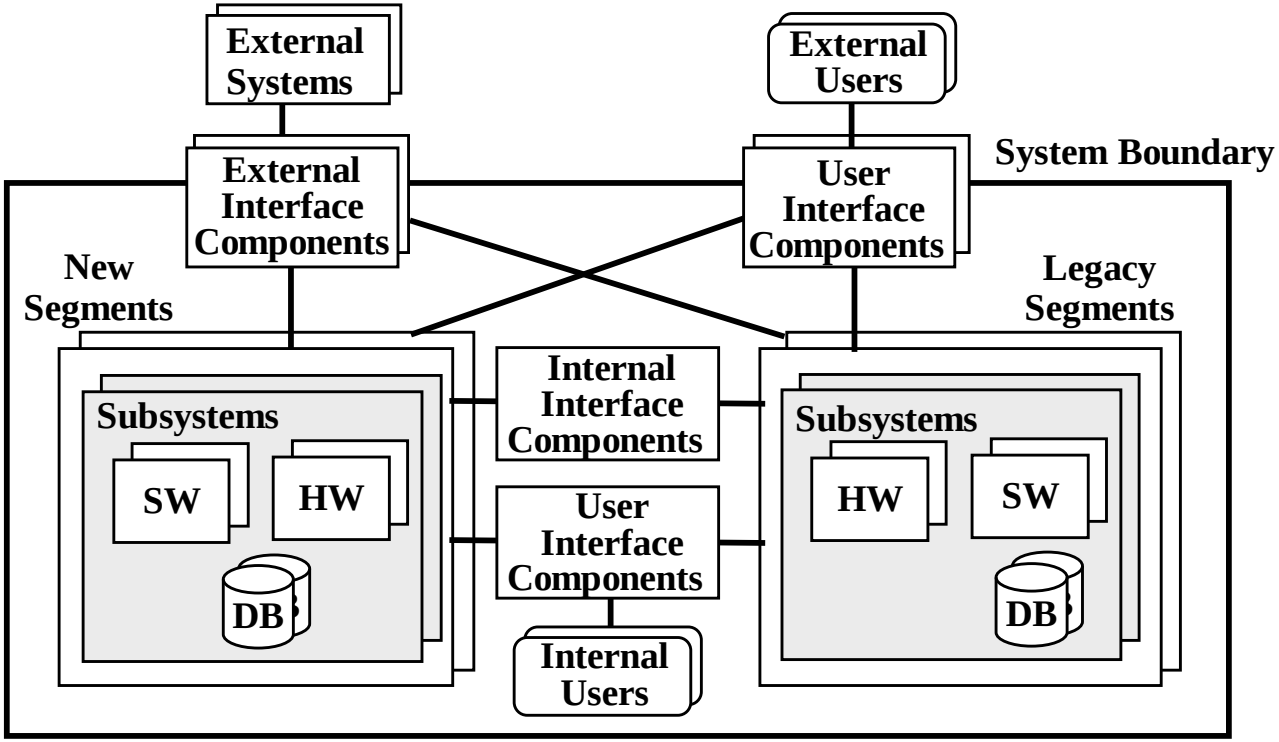
Good Logical Architecture Attributes

Common Mistakes - Effects

- **Incomplete – Missing design components, capabilities**
  - **Missing system requirements (e.g. system loads, throughput, response times, reliability, maintainability, failure modes, etc.)**
  - **Undefined interfaces, missing user groups (maintenance, external)**
- **Untraceable – Unnecessary cost, complexity**
  - **Functions without requirements, requirements without functions**
- **Unclear, inconsistent, unbalanced – Never ending change**
  - **Poorly defined functions, systems and interfaces used as functions**
  - **10+ functions at the same level**
  - **Primitive functions at high level (because we know what they are)**

Physical View

- **Physical Architecture View Identifies**
  - **System Boundary, Physical Components, Users, Standards**
  - **Physical Interfaces, Databases**



Good Physical Architectures Attributes

- **Complete**
  - Addresses all driving system requirements
  - Identifies all relevant external systems, interfaces and users
  - Identifies all components, interfaces, databases, internal users
  - Identifies all relevant standards
- **Traceable**
  - Each component traces to allocated system, derived requirements
- **Clear, consistent, balanced**
  - Uses common component naming throughout project
  - Preserves customer component terminology wherever feasible
  - Follows 7 ± 2 components rule of thumb at each level

Good Physical Architecture Attributes

Common Mistakes - Effects

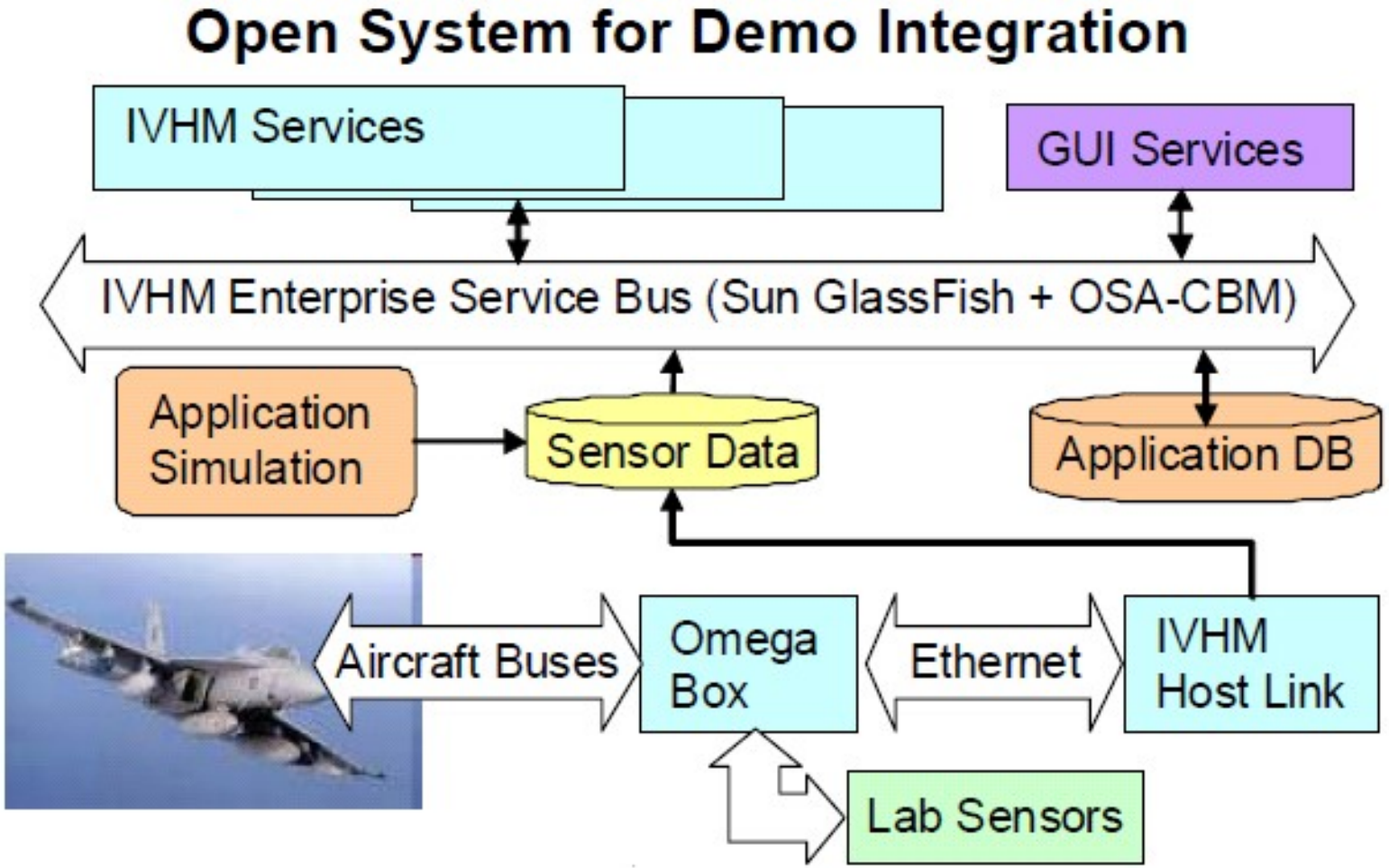
- **Incomplete - Costly changes late in development**
  - **Missing components (typically SW), interfaces, databases, users**
  - **Relevant standards not defined, adhered to**
  - **Does not adequately address Technical Performance Measures (e.g. system throughput, response times, failure modes, reliability, maintainability, etc)**

- **Untraceable - Unnecessary cost, complexity**
  - **Components without requirements or required functions (was relevant on last program, agreed to do this for the customer)**

- **Unclear, inconsistent, unbalanced – never ending change**
  - **Mixed component naming conventions causes confusion**
  - **Mixture of contractor terminology and customer terminology**
  - **Segments poorly partitioned (some are huge, some are small)**

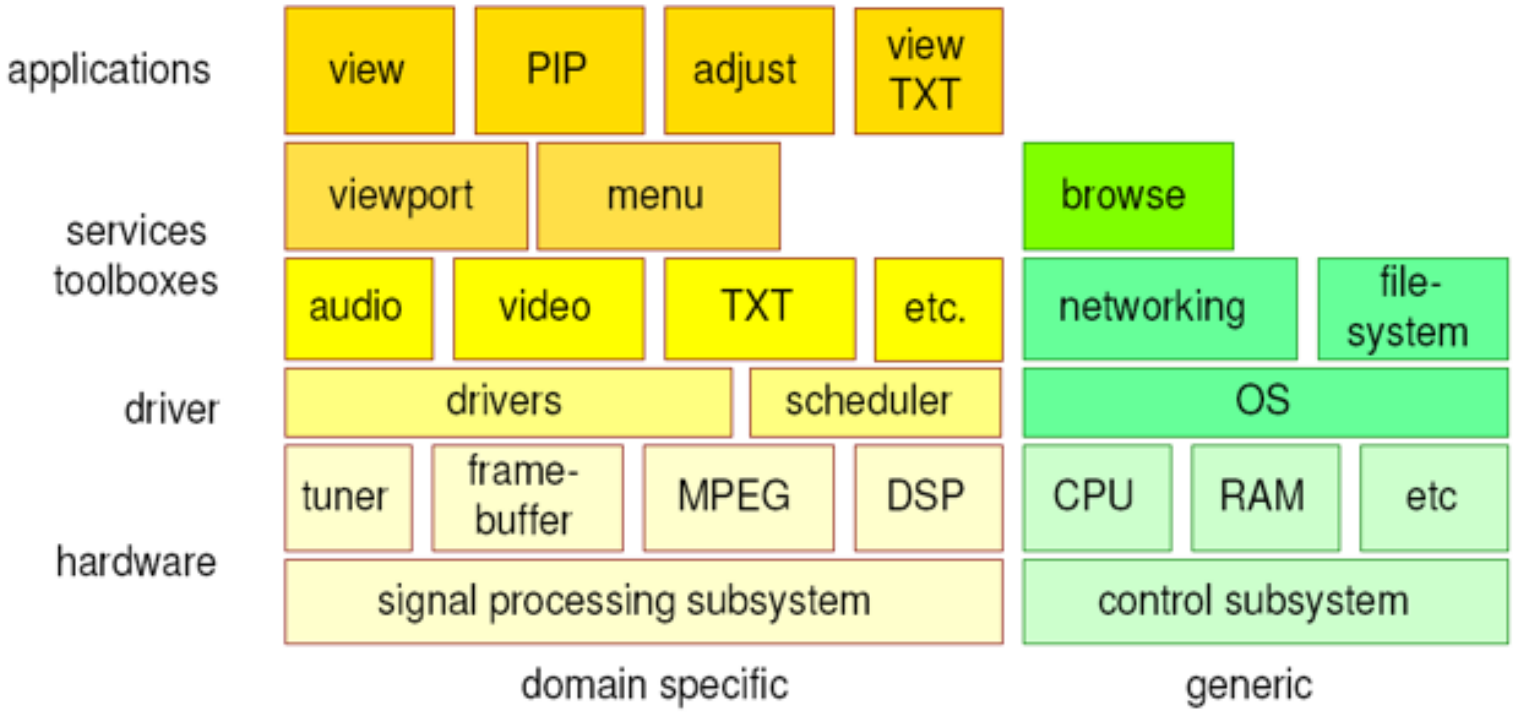
Physical View

NASA's Integrated Vehicle Health Management



Physical View

Decomposition for Simple TV



Enterprise versus System

- System
  - A construct or collection of different elements that together produce results not obtainable by the elements alone
- System of Systems
  - A construct or collection of different systems that together produce results not obtainable by the systems alone
- Enterprise
  - The whole collection of system of systems including all the people, information, technology, business operations.

Enterprise Architecture

- Explicit description of the current and desired relationships among the business and management processes and information technology

- "Enterprise architecture identifies the main components of an organization and how components in the organization's nervous system function together to achieve defined business objectives."
  - "These components include personnel, business processes, technology, financial information and other resources."
    - "Practical Guide to Enterprise Architecture" McGovern et al

Good Enterprise Architecture Attributes

- Addresses the organization's business and information needs
- Leverages the synergistic relationship between return on investment (ROI) and the total cost of ownership (TCO)
- Supports the ability to easily migrate from the current (as-is) state to the organization's desired future state (to-be) state
- Supports business objectives of reducing costs, improving the operational service, and increasing revenue

"Practical Guide to Enterprise Architecture" by McGovern *et al*

IEEE 1471

"Recommended Practice for Architectural Description for Software-Intensive Systems"

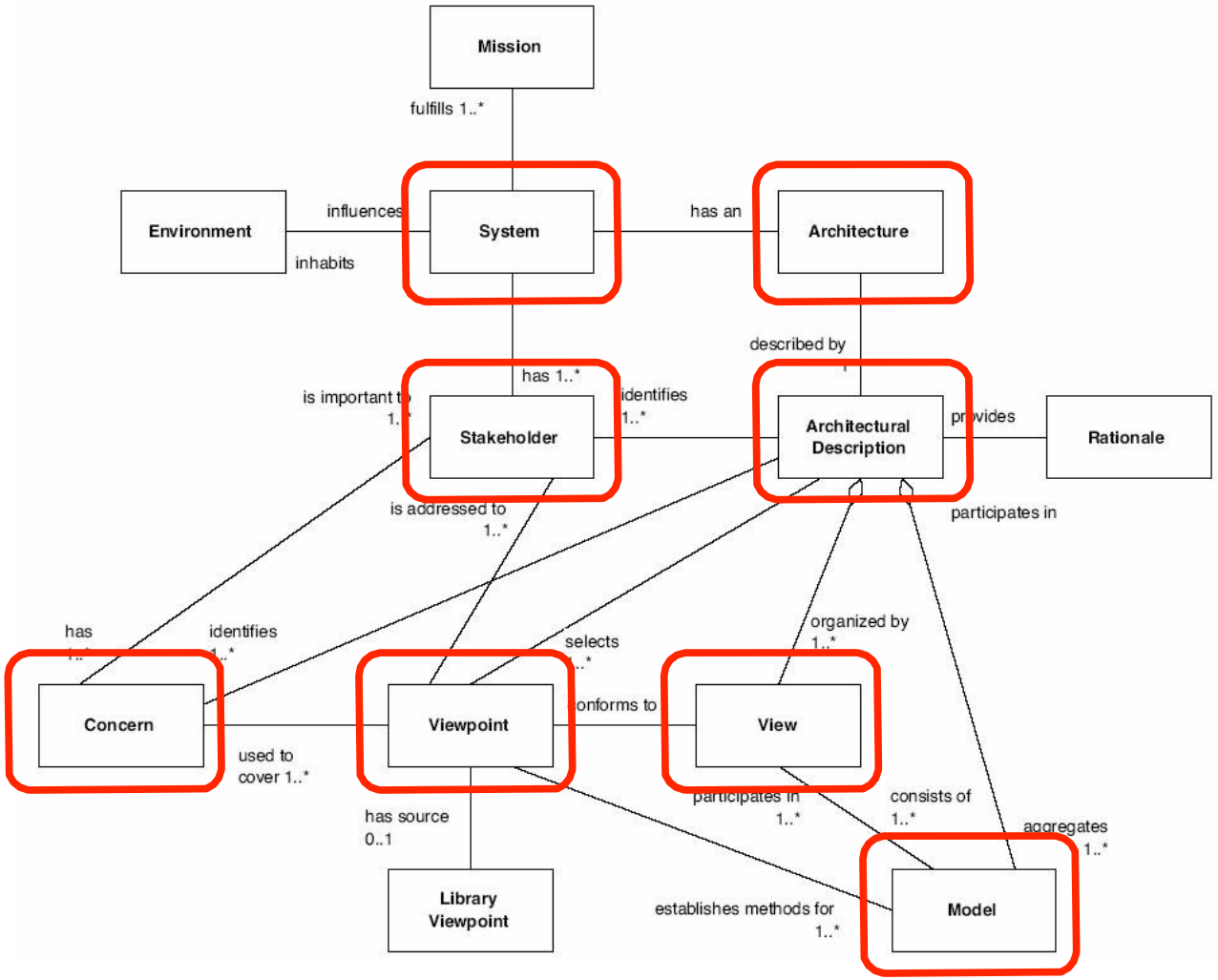


Image from http://arnon.me/2010/05/architectural-modeling-step/

Executable Architecture

- Architecture connected to simulation software
  - Model-Based Systems Engineering (MBSE)

- Executable architectures allow us to…
  - Verify the logic of the operational activities and the system functions
  - Estimate timing concerns
  - Estimate impact on resources
  - Conduct trade studies on different architectures

Why do we need architecture?

Let me count the ways…

- Provide Multiple Perspectives
- Identify Stakeholders
- Define Boundaries
- Estimate Efficiency
- Understand and Define Information
- Estimate Performance
- Identify Subsystems
- Map Execution
- Plan Integration
- Provide Measurement
- Refine Requirements



Summary - Why an architecture?

- Have technical consistency across programs and systems
- Establish and adhere to standards
- Use common language
- Communicate a common vision
- Manage risk
- Analyze tradeoffs
- Educate and train team
- Document decisions and design motivation

OnStar System Exercise:
CONOPS and Operational View



OnStar System Exercise:
CONOPS and Operational View

- **Develop several (2 – 3) operational scenarios**
  - **Keep them as solution-independent as possible**
  - **Include at least one good operational scenario, and one good support scenario**
- **Develop an Operational View patterned after one of the examples shown earlier in this lesson which illustrates your operational scenarios**
- **Be prepared to share your results with the class, including:**
  - **One of your scenarios,**
  - **How your operational view illustrates that scenario, and**
  - **Your observations for the exercise**



"Operational" must be interpreted as referring to support as well as operations.

Session 4 – System Architecture and Design

- Systems Architecture Context
- Architectural Levels and Views
- Architecture Frameworks

Lessons

1. **Program Overview Update**
2. **System Architecture Basics**
3. **Systems Design Process**
4. **Analysis of Alternatives**
5. **Model-Based Systems Engineering**
6. **Special Topics (if time permits)**

Appendices*

I. **SysML Diagrams and Language Concepts**

J. **Architecture Frameworks In-depth**

*Appendices A through H were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

Framework Goals

- Codify best practices for architectural description and therefore improve the state of the practice
- Ensure the sponsors of the framework receive architectural information in the format they desire
- Facilitate comparative evaluation of architectures through standardization of their means of description
- Improve the productivity of the development team by presenting basic designs in a standard way
- Improve interoperability of information systems by requiring that interoperation critical elements be described, and be described in a common way

What should an Architectural Framework include?

1. A purpose and audience for which the compliant architecture description is to be written

2. A set of viewpoints that when used should satisfy the purpose of #1

3. The normative requirement that a compliant architecture description provide a set of views of the architecture of the system using the required viewpoints

4. Tests for consistency and completeness among the views produced

5. As a practical matter, an architecture framework may contain other advisory information, like guidelines on its use

Architectural Framework Heuristics

- Focus on architecture decisions and let descriptions flow from the decisions
  - Ask yourself "What decisions about developing our system are contained in this document?"
- Always use an iterative process to do architecting
  - You can never get it done right the first time, so don't try
  - Easy to say "just get the problem right at the beginning" but don't really know if the problem is "right" until we've looked at several solutions
- Do not overstaff early – best architectures come from small teams
  - The decision-centric and strategic work gets swamped by visible document production
  - Mired in "as-is" data collection (this can expand to use all available funds)
  - Large teams may avoid hard choices and complex decisions

Framework Examples

- Department of Defense Architecture Framework (DODAF)
- Ministry of Defence Architecture Framework (MODAF)
- NATO Architecture Framework (NAF)
- The Open Group Architecture Framework (TOGAF)
- Zachman Framework
- Federal Enterprise Architecture Framework (FEAF)
- Reference Model for Open Distributed Processing (RM-ODP)
- ISO/IEC/IEEE 42010 “Systems and software engineering — Architecture description”
  - Formerly IEEE 1471 “Recommended Practice for Architecture Description of Software-Intensive Systems”

Detailed descriptions of these frameworks can be found in Appendix J.

Summary

Architecture Frameworks should…

- Codify best practices
- Ensure correct format for decision makers
- Facilitate architecture comparisons
- Improve development team productivity
- Improve system of system interoperability

System Architectures should be used to…

- Make decisions
- Communicate ideas
- Guide development

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		

All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?



Systems Engineering Technical Short Course

(Session 4: System Architecture and Design)

Lesson 3:
Design Processes

Session 4 – System Architecture and Design

Lessons

1. **Program Overview Update**
2. **System Architecture Basics**
3. **Systems Design Process**
4. **Analysis of Alternatives**
5. **Model-Based Systems Engineering**
6. **Special Topics (if time permits)**

- NASA Design Process
- Generic Design Process



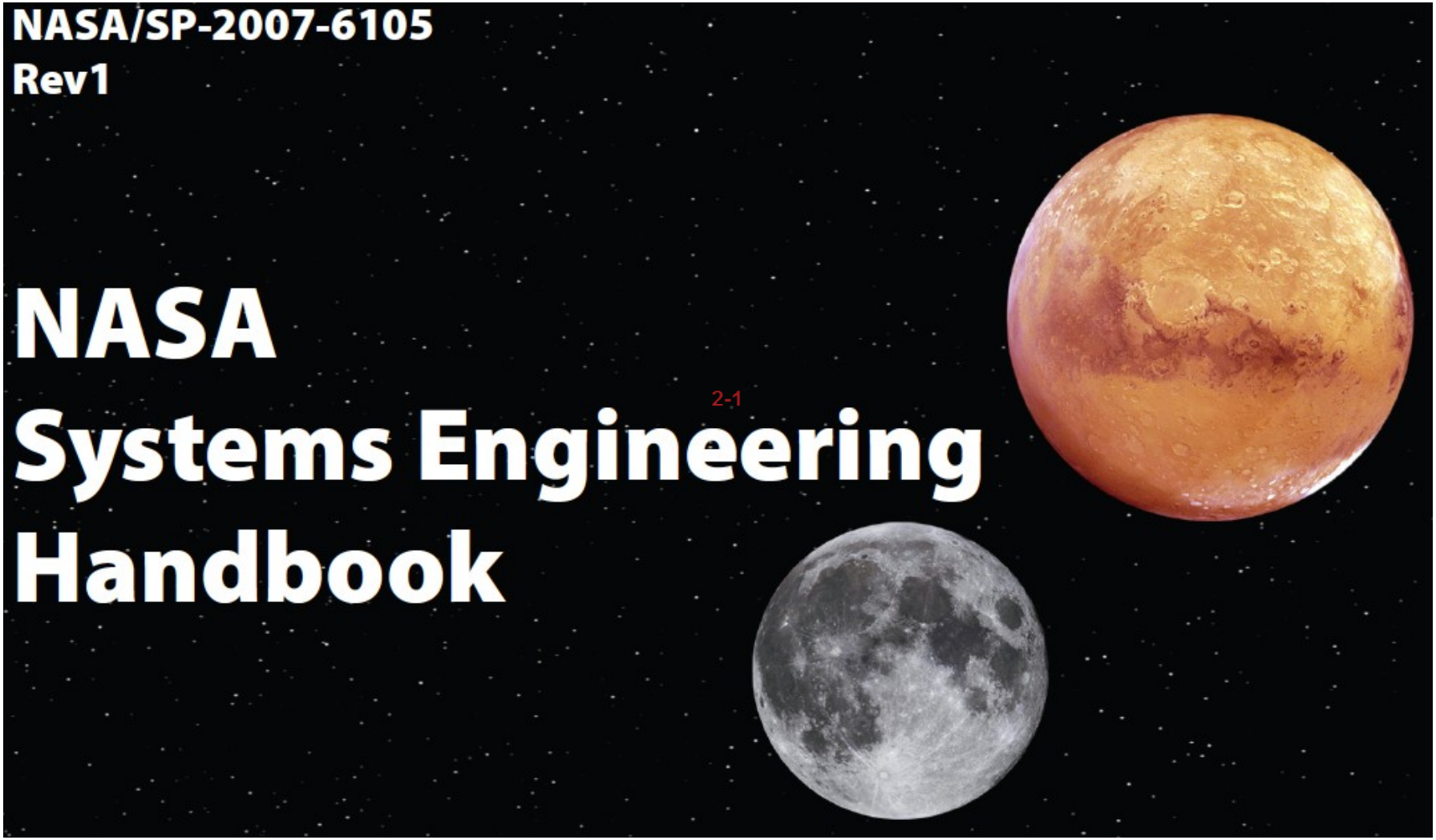
Appendices*

I. **SysML Diagrams and Language Concepts**

J. **Architecture Frameworks In-depth**

*Appendices A through H were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

NASA Design Process



http://www.acq.osd.mil/se/docs/NASA-SP-2007-6105-Rev-1-Final-31Dec2007.pdf

System Design Keys

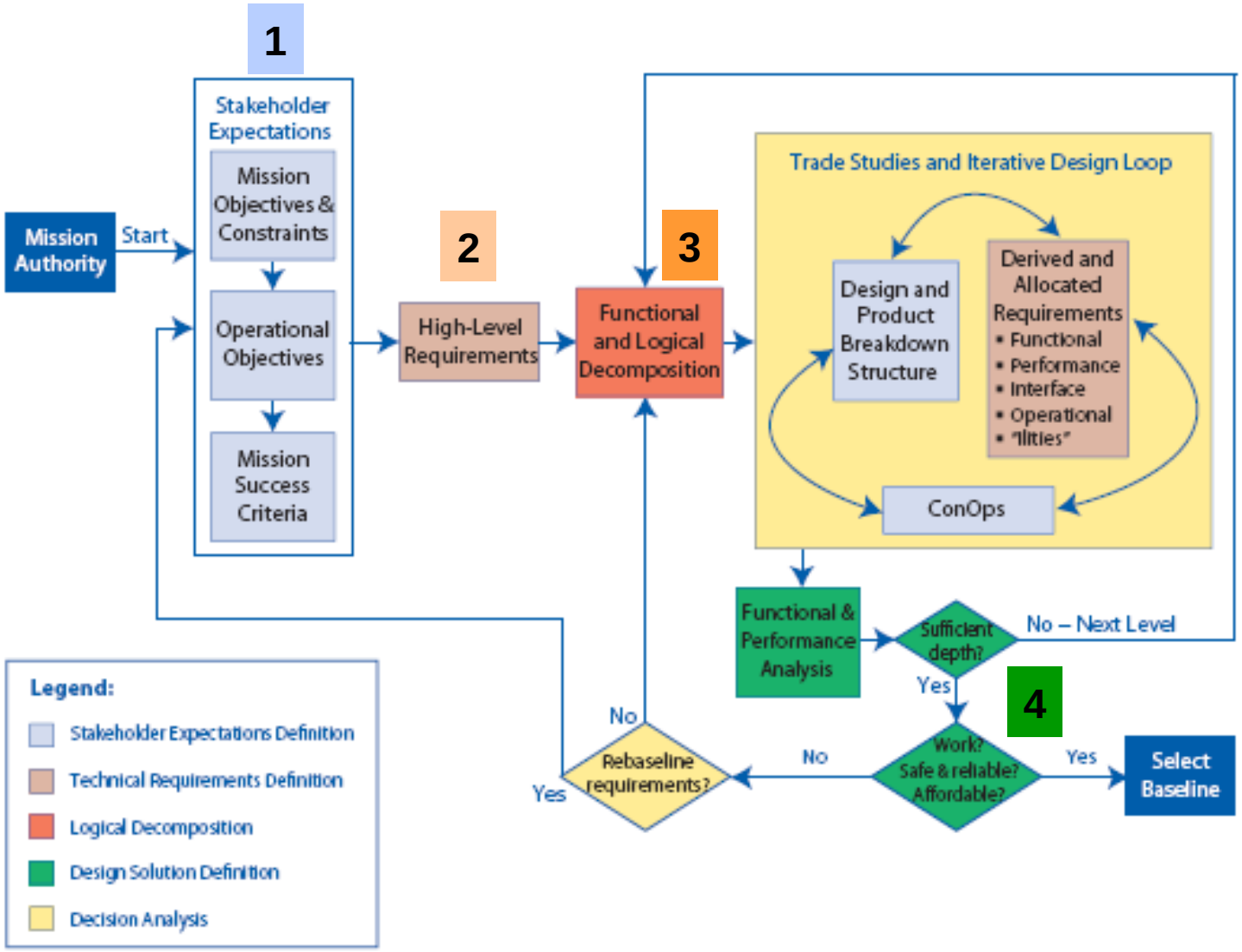
- Successfully understanding and defining the mission objectives and operational concepts
- Complete and thorough requirements traceability
- Clear and unambiguous requirements
- Documentation of all decisions made during the development of the original design concept in the technical data package
- Validation of a design solution through a continuing recursive and iterative process, during which the design solution is evaluated against stakeholder expectations

NASA's System Design Process

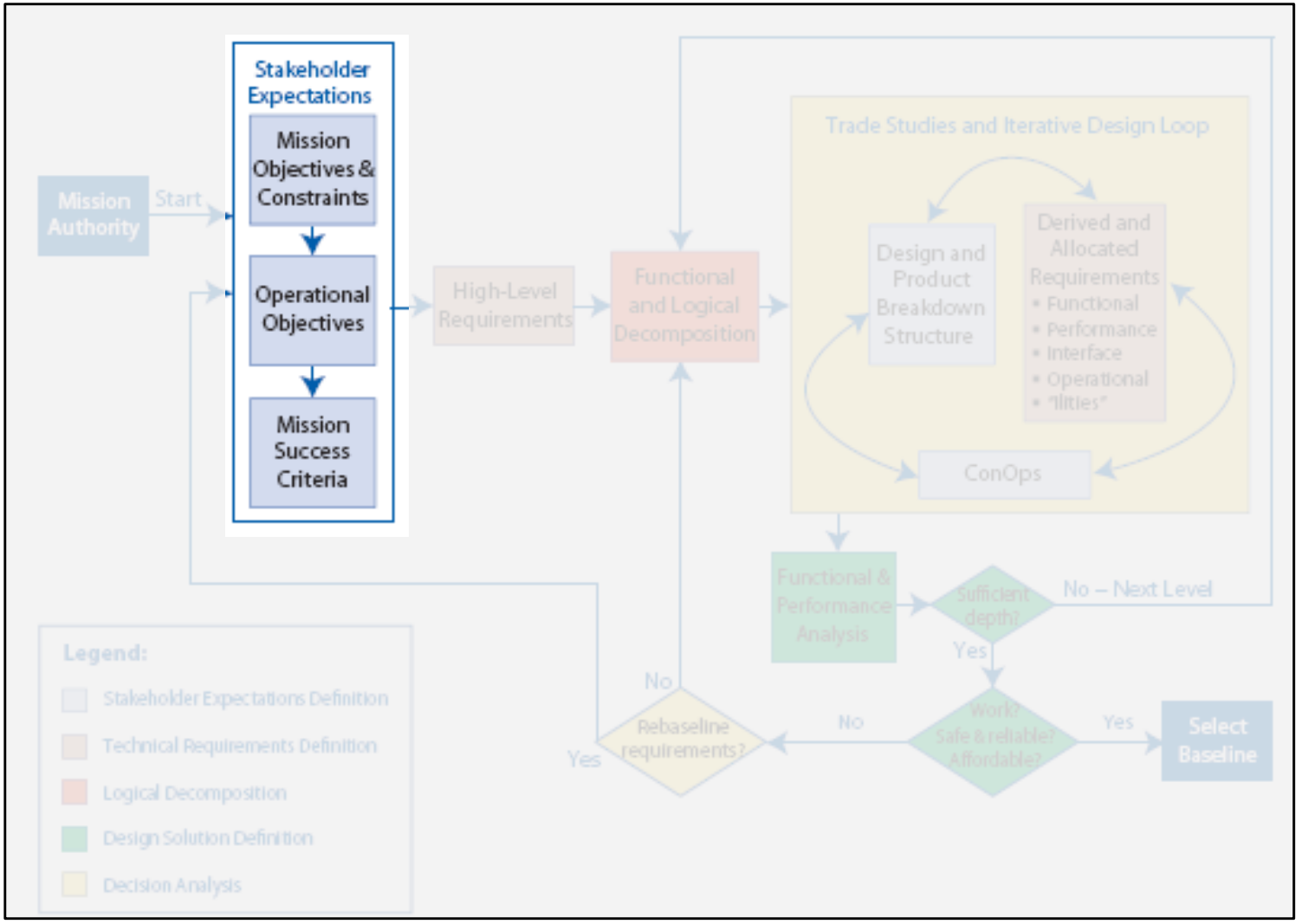
- **NASA's system design process has four interdependent, highly iterative, and recursive sub-processes, resulting in a validated set of requirements and a validated design solution that satisfies a set of stakeholder expectations, namely:**

1. **Stakeholder Expectations Definition (Problem Definition)**
2. **Technical Requirements Definition**
3. **Logical (Functional) Decomposition**
4. **Design Solution Definition**

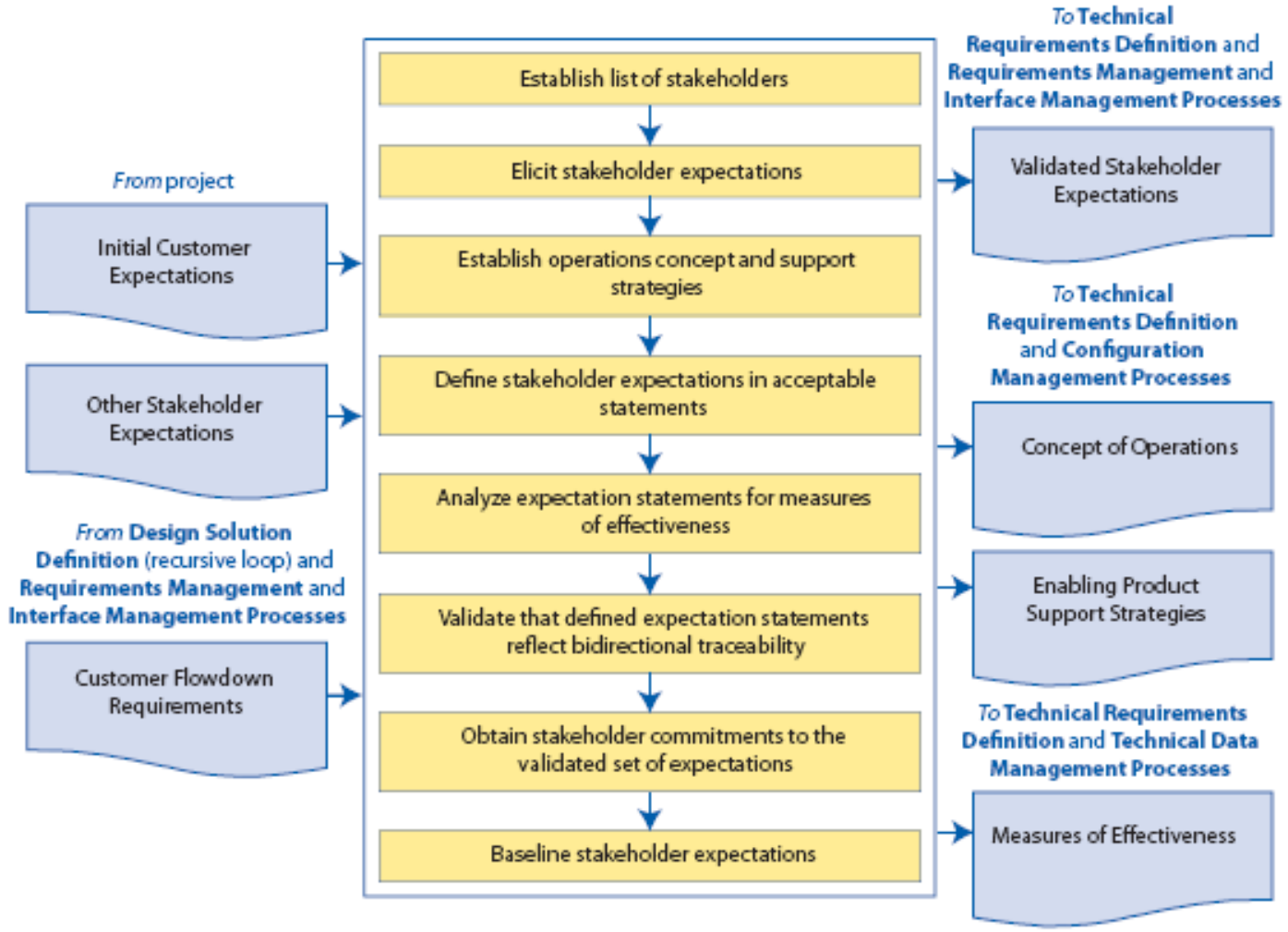
The System Design Process



Stakeholder Expectations



Stakeholder Expectations Process



Stakeholder Expectations

Inputs

- **Upper-level requirements and expectations**
  - **The needs, wants, desires, capabilities, constraints, external interfaces) that are being flowed down to a particular system of interest from a higher level (e.g., program, project, etc.)**
- **Identified customers and stakeholders**
  - **The organization(s) or individual(s) who have requested the system and those who are affected by or are in some way accountable for the system's outcome**

Stakeholder Expectations

Process Activities

- **Identifying stakeholders**
- **Identifying stakeholder expectations**

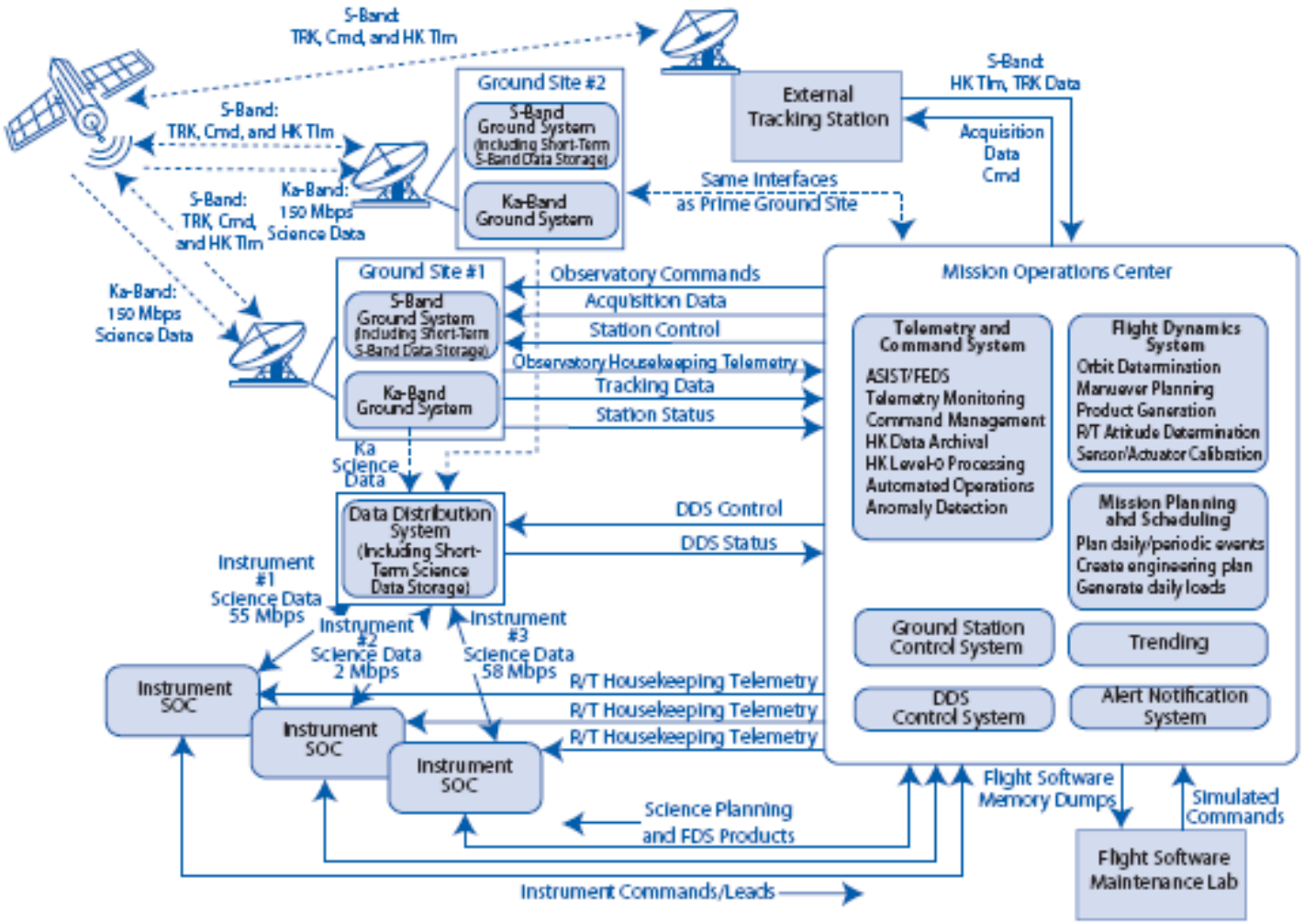
Stakeholder Expectations

Outputs

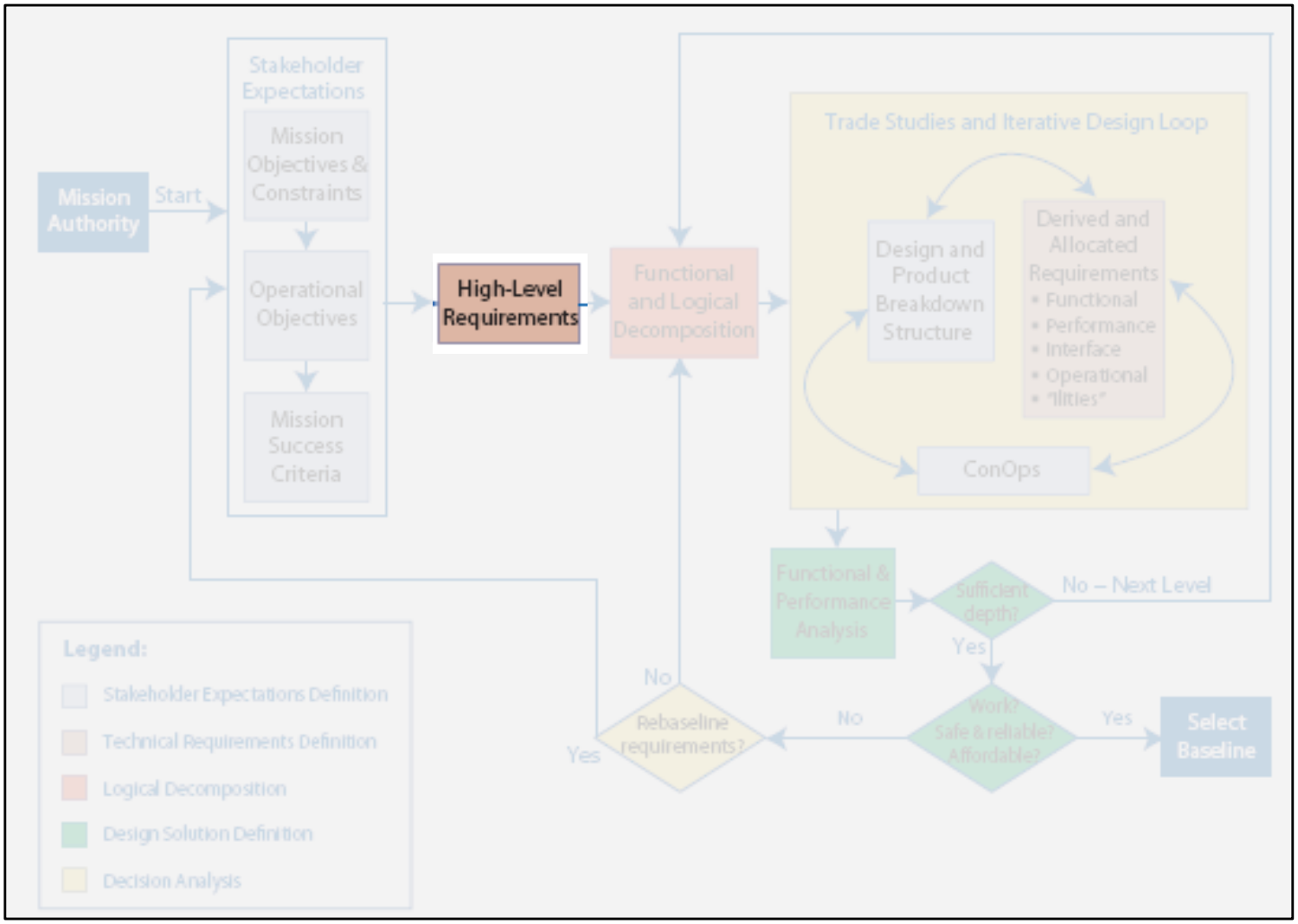
- **Top-Level Requirements and Expectations**
  - **These are the top-level requirements and expectations (e.g., needs, wants, desires, capabilities, constraints, and external interfaces) for the product(s) to be developed**

- **Concept of Operations (ConOps) –> Operational Architecture**
  - **This describes how the system will be operated during the life-cycle phases to meet stakeholder expectations**
  - **It describes the system characteristics from an operational perspective and helps facilitate an understanding of the system goals**
  - **Often expressed as an end-to-end operational architecture (DoDAF OV-1)**

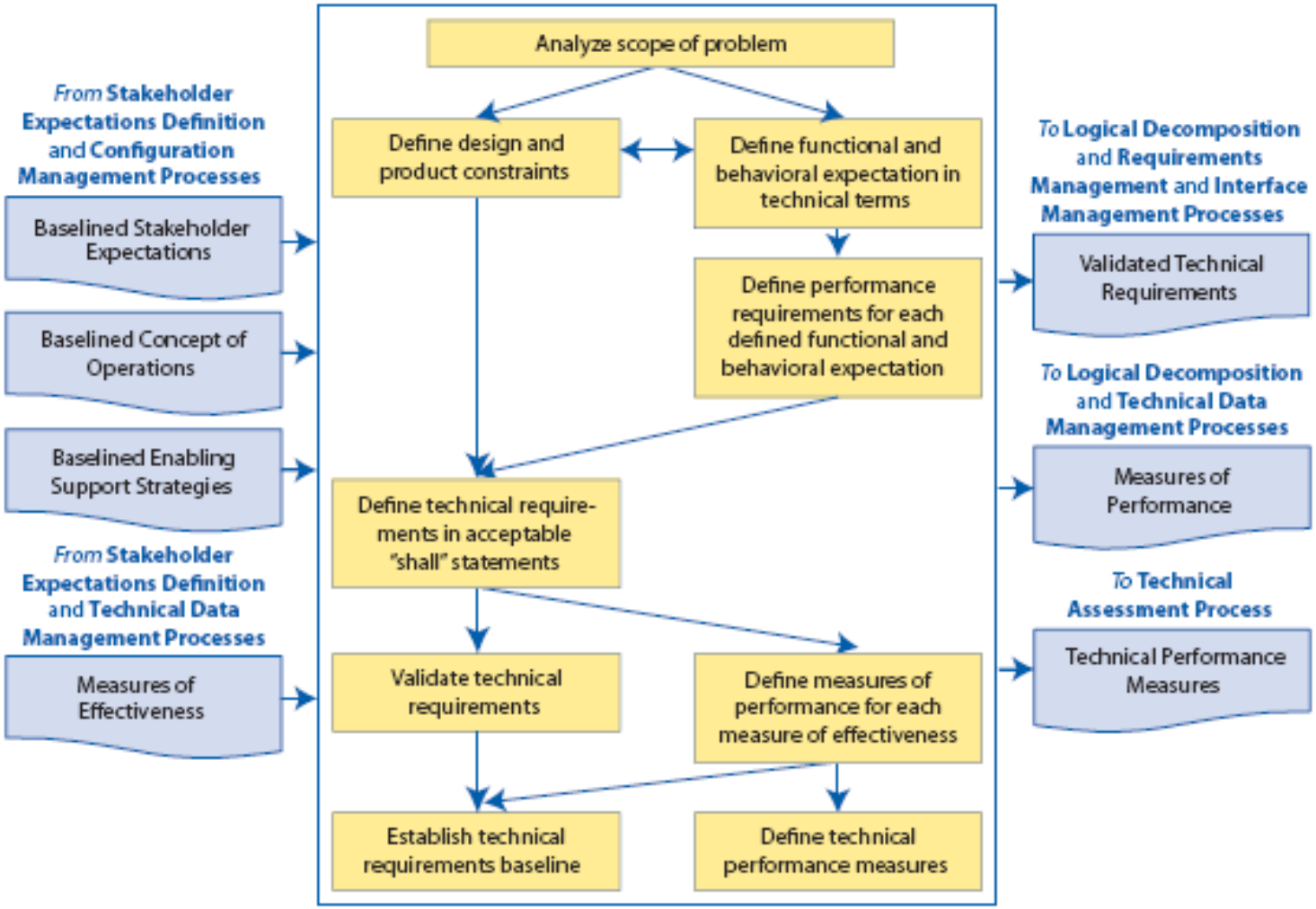
Example of an End-to-End Operational Architecture



High-Level Requirements



High-Level Requirements Process



High-Level Requirements

Inputs

- **Top-Level Requirements and Expectations**
  - **These are the top-level requirements and expectations (e.g., needs, wants, desires, capabilities, constraints, and external interfaces) for the product(s) to be developed**

- **Concept of Operations (ConOps) –> Operational Architecture**
  - **This describes how the system will be operated during the life-cycle phases to meet stakeholder expectations**
  - **It describes the system characteristics from an operational perspective and helps facilitate an understanding of the system goals**
  - **Often expressed as an end-to-end operational architecture (DoDAF OV-1)**

High-Level Requirements

Process Activities

- Defining constraints that the design must adhere to or how the system will be used
  - The constraints are typically not able to be changed based on tradeoff analyses
- Identifying those elements that are already under design control and cannot be changed
  - This helps establish those areas where further trades will be performed to narrow potential design solutions
- Establishing physical and functional interfaces (e.g., mechanical, electrical, thermal, human, etc.) with which the system must interact
- Defining functional and behavioral expectations for the range of anticipated uses of the system as identified in the ConOps

High-Level Requirements

Outputs

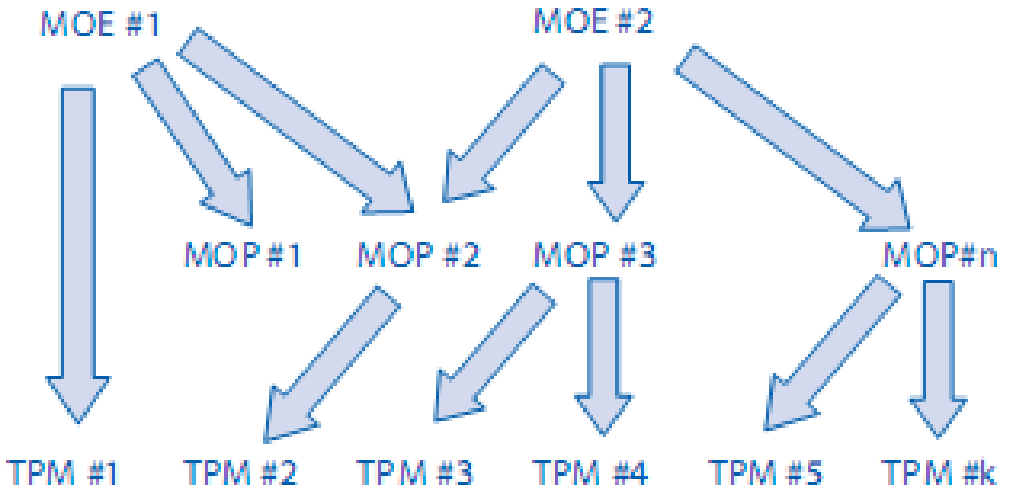
- **Technical Requirements**
  - **This would be the approved set of requirements that represents a complete description of the problem to be solved and requirements that have been validated and approved by the customer and stakeholders**

- **Technical Measures**
  - **An established set of measures based on the expectations and requirements that will be tracked and assessed to determine overall system or product effectiveness and customer satisfaction, such as Measures of Effectiveness (MOEs), Measures of Performance (MOPs), and Technical Performance Measures (TPMs)**

Some Common Types of Requirements

- **Functional**
  - **Functional requirements define what functions need to be done to accomplish the objectives**
- **Performance**
  - **Performance requirements define how well the system needs to perform the functions**
- **Interface**
- **Environmental**
- **Reliability**
- **Safety**
- **Human Factors**

MOE, MOP & TPM Relationships

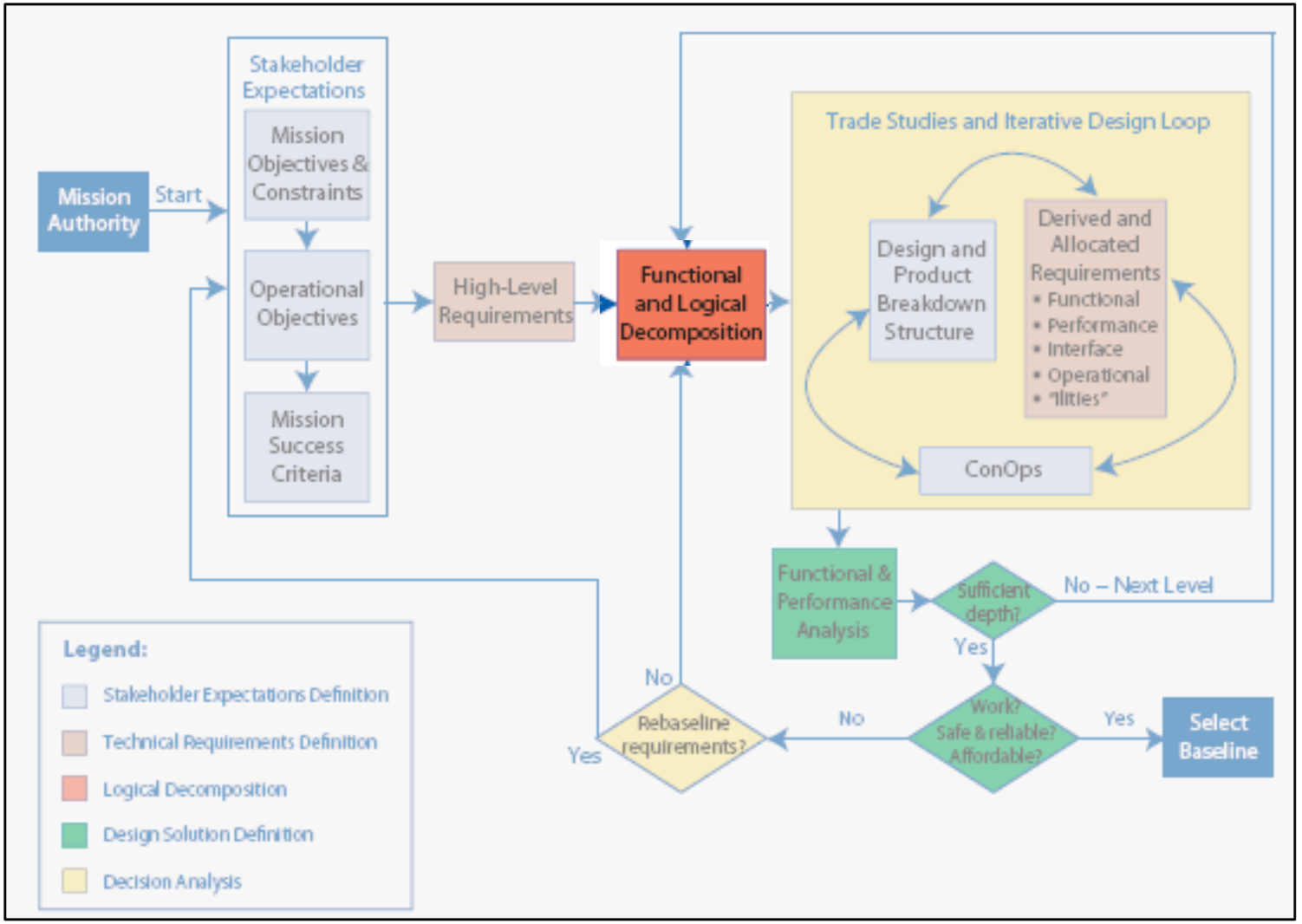


MOEs	Derived from stakeholder expectation statements; deemed critical to mission or operational success of the system
MOPs	Broad physical and performance parameters; means of ensuring meeting the associated MOEs
TPMs	Critical mission success or performance attributes; measurable; progress profile established, controlled, and monitored

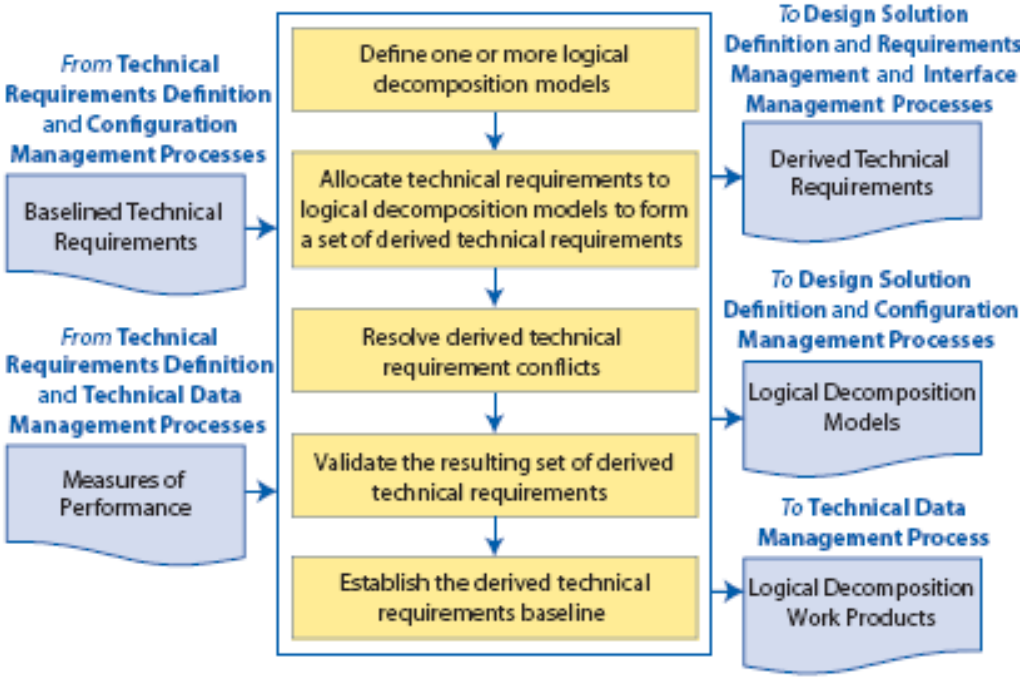
Importance of Technical Standards

- **Standards provide a proven basis for establishing common technical requirements across a program or project to avoid incompatibilities and ensure that at least minimum requirements are met**
- **Common standards can also lower implementation cost as well as costs for inspection, common supplies, etc.**
- **Typically, standards (and specifications) are used throughout the product life cycle to establish design requirements and margins, materials and process specifications, test methods, and interface specifications**
- **Standards are used as requirements (and guidelines) for design, fabrication, verification, validation, acceptance, operations, and maintenance**

Functional and Logical Decomposition



Functional and Logical Decomposition Process



Functional and Logical Decomposition

Inputs

- **Technical Requirements**
  - **This would be the approved set of requirements that represents a complete description of the problem to be solved and requirements that have been validated and approved by the customer and stakeholders**

- **Technical Measures**
  - **An established set of measures based on the expectations and requirements that will be tracked and assessed to determine overall system or product effectiveness and customer satisfaction, such as Measures of Effectiveness (MOEs), Measures of Performance (MOPs), and Technical Performance Measures (TPMs)**

Functional and Logical Decomposition

Process Activities

- ***Functional analysis*** **is the primary method used in system architecture development and functional requirement decomposition**
  - **It is the systematic process of identifying, describing, and relating the functions a system must perform to fulfill its goals and objectives**
  - **It identifies and links system functions, trade studies, interface characteristics, and rationales to requirements**

- **The process is recursive and iterative and continues until all desired levels of the architecture/system have been analyzed, defined, and baselined.**

Three Key Steps in Functional Analysis

1. Translate top-level requirements into functions that must be performed to accomplish the requirements

2. Decompose and allocate the functions to lower levels of the product breakdown structure

3. Identify and describe functional and subsystem interfaces

Requirements Flowdown

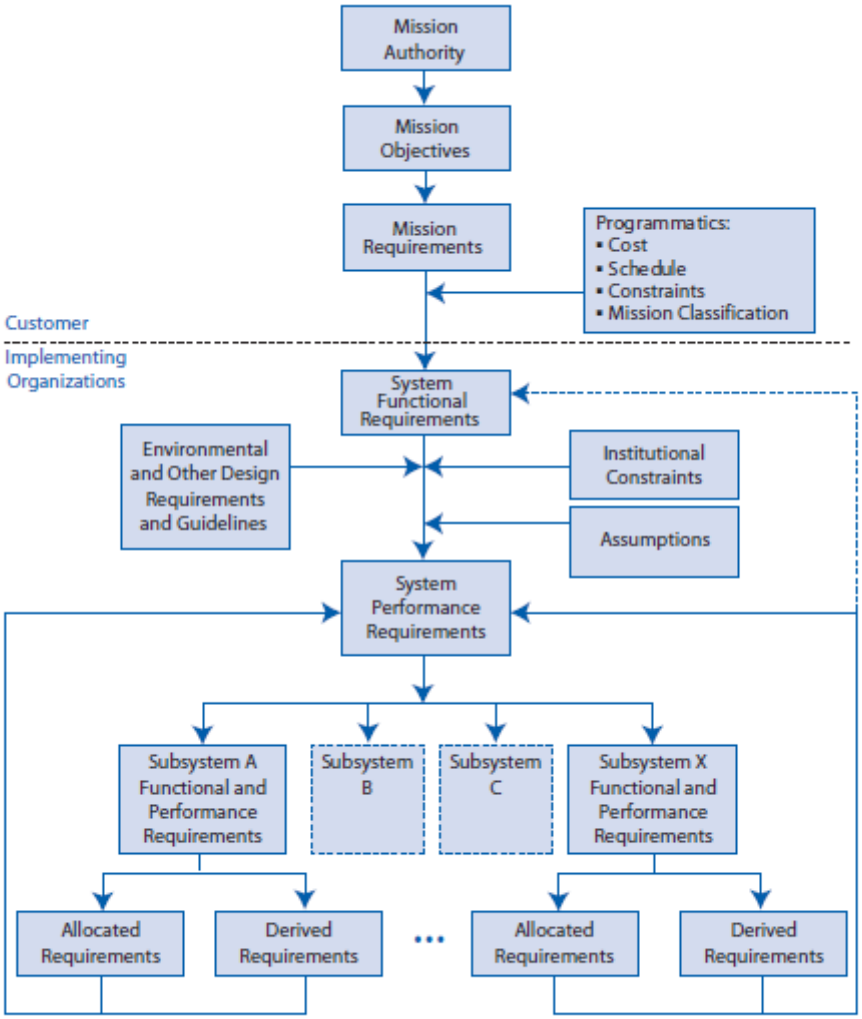


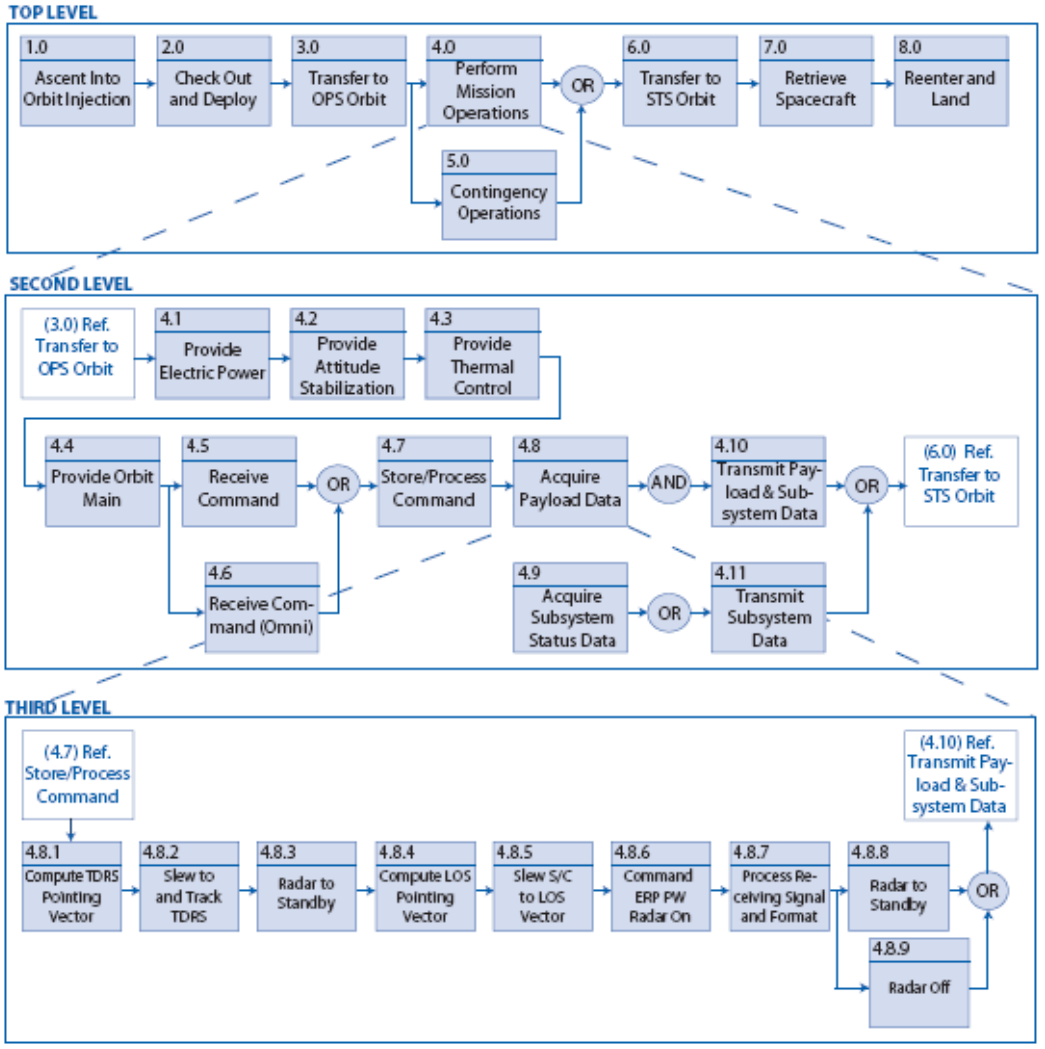
Figure 4.2-3 The flowdown of requirements

Requirements Flowdown Example

▪ User Needs	▪ My house must be comfortably cool, even on the hottest days	MOE
▪ Operational Requirement	▪ The system shall keep the house cooled down to 74 degrees F throughout the year	MOP
▪ System Requirement	▪ The system shall be capable of cooling a 2600 sq ft house down to 74 degrees F in a climatic environment of up to +120 deg F	TPM
▪ Subsystem Requirement	▪ The Chiller subsystem shall dissipate up to 22,000 BTU per hour	TPM
▪ Component Requirement	▪ The Compressor motor shall develop at least 2 HP.	TPM

Functional Analysis Techniques (1)

- **Functional Flow Block Diagrams**
  - **Indicates the sequential relationship of all functions that must be accomplished by a system**
  - **Show the entire network of actions that lead to the fulfillment of a function**



Functional Analysis Techniques (2)

- **N2 Diagrams**
  - **The system components or functions are placed on the diagonal**
  - **The remainder of the squares in the N x N matrix represent the interface inputs and outputs**
  - **Can be taken down into successively lower levels**
  - **Pinpoints areas where conflicts could arise in interfaces**
  - **Highlights input and output dependency assumptions and requirements**

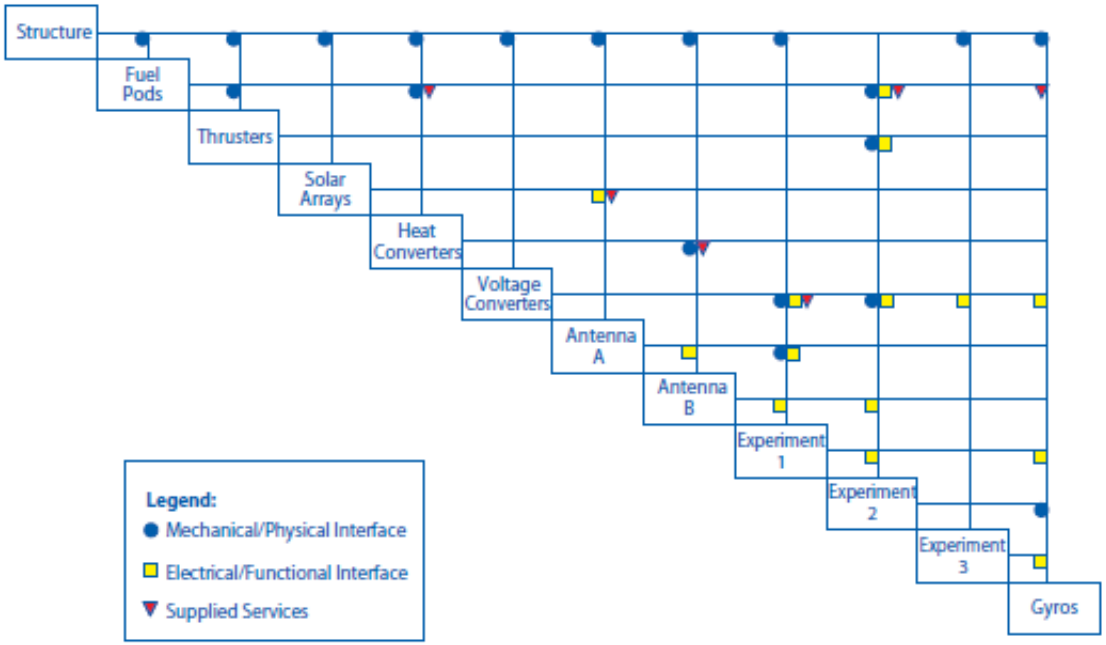
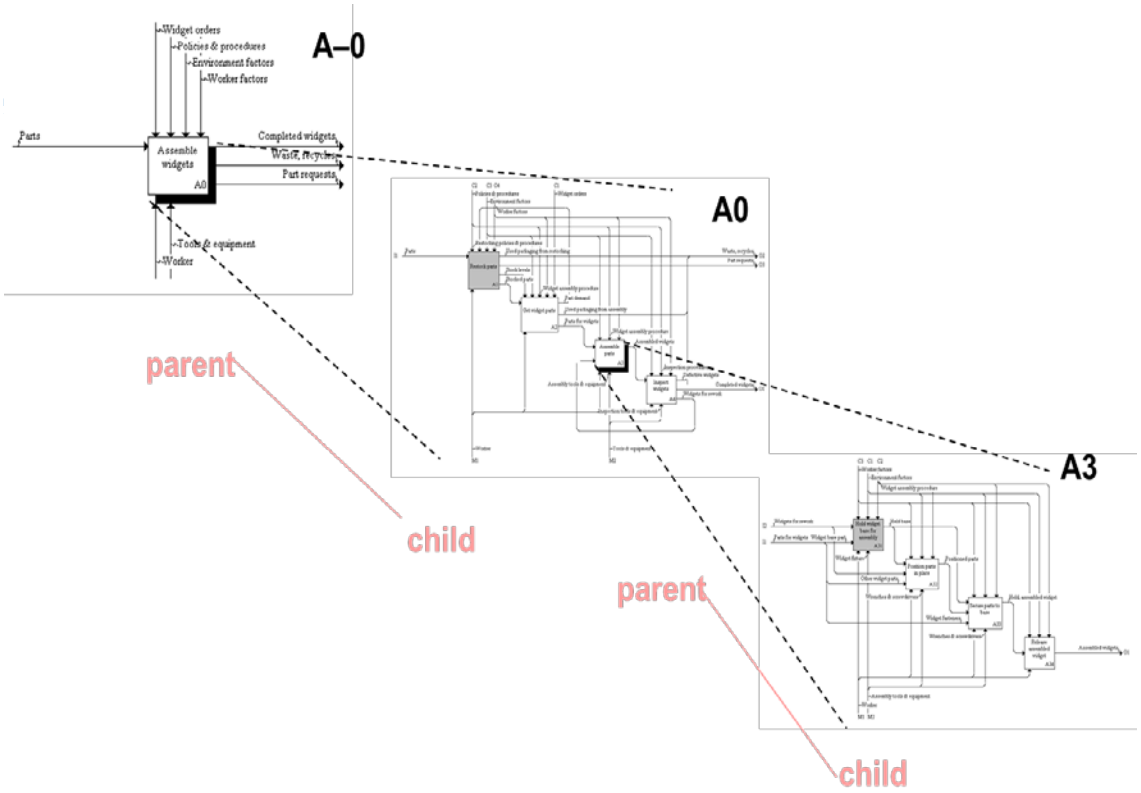


Figure F-6 N2 diagram for orbital equipment

Functional Analysis Techniques (3)

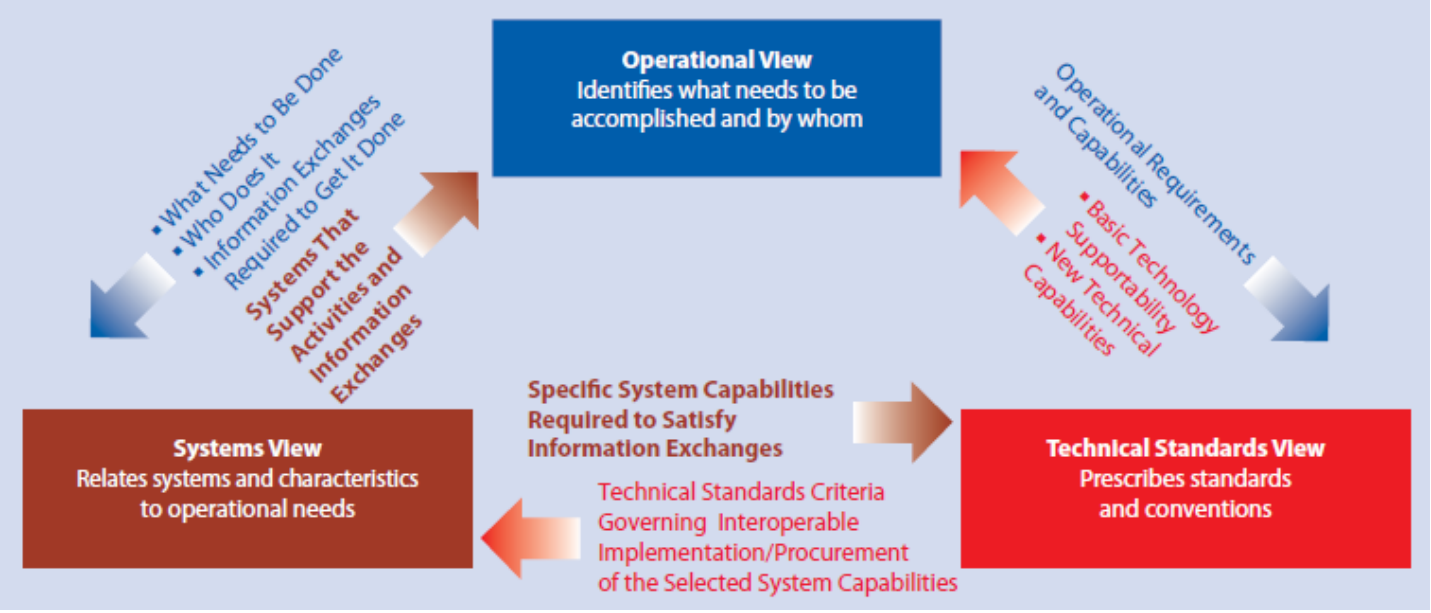
- **IDEF0 Diagrams**
  - **Provides rigor and precision in decomposing functions processes and designs**
  - **Enforces a gradual decomposition and exposition of detail**
  - **Can be taken down into successively lower levels of detail**
  - **Highlights input and output dependencies and requirements**



Functional and Logical Decomposition

Outputs (1)

- **System Architecture Model**
  - **Defines the underlying structure and relationship of the elements of the system (e.g., hardware, software, communications, operations, etc.) and the basis for the partitioning of requirements into lower levels to the point that design work can be accomplished**
  - **Example: DoDAF Operational View products**

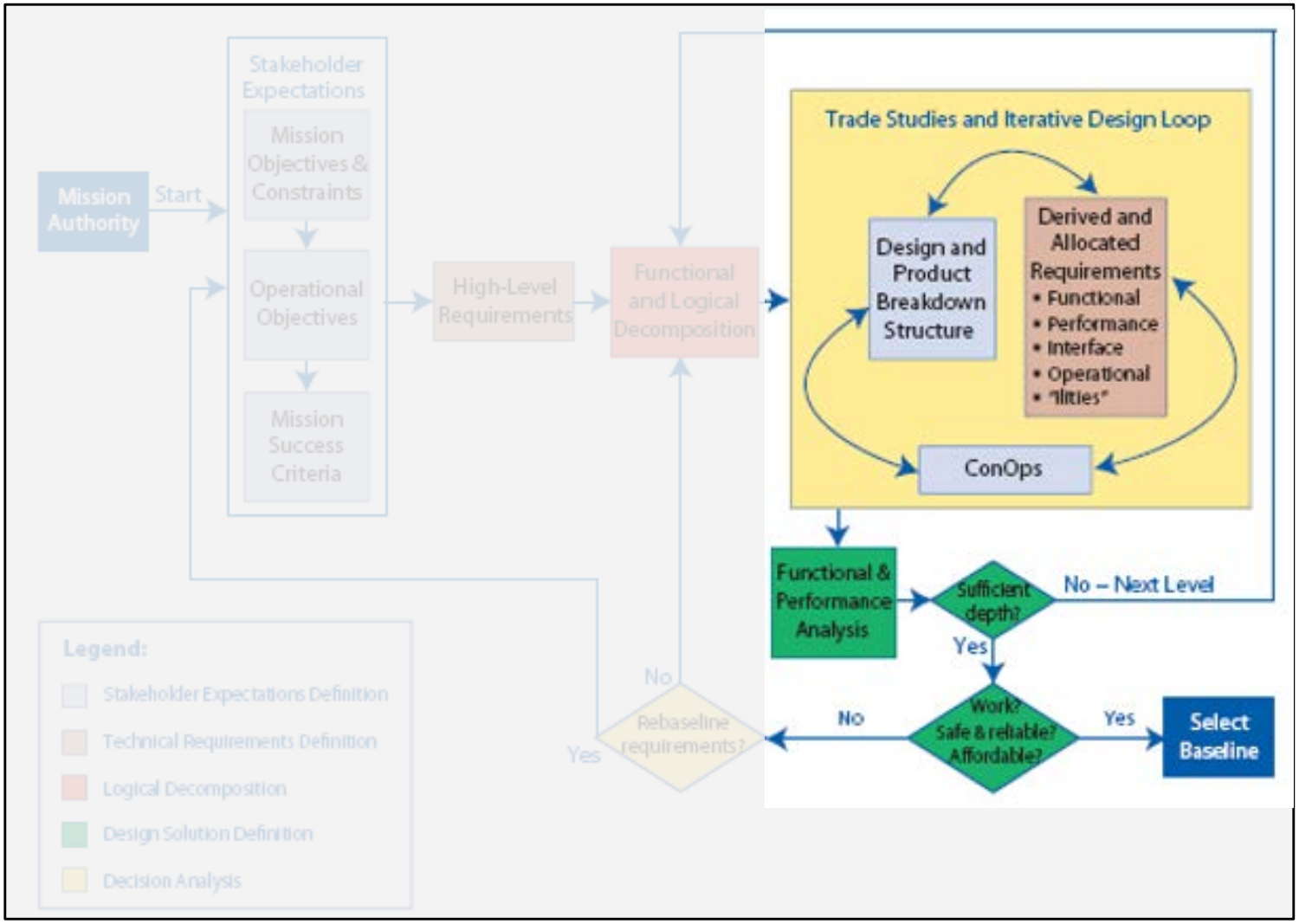


Functional and Logical Decomposition

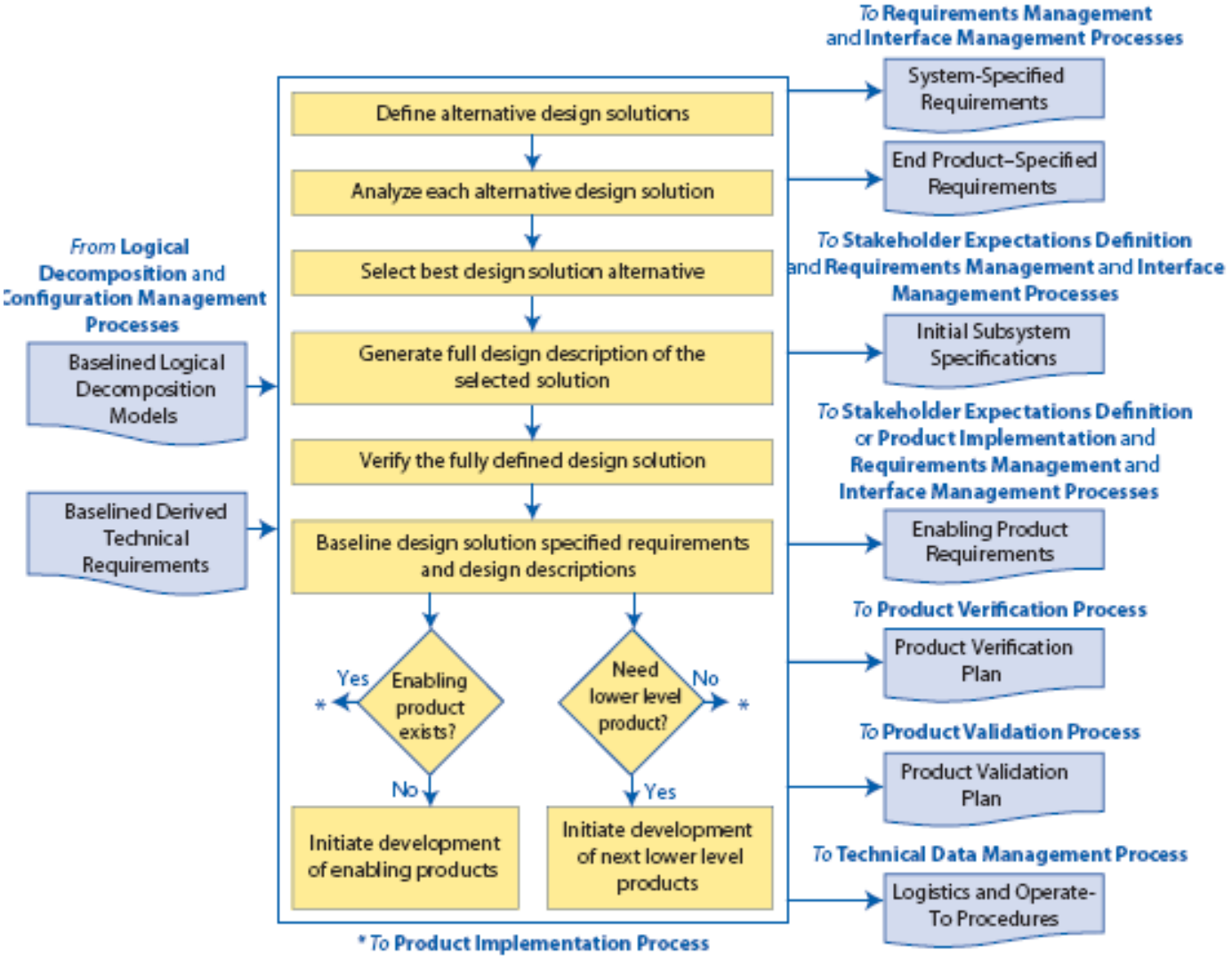
Outputs (2)

- **End Product Requirements**
  - **A defined set of make-to, buy-to, code-to, and other requirements from which design solutions can be accomplished**

Design Solution Definition



Design Solution Definition Process



Design Solution Definition

Inputs

- **Technical Requirements**
  - **The customer and stakeholder needs that have been translated into a reasonably complete set of validated requirements for the system, including all interface requirements**

- **Logical (Functional) Decomposition Models**
  - **Requirements decomposed by one or more different methods (e.g., function, time, behavior, data flow, states, modes, system architecture, etc.)**

Design Solution Definition Process

- **Define Alternative Design Solutions**
- **Create Alternative Design Concepts**

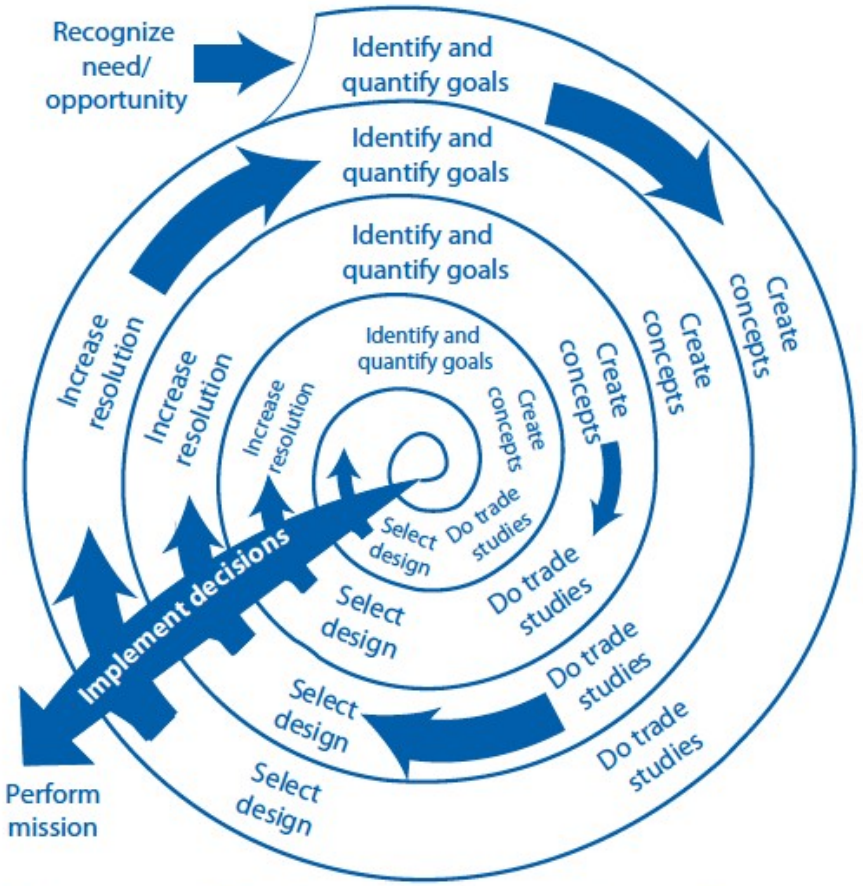


Figure 4.4-2 The doctrine of successive refinement

Design Solution Definition Process

- **Analyze Each Alternative Design Solution**
  - **Evaluate these alternatives in terms of the MOEs, MOPs, TPMs and system cost**
    - **Mathematical models are useful in this step not only for forcing recognition of the relationships among the outcome variables, but also for helping to determine what the measures of performance must be quantitatively**
  - **Rank the alternatives according to appropriate selection criteria**
  - **Drop less promising alternatives and proceed to the next level of resolution, if needed**

Design Solution Definition Process

- **Increase the Resolution of the Design**
- **Fully Describe the Design Solution**
- **Verify the Design Solution**
  - **Realizable within the constraints imposed on the technical effort**
  - **Requirements that are stated in acceptable statements and have bidirectional traceability with the derived technical requirements, technical requirements, and stakeholder expectations**
  - **Decisions and assumptions made in forming the solution are consistent with its set of derived technical requirements, separately allocated technical requirements, and identified system product and service constraints**

Design Solution Definition Process

- **Validate** the Design Solution
- **Identify** Enabling Products
  - Enabling products are the **life-cycle support products** and **services** that facilitate the progression and use of the operational end product through its lifecycle
    - Production
    - Test
    - Deployment
    - Training
    - Maintenance
    - Disposal
- **Baseline** the Design Solution

Design Solution Definition

Outputs (1/4)

- **System Specification**
  - **The system specification contains the functional baseline for the system that is the result of the Design Solution Definition Process**
  - **The system design specification provides sufficient guidance, constraints, and system requirements for the design engineers to execute the design**

- **System External Interface Specifications**
  - **The system external interface specifications describe the functional baseline for the behavior and characteristics of all physical interfaces that the system has with the external world**
  - **These include all structural, thermal, electrical, and signal interfaces, as well as the human-system interfaces.**

Design Solution Definition

Outputs (2/4)

- **End-Product Specifications**
  - **Contain the detailed build-to and code to requirements for the end product. They are detailed, exact statements of design particulars, such as statements prescribing materials, dimensions, and quality of work to build, install, or manufacture the end product.**

- **End-Product Interface Specifications**
  - **Contain the detailed build-to and code-to requirements for the behavior and characteristics of all logical and physical interfaces that the end product has with external elements, including the human-system interfaces.**

- **Initial Subsystem Specifications**
  - **The end-product subsystem initial specifications provide detailed information on subsystems if they are required.**

Design Solution Definition

Outputs (3/4)

- **Enabling Product Requirements**
  - **The life-cycle support products and services that facilitate the progression and use of the operational end product through its life cycle**

- **Product Verification Plan**
  - **The verification plan provides the content and depth of detail necessary to provide full visibility of all verification activities for the end product**
  - **Depending on the scope of the end product, the plan encompasses qualification, acceptance, operational, and disposal verification activities**

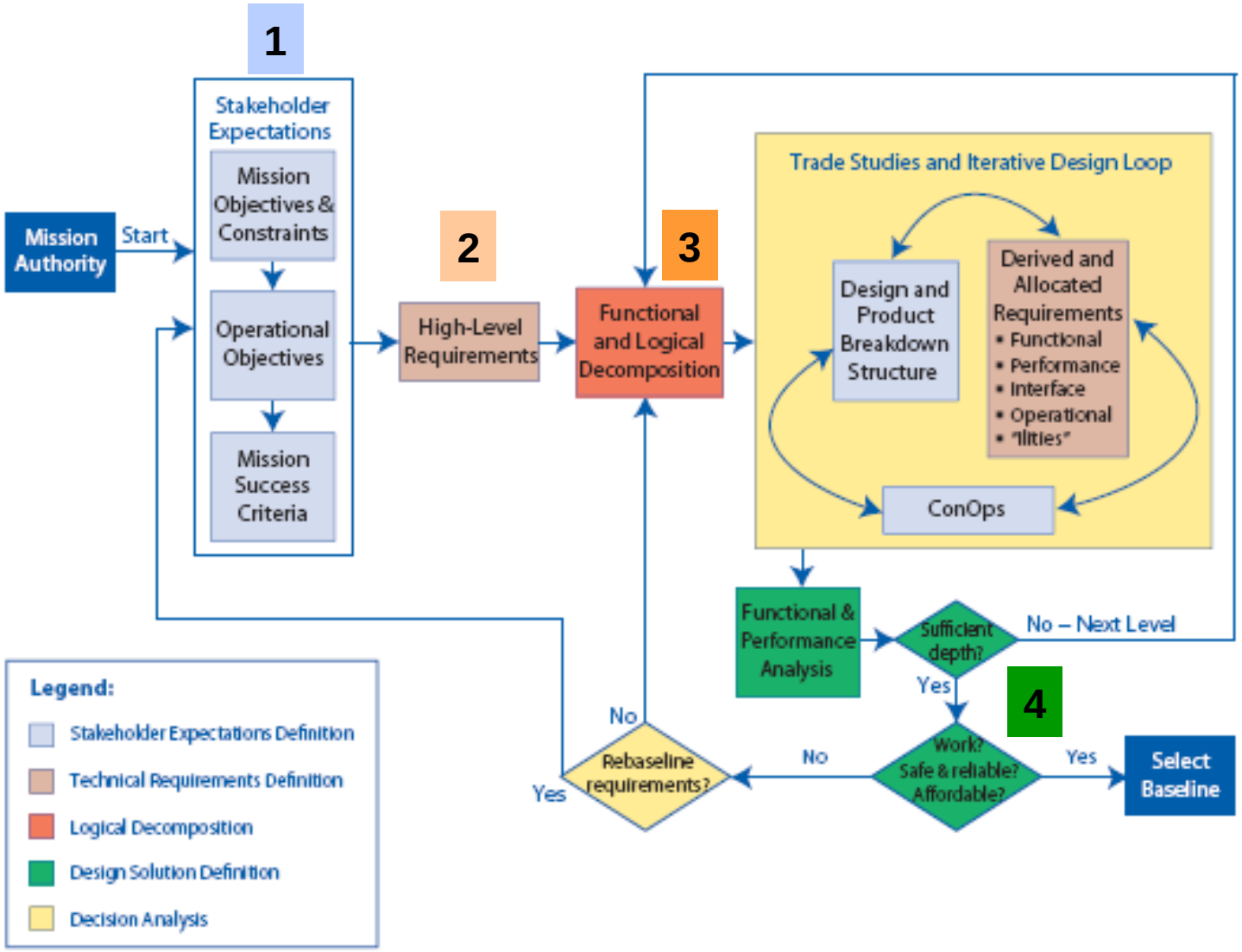
Design Solution Definition

Outputs (4/4)

- **Product Validation Plan**
  - **Provides the content and depth of detail necessary to provide full visibility of all activities to validate the realized product against the baselined stakeholder expectations**
  - **The plan identifies the type of validation, the validation procedures, and the validation environment that are appropriate to confirm that the realized end product conforms to stakeholder expectations**

- **Logistics and Operate-to Procedures**
  - **The applicable logistics plan for the system describes such things as handling, transportation, maintenance, long-term storage, and operational considerations for the particular design solution**

The System Design Process



OnStar System Exercise:

Design Selection Criteria - MOEs, MOPs and TPMs



OnStar System Exercise:
Design Selection Criteria - MOEs, MOPs and TPMs

- **Develop at least one MOE for each of your operational scenarios from your exercise in Lesson 2**
  - **At least one for your operational scenario, and one for your support scenario**
- **Develop at least one supporting MOP for each of your MOEs**
- **Develop at least one supporting TPM for each of your MOPs**
- **Be prepared to share your results with the class, including:**
  - **One of your MOEs, derived MOP(s) and TPM(s), and**
  - **Your observations for the exercise**



Session 4 – System Architecture and Design

Lessons

1. **Program Overview Update**
2. **System Architecture Basics**
3. **Systems Design Process**
4. **Analysis of Alternatives**
5. **Model-Based Systems Engineering**
6. **Special Topics (if time permits)**

- NASA Design Process
- Generic Design Process



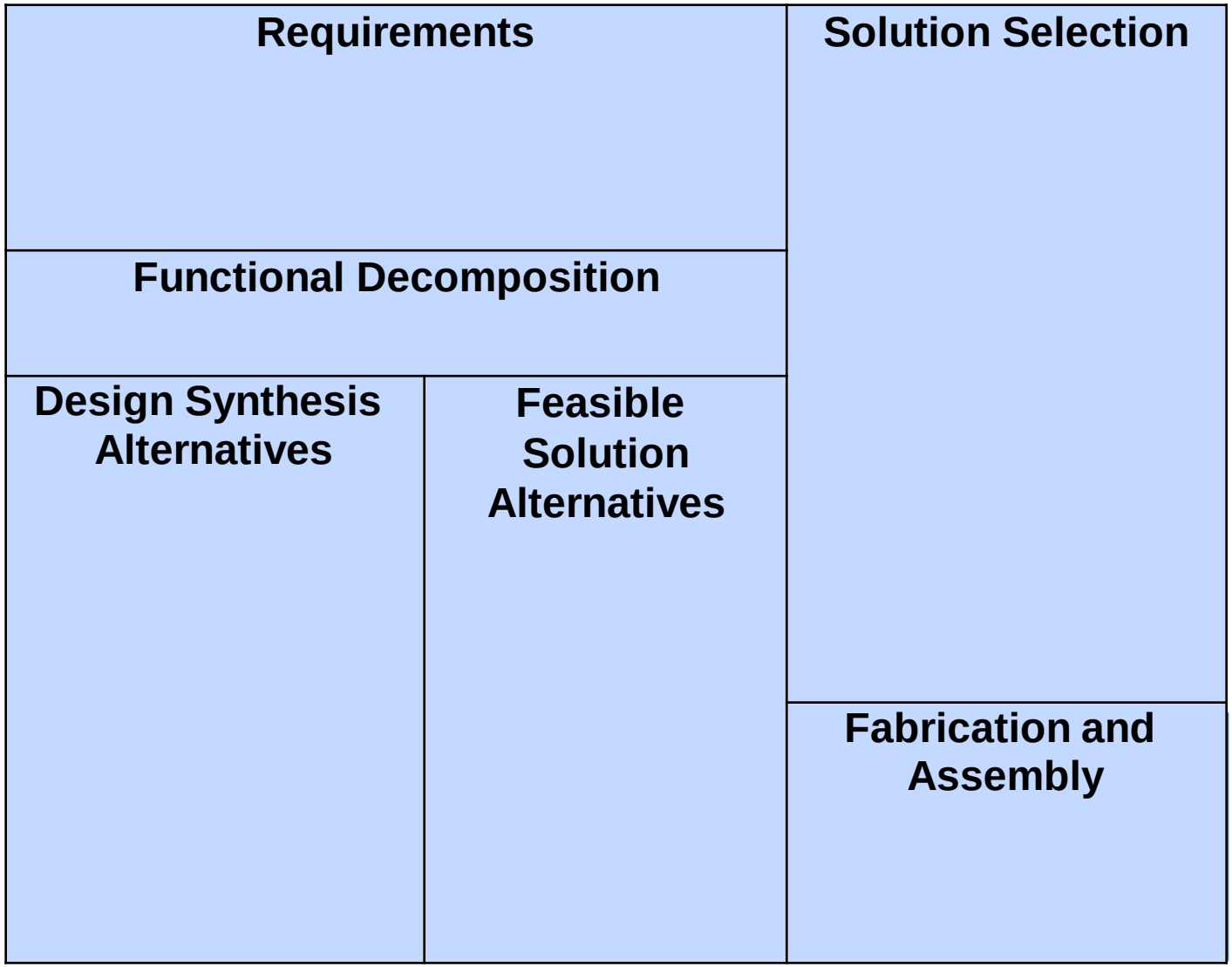
Appendices*

I. **SysML Diagrams and Language Concepts**

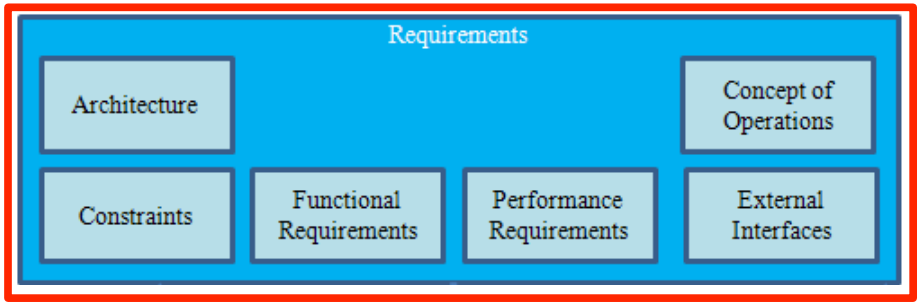
J. **Architecture Frameworks In-depth**

*Appendices A through H were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

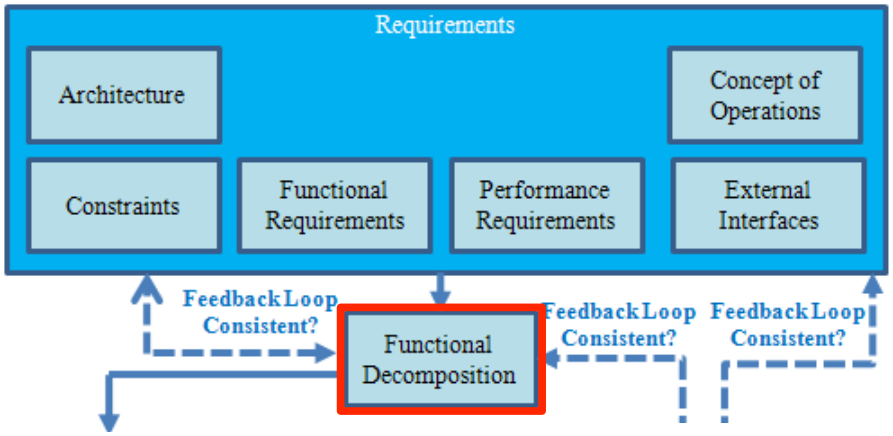
Generic Design Process



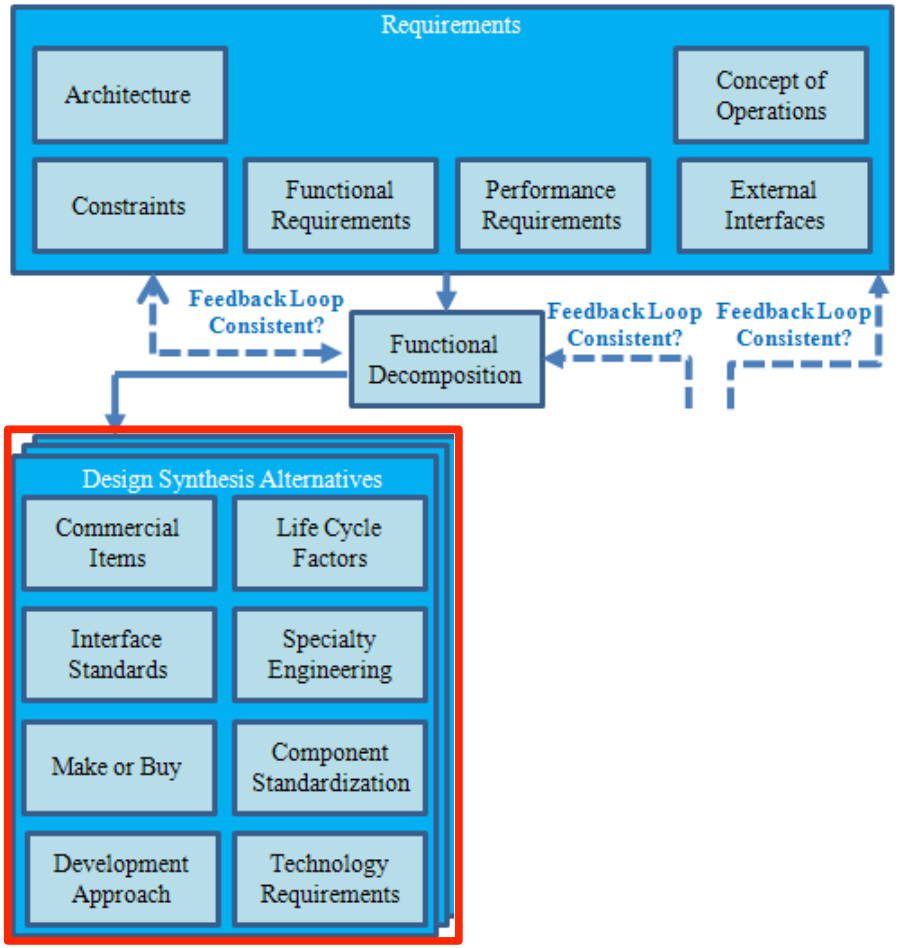
Generic Design Process



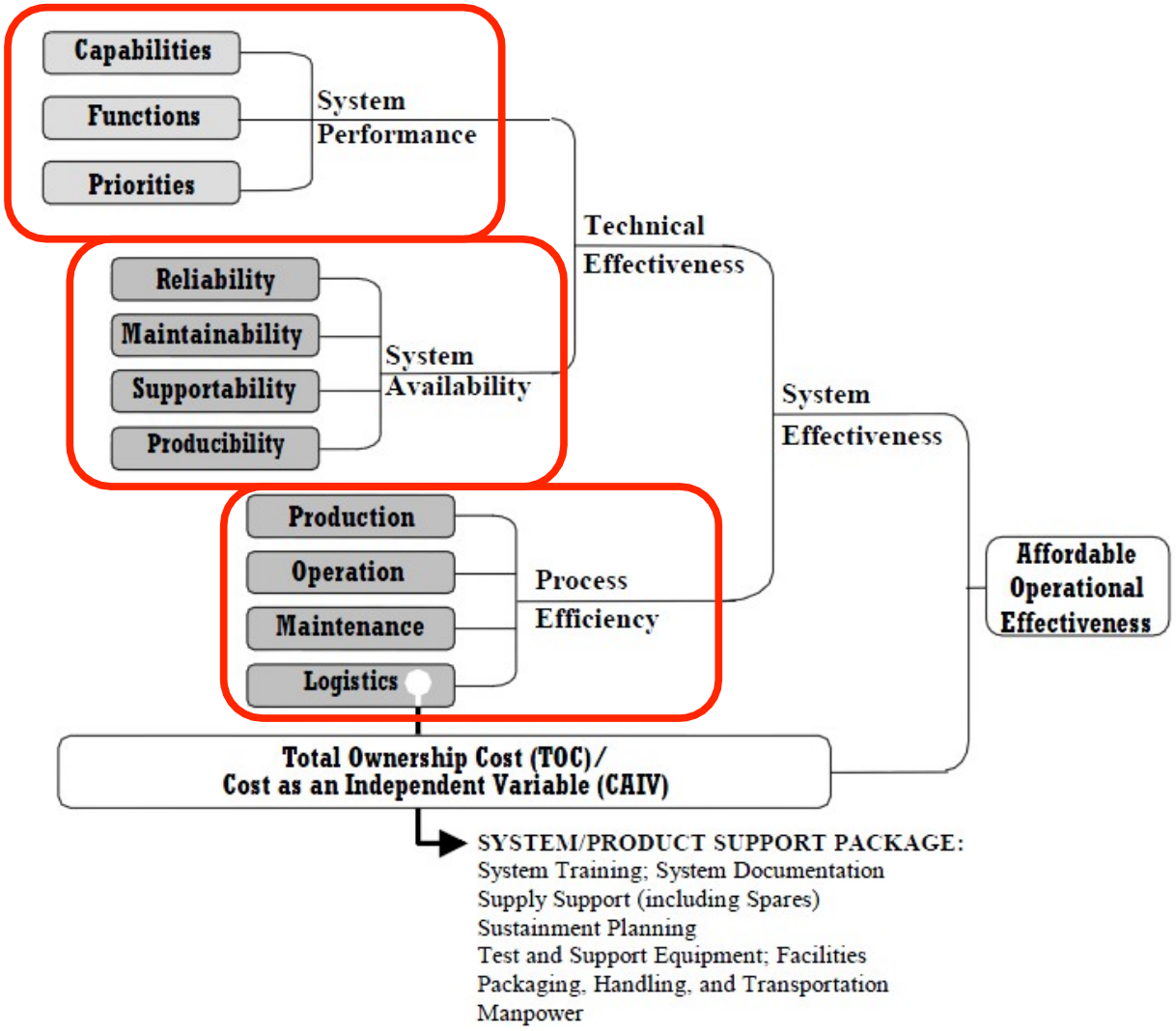
Generic Design Process



Generic Design Process



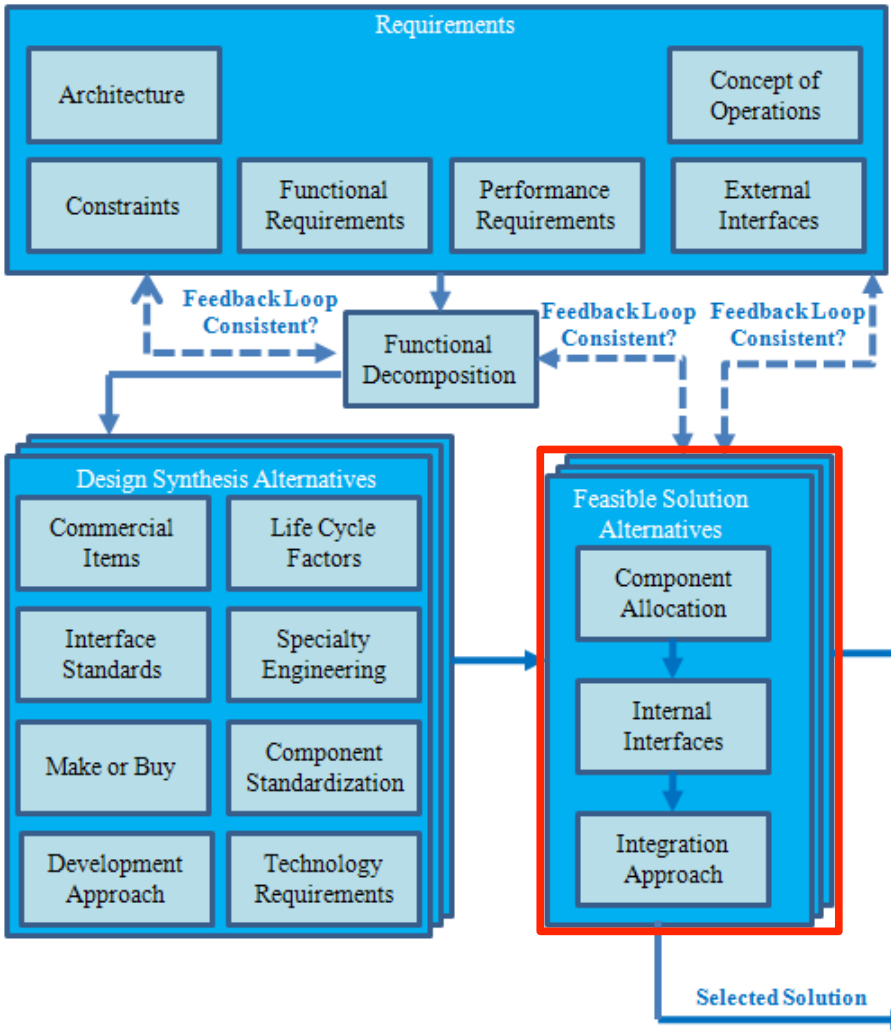
“ilities” – Design for Effectiveness



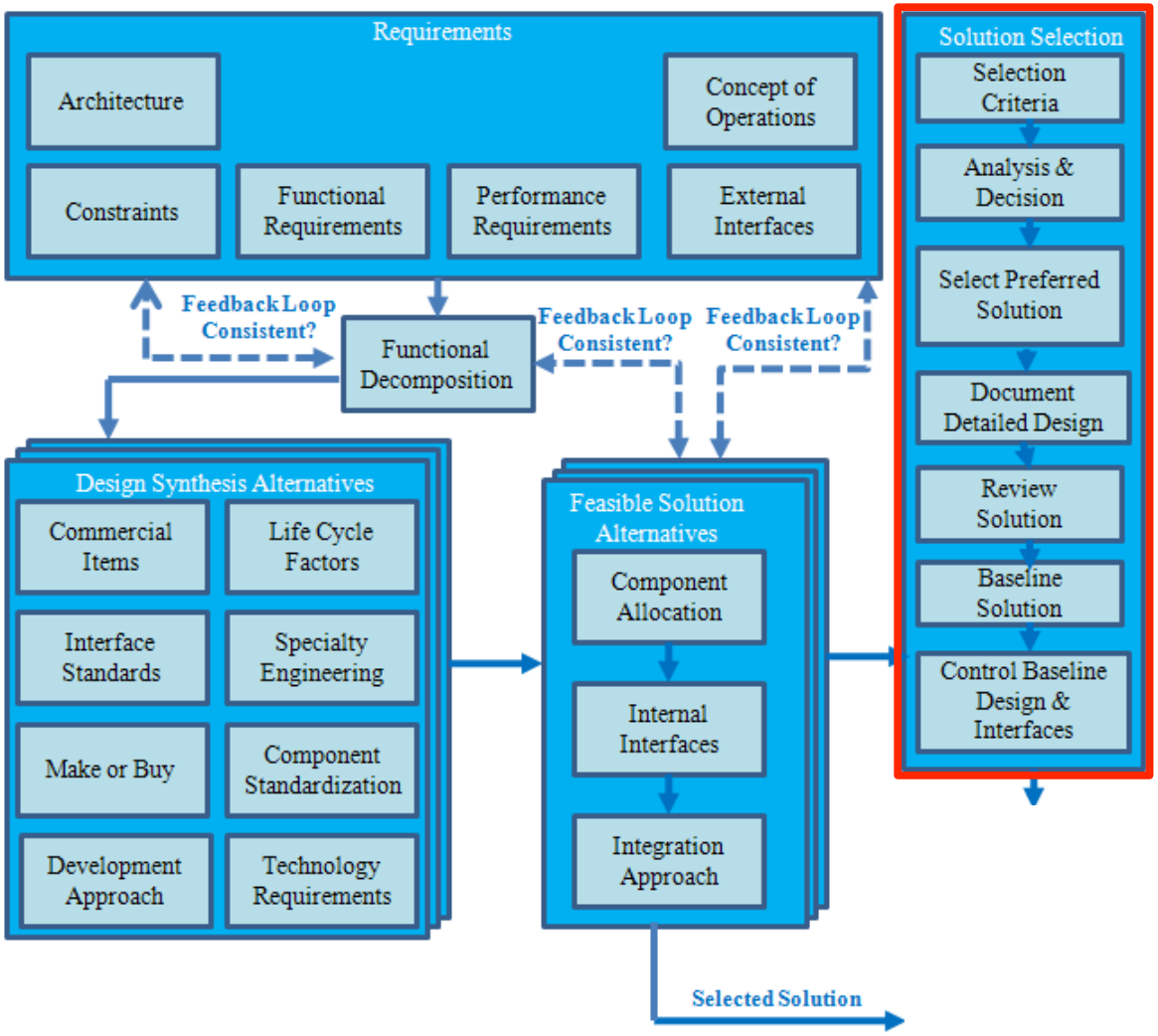
Design Synthesis Alternatives

1. accessibility
2. accountability
3. accuracy
4. adaptability
5. administrability
6. affordability
7. auditability
8. availability
9. credibility
10. standards compliance
11. process capabilities
12. compatibility
13. composability
14. configurability
15. Correctness
16. customizability
17. determinability
18. degradability
19. demonstrability
20. dependability
21. deployability
22. distributability
23. durability
24. effectiveness
25. efficiency
26. evolvability
27. extensibility
28. fidelity
29. flexibility
30. installability
31. Integrity
32. interchangeability
33. interoperability
34. learnability
35. maintainability
36. modularity
37. nomadicity
38. operability
39. orthogonality
40. portability
41. precision
42. predictability
43. provability
44. recoverability
45. relevance
46. reliability
47. repeatability
48. reproducibility
49. responsiveness
50. reusability
51. robustness
52. safety
53. scalability
54. seamlessness
55. serviceability (a.k.a. supportability)
56. securability
57. simplicity
58. stability
59. survivability
60. sustainability
61. tailorability
62. testability
63. timeliness
64. traceability
65. ubiquity
66. understandability
67. upgradability
68. usability

Generic Design Process



Generic Design Process



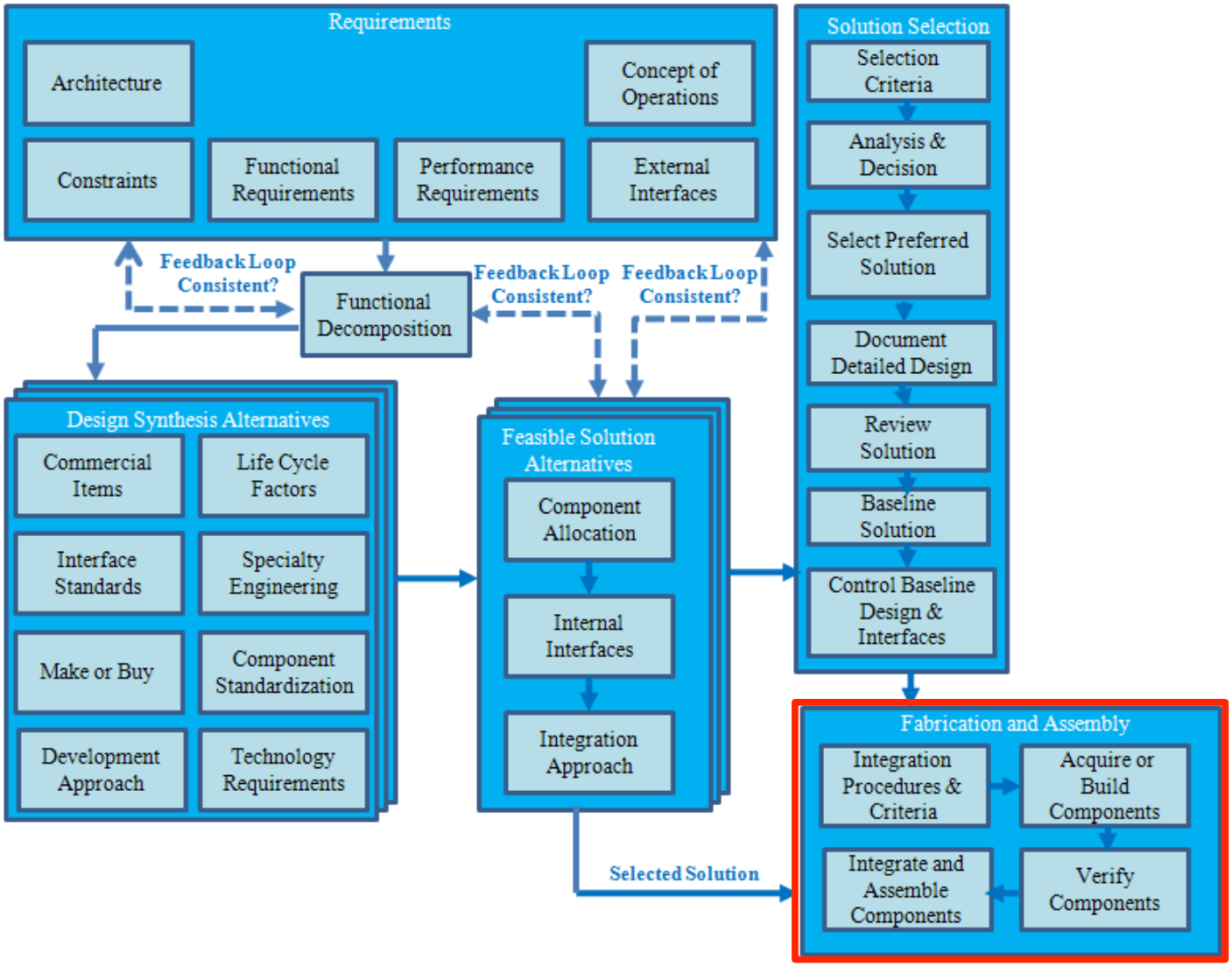
Solution Selection

- **Selection Criteria**

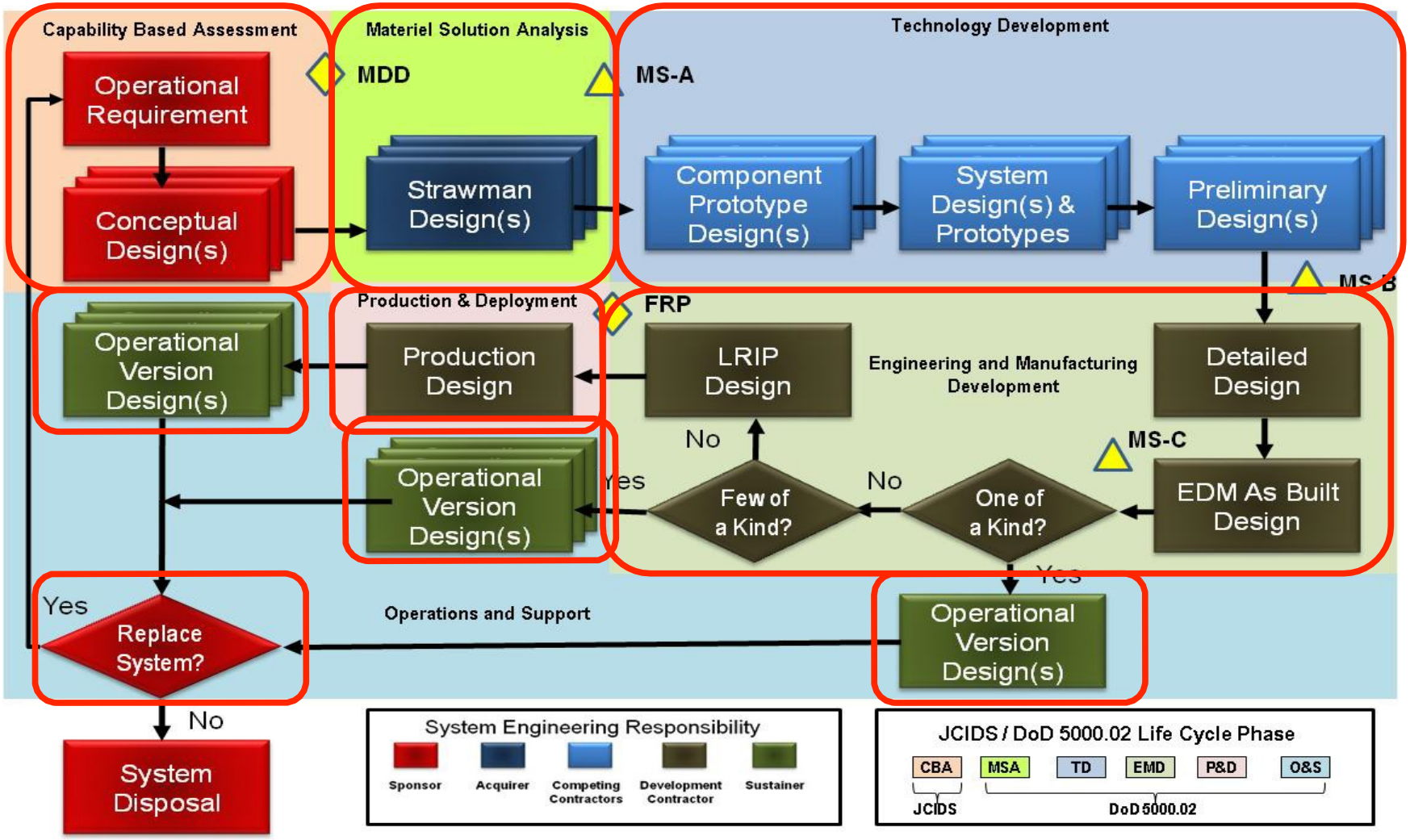
	Mandatory Requirements				Desirable Requirements			Thousands	Months	
	MOP 1	MOP 2	MOP 3	MOP 4	MOP 5	MOP 6	MOP 7	PRICE	SCHEDULE	RISK
Atternative 1								180	15	High
Alternative 2								170	12	Mod
Alternative 3								150	10	High
Alternative 4								130	12	Low

Excpetional — No Risk
Meets Req — Low Risk
Marginal — Mod Risk
Fails Req — High Risk

Generic Design Process



DoD System Design Evolution



Summary

Generic Design Process

- Requirements
- Functional Decomposition
- Design Synthesis Alternatives
- Feasible Solution Alternatives
- Solution Selection
- Fabrication and Assembly

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		

All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?

Back-up slides

The following slides provide more information on each of the phases and elements within each phase of the Generic Design Process.

Requirements

- **Architecture**
  - **High-Level Operational Graphic (OV-1)**
  - **Operational Resource Flow Description (OV-2)**
  - **Operational Resource Flow Matrix (OV-3)**
  - **Organizational Relationships Chart (OV-4)**
  - **Operational Activity Decomposition Tree (OV-5a)**
  - **Operational Activity Model (OV-5b)**
  - **Event-Trace Description (OV-6c)**
  - **Systems Resource Flow Description (SV-2)**
  - **Services Resource Flow Description (SvcV-2)**
  - **Systems Functionality Description (SV-4)**
  - **Services Functionality Description (SvcV-4)**

Requirements

- **Architecture**
  - **Operational Activity to Systems Function Traceability Matrix (SV-5a)**
  - **Operational Activity to Systems Traceability Matrix (SV-5b)**
  - **Operational Activity to Services Traceability Matrix (SvcV-5)**
  - **Systems Resource Flow Matrix (SV-6)**
  - **Services Resource Flow Matrix (SvcV-6)**
  - **Standards Profile (StdV-1)**
  - **Standards Forecast (StdV-2)**

Requirements

- **Constraints**
  - **Use of a particular system or subsystem architecture or requirements on the architecture, e.g., required subsystems; use of standard, military, or existing components; or use of Government furnished property (equipment, information, or software)**
  - **Use of particular design or construction standards; use of particular data standards; use of a particular programming language; use of existing software; workmanship requirements and production techniques**
  - **Physical characteristics of the system or subsystem (such as weight limits, dimensional limits, color, protective coatings); interchangeability of parts; ability to be transported from one location to another; ability to be carried or set up by one or a given number of people.**

Requirements

- **Constraints**
  - **Materials that can and cannot be used; requirements on handling of toxic materials; limits on electromagnetic radiation that the system is permitted to generate**
  - **Use of nameplates, part marking, serial and lot number marking, and other identifying markings**
  - **Flexibility and expandability should be provided to support anticipated areas of growth or changes in technology, threat, or mission**
  - **Manufacturing requirements and constraints associated with producing the system or subsystem**

Requirements

- **Functional Requirements**
  - **MIL-HDBK-520, "A functional requirement is a necessary task, action or activity that should be accomplished."**
  - **EIA-632 defines a functional requirement as "A requirement that defines what system products must do and their desired behavior in terms of an effect produced, or an action or service to be performed."**

- **Examples**
  - **An example of a behavior is "the system shall switch from standby mode to run mode;"**
  - **An example of an effect produced is "the subsystem shall cause an alert signal;"**
  - **An example of an action or service to be performed is "the system shall produce a signal that opens the valve."**

Requirements

- **Performance Requirements**
  - **Key Performance Parameters (KPPs)**
  - **"KPPs are those attributes or characteristics that are considered critical or essential to the development of an effective military capability that make a significant contribution to the characteristics of the future joint force."**

- **Examples**
  - **An example of a behavior is "the system shall switch from standby mode to run mode within 5 seconds;"**
  - **An example of an effect produced is "the subsystem shall cause an alert signal within 4 microseconds;"**
  - **An example of an action or service to be performed is "the system shall produce a signal that opens the valve within 2 milliseconds."**

Requirements

- **Performance Requirements (Con't)**
  - **EIA-632, "A requirement that defines how well the system products are required to perform a function, along with the conditions under which the function is performed."**

- **Examples**
  - **An example of a behavior is "the system shall switch from standby mode to run mode within 5 seconds when the primary system fails;"**
  - **An example of an effect produced is "the subsystem shall cause an alert signal within 4 microseconds when the operating temperature exceeds 160 degrees Fahrenheit;"**
  - **An example of an action or service to be performed is "the system shall produce a signal that opens the valve within 2 milliseconds when the pressure in the vat exceeds 300 pounds per square inch."**

Requirements

- **External Interfaces**
  - **EIA-632 defines an interface requirement as a requirement that defines the conditions of interaction between items.**
    - **Interface requirements include both logical and physical interfaces. They include, as necessary, physical measurements, definitions of sequences of energy or information transfer, and all other significant interactions between items.**

- **Concept of Operations (CONOPS)**
  - **IEEE STD 1362:1998 states "A ConOps is a user-oriented document that describes system characteristics for a proposed system from the users' viewpoint.**

Functional Decomposition

- **“This step involves allocating functional performance specifications into system functional and performance requirements allocated across the system down to a configuration item level.”**

- **“Functions are allocated to each configuration item yielding "design to" preliminary design specifications and verification plans at the configuration item level.”**
  - **Defense Acquisition Guide**

Functional Decomposition

- **Develop logical solution representations to provide an abstract definition of the solution to the system technical requirements**
  - **Functional flows**
  - **Behavioral responses**
  - **State and mode transitions**
  - **Timelines**
  - **Control flows**
  - **Data flows**
  - **Information models**
  - **Object services and attributes**
  - **Context diagrams**
  - **Threads**
  - **Data structures**
  - **Functional failure modes and effects**

Design Synthesis Alternatives

- **Commercial Items**
  - **Technology Maturity**
  - **Unanticipated changes and the natural evolution of commercial items may drive reconsideration of engineering decisions throughout the life cycle**
  - **Consider the logistics implications of supporting commercial items in a military environment**
  - **COTS items have a relatively short manufacturing life, a proactive diminishing manufacturing sources and material shortages/obsolescence approach should also be considered**
  - **Assess the long- term sustainability of COTS options and to avoid or minimize single source options**
- **Interface Standards**
- **Make or Buy**

Design Synthesis Alternatives

- **Development Approach**
  - **The classic development approach is the "waterfall" or linear approach that consists of feasibility study, analysis, design, implementation, testing and maintenance phases.**
  - **Other**
    - **Prototyping**
    - **Spiral**
    - **Object Oriented**
    - **Client Server**
    - **Joint Application Development**
    - **Information Engineering (Data Driven)**
    - **Reuse**
    - **Business Process Reengineering**
    - **Outsourcing**
    - **Evolutionary**

Design Synthesis Alternatives

- **Life Cycle Factors**
  - **Technology Turnover**
  - **Sustainment Approach**
- **Specialty Engineering**
  - **“ilities”**

Design Synthesis Alternatives

- Component Standardization
- Technology Requirements

Feasible Solution Alternatives

- Component Allocation
- Internal Interfaces
- Integration Approach

Solution Selection

- Analysis and Decision
- Select Preferred Solution
- Document Detailed Design
- Review Solution

Solution Selection

- **Baseline Solution**
  - **MIL-HDBK-61A defines a configuration baseline as**
    - **An agreed-to description of the attributes of a product, at a point in time, which serves as a basis for defining change**
    - **An approved and released document, or a set of documents, each of a specific revision; the purpose of which is to provide a defined basis for managing change**
    - **The currently approved and released configuration documentation**
    - **A released set of files comprising a software version and associated configuration documentation**
- **Control Baseline and Interfaces**

Fabrication and Assembly

- Integration Procedures & Criteria
- Acquire or Build Components
- Verify Components
- Integrate and Assemble Components



Systems Engineering Technical Short Course

(Session 4: System Architecture and Design)

Lesson 4:
Analysis of Alternatives (AoA)
(AKA: Trade Studies)

Session 4 – System Architecture and Design

Lessons

1. **Program Overview Update**
2. **System Architecture Basics**
3. **Systems Design Process**
4. **Analysis of Alternatives**
5. **Model-Based Systems Engineering**
6. **Special Topics (if time permits)**

- Formal Decision Process
- Evaluation Methods

Appendices*

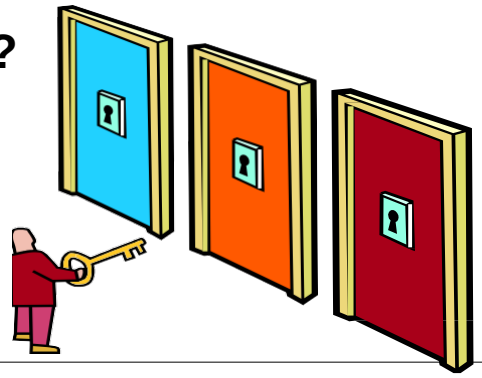
I. **SysML Diagrams and Language Concepts**

J. **Architecture Frameworks In-depth**

*Appendices A through H were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

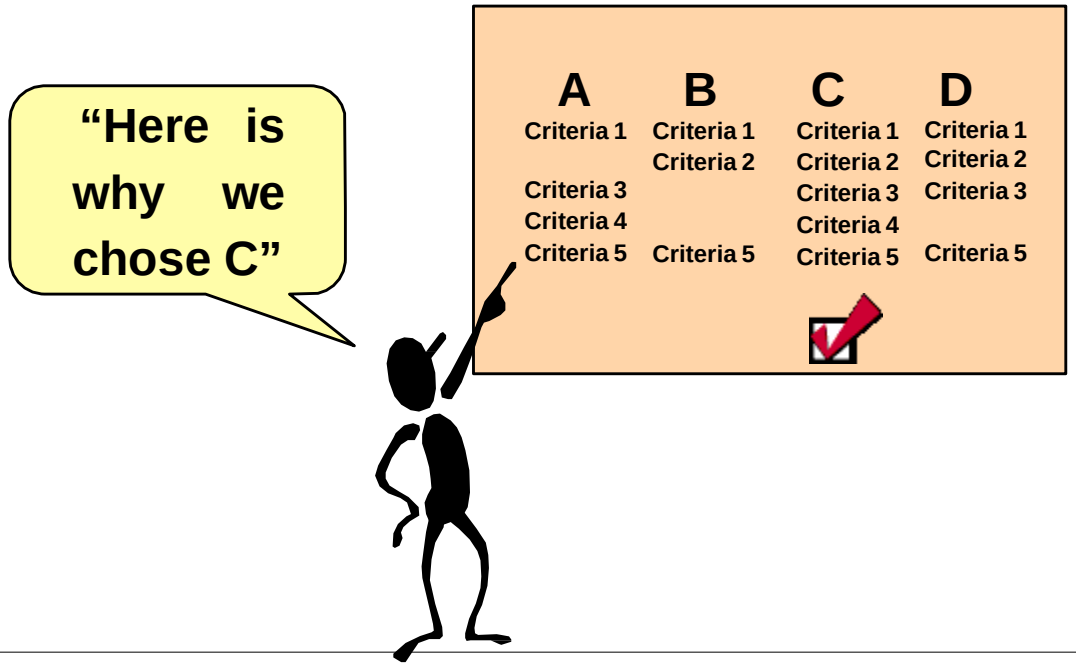
Comparing Alternatives

- The systems engineer is often involved in making design decisions
  - Is there a need for a new system or is there another solution that does not require a development?
    - Compare the existing system and the proposed improved systems against the user's stated requirements
  - What design concept(s) should be pursued further?
    - Example: Develop a new satellite or a new mobile radar system?
  - How should the system be broken up into subsystems?
    - Example: Decide how many trucks comprise the mobile radar system and how are the major components allocated to them
  - Which off-the-shelf component should we select?



Why Follow a Formal Decision Process?

- **Objective process**…reduce or eliminate biased opinion
  - …or even perception of biased opinion
- Provides an **explanation** on how the decision was reached
  - Important on large scale systems with many stakeholders and/ or interested observers
- **Repeatable**



AoA is about…

- **Sound, defensible, value-based decision-making**
- **Providing essential functions for an appropriate cost**
- **Benefit-to-cost relationships and working both sides of the equation**
- **Managing the decision points**
- **Making better decisions!!!**

AoA Levels of Maturity

- **Level 0 – Propose one alternative and justify it.**
- **Level 1 – Propose multiple alternatives and provide a one-dimensional comparative analysis with weak inclusion of uncertainty effects.**
- **Level 2 – Propose multiple alternatives and provide multi-dimensional comparative analysis with weak inclusion of uncertainty effects.**
- **Level 3 – Propose multiple alternatives, and provide multi-dimensional comparative analysis and support robust resource allocation decisions with the inclusion of uncertainty effects.**

Source: Decisions Based on Analysis of Alternatives (AoA) by Dr. David G. Ullman, January 2009

One Approach*

- Define the objective
- Identify alternatives
- Determine criteria for selection and relative importance
- Determine how to score each criterion
- Score the alternatives
- Make a selection
- Analyze the basis and robustness of the selection

* There are many other techniques. See, for example, Buede, Chapter 13

Define the Objective

- Before setting out, determine what the goal is (the purpose for doing the trade study)
  - Choose the best overall candidate
  - Choose the top *x* candidates
    - e.g., narrow the field for further investigation
  - Eliminate the least desirable candidate(s)
  - Determine the “least unappealing” candidates

- Define the scope/formality
  - Decision to be presented to an executive or sponsor
    - More formal, capture more documentation
  - Lower level decision within the team
    - Less formal, may spend less time on some steps

Identify Alternatives

- To make a choice between alternatives, one must have alternatives to choose from
  - 1. Identify the main attributes that are most important
    - Performance requirement values (MOPs, TPMs, KPPs)
    - Specific functionality present
  - 2. Brainstorm for alternatives – "cast the net wide"
    - Systems or subsystems that fulfills the need now
    - Similar systems
    - New technologies
    - New products to consider
    - Innovative ideas
  - 3. Select at least 3 - 4 alternatives that meet all or most of the selected attributes
    - Too many alternatives: adds time to gather the data and often results in less differentiation between alternatives
    - Too few alternatives: Leaves little to choose from

Roadblocks to new ideas….

- It needs higher approval
- Its too ambitious for us
- We did it this way the last time
- We have no problems now…why change
- If you want a quick answer it's…NO!!
- It does not agree with the specs
- It's against our policy
- That's not the way it's done around here
- It won't work
- It's too much trouble to phase in
- We tried that 10 years ago, and it didn't…
- …

Determine Criteria for Selection

- **Criteria:** **"what you are going to base the decision on"**
- **Best – Values that can be compared**
  - **Performance values (MOPs, TPMs, KPPs)**
  - **Cost (Lifecycle costs, other)**
  - **Schedule**
  - **Risk**
- **Yes/No**
  - **Required functionality present**
  - **Meets programmatic needs (can meet schedule)**
  - **Do not want too many yes/no answers – results in less differentiation between alternatives**
- **Again – choose multiple criteria but not too many**
  - **Too many criteria: Results in less differentiation between alternatives when scoring is applied**
  - **Too few criteria: May not address all the important differentiators**
- **Rank the Criteria – assign weights to each criterion**

Determine How to Score Each Criterion (1 of 2)

1. **Pugh Matrix – useful when evaluating and selecting system concepts at early stage of design, for each criterion for each concept, assign a '+', '-', or 'S':**
   - **'+' is better than required performance**
   - **'-' is does not meet required performance**
   - **'S' is performance on par with requirement**
   - **Sum up the number of '+', '-', 'S' for each concept.**
   - **Use Pugh matrix to 'inform', not 'determine', concept selection**
2. **Discrete values – useful for comparing criterion on different scales:**
   - **Ranking – Choose a range for rating (e.g., 0-4 or: 0, 1, 2, 3, 4)**
   - **Weighted percentages of utility (e.g., 0%, 50%, 80%, 100%)**
   - **Yes/No -> also can be expressed as 0 [0%] or Max value [100%]**
   - **Subjective (high, medium, low) -> translate to rankings**

Determine How to Score Each Criterion (2 of 2)

Gas Mileage	**0**	**1**	**2**	**3**	**4**	
Risk	<=20	21-24	25-28	29-32	33+	
	High		Medium		Low	
Reliability (MTBF)*	<2000 hrs	2000-2200	2201-2500	2501-3000	3000+	Unevenly spaced ranges depending on value (utility) of better reliability

3. Numerical values

- **Harder to weight dissimilar scales**
- **Need to normalize to a common range, define minimum and maximum values**

***Reliability (MTBF)** (range is 0-10): <2000=0, >3000=10, otherwise value = $\frac{\text{MTBF - 2000}}{100}$

Score the Alternatives Using the Criteria

Example: Choosing a car

Alternative 1: ToyCar Corollya

Criteria	Value	Weight	Assigned Score	Total Points
Gas Mileage	30 MPG	20%	3	0.6
Reliability	4 (based on famous magazine survey)	20%	4	0.8
Cost	$14500	15%	2	0.3
Safety	Air Bags = Yes	15%	4	0.6
Warranty	3/36000	10%	3	0.3
Options	AC=yes, CD=yes	10%	4	0.4
Style	Good (subjective)	10%	2	0.2

Grand Total Points = 3.2

Do this for each Alternative

Prepare to Make a Selection

Alternative	Total Points
ToyCar Corollya	3.2
Crystal Xenon	3.0
General Engines Vibration	2.8
Hounder Civilian	3.0
Missin Versatile	2.6



This is a process to help get an answer: you still make the decision

Analyze the Basis and Robustness of the Decision

- Does the answer make sense?
- Can you pick out the differentiator(s)?
- Are all the values very close together or do most of the results all have the same value?
  - Are there criteria where all the scores were the same?
  - Could indicate that your criteria and/or scoring do not differentiate very well
- Robustness
  - Are there special cases that just looking at total scores do not catch very well?
    - Fail one specific criteria and it should be an automatic rule-out (vice a zero point scoring that may get diluted in an overall grade)
  - Look at the scores and criteria values: Does the answer change significantly if one criteria is rated differently?

This is a process to help get an answer: you still make the decision

DoD Analysis of Alternatives (AoA)

- **A DoD AoA is a very formal trade study**
  - **Goal: Explore numerous solutions (6 – 8) in the concept definition phase to identify the most promising options**
  - **Typically seen by high level decision makers, especially for large projects**
  - **Used to justify why a project should be started and money spent/invested**
- **Formal requirement in the Department of Defense acquisition process at key decision points**
- **NASA has a similar process**

DoD AoA Sample Content

- Introduction
  - Background
  - Purpose
  - Scope
- Ground Rules
  - Scenarios
  - Threats
  - Environment
  - Constraints and Assumptions
- Alternatives
  - Description of Alternatives
  - Nonviable Alternatives
  - Operations Concepts
  - Support Concepts
- Determination of Effectiveness Measures
  - Mission Tasks
  - Measures of Effectiveness
  - Measures of Performance
- Effectiveness Analysis
  - Effectiveness Methodology
  - Models, Simulations, and Data
  - Effectiveness Sensitivity Analysis
- Cost Analysis
  - Lifecycle Cost Methodology
  - Models and Data
  - Cost Sensitivity and/or Risk Analysis
- Cost-Effectiveness Comparisons
  - Cost-Effectiveness Methodology
  - Displays or Presentation Formats
  - Criteria for Screening Alternatives
- Organization and Management
  - Study Team/Organization
  - AoA Review Process
  - Schedule

AoA Exercise:
Concept Selection

A 10-week trade study was performed involving four alternative system concepts (I, II, III, IV). Five Selection Criteria were used, each weighted equally. Each of the selection criteria were evaluated for each of the concepts and values were assigned on a scale of 1 to 5 for each criterion. The results of the weighted utility scores are shown below. A simplified addition function was used to generate an overall score for each concept.

	A	B	C	D	E	Score
Concept I	1	2	2	4	2	11
Concept II	3	1	2	5	5	16
Concept III	2	2	1	2	2	9
Concept IV	3	3	3	3	3	15

Based on the above table of rating profiles:

1. Which concept would you select as the best concept? Explain your selection.

2. After considering the above results, the project manager decides to take the two concepts with the highest scores (II and IV) and conduct a “run off,” and instructs your to team to perform a run-off trade study between the two concepts. However, you only have one week to perform this additional study. Describe what you would do to perform the run-off study.

3. Be prepared to share your results with the class!

Session 4 – System Architecture and Design

Lessons

1. **Program Overview Update**
2. **System Architecture Basics**
3. **Systems Design Process**
4. **Analysis of Alternatives**
5. **Model-Based Systems Engineering**
6. **Special Topics (if time permits)**

- Formal Decision Process
- Evaluation Methods

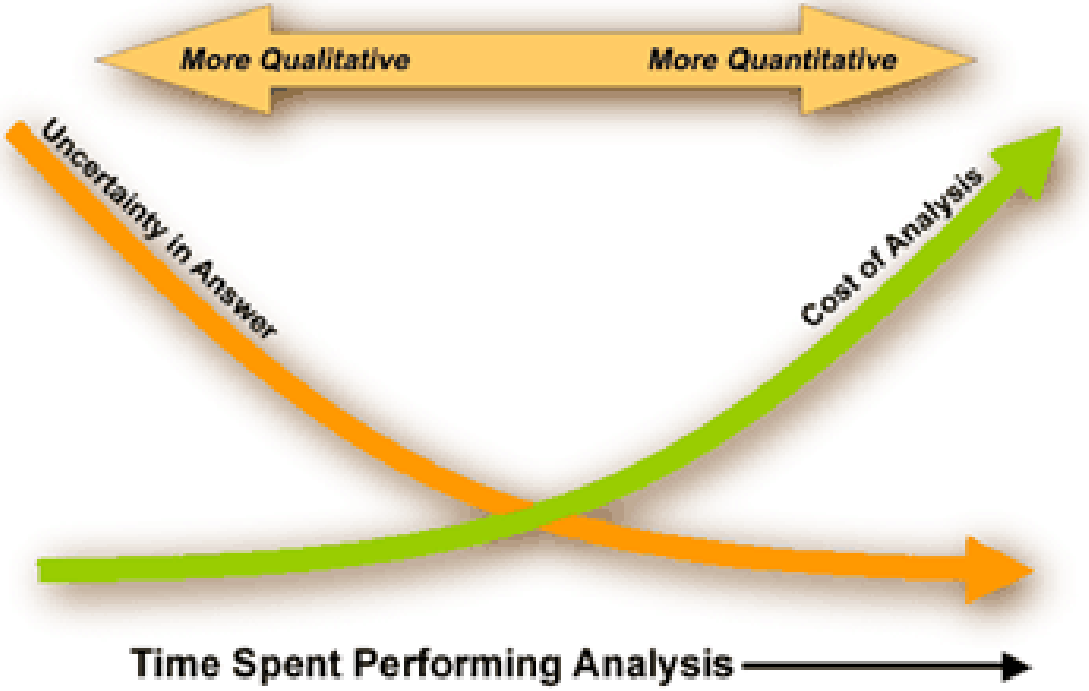
Appendices*

I. **SysML Diagrams and Language Concepts**

J. **Architecture Frameworks In-depth**

*Appendices A through H were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

Hard vs. Soft Analyses



- **Hard:** based on ability to describe issues in terms of mathematical relationships, allowing use of quantitative modeling and simulation
- **Soft:** rely on judgments based on experience by "experts"

What are the relative advantages and disadvantages of each?

Hard vs. Soft Analyses (continued)

Hard Analysis

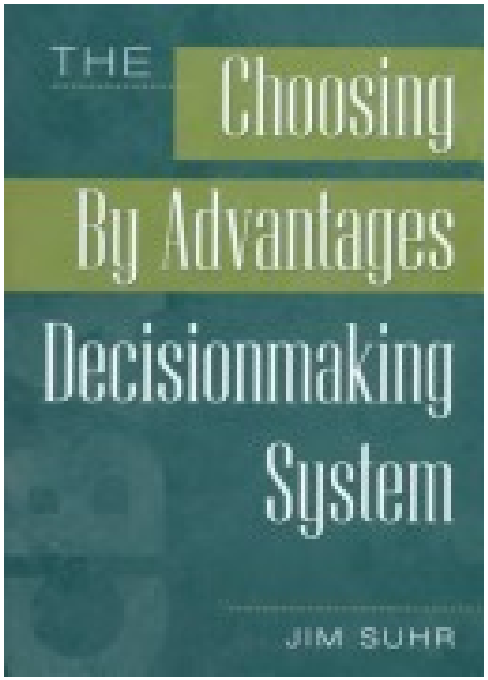
- Advantages
  - Repeatable
  - Supports parametric analysis
  - Reduces bias
  - Makes existing biases more visible
- Disadvantages
  - Requires significant input data
  - Requires significant time and skill
  - Requires understanding mathematical relationships

Soft Analysis

- Advantages
  - Provides quick answers
  - Little quantitative input data
  - No complicated math
  - Complex subjective issues
  - Effort independent of complexity
- Disadvantages
  - Expert dependent/not repeatable
  - Not well suited to parametric analysis
  - Results difficult to interpret relative to quantitative goals

Choosing By Advantages

- To simplify, clarify, and unify decision-making
- To quantify non-monetary benefits or advantages
- Tenets
  - Decisions must be anchored to the relevant facts
  - Base decisions *only* on the importance of advantages, i.e., the relevant and perceivable differences between alternatives



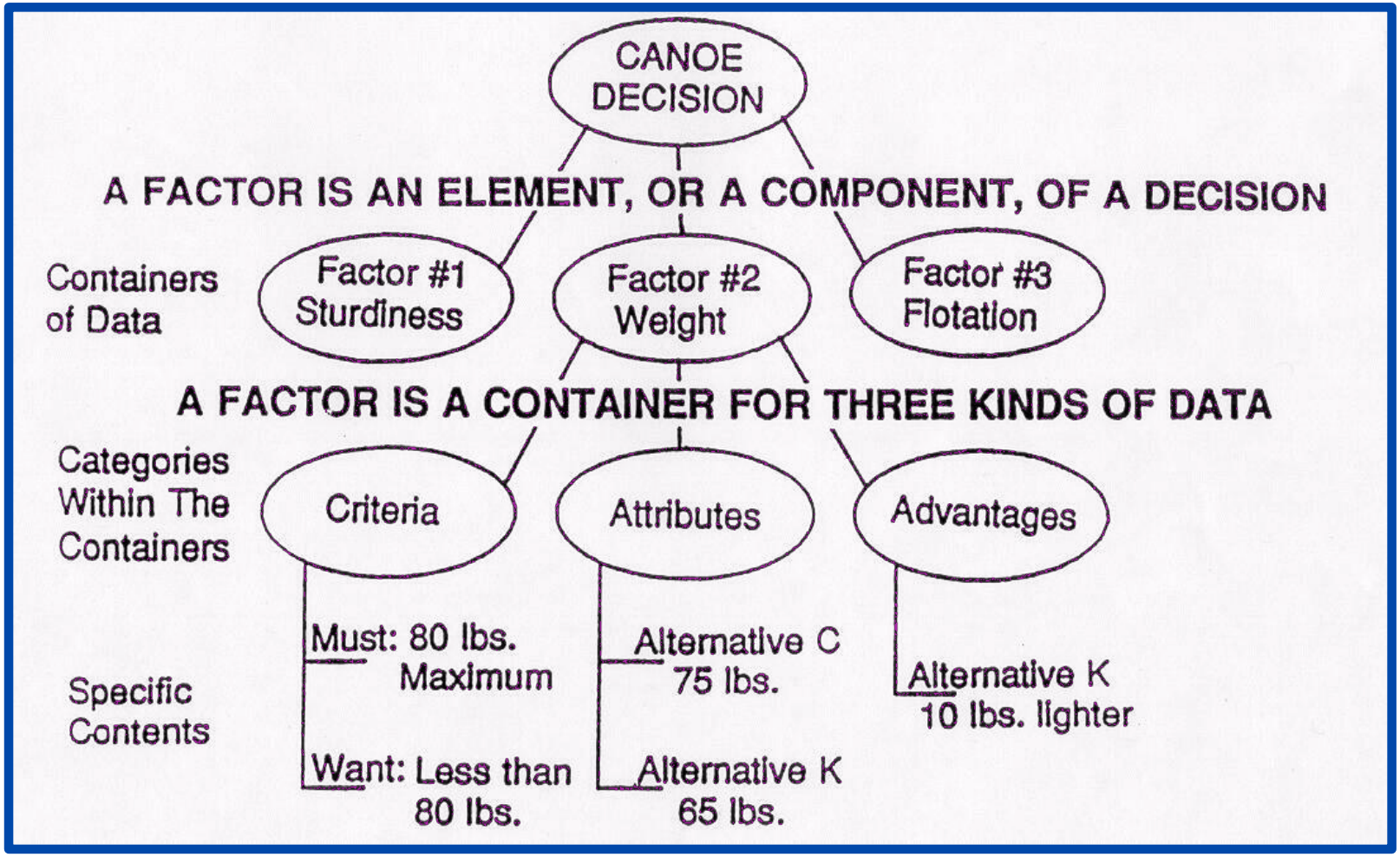
Example: Buying a Canoe







Example: Buying a Canoe



Example: Picking a Tent Site













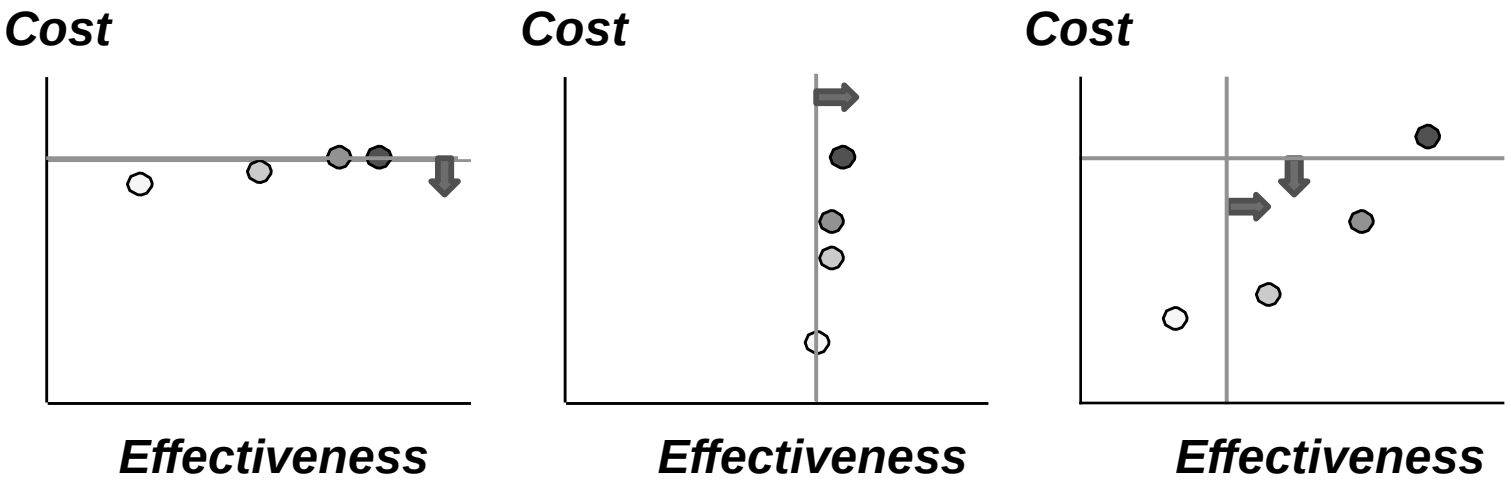
Picking a Tent Site (CBA Matrix)

	Alternatives		
Factors	**Site No. 8**	**Site No. 19**	**Site No. 23**
Water			
Attribute	60 ft away	260 ft away	150 ft away
Advantage	200 ft closer	None	110 ft closer
Tent Spot	40		30
Attribute	Moderately level	Almost level	Quite sloping
Advantage	Moderately more level	Much more level	None
Table	30	70	
Attribute	Without	Without	With
Advantage	None	None	With versus without
Privacy			65
Attribute	Not very private	Very private	Moderately private
Advantage	None	Much more private	Mod. More privacy
		100	45
Importance of	70	170	140
Advantage	$3.00	$4.50	$20.00

Cost Benefit Analysis

Equal Cost ***Variable Cost*** ***Variable Cost***

Variable Effectiveness ***Equal Effectiveness*** ***Variable Effectiveness***



- **Equal Cost-Variable Effectiveness**
- **Variable Cost-Equal Effectiveness**
- **Variable Cost-Variable Effectiveness**

Knapsack Analysis

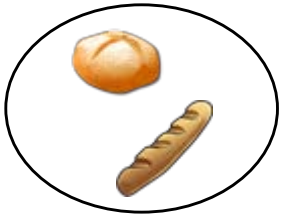
Multiple-choice KP: Select one item from each of *n* classes of items

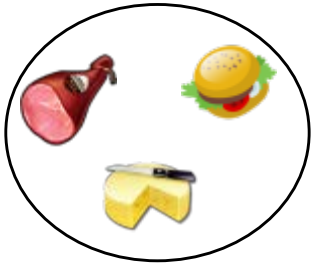
$$\text{maximize} \sum_{i=1}^{n} \sum_{j \in N_i} v_{ij} x_{ij} \quad \text{such that} \quad \sum_{i=1}^{n} \sum_{j \in N_i} c_{ij} x_{ij} \leq C,$$

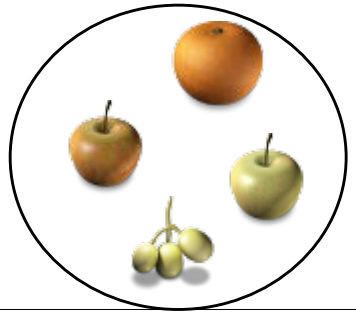
$$\sum_{j \in N_i} x_{ij} = 1, \quad \text{and} \quad x_{ij} \in \{0,1\}, i = 1,2,\ldots,n$$













Automotive Powertrain Selection as a Knapsack Problem



Automotive Powertrain Architecting as a Knapsack Problem

Engine Block
• V8
• V6
• I4

Valve Train
• Fixed 2V
• Fixed 4V
• Variable 4V

Boosting
• None
• Turbocharger
• Supercharger

Throttle
• Mechanical Coupling
• Mech with Cruise Ctrl
• Electronic

Transmission
• Manual 5Sp
• Auto 5Sp
• CVT

Power Split
• 2WD
• 50-50 4WD
• 80-20 AWD

Automotive Powertrain Architecting as a Knapsack Problem

- **Metrics**
  - Fuel economy
  - Torque delivery
  - Drive quality
  - Cost
  - Others?
- **Stakeholders**
  - Customer
  - Engine designers
  - Transmission designers
  - Drive train designers
  - Integration engineers

Automotive Powertrain Architecting as a Knapsack Problem

	V8	V6	I4	F2V	F4V	V4V	No Boost	Tbo	Spr
Fuel	25	30	35	-2	0	+2	0	-3	-5
Torque	250	200	150	-10	+5	+15	0	+25	+35
Drive	2	1	0	0	0	0	0	+1	+2
Cost	3500	3000	2500	250	500	1000	0	500	1000

Automotive Powertrain Architecting as a Knapsack Problem

	Mech	M/CC	ET	M5S	A5S	CVT	2WD	4WD	AWD
Fuel	0	+1	+2	+2	0	+2	0	-3	-2
Torque	0	0	0	0	+20	+5	0	0	0
Drive	0	+1	+2	0	+1	+2	0	+1	+2
Cost	150	225	200	500	1500	3000	0	500	350

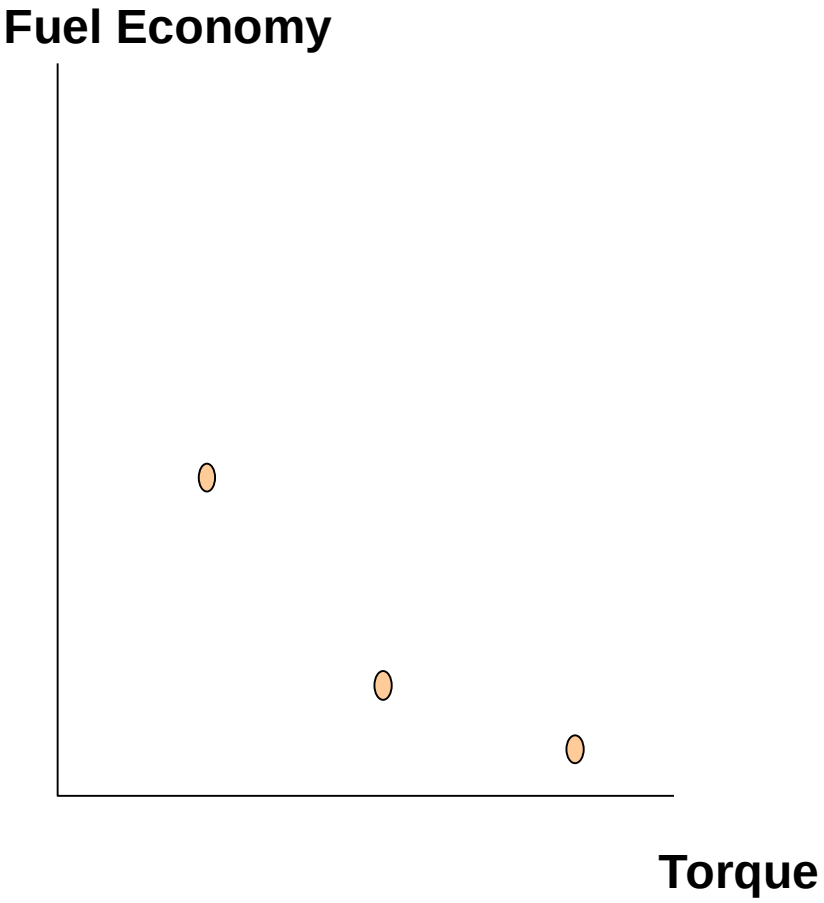
Automotive Powertrain Architecting as a Knapsack Problem

	Block	Valve Train	Boosting	Throttle	Tranny	Power split	Total
	V6	F4V	Tbo	Mech	A5S	2WD	
Fuel	30	0	-3	0	0	0	27
Torque	200	5	25	0	20	0	250
Drive	1	0	1	0	1	0	3
Cost	3000	500	500	25	1500	0	5525

Automotive Powertrain Architecting as a Knapsack Problem

Evaluate Alternatives

	Bill	Matt	Dan's
Fuel	27	36	23
Torque	250	140	305
Drive	3	1	9
Cost	5525	3325	9150



Multi-Attribute Utility Estimation

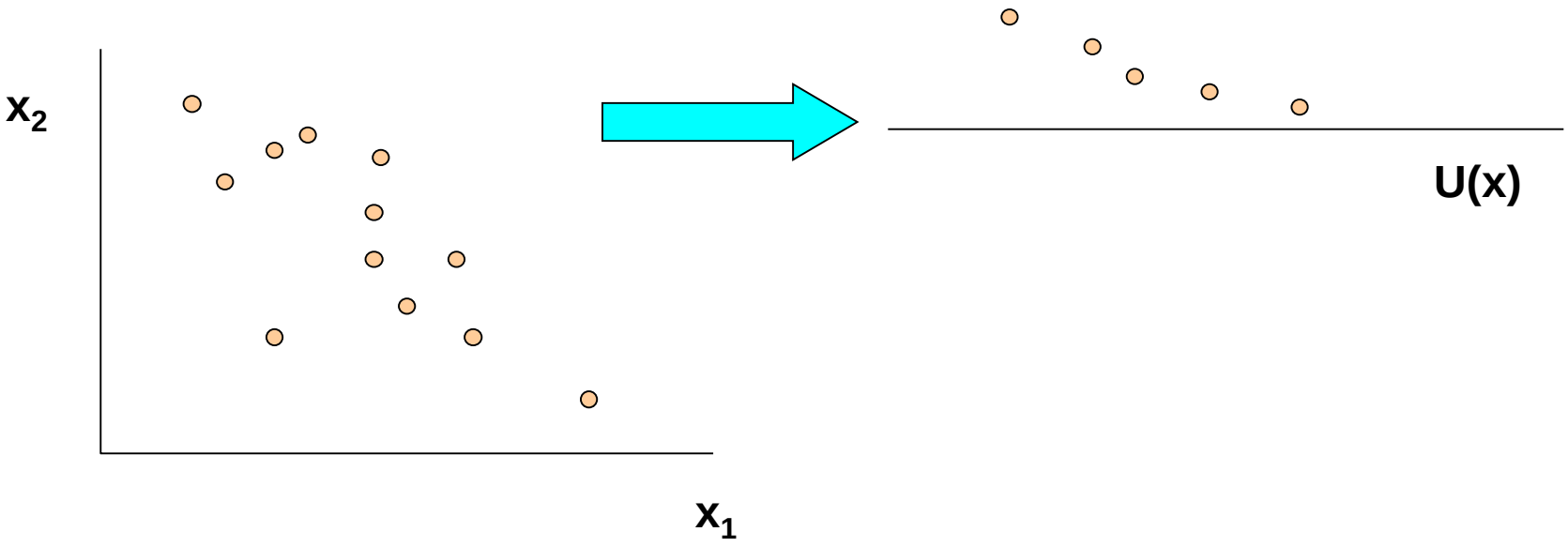
- **Simplest form: additive utility functions**
- **Sometimes called "Value Modeling"**
  - $U(x) = \alpha_1 * x_1 + \alpha_2 * x_2 + \ldots + \alpha_n * x_n$
  - $\alpha_1 + \alpha_2 + \ldots + \alpha_n = 1$
- **Identifies uniquely optimal solutions**
- **Advantages**
  - **Can deal with large number of attributes**
  - **Simple to communicate**
- **Disadvantages**
  - **Often difficult to assess α_1, α_2, and α_n, particularly when multiple stakeholders are involved**
  - **Values of α_1, α_2, and α_n can change with context**
    - **Political or strategic conditions**
    - **System of Systems mission or configuration**

Multi-Attribute Utility Estimation

- **Two-attribute example**

$$U(x) = \alpha_1 * x_1 + \alpha_2 * x_2$$

$$\alpha_1 + \alpha_2 = 1$$

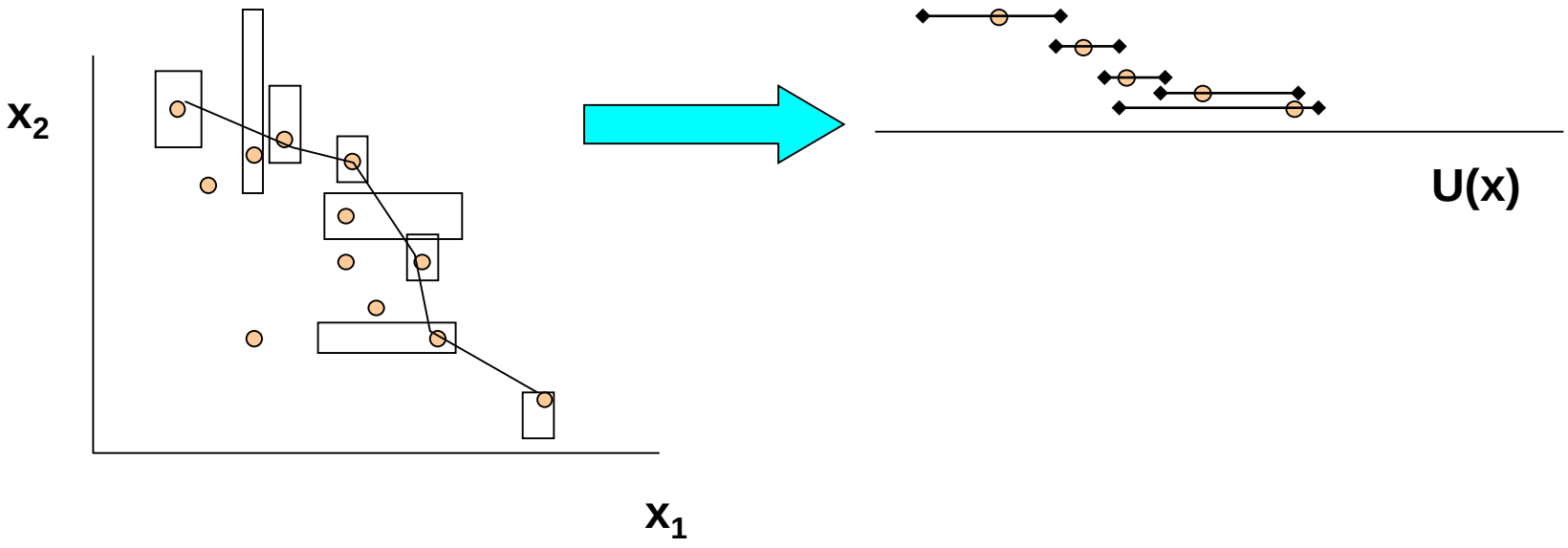


Multi-Attribute Utility Estimation

- **Two-attribute example with uncertainty**

$$U(x) = \alpha_1 * x_1 + \alpha_2 * x_2$$

$$\alpha_1 + \alpha_2 = 1$$



Summary

- Document how you reached you decision so you can explain it now and defend it later if needed
- The process described here will support trades at many different levels (big to small)
- The presented method is only one of many
  - There is a whole field of Decision Analysis to support making trades and making decisions
- Analysis of Alternatives is a name for a very formal trade study to choose the overall project concept(s) for further study or implementation

Analysis Exercise:
Knapsack Problem

- **Each hiker carries a knapsack of capacity 34 and can store 3 kinds of food (A, B, C) which respectively supply energy values (6,4,2) for a consumed capacity of (7,5,3)**
- **Find which food is to be put in the knapsack so that the energy value is maximal**
- **Be prepared to share your results with the class!**



Other References for Further Reading

- **DoD Defense Acquisition Guidebook**
  - **https://dag.dau.mil/ -- latest version**
- **NASA Approaches for Conducting Alternatives Analysis**
  - **http://nodis3.gsfc.nasa.gov/displayDir.cfm?Internal_ID=N_PR_2830_0001_&page_name=AppendixE**

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		

All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?



Systems Engineering Technical Short Course

(Session 4: System Architecture and Design)

Lesson 5: Model-Based Systems Engineering

Session 4 – System Architecture and Design

Lessons

1. **Program Overview Update**
2. **System Architecture Basics**
3. **Systems Design Process**
4. **Analysis of Alternatives**
5. **Model-Based Systems Engineering**
6. **Special Topics (if time permits)**

- Motivation
- SysML Overview
- Distilled Example

Appendices*

I. **SysML Diagrams and Language Concepts**

J. **Architecture Frameworks In-depth**

*Appendices A through H were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

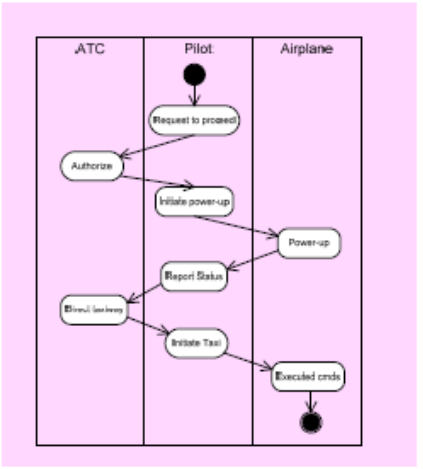
SE Practices for Describing Systems

Past



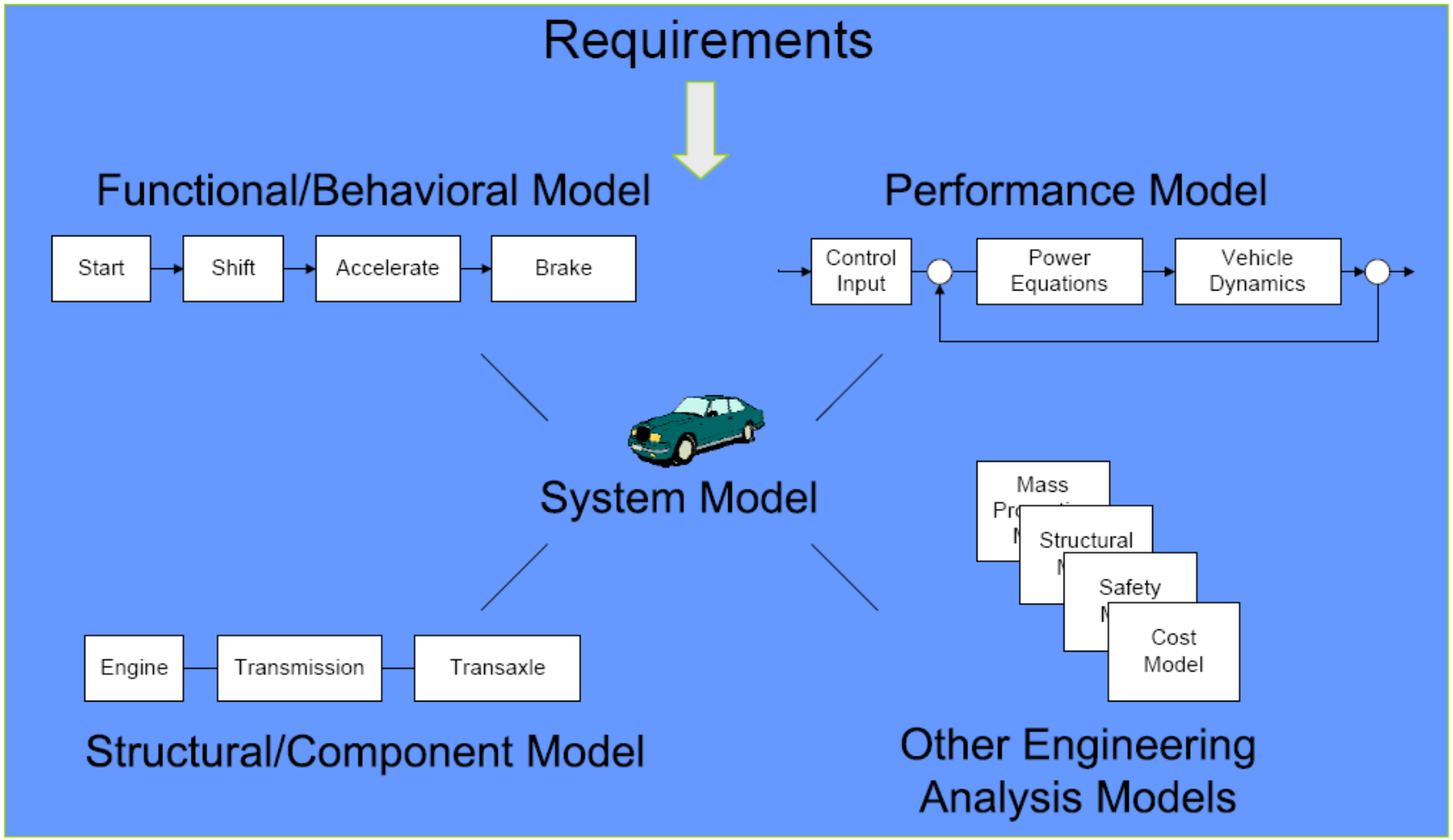
- Specifications
- Interface requirements
- System design
- Analysis & Trade-off
- Test plans

Future



Moving from Document centric to Model centric

System Modeling



Integrated System Model Must Address Multiple Aspects of a System

Model-Based Systems Engineering Benefits

- **Shared understanding of system requirements and design**
  - **Validation of requirements**
  - **Common basis for analysis and design**
  - **Facilitates identification of risks**

- **Assists in managing complex system development**
  - **Separation of concerns via multiple views of integrated models**
  - **Supports traceability through hierarchical system models**
  - **Facilitates impact analysis of requirements and design changes**
  - **Supports incremental development and evolutionary acquisition**

Model-Based Systems Engineering Benefits (concluded)

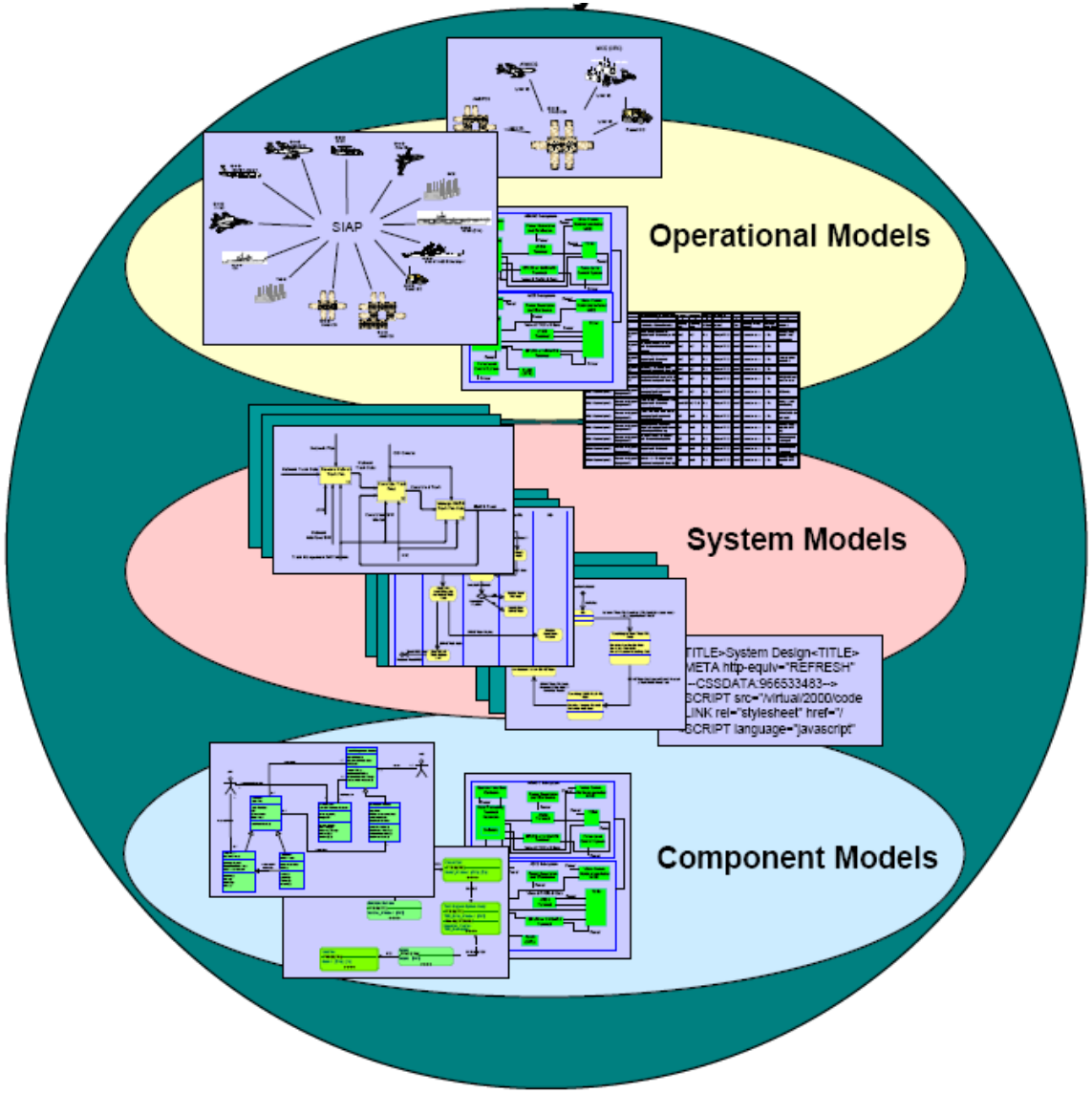
- Improved design quality
  - Reduced errors and ambiguity
  - More complete representation
- Supports early and on-going verification and validation to reduce risk
- Provides value through life-cycle (e.g., training)
- Enhances knowledge capture and retention

System of Systems (SoS)

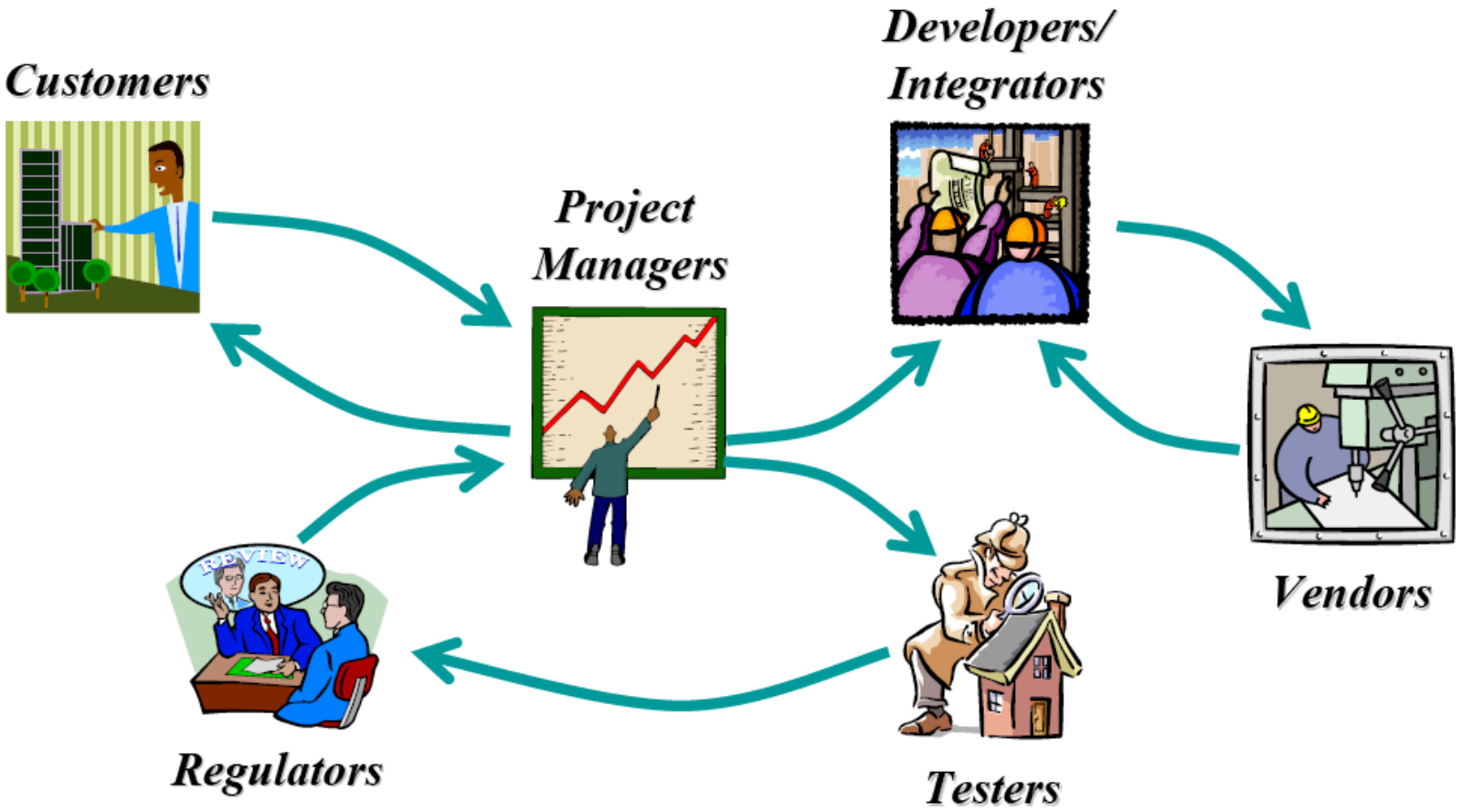


Modeling Needed to Manage System Complexity

Modeling at Multiple Levels



Stakeholders involved in System Acquisition



Modeling Needed to Improve Communications

Session 4 – System Architecture and Design

Lessons

1. **Program Overview Update**
2. **System Architecture Basics**
3. **Systems Design Process**
4. **Analysis of Alternatives**
5. **Model-Based Systems Engineering**
6. **Special Topics (if time permits)**

- Motivation
- SysML Overview
- Distilled Example

Appendices*

I. **SysML Diagrams and Language Concepts**

J. **Architecture Frameworks In-depth**

*Appendices A through H were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

What is SysML?

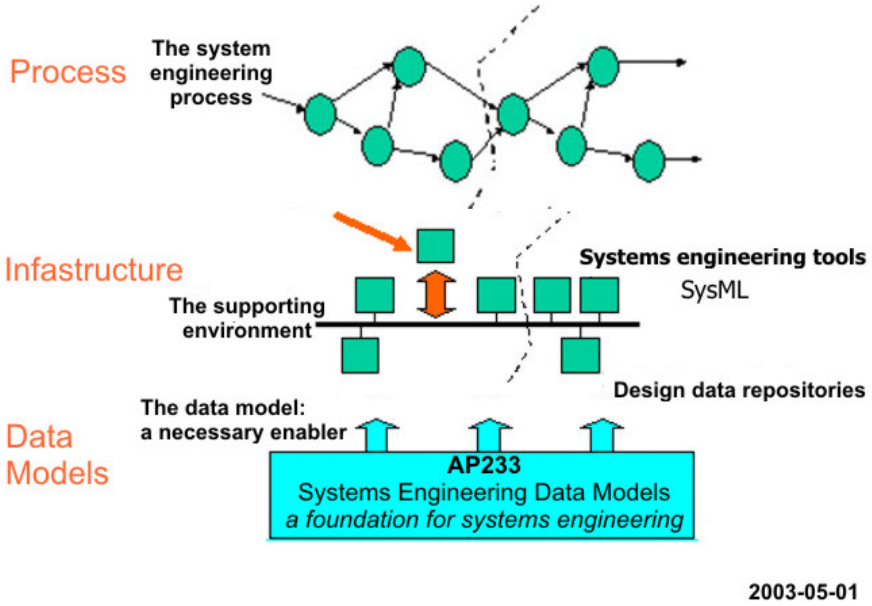
- A graphical modeling language in response to the UML for Systems Engineering RFP developed by OMG, INCOSE and AP233
- It IS a visual modeling language that provides…
  - Semantics (meaning)
  - Notation (representation of meaning)
- It IS NOT a methodology or a tool
  - SysML is methodology and tool independent

SysML is Critical Enabler for Model Driven SE

AP233 and ISO 10303 (STEP)

- **AP233 is a data exchange protocol for systems engineering data based on ISO 10303 (STEP)**

- **STEP is a standard to describe, represent and exchange industrial data in a computer interpretable format**
  - **The intent of AP233 is to support the whole system development life cycle ranging from requirements definition to system verification and validation**



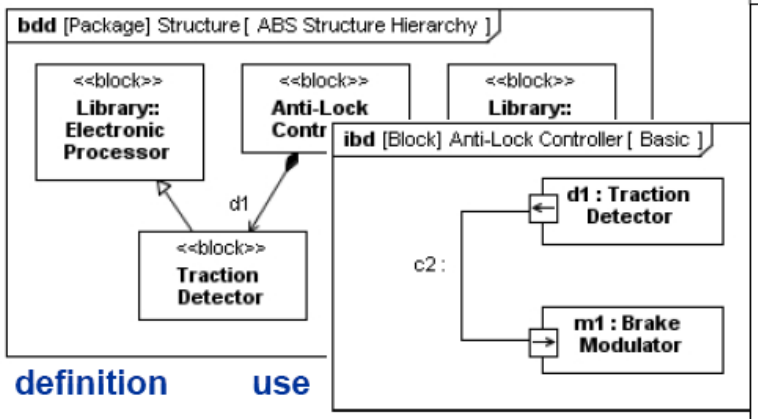
See http://pdesinc.aticorp.org and http://www.ap233.org

SysML Describes System Aspects

- Structural composition, interconnection, and classification
- Function-based, message-based, and state-based behavior
- Constraints of the physical and performance properties
- Allocations between behavior, structure, and constraints (e.g. functions allocated to components)
- Requirements and their relationship to other requirements, design elements, and test cases

Four Pillars of SysML

1. Structure

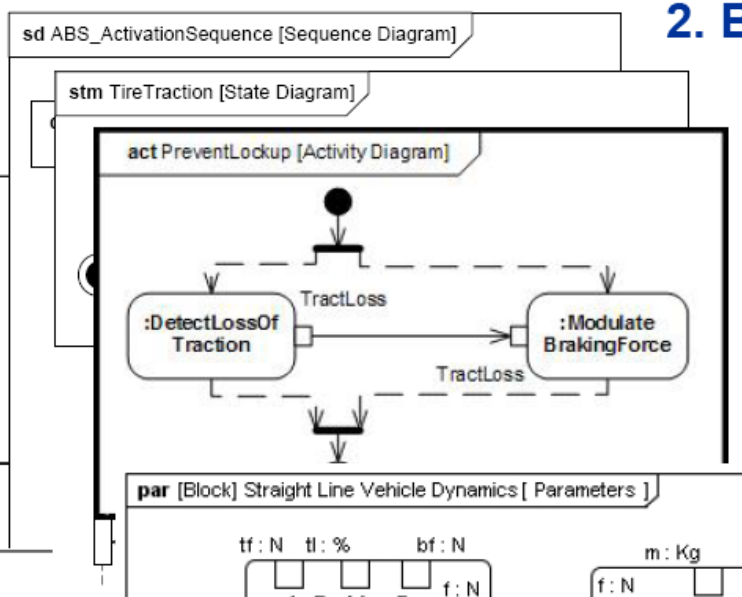


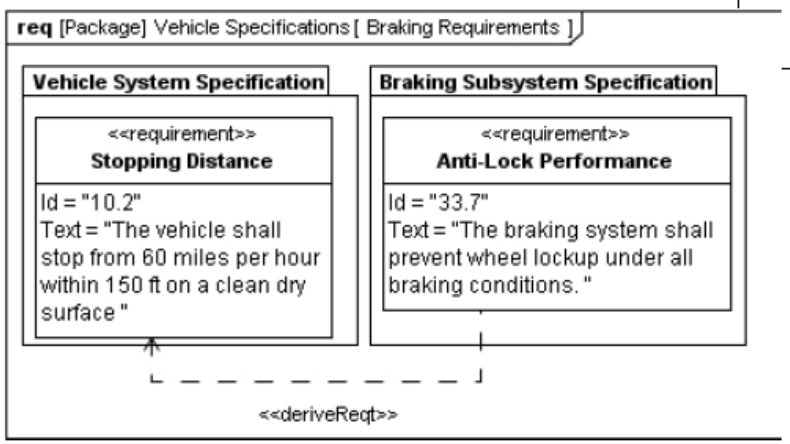
2. Behavior

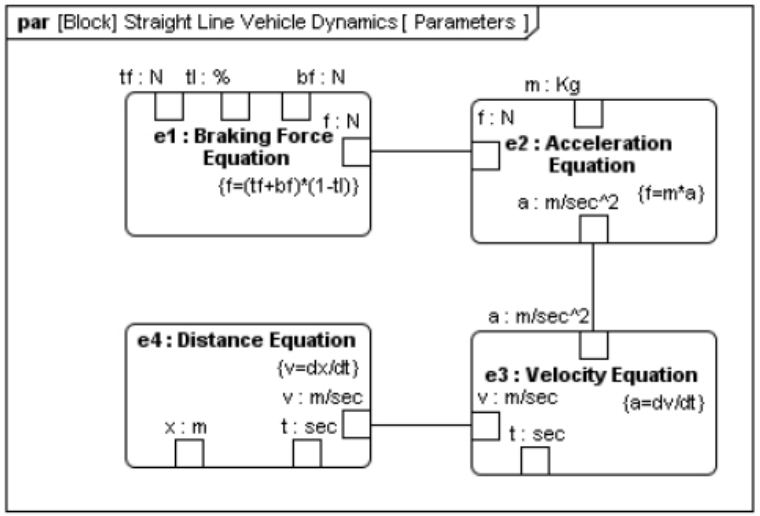
interaction

state machine

activity/ function





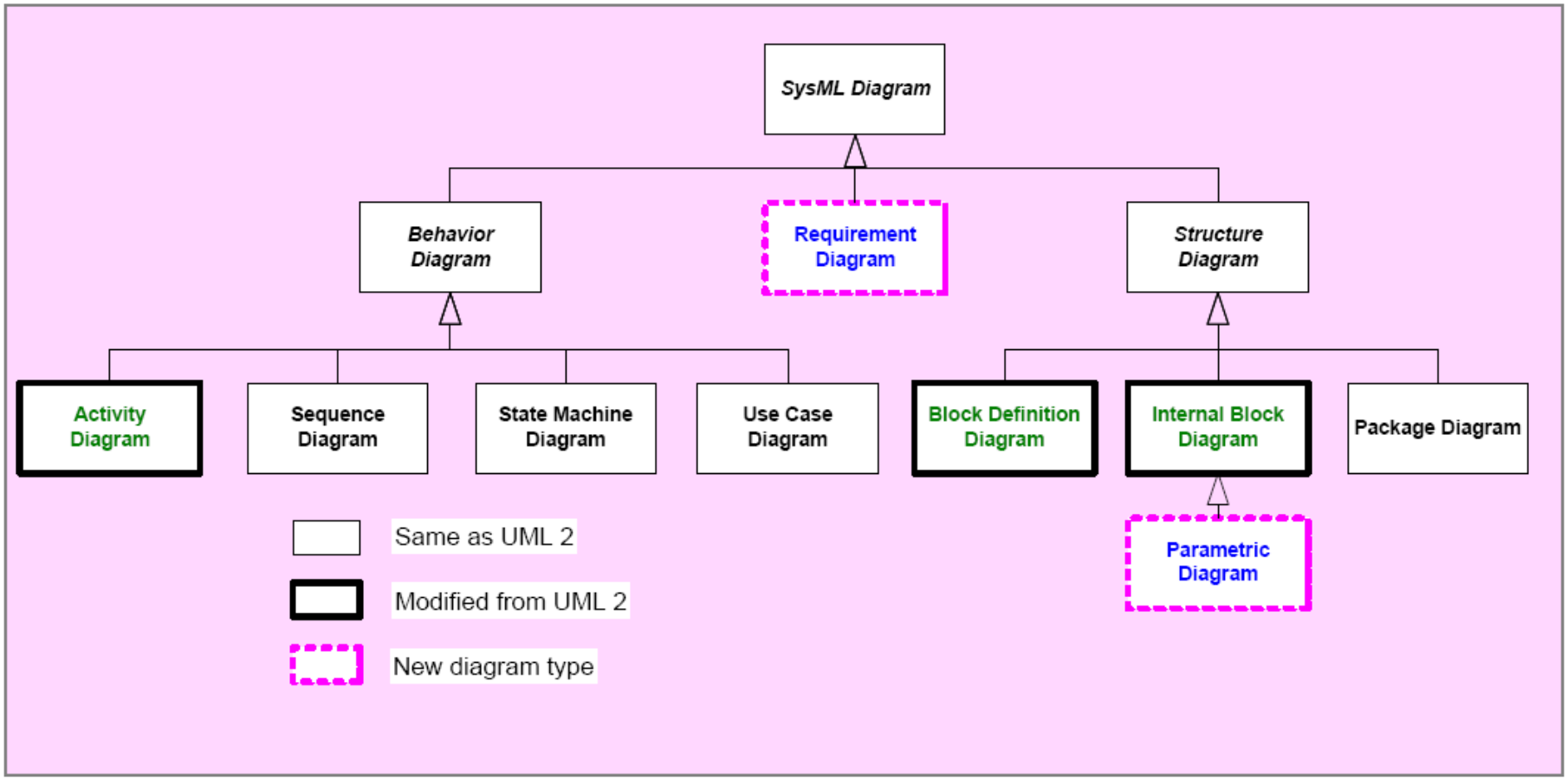


3. Requirements

4. Parametrics

Antilock Braking System

SysML Diagram Taxonomy



Detailed descriptions and illustrations of these SysML diagrams can be found in Appendix J, along with practical examples of their applications.

SysML Structure Diagrams

- **Block Definition Diagram (bdd)**
  - Represents structural elements called blocks, and their composition and classification
- **Internal Block Diagram (ibd)**
  - Represents interconnections and interfaces between the parts of a block
- **Parametric Diagram (par)**
  - Represents constraints on property values, such as ***F = ma***, used to support engineering analysis
- **Package Diagram (pkg)**
  - Represents the organization of a model in terms of packages that contain model elements

SysML Behavior Diagrams

- **Activity Diagram (act)**
  - **Represents behavior in terms of ordering actions based on the availability of inputs, outputs, and control, and how the actions transform the inputs to outputs**
- **Sequence Diagram (sd)**
  - **Represents behavior in terms of a sequence of messages exchanged between parts**
- **State Machine Diagram (stm)**
  - **Represents behavior of an entity in terms of its transitions between states triggered by events**
- **Use Case Diagram (uc)**
  - **Represents functionality in terms of how a system or other entity is used by external entities (i.e. actors) to accomplish a set of goals.**

SysML Requirements Diagram

- **Requirements Diagram (req)**
  - **Represents text-based requirements and their relationship to other requirements, design elements, and test cases to support requirements traceability**

SE Activities and SysML

- Capture and analyze black box system requirements
  - Capture text-based requirements in a requirements management tool
  - Import requirements into the SysML tool
  - Identify top-level functionality in terms of system use cases
  - Capture the traceability between the use cases and requirements
  - Model the use case scenarios as activity diagrams, sequence diagrams and/or state machine diagrams
  - Create a context diagram
  - Identify test cases to support system verification

SE Activities and SysML

- Develop one or more candidate system architectures to satisfy the requirements
  - *Decompose* the system using the block definition diagram
  - Define the *interaction* among the parts using activity or sequence diagrams
  - Define the *interconnection* among the parts using the internal block diagram
- Perform engineering and trade-off analysis to evaluate and select the preferred architecture
  - Capture the *constraints* on system properties using the parametric diagram to support analysis of performance, reliability, cost, and other critical properties
  - Perform the *engineering analysis* to determine the budgeted values to the system properties

SE Activities and SysML

- Specify component requirements and their traceability to system requirements
  - Capture the functional, interface, and performance requirements for each component (block) in the architecture
  - Trace component requirements to the system requirements
- Verify that the system design satisfies the requirements by executing system-level test cases

Session 4 – System Architecture and Design

Lessons

1. **Program Overview Update**
2. **System Architecture Basics**
3. **Systems Design Process**
4. **Analysis of Alternatives**
5. **Model-Based Systems Engineering**
6. **Special Topics (if time permits)**

- Motivation
- SysML Overview
- Distilled Example



Appendices*

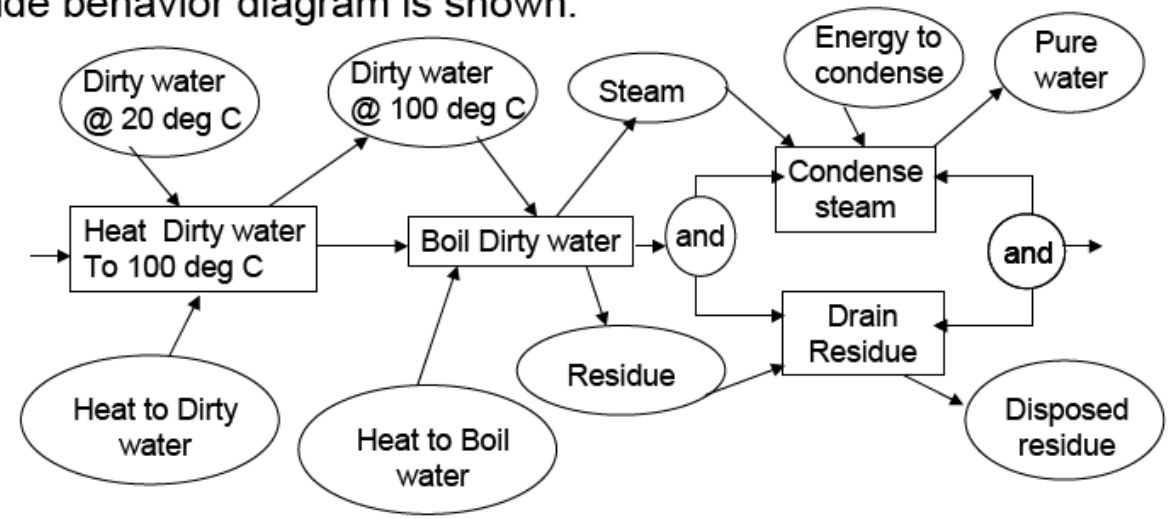
I. **SysML Diagrams and Language Concepts**

J. **Architecture Frameworks In-depth**

*Appendices A through H were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

Problem Statement

- Describe a system for purifying dirty water.
  - Heat dirty water and condense steam are performed by a Counter Flow Heat Exchanger
  - Boil dirty water is performed by a Boiler
  - Drain residue is performed by a Drain
  - The water has properties: vol = 1 liter, density 1 gm/cm3, temp 20 deg C, specific heat 1cal/gm deg C, heat of vaporization 540 cal/gm.
- A crude behavior diagram is shown.



What are the real requirements?
How do we design the system?

Distiller Types

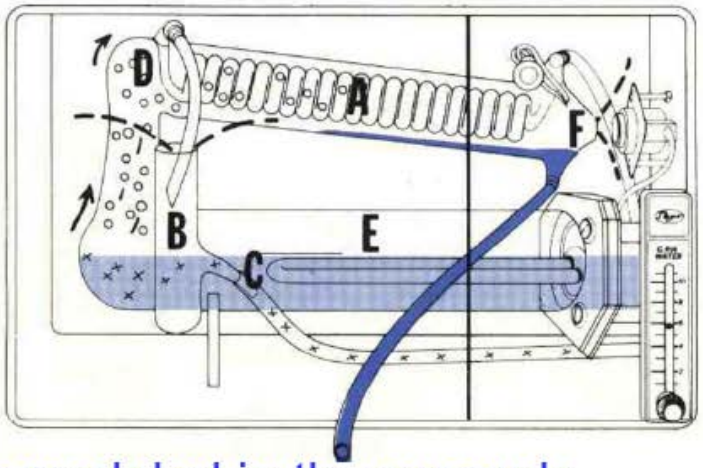
Batch Distiller





Continuous Distiller





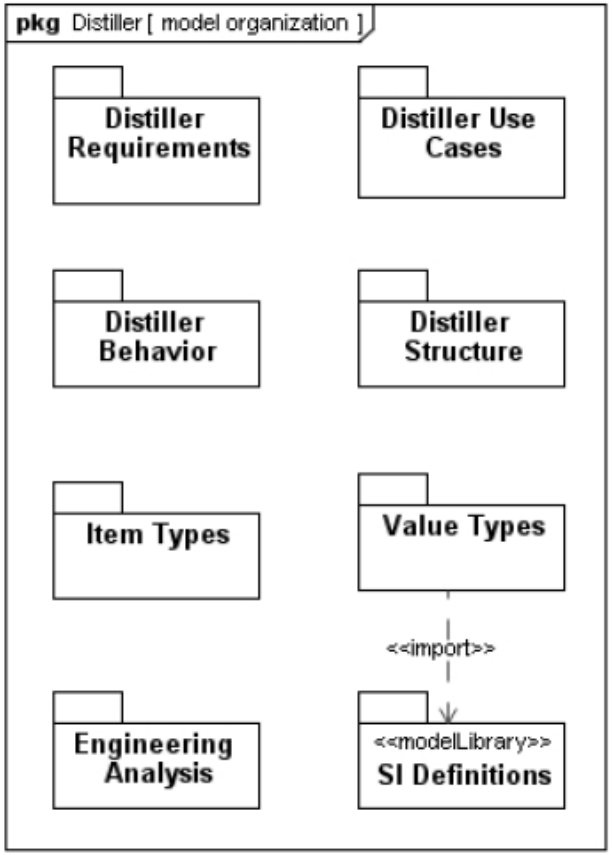
Note: Not all aspects of the distiller are modeled in the example

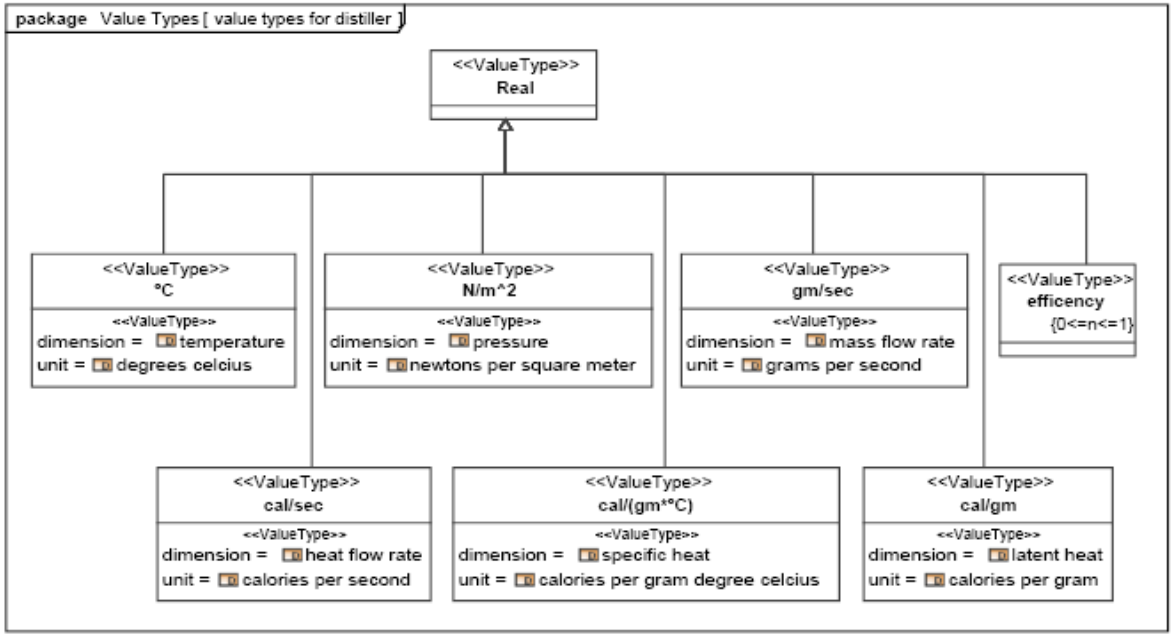
Modeling Process Used

- Organize the model, identify libraries needed
- List requirements and assumptions
- Model behavior
  - In similar form to problem statement
  - Elaborate as necessary
- Model structure
  - Capture implied inputs and outputs
    - segregate I/O from behavioral flows
  - Allocate behavior onto structure, flow onto I/O
- Capture and evaluate parametric constraints
  - Heat balance equation
- Modify design as required to meet constraints
- Model the user interaction
- Modify design to reflect user interaction

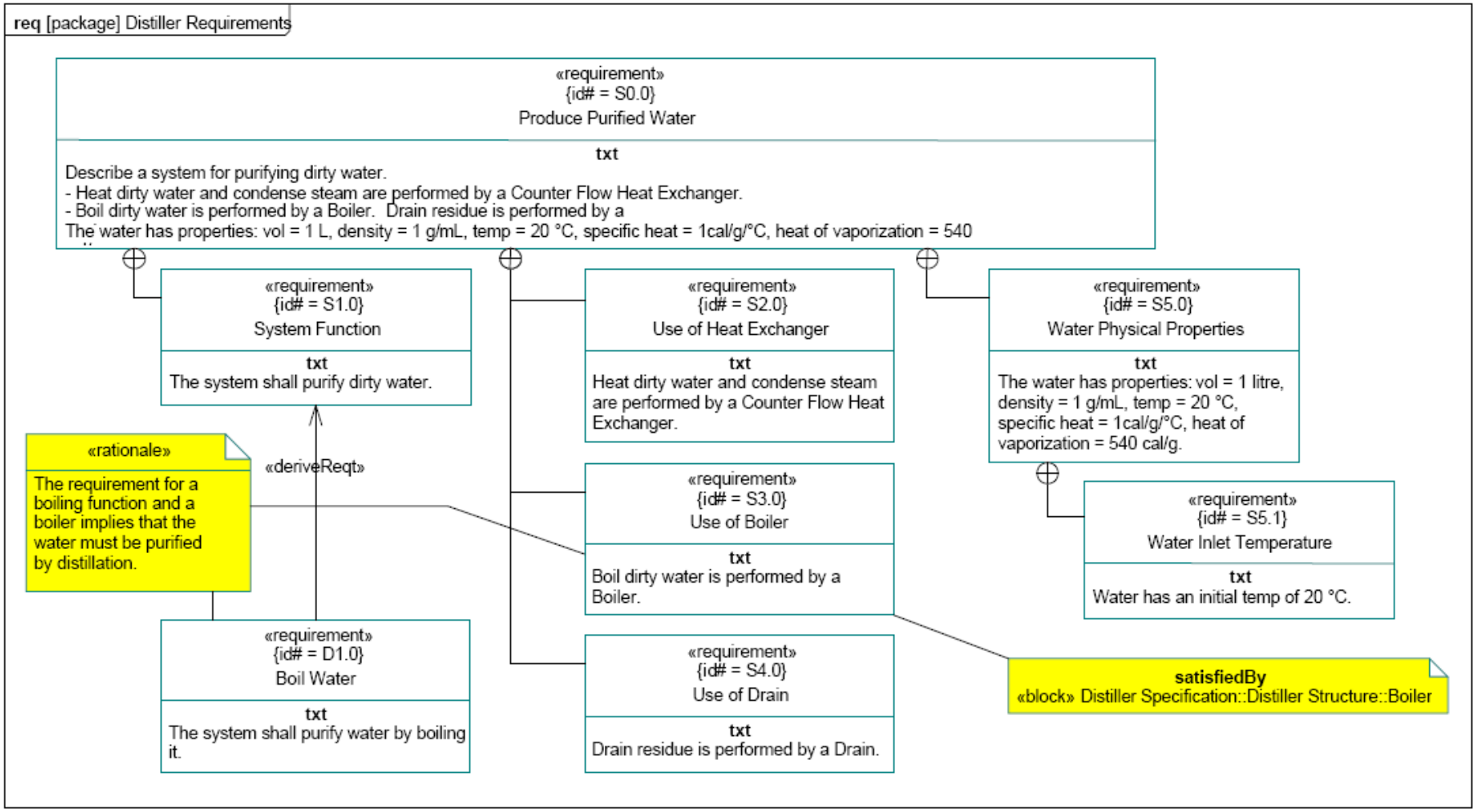
Package Diagrams

Model Organization and Value Types





Original Requirements



Tool: ARTiSAN Studio 6.1

Requirements Tables

table [requirement]OriginalStatement[Decomposition of OriginalStatement]

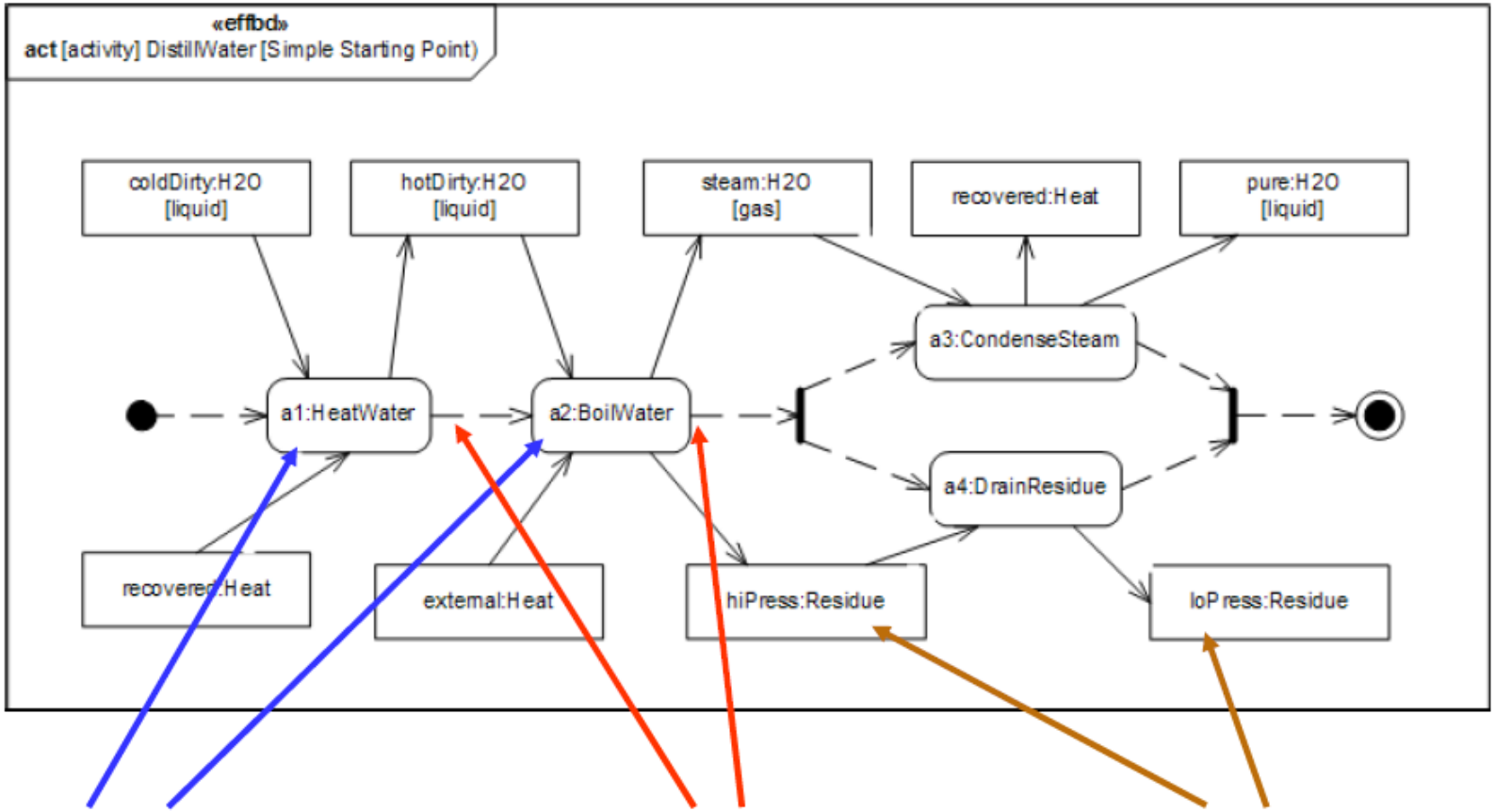
id	name	text
S0.0	OriginalStatement	Describe a system for purifying dirty water. …
S1.0	PurifyWater	The system shall purify dirty water.
S2.0	HeatExchanger	Heat dirty water and condense steam are performed by a …
S3.0	Boiler	Boil dirty water is performed by a Boiler.
S4.0	Drain	Drain residue is performed by a Drain.
S5.0	WaterProperties	water has properties: density 1 gm/cm3, temp 20 deg C, …
S5.1	WaterInitialTemp	water has an initial temp 20 deg C

table [requirement] PurifyWater[Requirements Tree]

id	name	relation	id	name	Rationale
S1.0	PurifyWater	deriveReqt	D1.0	DistillWater	The requirement for a boiling function and a boiler implies that the water must be purified by distillation

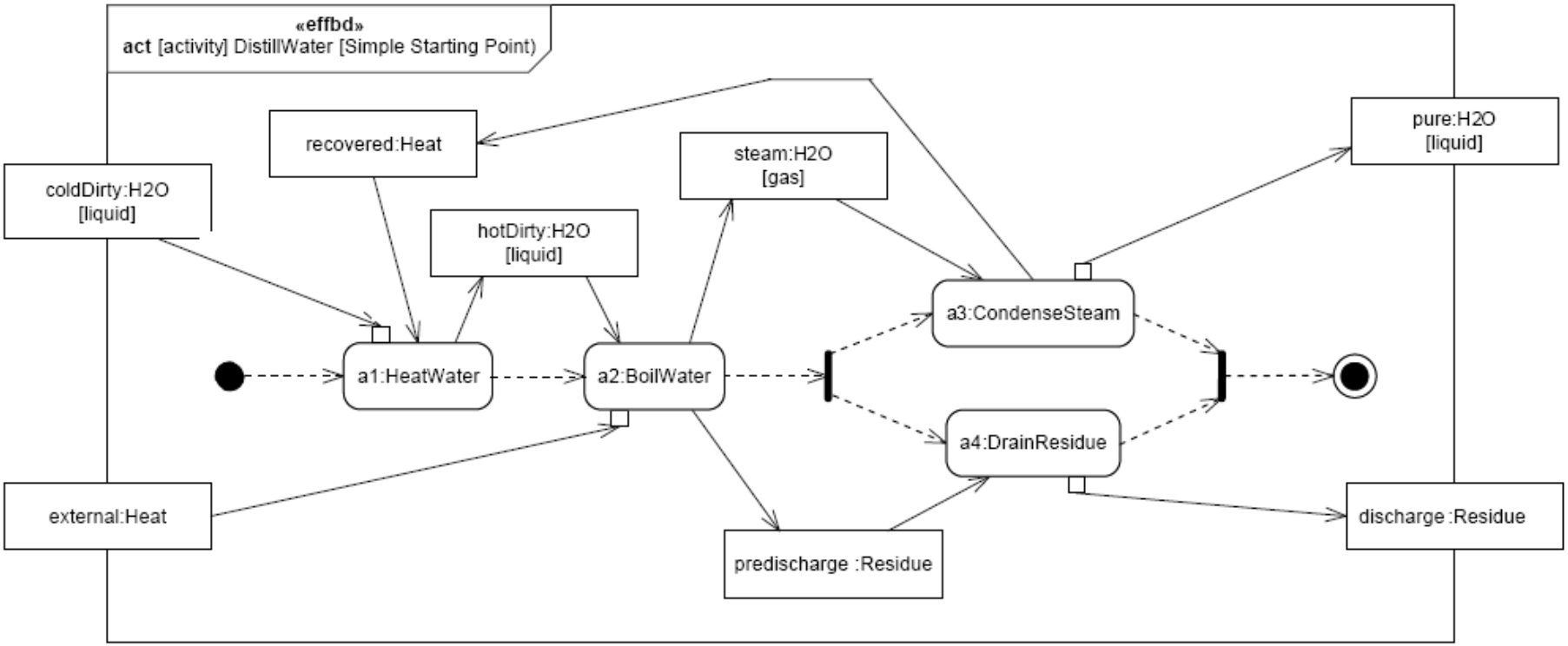
Initial Activity Diagram for DistillWater

This activity diagram applies the SysML EFFBD profile, and formalizes the diagram in the problem statement.



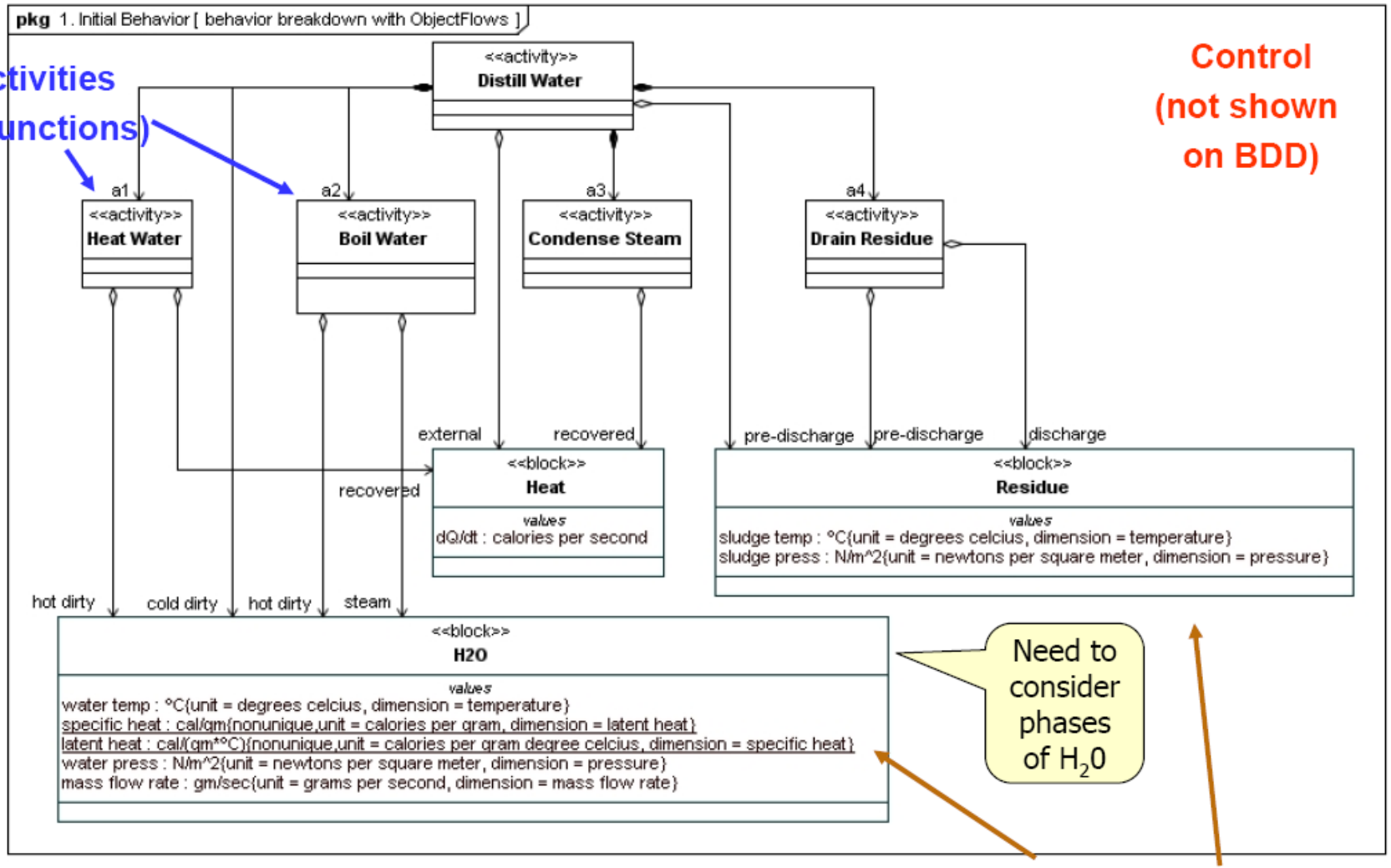
Activity Diagram

(Serial Behavior of Batch Distiller)

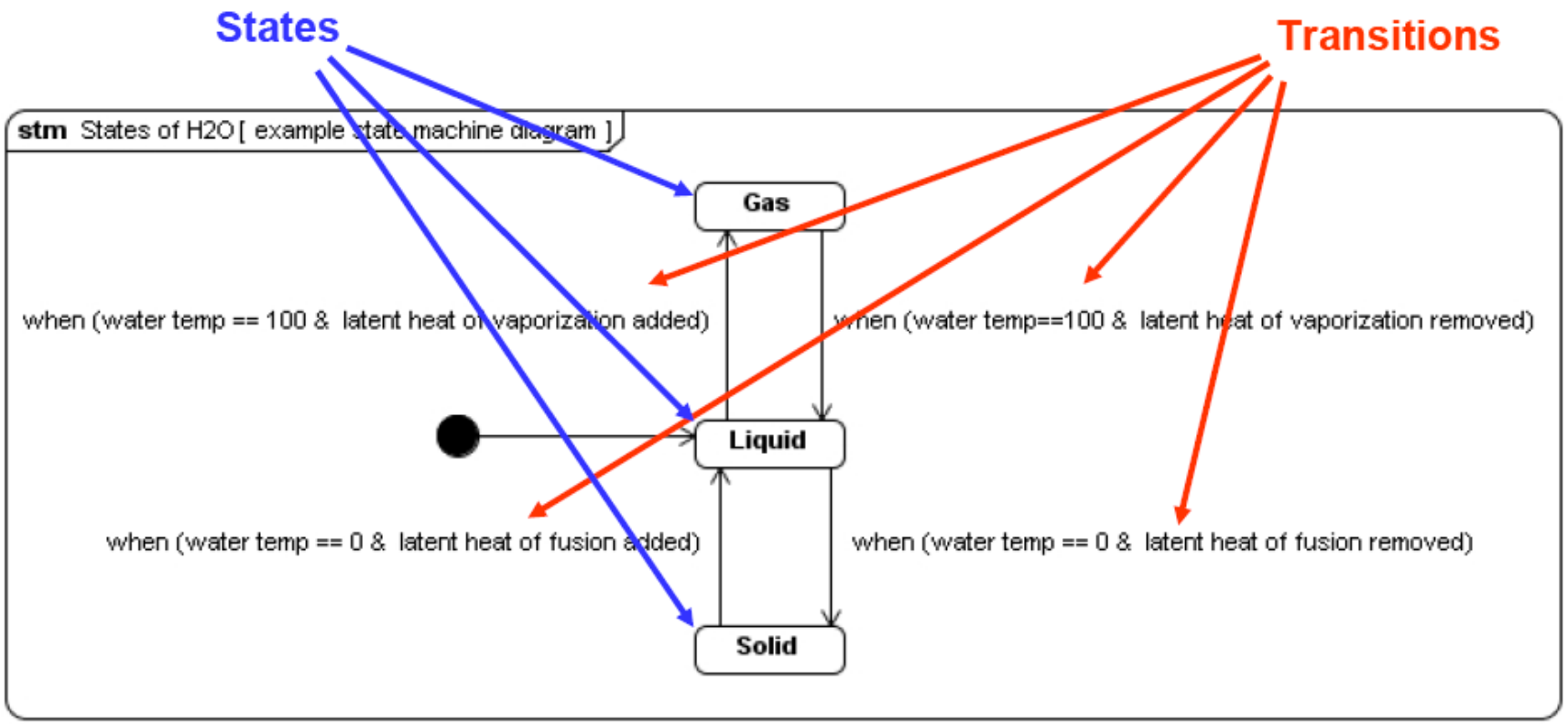


Block Definition Diagram

Distiller Behavior

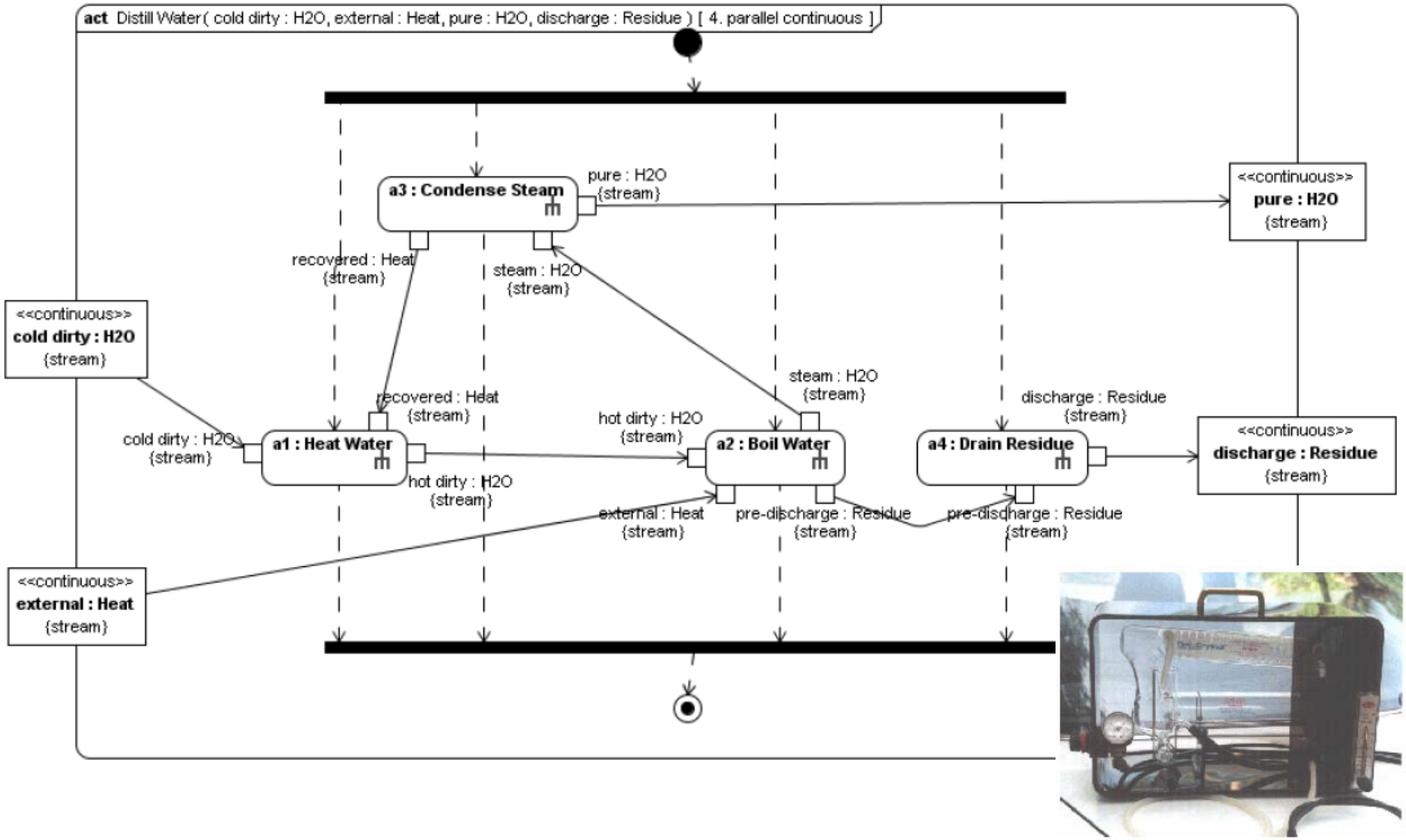


State Diagram



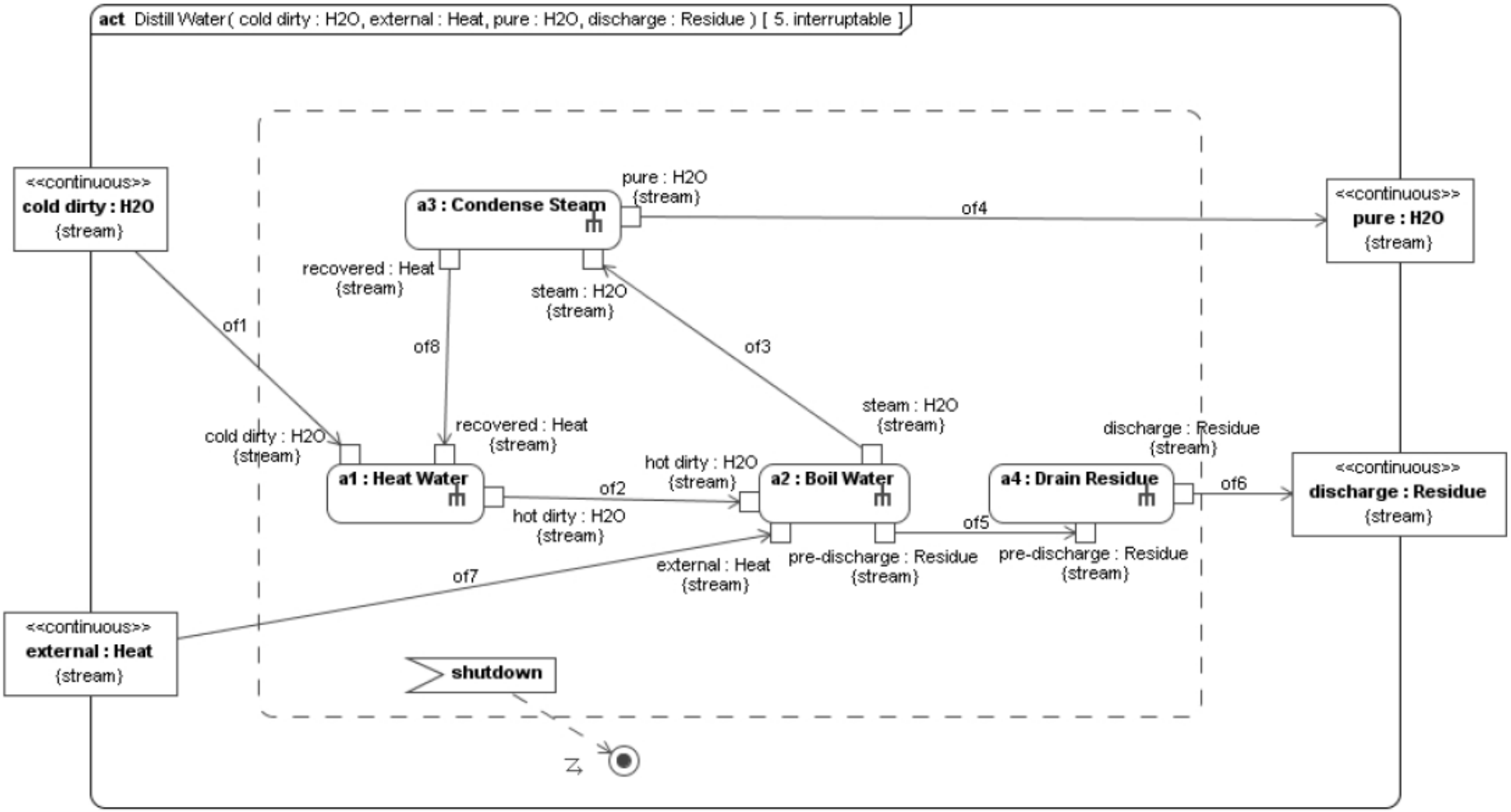
Activity Diagram

(Continuous Distiller)

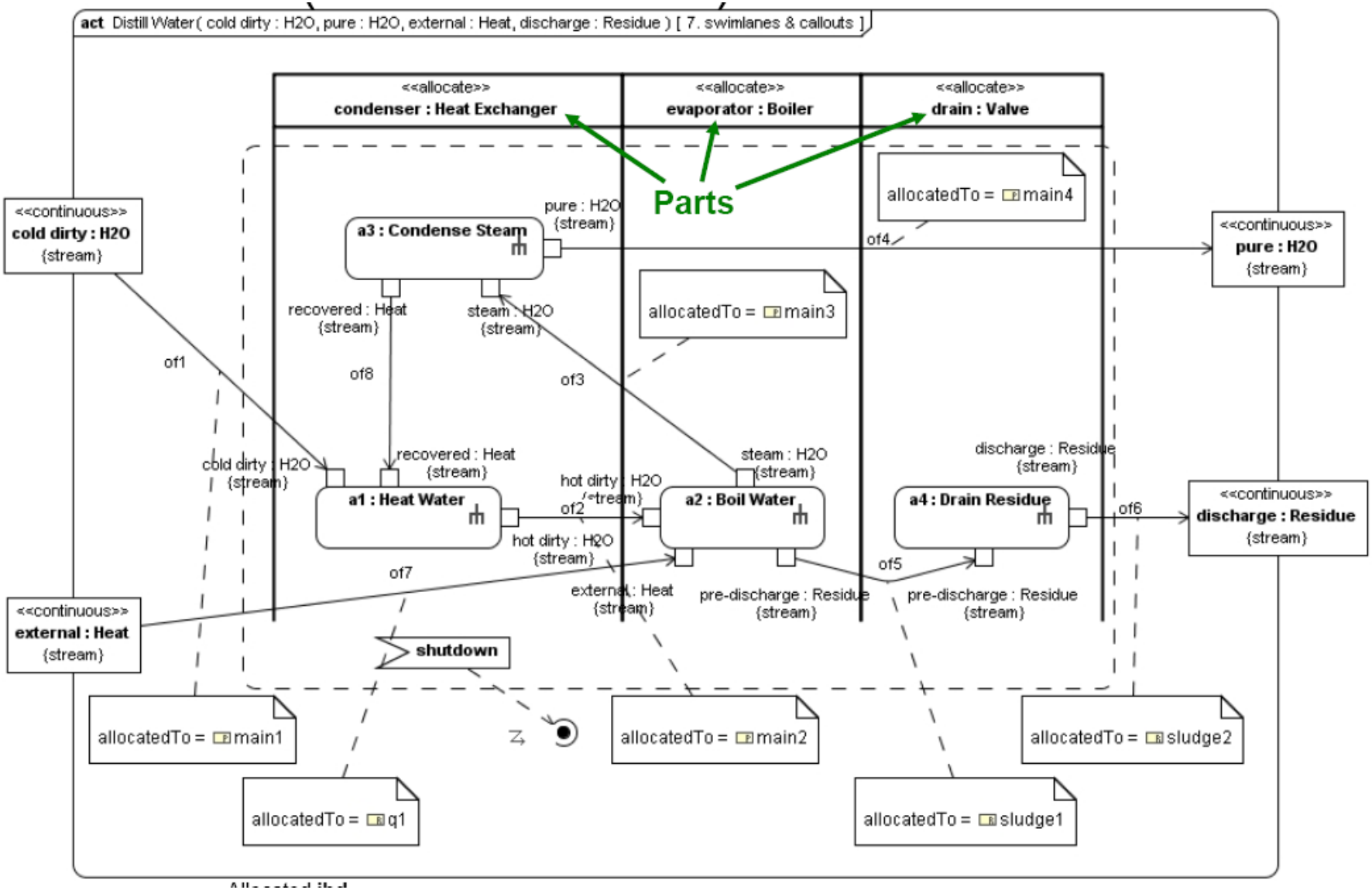


Activity Diagram

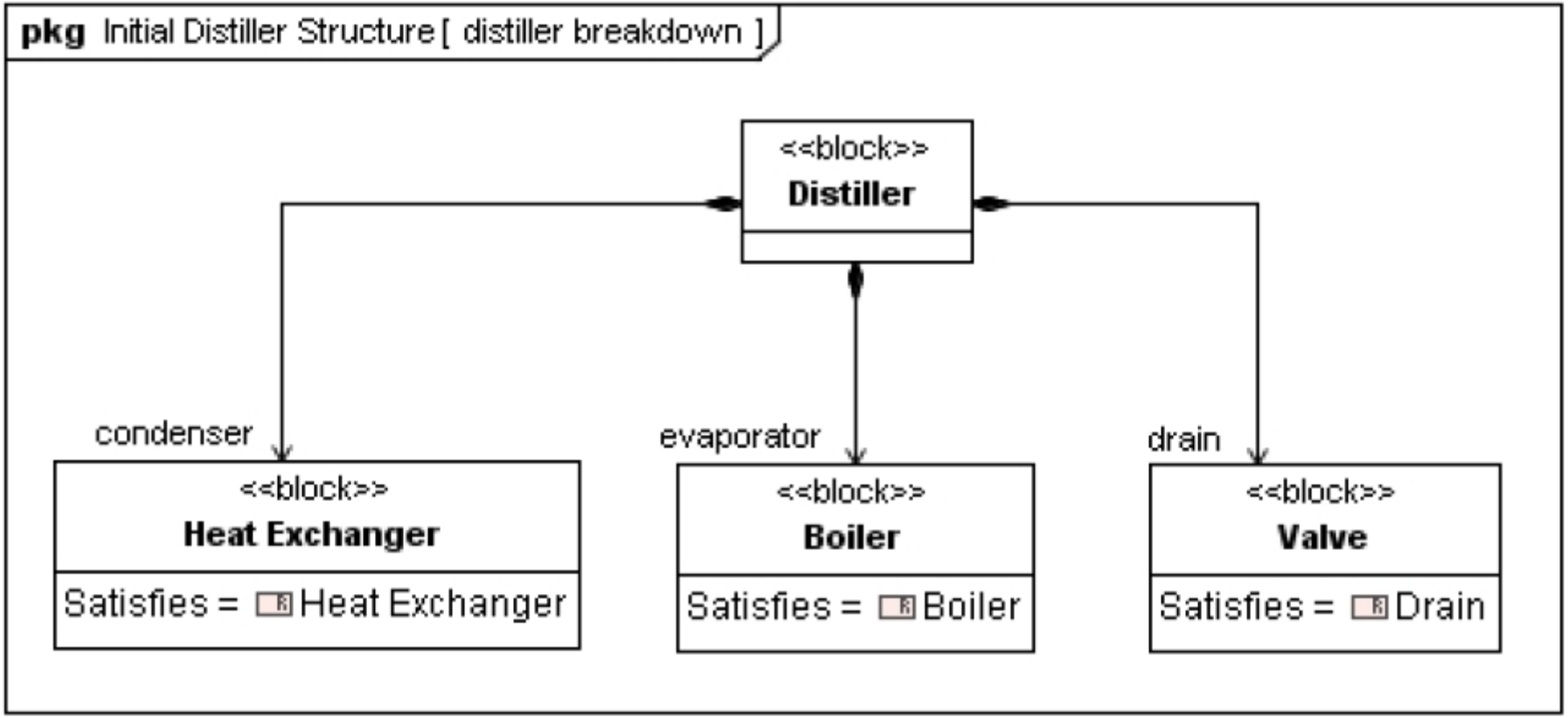
No Control Flow, ActionPin Notation, Continuous



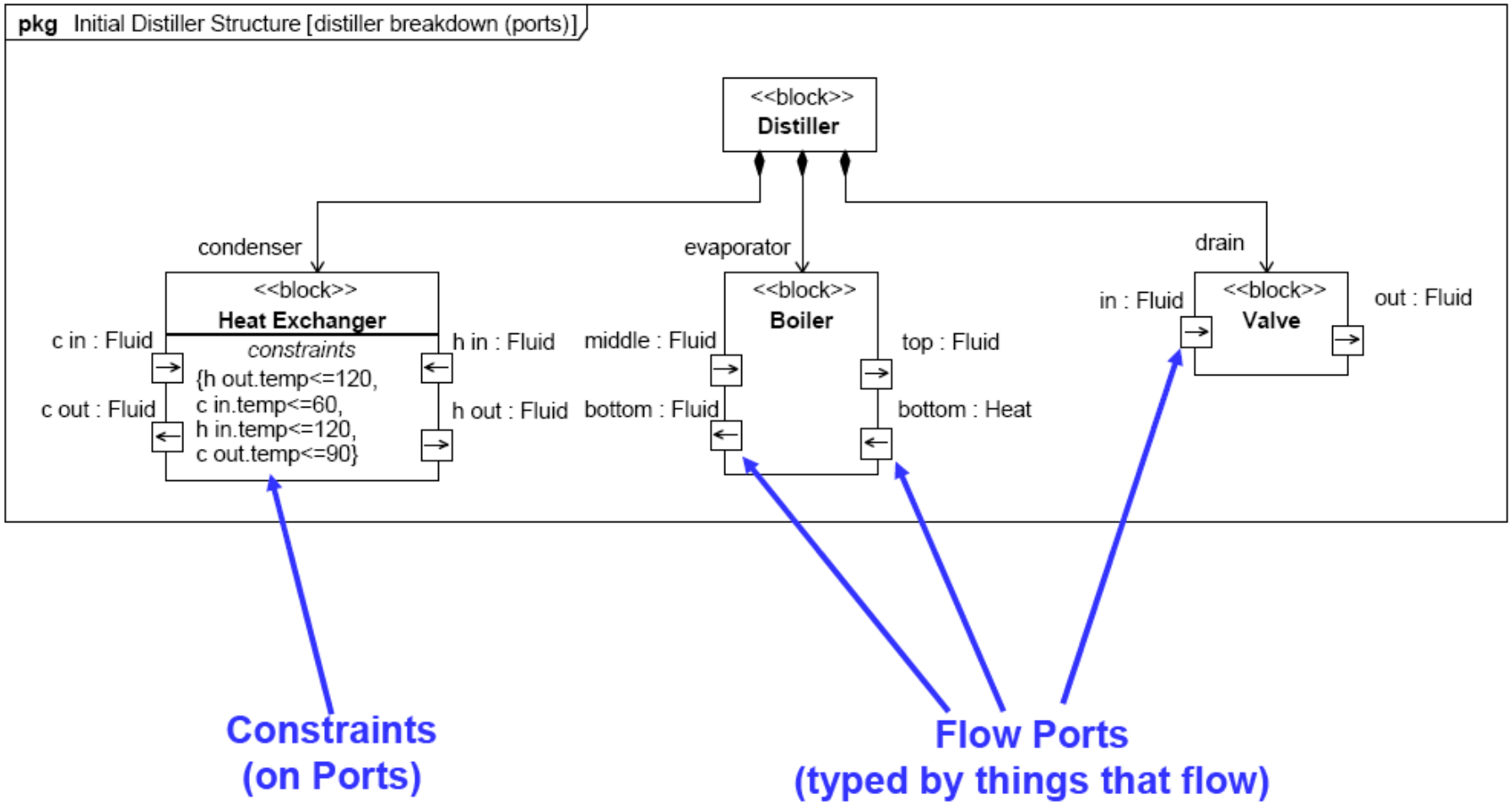
Activity Diagram with Swim Lanes



Block Definition Diagram

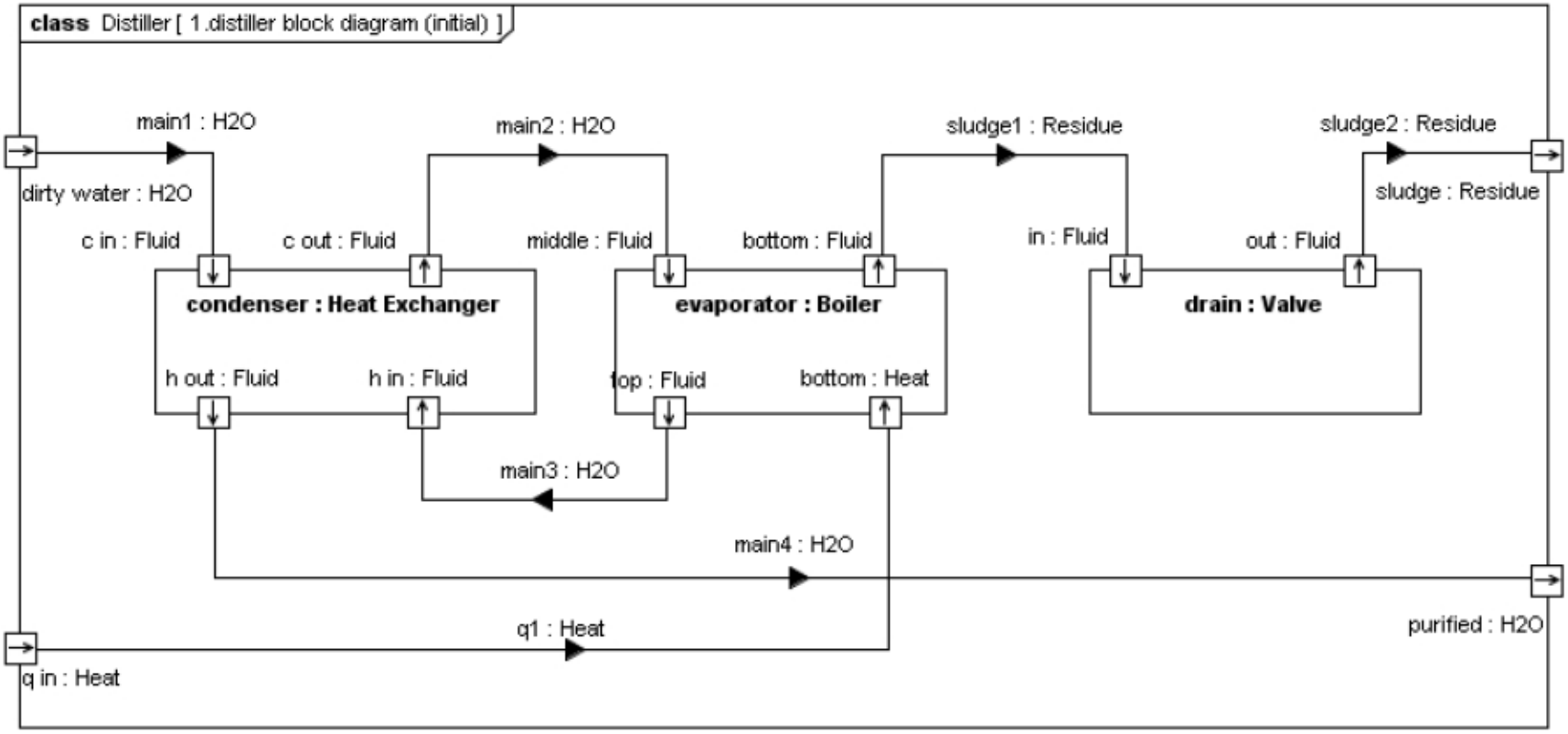


Heat Exchanger Flow Ports

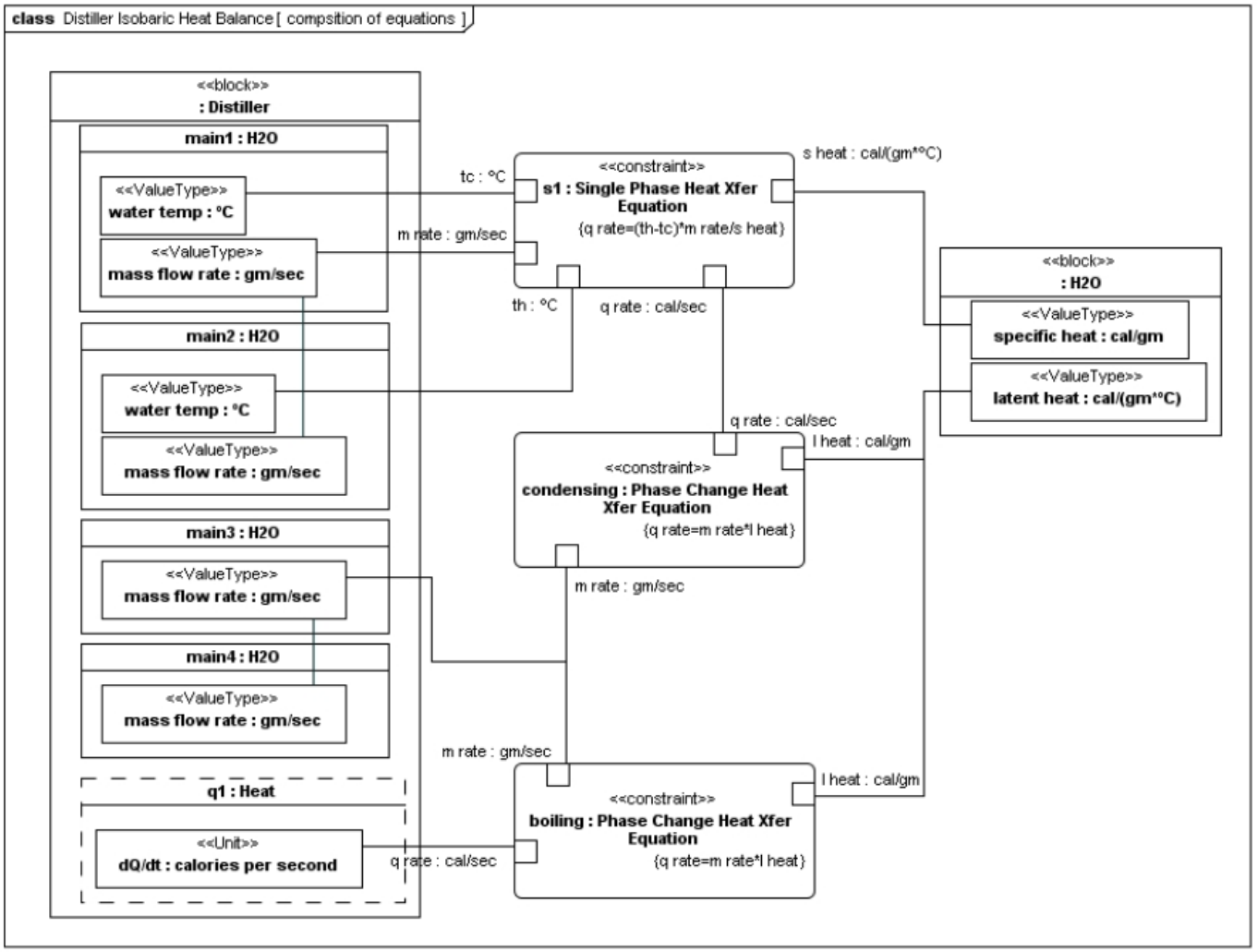


Internal Block Diagram

Initial Design



Heat Balance



Heat Balance Results (leading to design change)

table IsobaricHeatBalance1 [Results of Isobaric Heat Balance]

	main1 : H2O	main2 : H2O frm condenser	main2 : H2O into evap	main3 : H2O	main4 : H2O
specific heat cal/gm-°C	1				
latent heat cal/cm	540				
mass flow rate gm/sec	6.8	6.8	1	1	1
temp °C	20	100	100	100	100

Satisfies «requirement» WaterSpecificHeat

Satisfies «requirement» WaterHeatOfVaporization

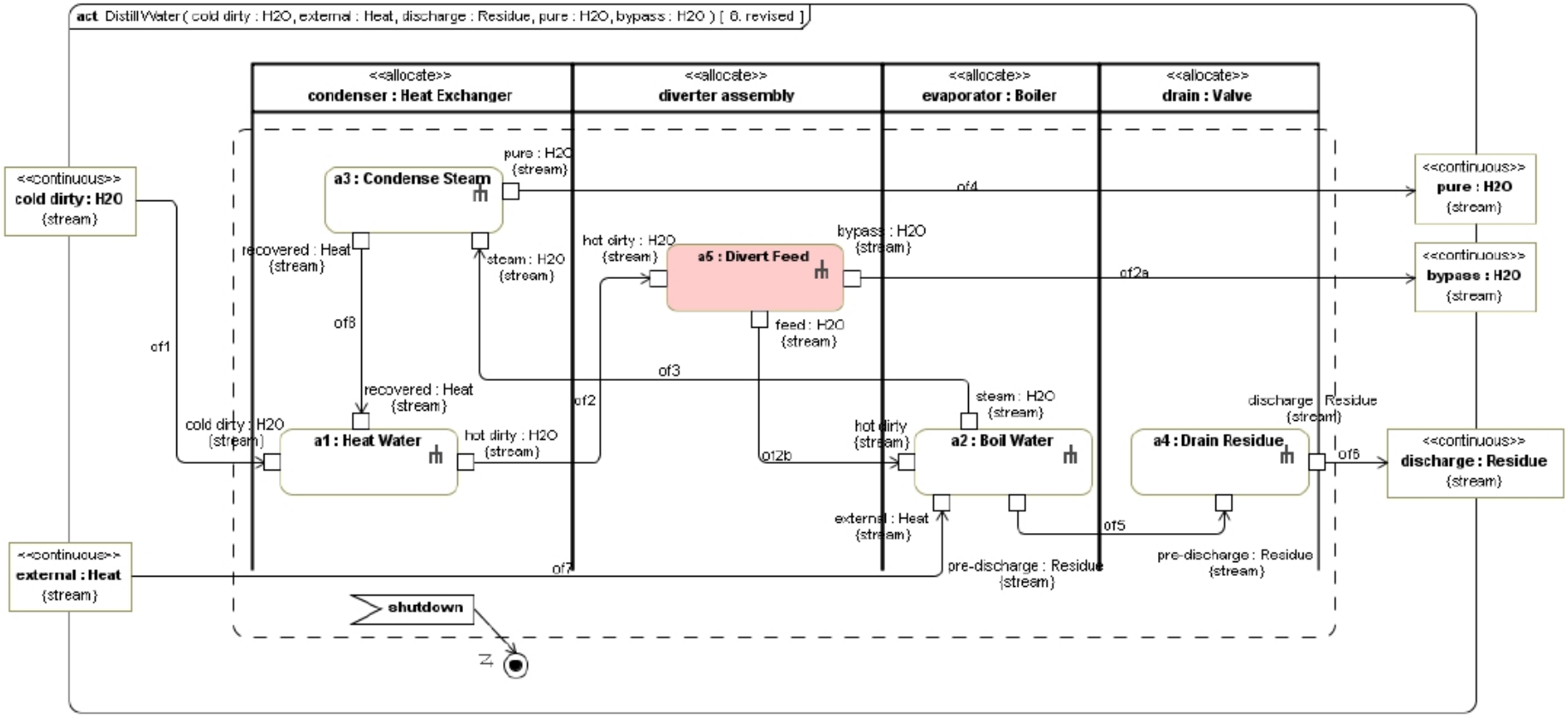
Satisfies «requirement» WaterInitialTemp

dQ/dt cooling water cal/sec	540
dQ/dt steam-condensate cal/sec	540
condenser efficency	1
heat deficit	0

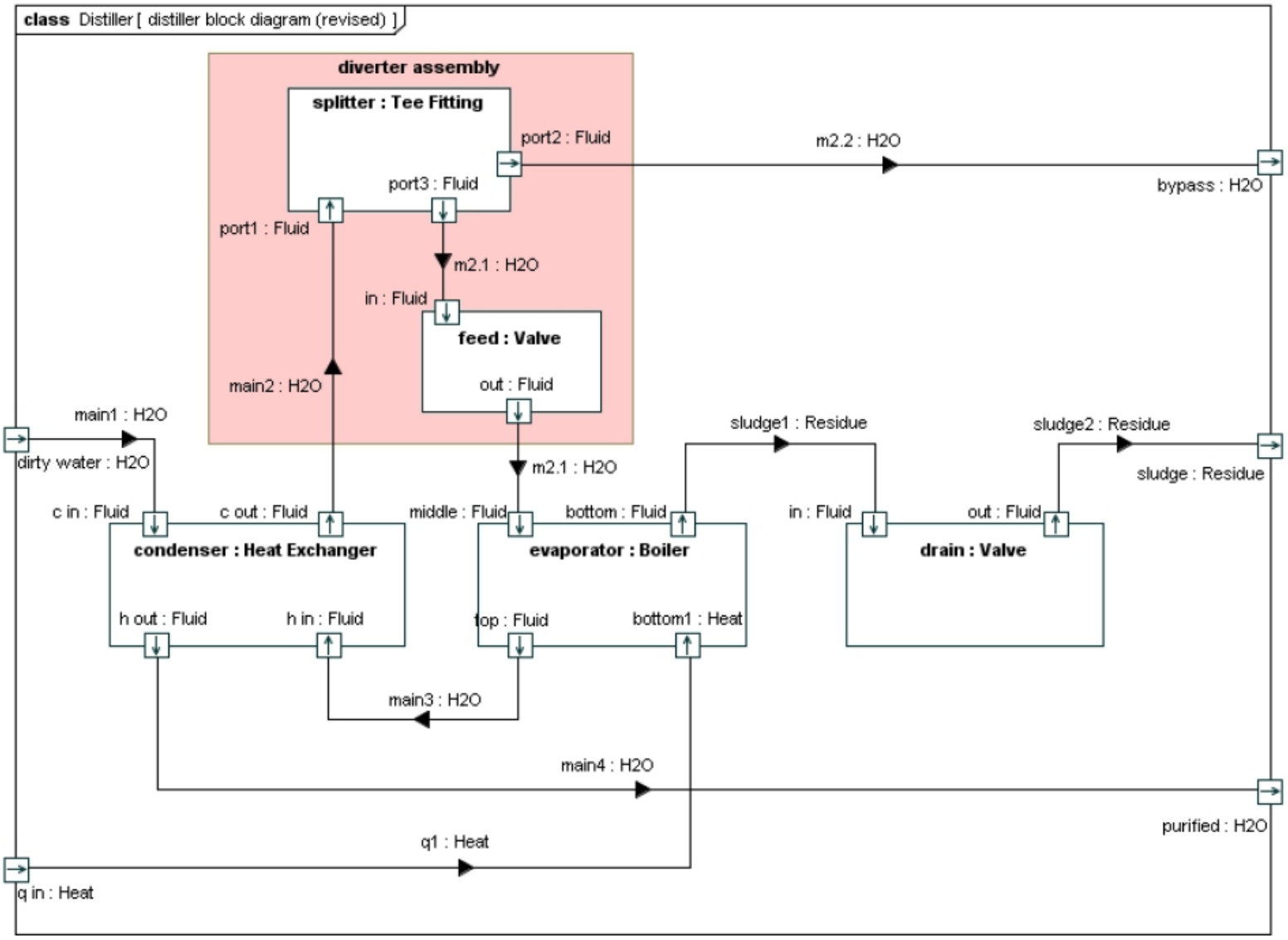
dQ/dt condensate-steam cal/sec	540
boiler efficiency	1
dQ/dt in boiler cal/sec	540

Note: Cooling water needs to have 6.75x flow of steam! Need bypass between hx_water_out and bx_water_in!

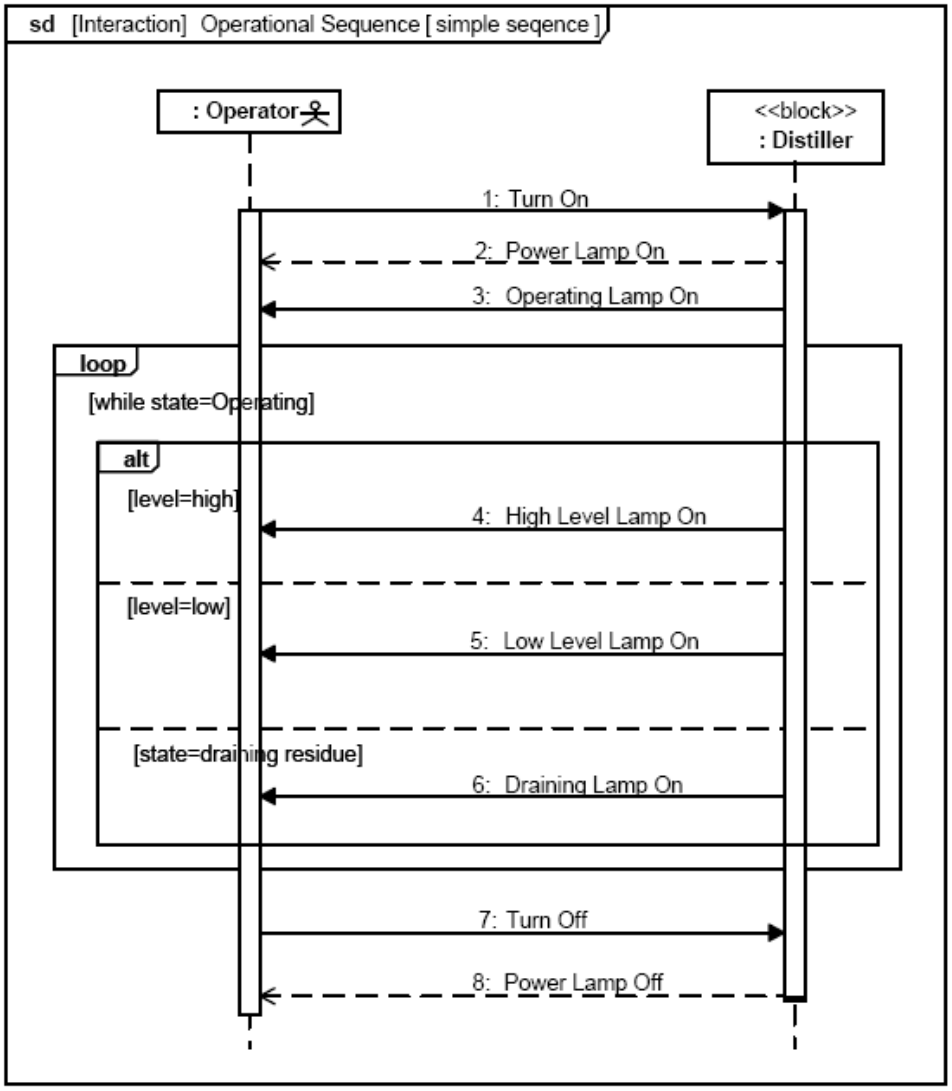
Updated Activity Diagram

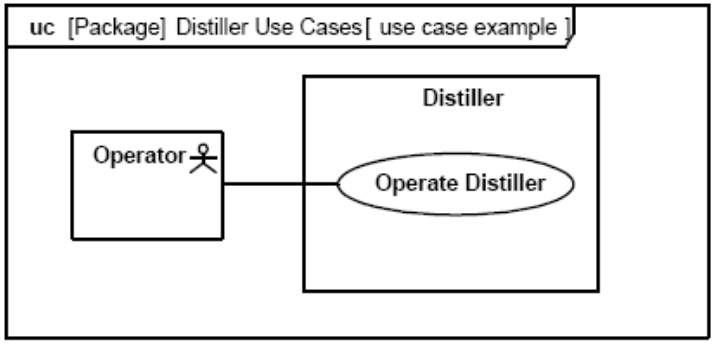


Updated Internal Block Diagram



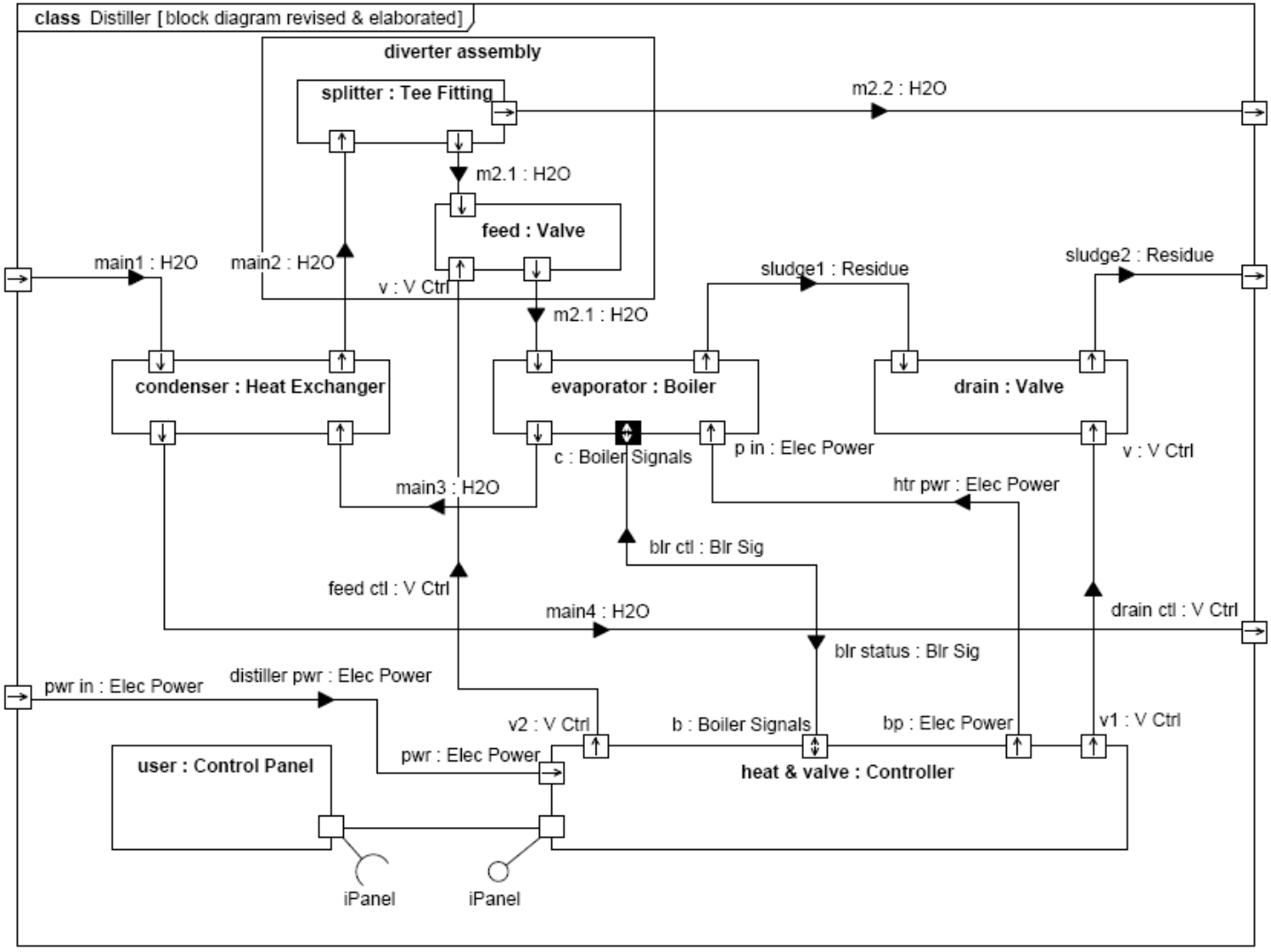
Sequence Diagram and Use Case



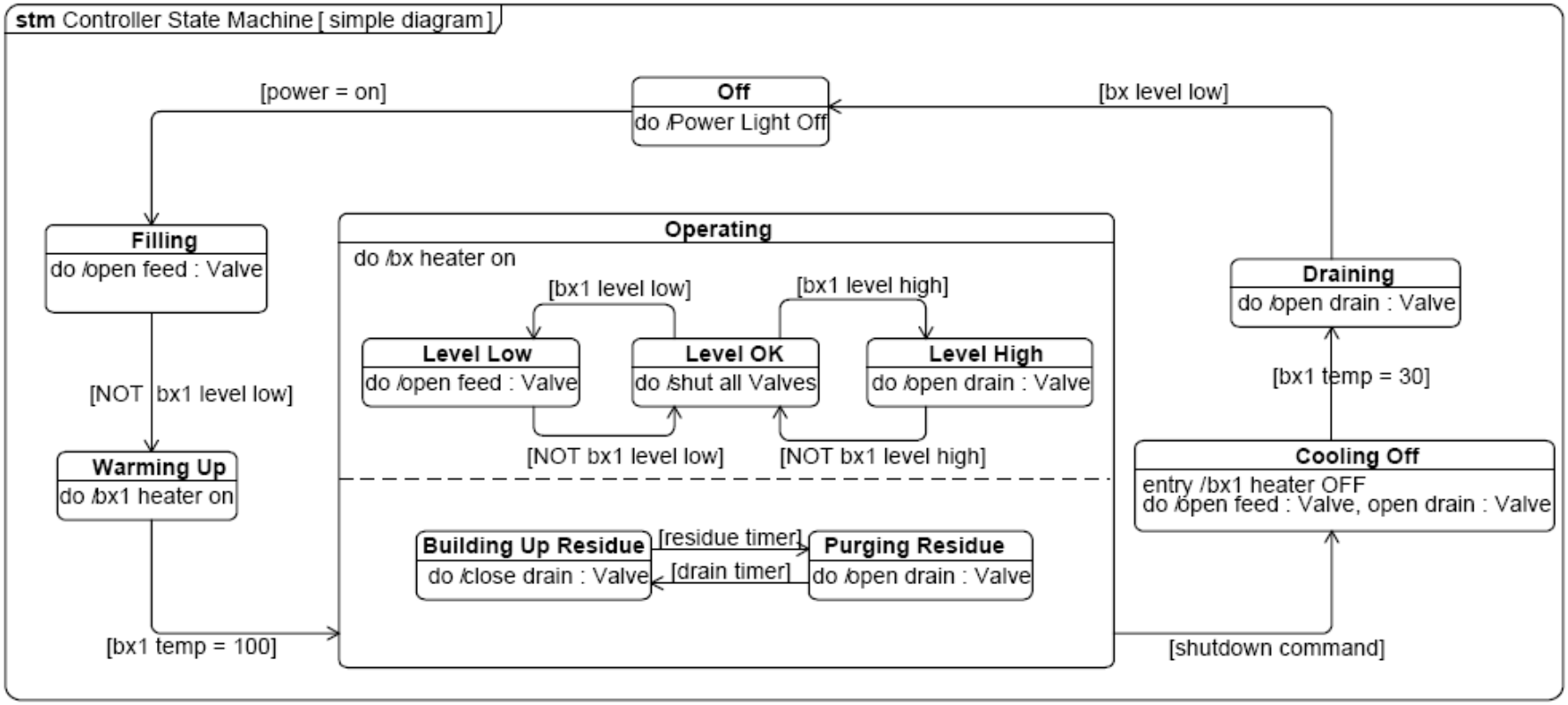


Internal Block Diagram

Updated Controller



State Machine: Controller



References

Reference Papers

- **Exploring Model Based Engineering for Large Telescopes**
  Karban, Robert, M Zamparelli, B Bauvir, B Koehler, L Noethe, A Balestra, SPIE Astronomical Telescopes and Instrumentation, 2008
- **Building Bridges Between Systems and Software With SysML and UML**
  Matthew Hause, Francis Thom, INCOSE Intl. Symposium, June ' 2008
- **Integrating Models and Simulations of Continuous Dynamics into SysML**
  Thomas Johnson, Christiaan Paredis, Roger Burkhart, Jan ' 2008
- **Modeling Continuous System Dynamics in SysML**
  Thomas Johnson, Jonathan Jobe, Christiaan Paredis, Roger Burkhart, Proceedings of the IMECE 2007, Nov ' 2007
- **Survey of Model Based Systems Engineering Methodologies (MBSE) revB**
  Jeff A Estefan, May 23 ' 2008
- **Simulation-Based Design Using SysML—Part 1: A Parametrics Primer,**
  RS Peak, RM Burkhart, SA Friedenthal, MW Wilson, M Bajaj, I Kim, INCOSE Intl. Symposium, June ' 2007
- **Simulation-Based Design Using SysML--Part 2: Celebrating Diversity by Example,**
  RS Peak, RM Burkhart, SA Friedenthal, MW Wilson, M Bajaj, I Kim, INCOSE Intl. Symposium, June ' 2007
- **An Overview of the Systems Modeling Language for Products and Systems Development,**
  Laurent Balmelli, Oct ' 2006.
- **Model-driven systems development,**
  L. Balmelli, D. Brown, M. Cantor, M. Mott, July ' 2006.
- **The SysML Modelling Language,**
  Matthew Hause, Fifteenth European Systems Engineering Conference, Sept ' 2006.
- **SysML and UML 2.0 Support for Activity Modeling**
  Bock. C., vol. 9 no.2, pp. 160-186, Journal of International Council of Systems Engineering, 2006.
- **Formal Performance Analysis and Simulation of UML/SysML Models for ESL Design**
  Alexander Viehl , Timo Schönwald, Oliver Bringmann, Wolfgang Rosenstiel
- **A SysML Extension for Bond Graphs Support**
  Turki S, Soriano T
- **SysML and Systems Engineering Applied to UML-Based SoC Design**
  Yves Vanderperren, Wim Dehaene
- **From UML/SysML to Matlab/Simulink: Current State and Future Perspectives**
  Yves Vanderperren, Wim Dehaene

Reference Books

- Systems Engineering with SysML/UML by Tim Weilkiens.
- A Practical Guide to SysML by Sanford Friedenthal, Alan Moore, and Rick Steiner.
- SysML for Systems Engineering: The Emperor's New Modelling Language by John Holt and Simon Perry.

SysML Vendors

- **See http://sysml-directory.omg.org/vendor/list.htm for a current complete list**

- Armstrong Process Group, Inc.
- **Artisan Software Tools**
- **EmbeddedPlus Engineering (3rd party to IBM Rational)**
- **IBM Telelogic (Rhapsody & Tau)**
- **No Magic, Inc.**
- North State Software
- oose Innovative Informatik GmbH
- Pathfinder Solutions
- SODIUS
- **Software Stencils** (Visio, SysML, and UML Templates)
- **Sparx Systems**

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		

All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?

Systems Engineering Technical Short Course

(Session 4: System Architecture and Design)

Lesson 6:
Special Topics
(as time permits)

Session 4 – System Architecture and Design

Lessons

1. **Program Overview Update**
2. **System Architecture Basics**
3. **Systems Design Process**
4. **Analysis of Alternatives**
5. **Model-Based Systems Engineering**
6. **Special Topics (if time permits)**

Appendices*

I. **SysML Diagrams and Language Concepts**

J. **Architecture Frameworks In-depth**

- IDEF0 In-depth
- Architecture Framework Examples

*Appendices A through H were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

IDEF0 Standard

Draft Federal Information
Processing Standards Publication 183

1993 December 21

Announcing the Standard for

INTEGRATION DEFINITION FOR FUNCTION MODELING (IDEF0)

Federal Information Processing Standards Publications (FIPS PUBS) are issued by the National Institute of Standards and Technology after approval by the Secretary of Commerce pursuant to Section 111(d) of the Federal Property and Administrative Services Act of 1949 as amended by the Computer Security Act of 1987, Public Law 100-235.

References: Beude, p. 85 ff. and http://en.wikipedia.org/wiki/IDEF0

IDEF0

- **Integrated DEFinition language 0**
  - **Originally System Analysis and Design Technique (SADT)**
    - **Developed by Douglas T. Ross, SofTech**
  - **Used in USAF's Integrated Computer Aided Manufacturing (ICAM) Program to model manufacturing and logistics activities**

- **Generally used for modeling complex processes (activities, functions)**

Primary IDEF0 Concepts

- "Box and arrows" language with syntax and semantics

- Provides systems engineering approach to:
  - Performing system/process analysis at all levels (multiple levels of abstraction)
  - Producing reference documentation
  - Communicating among analysts, designers, users, and managers
  - Promoting shared understanding
  - Managing large, complex projects
  - Providing a reference architecture for enterprise analysis, information engineering, and resource management

Other Concepts and Characteristics

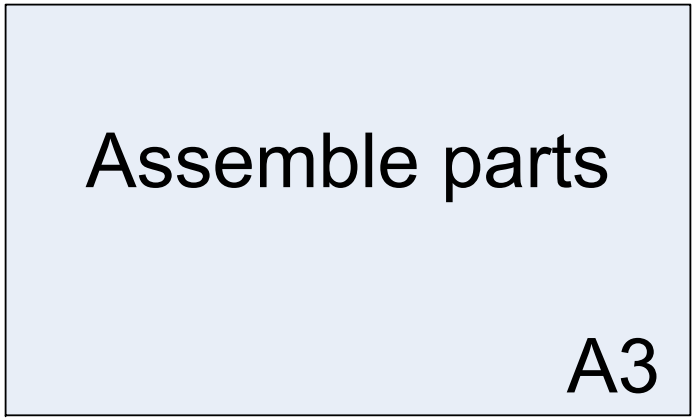
- Gradual exposition of detail
- Limitation of detail
- Diagrams supported with text
- Rigor and precision
  - Exposition control
  - Bounded context (no omissions or additional out-of-scope detail)
  - Syntax rules for graphics (boxes and arrows)
  - Uniqueness of names and labels on a diagram
  - Diagram connectivity
  - Data/object connectivity
  - Rules for determining role of data or objects
  - Arrow label requirements (minimum labeling rules)
  - Purpose and viewpoint

Syntax and Semantics

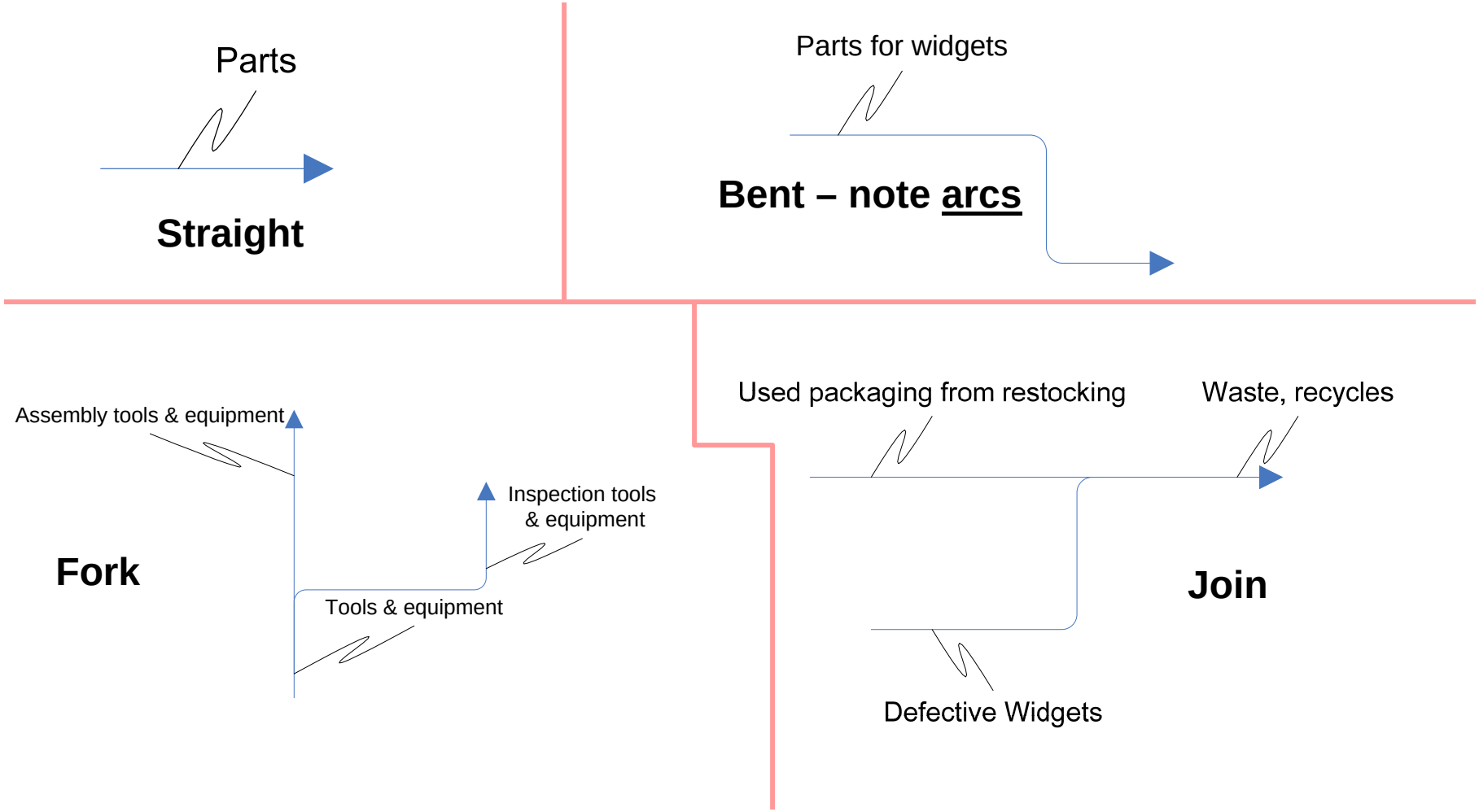
- **Syntax of a representation (~grammar)**
  - **Elements**
  - **Structure**
  - **Form**
- **Semantics (~meaning)**

Syntax: Boxes

- **Solid lines**
- **Verb or verb phrase**
- **Box number**



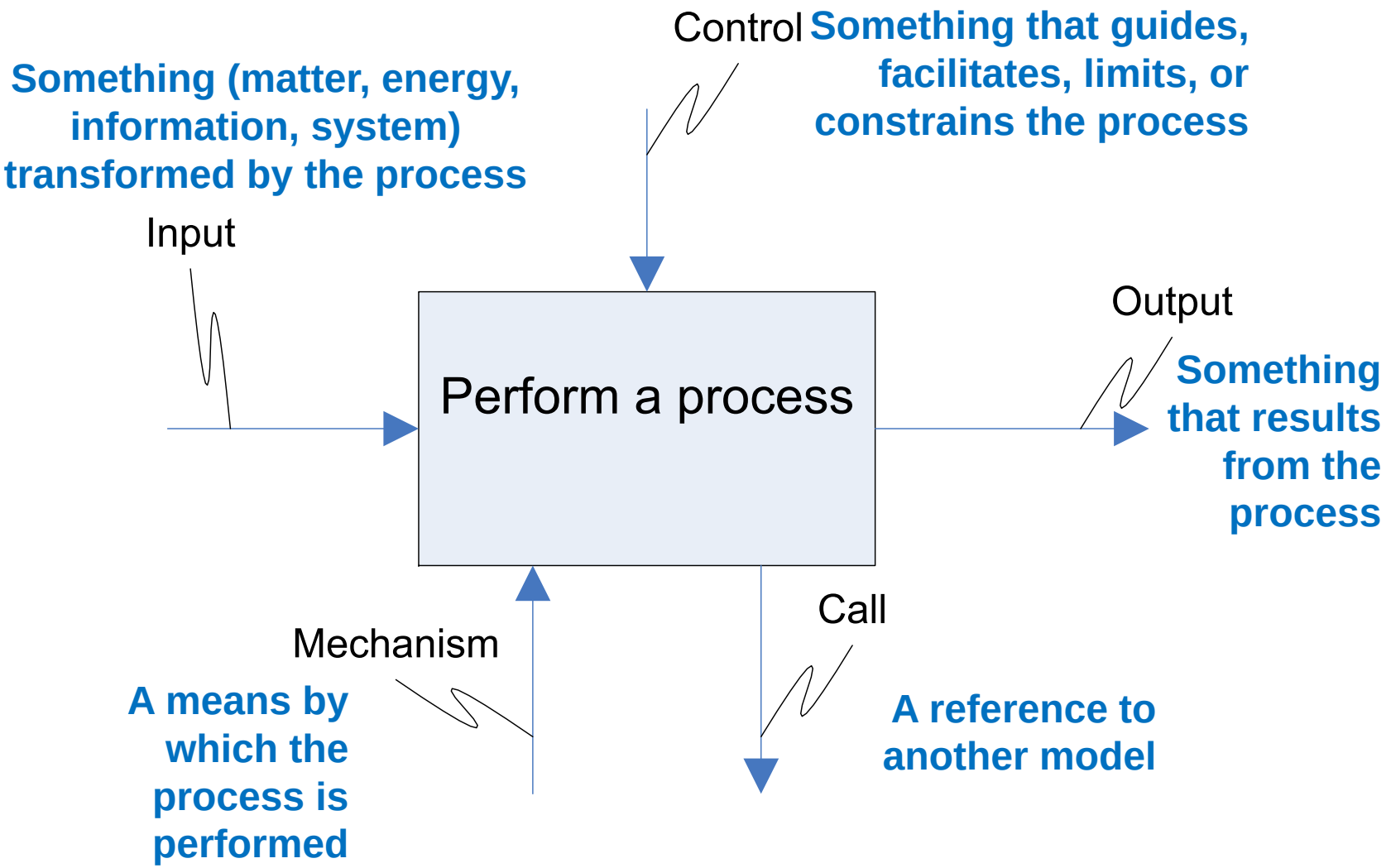
Syntax: Arrows



Syntax: Box and Arrow Rules

- **Boxes**
  - Boxes shall be sufficient in size to insert box name
  - Boxes shall be rectangular in shape, with square corners
  - Boxes shall be drawn with solid lines
- **Arrows**
  - Arrows that bend shall be curved using only 90 degree arcs
  - Arrows shall be drawn in solid line segments
  - Arrows shall be drawn vertically or horizontally, *not diagonally*
  - Arrow ends shall touch the outer perimeter of the function box and shall not cross into the box
  - Arrows shall attach at box sides, *not at corners*

Semantics

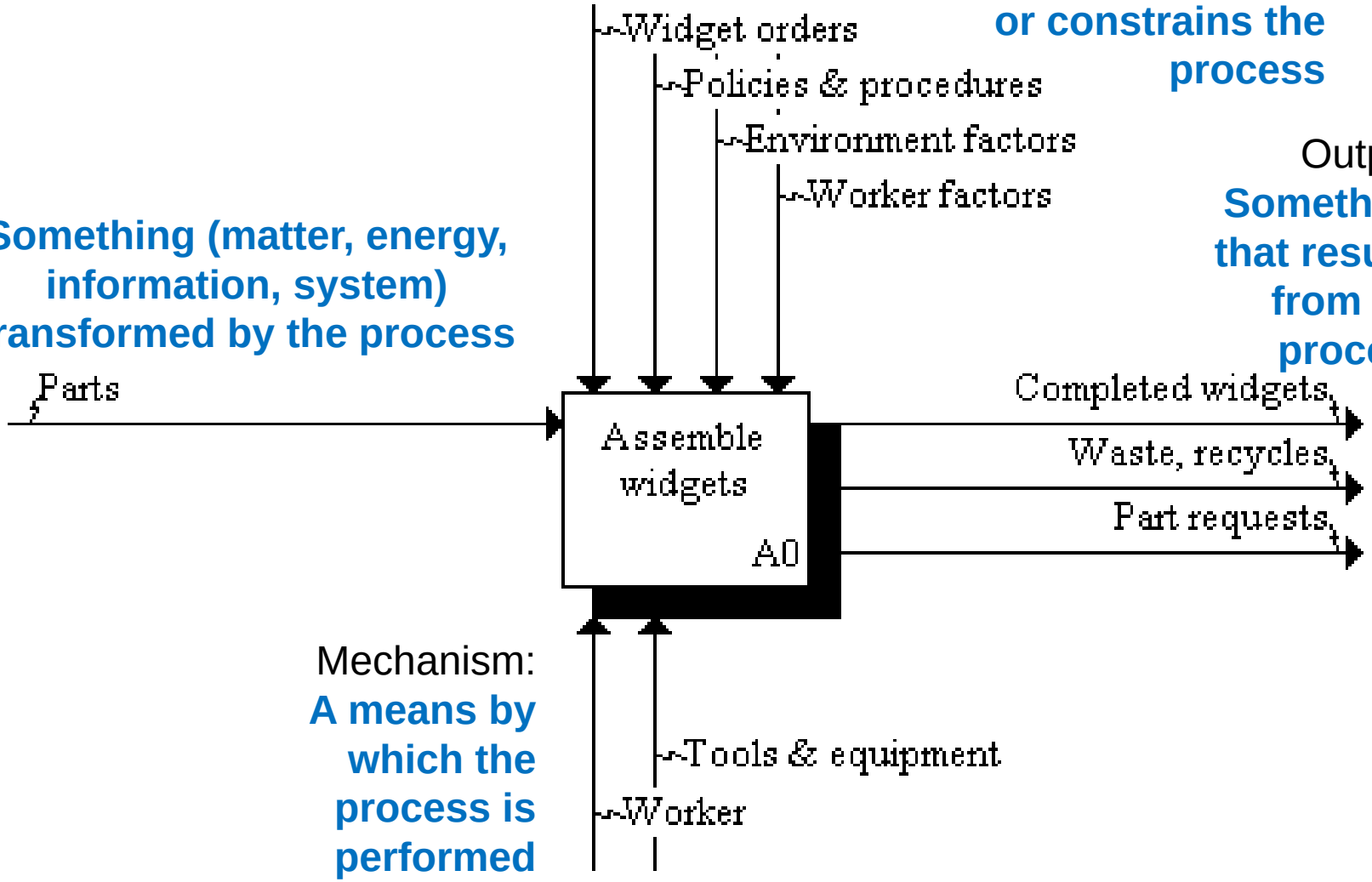

Control
Something that guides, facilitates, limits, or constrains the process
Something (matter, energy, information, system) transformed by the process
Input
Perform a process
Output
Something that results from the process
Call
Mechanism
A means by which the process is performed
A reference to another model

Example

Control: **Something that guides, facilitates, limits, or constrains the process**

Output: **Something that results from the process**

Something (matter, energy, information, system) transformed by the process



Mechanism: **A means by which the process is performed**

Syntax: More Box and Arrow Rules

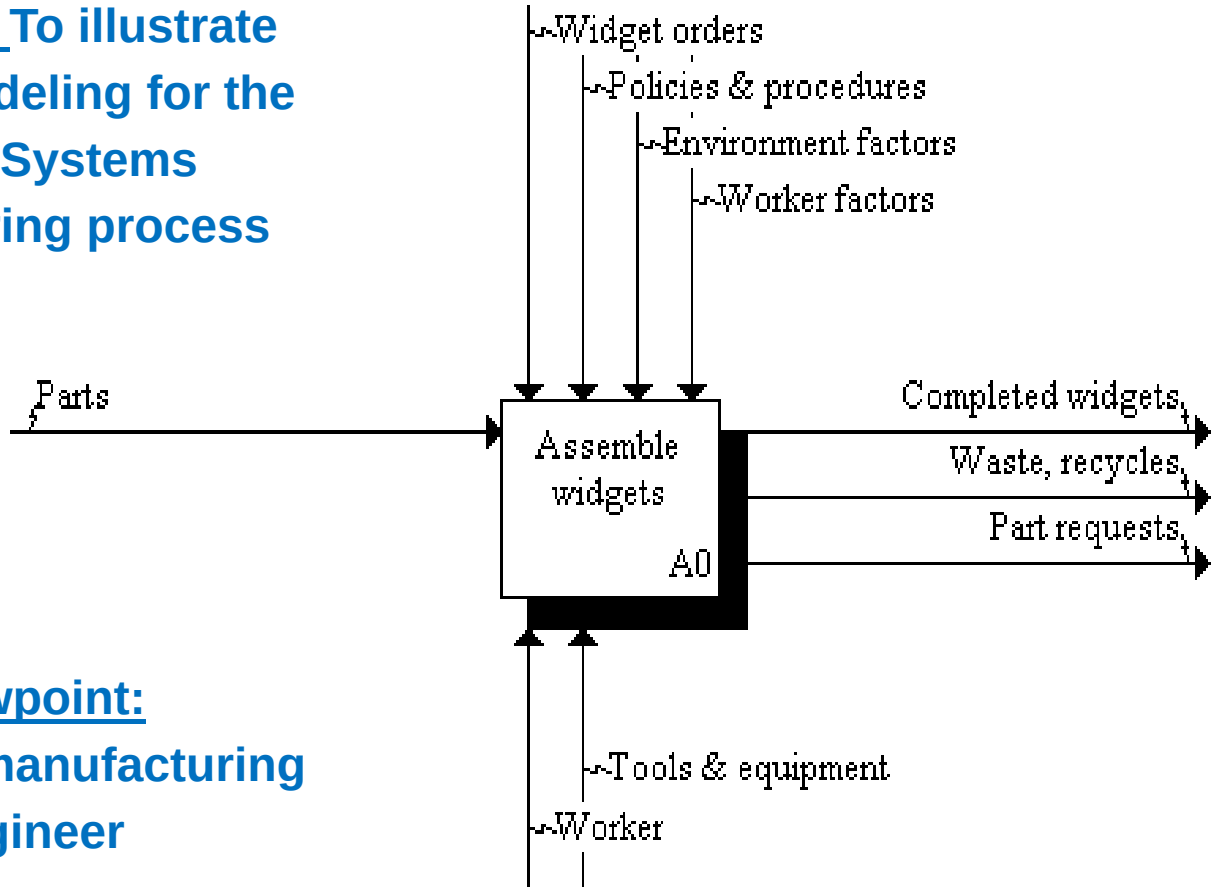
1. A box shall be named with an *active verb or verb phrase*
2. Each side of a function box shall have a standard box/arrow relationship:
   a. Input arrows shall interface with the left side of a box
   b. Control arrows shall interface with the top side of a box
   c. Output arrows shall interface with the right side of the box
   d. Mechanism arrows (except call arrows) shall point upward and shall connect to the bottom side of the box
   e. Mechanism call arrows shall point downward, shall connect to the bottom side of the box, and shall be labeled with the reference expression for the box which details the subject box
3. Arrow segments, except for call arrows, shall be labeled with a *noun or noun phrase* unless a single arrow label clearly applies to the arrow as a whole
4. A "squiggle" (⁄) shall be used to link an arrow with its associated label, unless the arrow/label relationship is obvious
5. Arrow labels shall not consist solely of any of the following terms: function, input, control, output, mechanism, or call

Top-Level Context Diagram

- **Subject of model represented by single box with bounding arrows.**
- **Called A – 0 (“A minus zero”)**
- **Box and arrows are very general**
- **Sets model scope or boundary and orientation**
- **Should include**
  - **Purpose**
  - **Viewpoint**

Example Context Diagram: A–0 Assemble widgets

Purpose: To illustrate IDEF0 modeling for the Work Systems Engineering process



Viewpoint: Industrial/manufacturing engineer

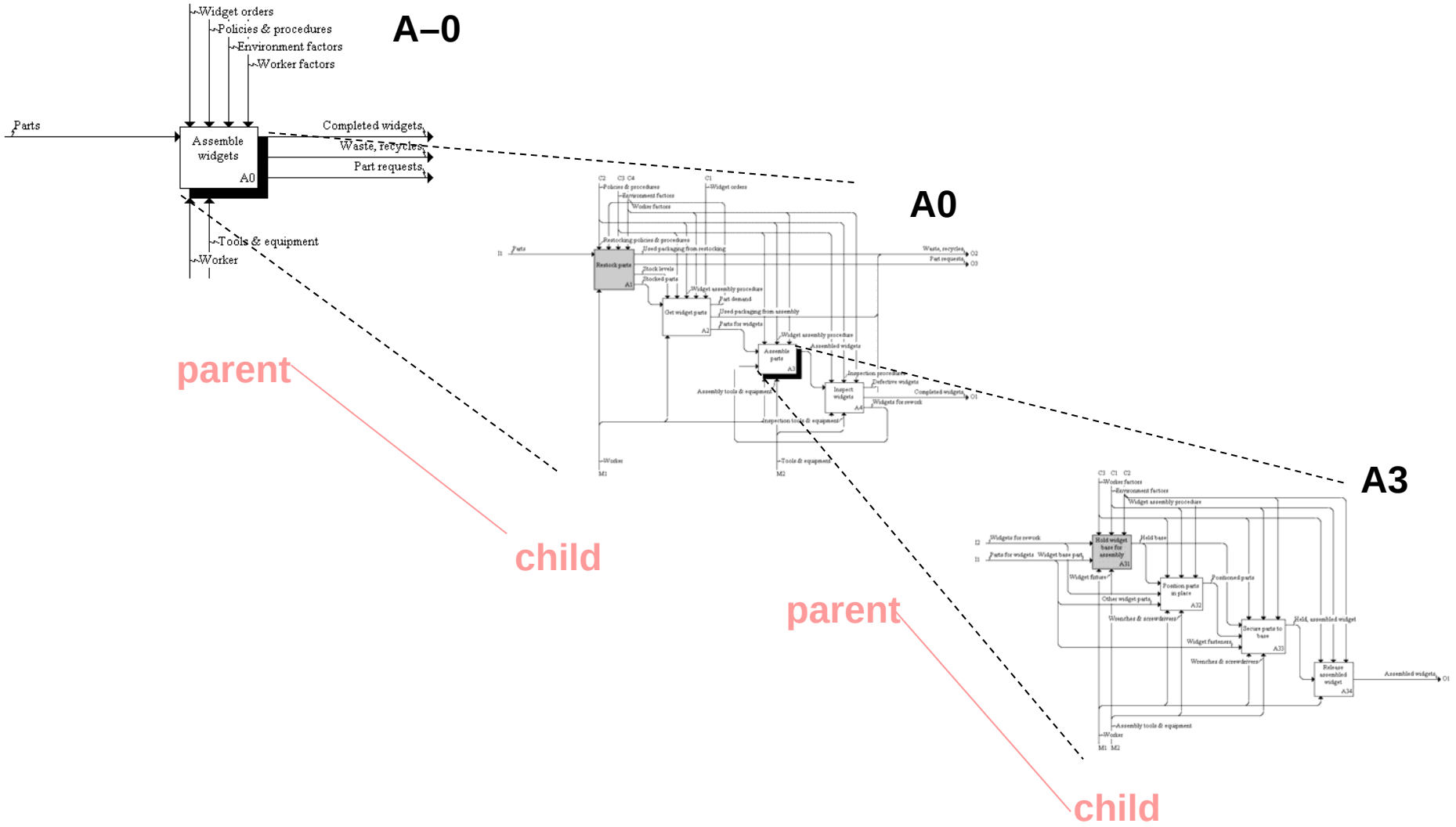
Parent Diagram

- **Diagram that contains one or more parent boxes, i.e., boxes detailed on child diagrams.**

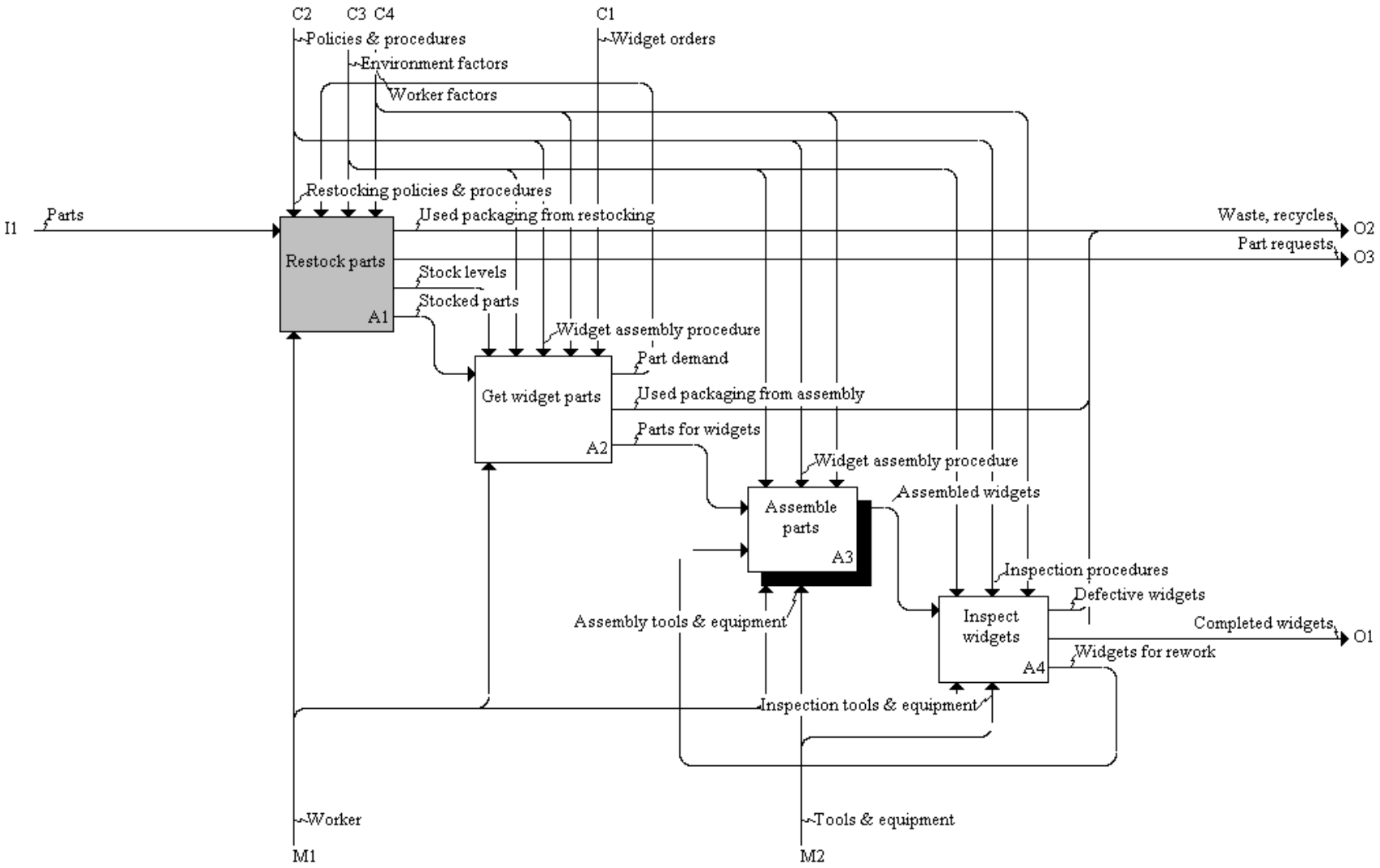
Child Diagram

- The single process in Top-Level Context Diagram (A–0) may be decomposed into sub-processes and modeled in a child (A0) diagram
- Each process in the A0 diagram may be decomposed further into sub-processes and modeled in (grand-) child (A1, A2, … A6) diagrams
- Each (grand-) child process may be decomposed further into sub-processes and modeling (great-grand-) child diagrams
- And so on …

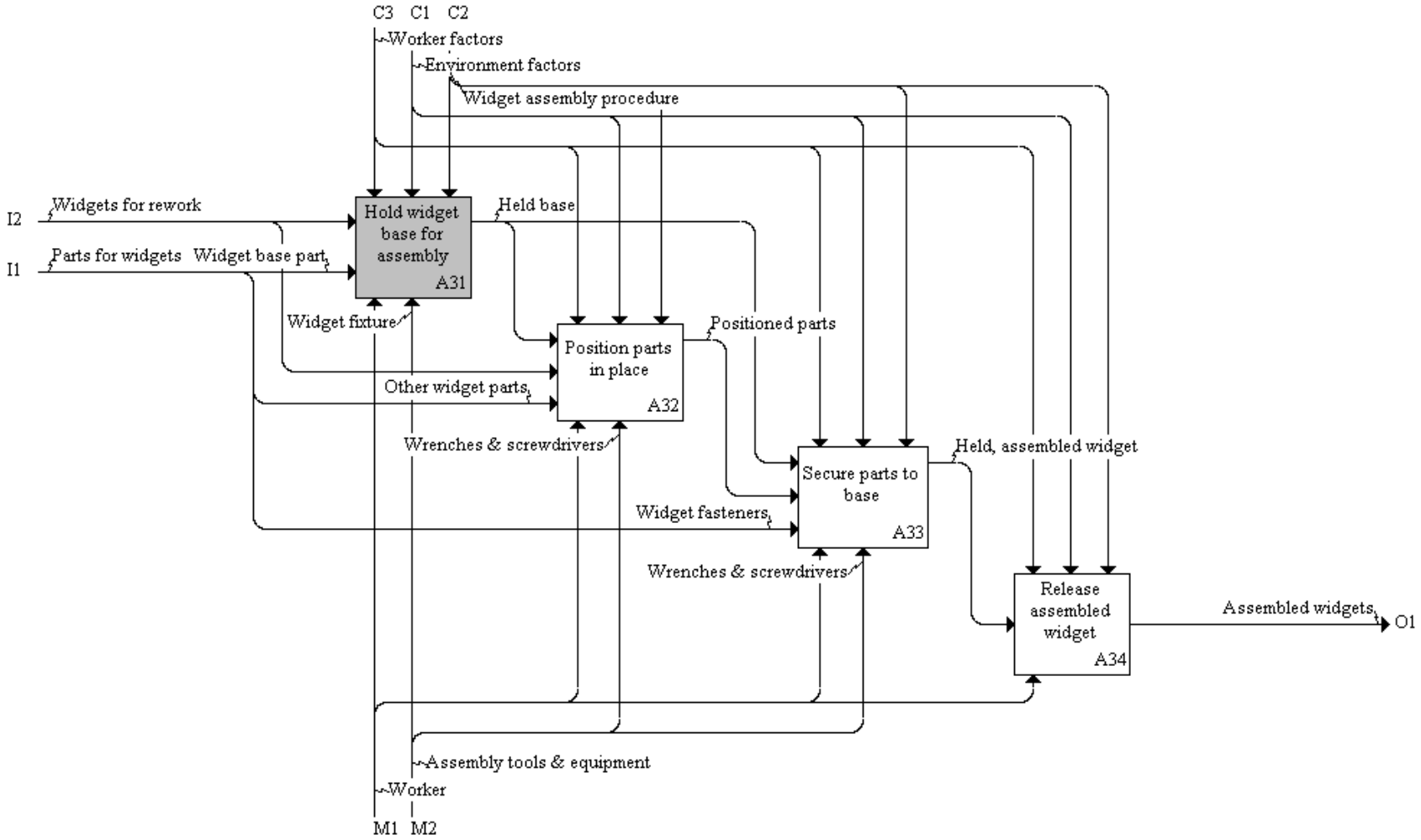
Process Decomposition



A0 Diagram


C2
C3 C4
C1
Policies & procedures
Environment factors
Worker factors
Widget orders
Restocking policies & procedures
I1
Parts
Restock parts
A1
Used packaging from restocking
Waste, recycles
O2
Part requests
O3
Stock levels
Stocked parts
Widget assembly procedure
Part demand
Get widget parts
A2
Used packaging from assembly
Parts for widgets
Widget assembly procedure
Assembled widgets
Assemble parts
A3
Inspection procedures
Defective widgets
Assembly tools & equipment
Inspect widgets
Completed widgets
O1
Widgets for rework
A4
Inspection tools & equipment
Worker
M1
Tools & equipment
M2

A3 Diagram

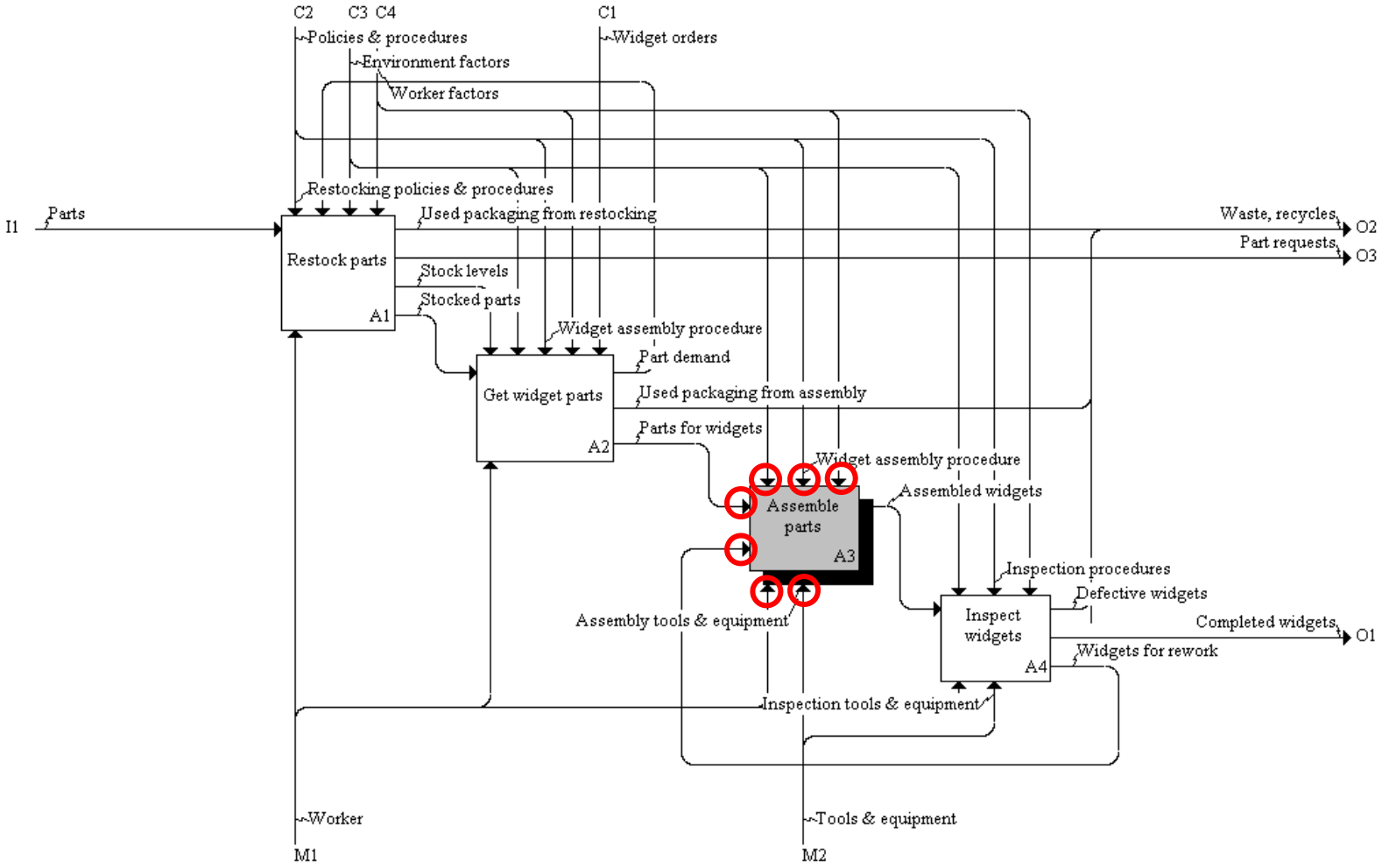

C3
C1
C2
Worker factors
Environment factors
Widget assembly procedure
I2
Widgets for rework
I1
Parts for widgets
Widget base part
Hold widget base for assembly
A31
Held base
Widget fixture
Position parts in place
A32
Positioned parts
Other widget parts
Wrenches & screwdrivers
Secure parts to base
A33
Held, assembled widget
Widget fasteners
Wrenches & screwdrivers
Release assembled widget
A34
Assembled widgets
O1
Assembly tools & equipment
Worker
M1
M2

Inter-Box Connections

- **Except for A–0, diagrams contain 3 – 6 boxes**
- **Normally organized on diagonal (“staircase”)**
- **Any output of one box may be input, control, or mechanism of another box**
- **If a box is detailed on child diagram, every arrow connected to the box appears on the child diagram (unless it is *tunneled*, discussed later)**

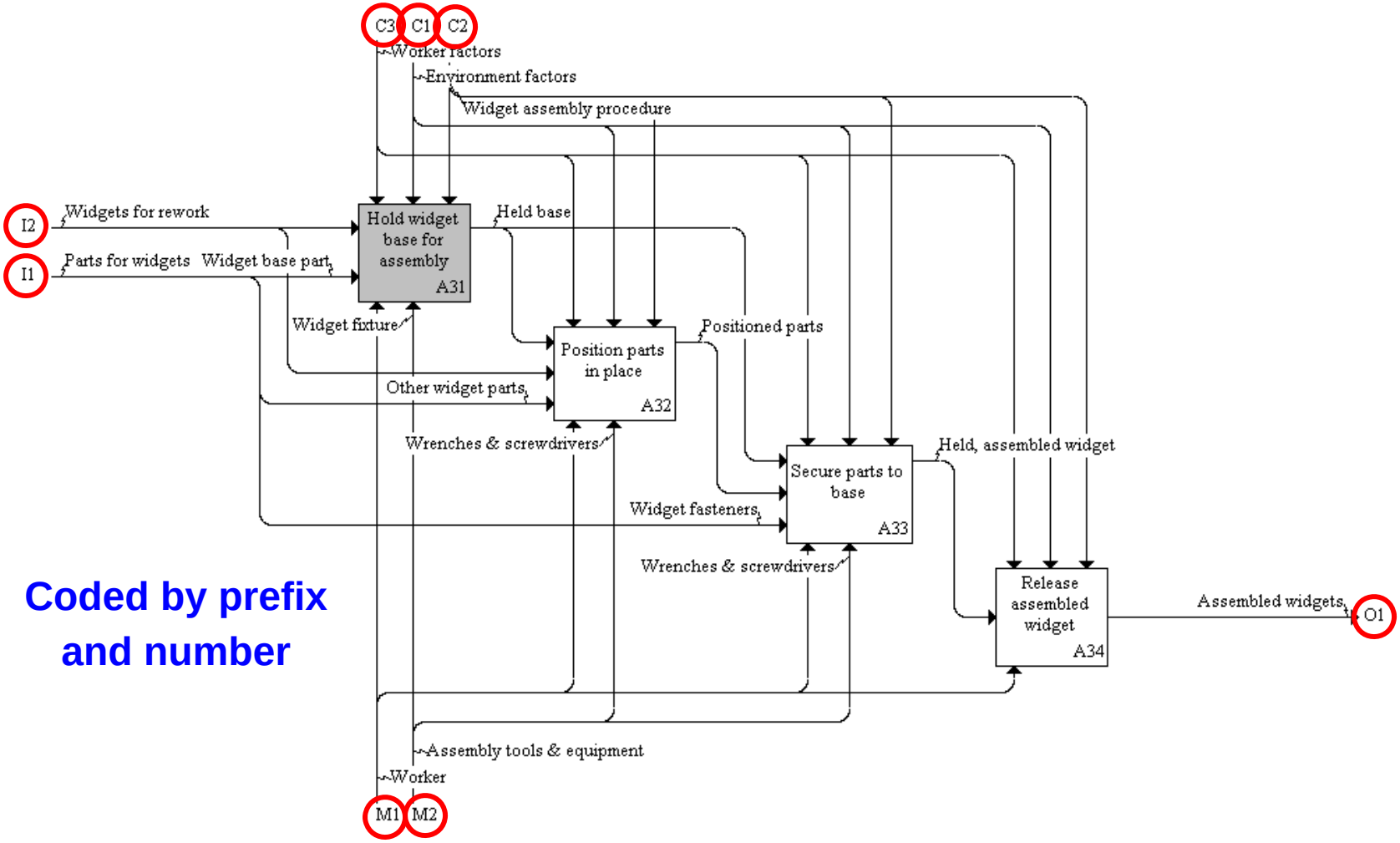
Inter-Box Connections

(arrows for child diagram)



Boundary Arrows:

(Arrows from parent box on parent diagram)



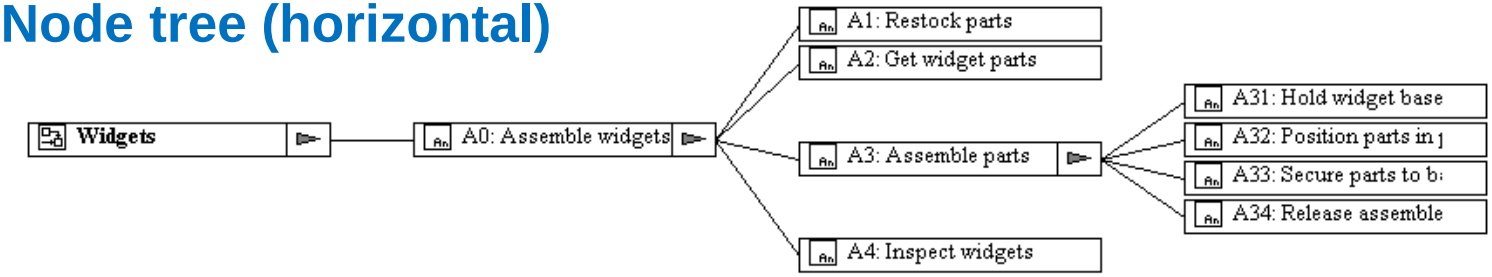
Coded by prefix and number

Box Numbers and Node Numbers

- **Box numbers**
  - Single box in context (A-0) diagram numbered A0 (“Activity” 0)
  - Boxes in context diagram’s child numbered A1, A2, A3, … [A6]
  - Boxes in A1’s child diagram numbered A11, A12, …
  - Boxes in A2’s child diagram numbered A21, A22, …
  - Boxes in A21’s child diagram numbered A211, A212, …
  - and so on …
- **Node** – for our purposes, another name for a diagram
- **Node numbers**
  - Context diagram is node A-0
  - A-0’s child node is node A0
  - A0’s children are nodes A1, A2, …
  - In general, a node bears the same number as the box in the parent node it details

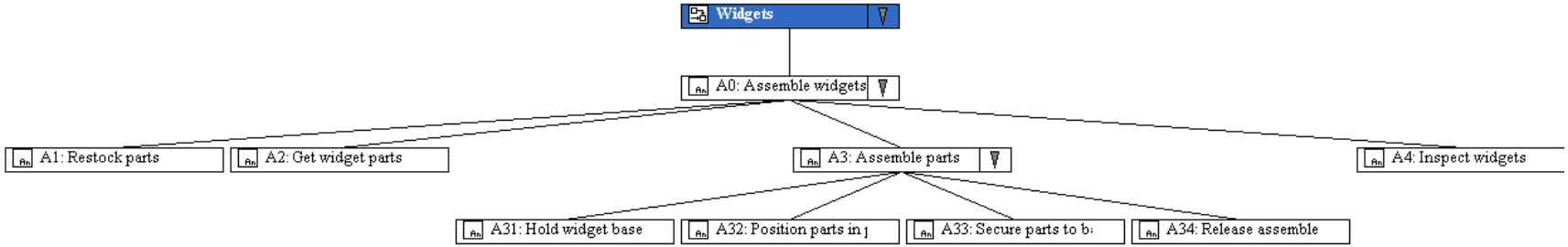
Node List, Node Tree

- Node List
  - A-0: Assemble widgets
  - A0: Assemble widgets
    - A1: Restock parts
    - A2: Get widget parts
    - A3: Assemble parts
      - A31: Hold widget base
      - A32: Position parts in place
      - A33: Secure parts to base
      - A34: Release assembled widget
    - A4: Inspect widgets
- Node tree (horizontal)

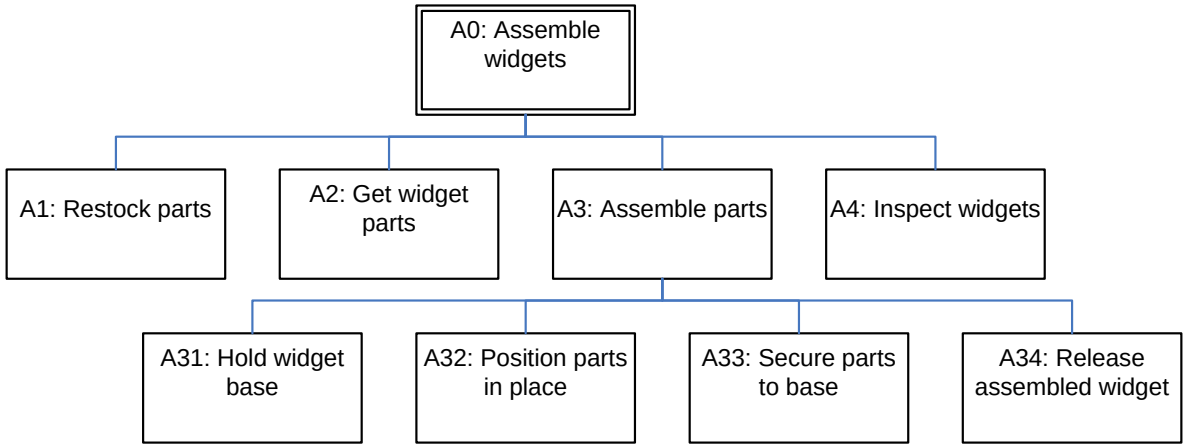


Node List, Node Tree

- **Node tree (vertical – AI0Win)**



- **Node tree (vertical – Visio)**



Reading IDEF0 Diagrams

- Scan boxes of diagram to gain impression of what is being described
- Refer to parent diagram
  - Note arrow connections to parent box
  - Try to identify "most important" input, control, output
- Consider arrows of current diagram
- Try to determine if there is a main path linking "most important" input or control and "most important" output
- Mentally walk through the diagram, from upper left to lower right
  - Note how arrows interact with each box
  - Determine if there are secondary paths
  - Check the story being told by diagram by considering how familiar situations are handled
- Check to see if a related FEO diagram exists
- Read text and glossary, if provided

Creating IDEF0 Diagrams

- Select Context, Viewpoint, Purpose
- Create Context (A–0) Diagram
- Create topmost (A0) diagram
  - 3-6 sub-process boxes per diagram
  - May be necessary to switch back and forth between A0 and A-0 several times to get a good start
- Create child diagrams
  - Detail each A0 process box into 3-6 sub-process boxes
  - May be necessary to redraw several times
  - Split and cluster boxes until satisfied
- Create supporting materials: text, glossary, FEOs

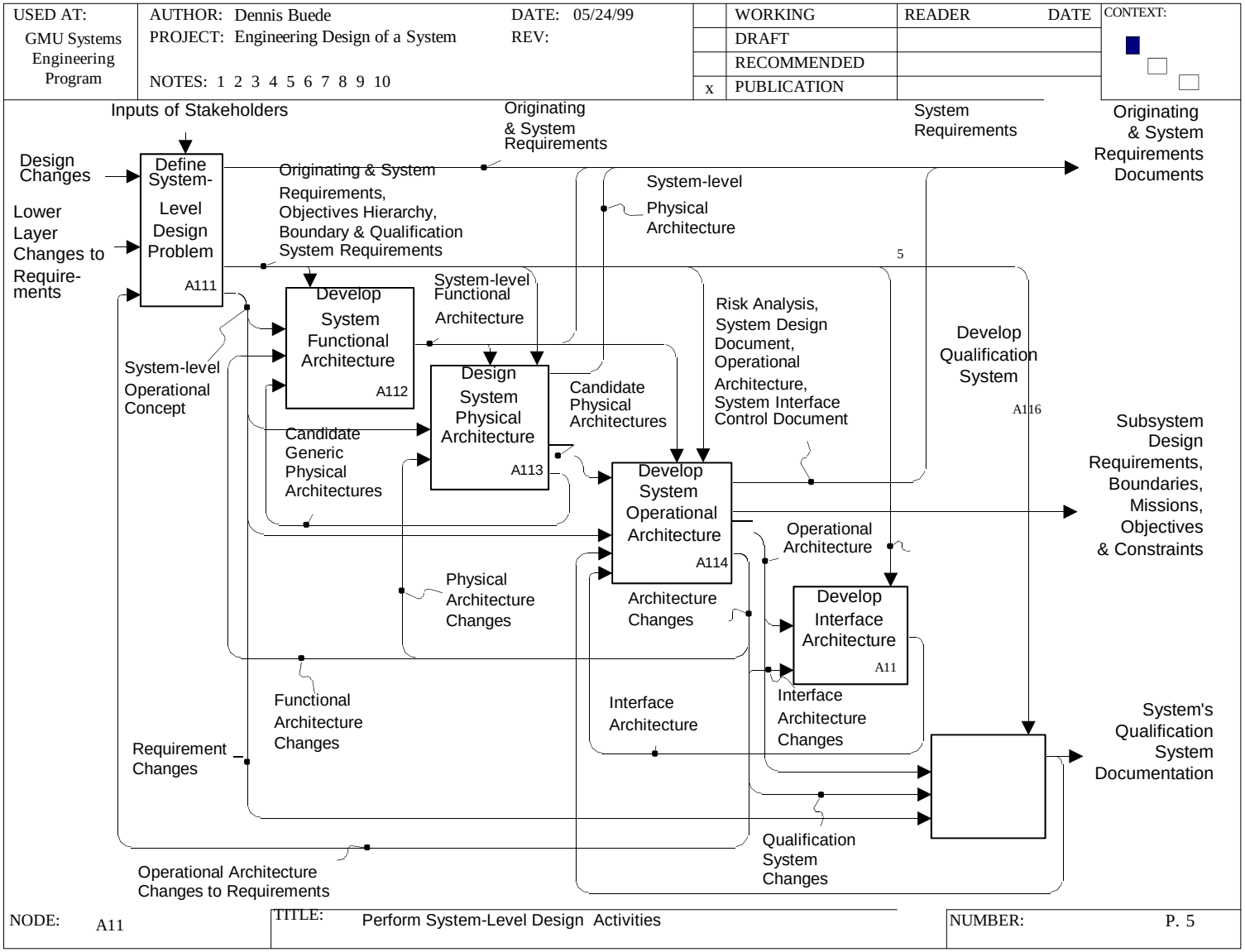
Detailing Boxes

- **Select a box:**
  - **Select the hardest part.**
  - **Select the box whose detail will give the most information.**
- **Gather data**
- **Structure: draw and re-draw**
- **Write text carefully**
- **Draw boxes**
  - **Make process box names verbs or verb phrases**
  - **Lay out diagonally, upper left to lower right**
  - **Place boxes that "dominate" in the upper left**
  - **All boxes should have a consistent "flavor" (level of complexity)**

Drawing Interface Arrows

- Think control and constraint, not flow
  - Don't worry about sequence
  - All boxes may be active simultaneously
- Bundle groups of arrows, when possible
- Don't clutter with arrows
- All boxes must have control arrows, but they don't require input arrows
- Give arrows noun or noun phrase names

IDEF0 Architecture Example



Optional Exercise

Path Tracing an IDEF0 Diagram

- **Using the guidelines for reading an IDEF0 diagram provided earlier in this lesson:**
  - **Mentally walk through the diagram, from upper left to lower right**
  - **Try to identify “most important” input(s), control(s) and output(s)**
  - **Try to determine if there is a main path linking the “most important” input or control and "most important" output**
  - **Check story being told by diagram by considering how familiar situations are handled**
- **Be prepared to share your results with the class, including:**
  - **The main path and most important input, control and output you discovered**
  - **The story being conveyed by the diagram**
  - **Your observations for the exercise**

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		

All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?

Session 4 – System Architecture and Design

Lessons

1. **Program Overview Update**
2. **System Architecture Basics**
3. **Systems Design Process**
4. **Analysis of Alternatives**
5. **Model-Based Systems Engineering**
6. **Special Topics (if time permits)**

Appendices*

I. **SysML Diagrams and Language Concepts**

J. **Architecture Frameworks In-depth**

- IDEF0 In-depth
- Architecture Framework Examples

*Appendices A through H were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered

Architecture Frameworks

- **A structured way to express an architecture**
  - Aids in documenting similar content
  - Useful for
    - Communicating between projects or across the enterprise
    - Comparing architectures

- **Common Frameworks**
  1. Department of Defense Architecture Framework (DoDAF)
  2. Enterprise Architecture Planning (EAP)
  3. Federal Enterprise Architecture Framework (FEAF)
  4. The Open Group Architecture Framework (TOGAF)
  5. Zachman Framework

A framework is not an architecture!

1. DoD Architecture Framework (DoDAF)

- **DoD mandated framework for documenting architecture**
- **Expresses architecture in 8 types of views**
  - **Capability Views (CV-1 to -7)**
  - **Data and Information Views (DIV -1 to -3)**
  - **Operational Views (OV-1 to -5a,b; 6a,b,c)**
  - **Project Views (PV -1 to -3)**
  - **Services Views (SvcV -1 to 3a,b; 4 to 10a,b c)**
  - **Standards Views (StdV -1 & 2)**
  - **System Views (SV-1 to -5a,b; 6 to 10a,b,c)**
  - **All Views (AV-1 & -2)**
- **There are also relationships between views**

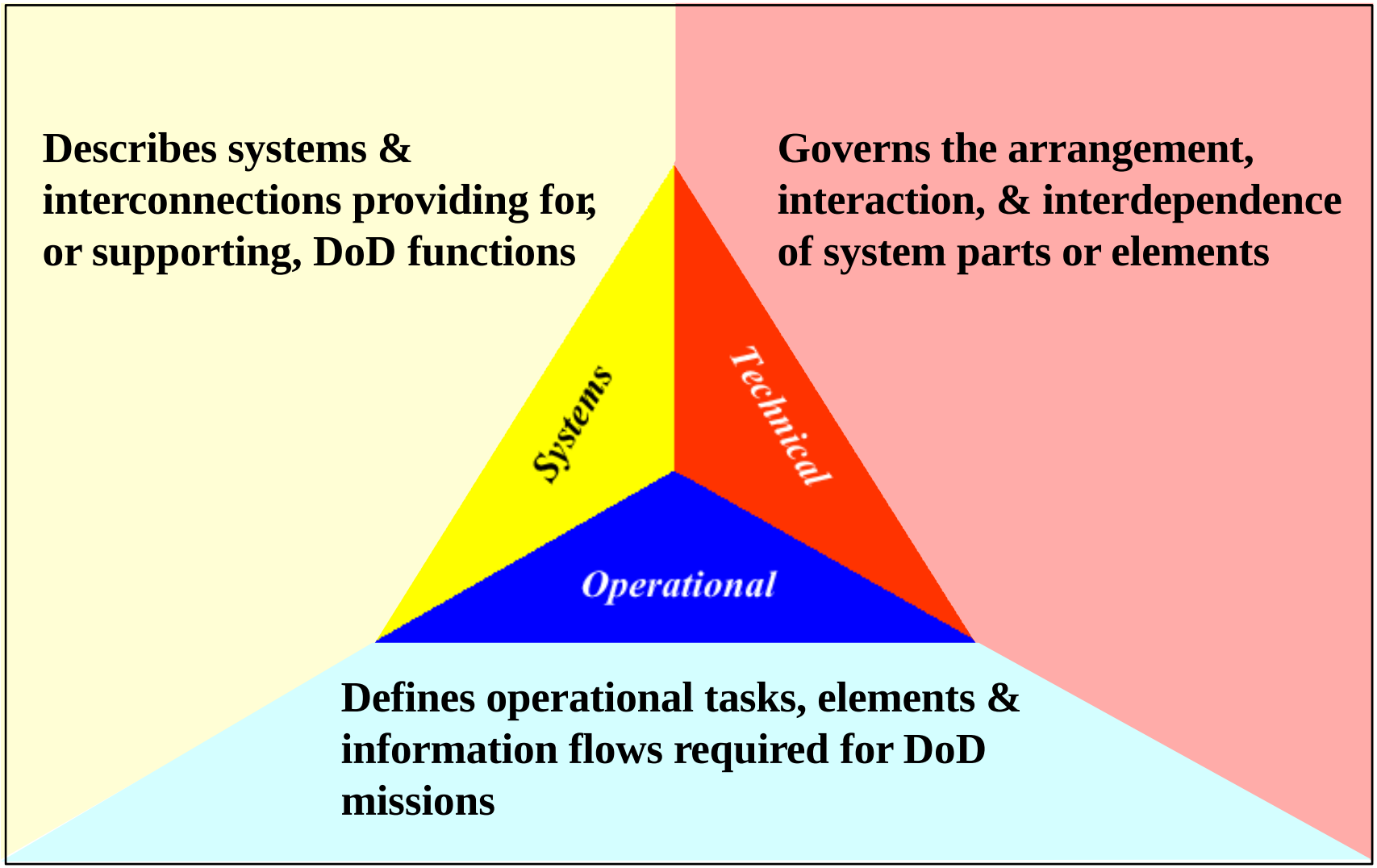
Definition for the DoD Architecture Framework



The DoD Architecture Framework is the structure for organizing architecture concepts, principles, assumptions, and terminology about operations and technology into meaningful patterns to satisfy specific DoD purposes.

CIO/NII Enabling Net-Centric Operations

DoD Architecture Framework (DoDAF)



DoD Architecture Framework (DoDAF)

All Viewpoint
The All Viewpoint describes the overarching aspects of architecture context that relate to all viewpoints.

Capability Viewpoint
The Capability Viewpoint ar<culates the capability requirements, the delivery <ming, and the deployed capability.

Data and Informa6on Viewpoint
The Data and Informa<on Viewpoint ar<culates the data rela<onships and alignment structures in the architecture content for the capability and opera<onal requirements, system engineering processes, and systems and services.

Opera6onal Viewpoint
The Opera<onal Viewpoint includes the opera<onal scenarios, ac<vi<es, and requirements that support capabili<es.

Project Viewpoint
The Project Viewpoint describes the rela<onships between opera<onal and capability requirements and the various projects being implemented. The Project Viewpoint also details dependencies among capability and opera<onal requirements, system engineering processes, systems design, and services design within the Defense Acquisi<on System process. An example is the Vcharts in Chapter 4 of the Defense Acquisi<on Guide.

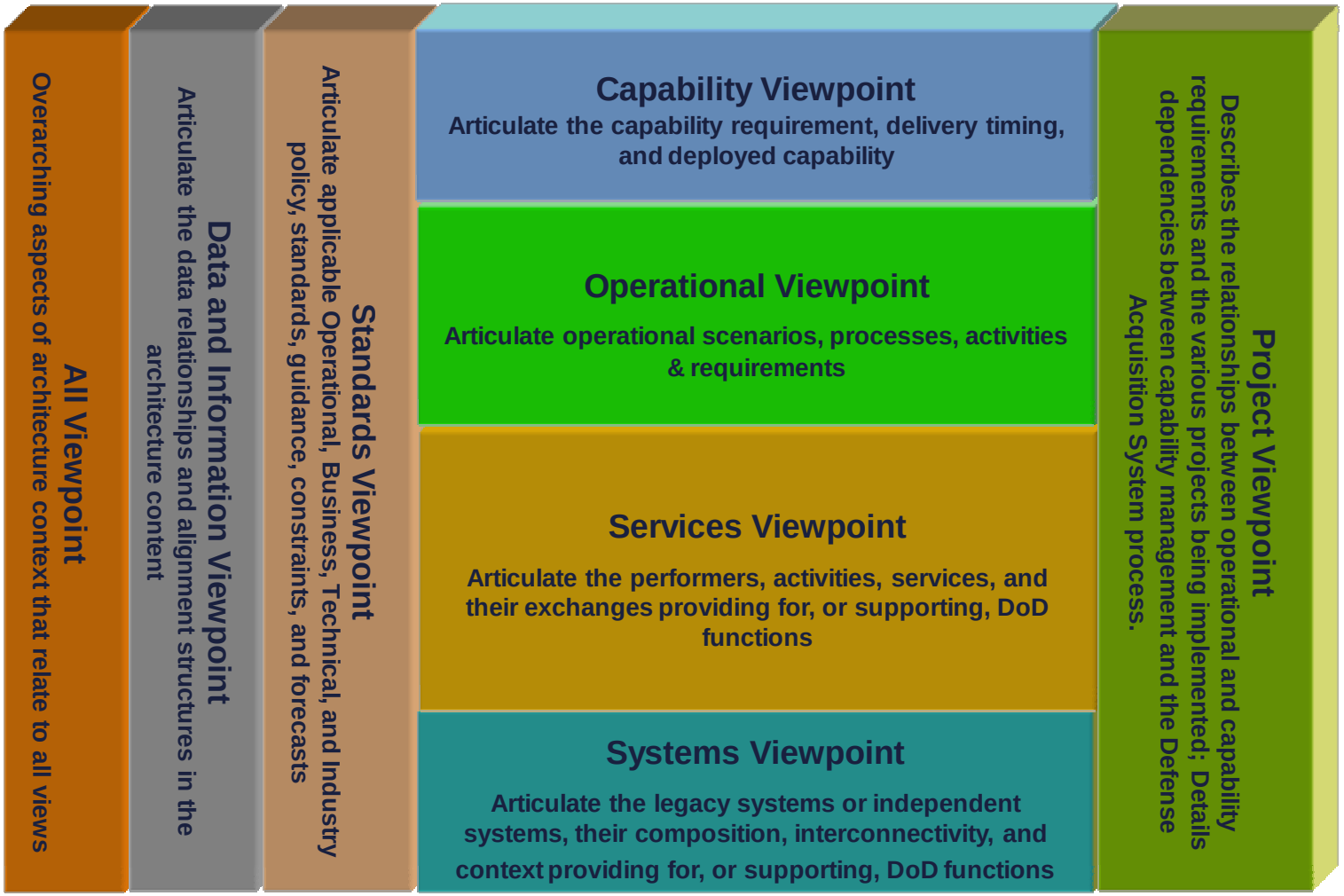
Services Viewpoint
The Services Viewpoint is the design for solu<ons ar<cula<ng the Performers, Ac<vi<es, Services, and their Exchanges, providing for or suppor<ng opera<onal and capability func<ons.

Standards Viewpoint
The Standards Viewpoint ar<culates the applicable opera<onal, business, technical, and industry policies, standards, guidance, constraints, and forecasts that apply to capability and opera<onal requirements, system engineering processes, and systems

Systems Viewpoint
The Systems Viewpoint, for Legacy support, is the design for solu<ons ar<cula<ng the systems, their composi<on, interconnec<vity, and context providing for or suppor<ng opera<onal and capability func<ons.

- **Broken Down into Viewpoints**
- **Broad Summary Information for the Whole Enterprise**
- **Narrowly Focused Information for Specialized Purpose**
- **Information About How Aspects of the Enterprise are Connected**

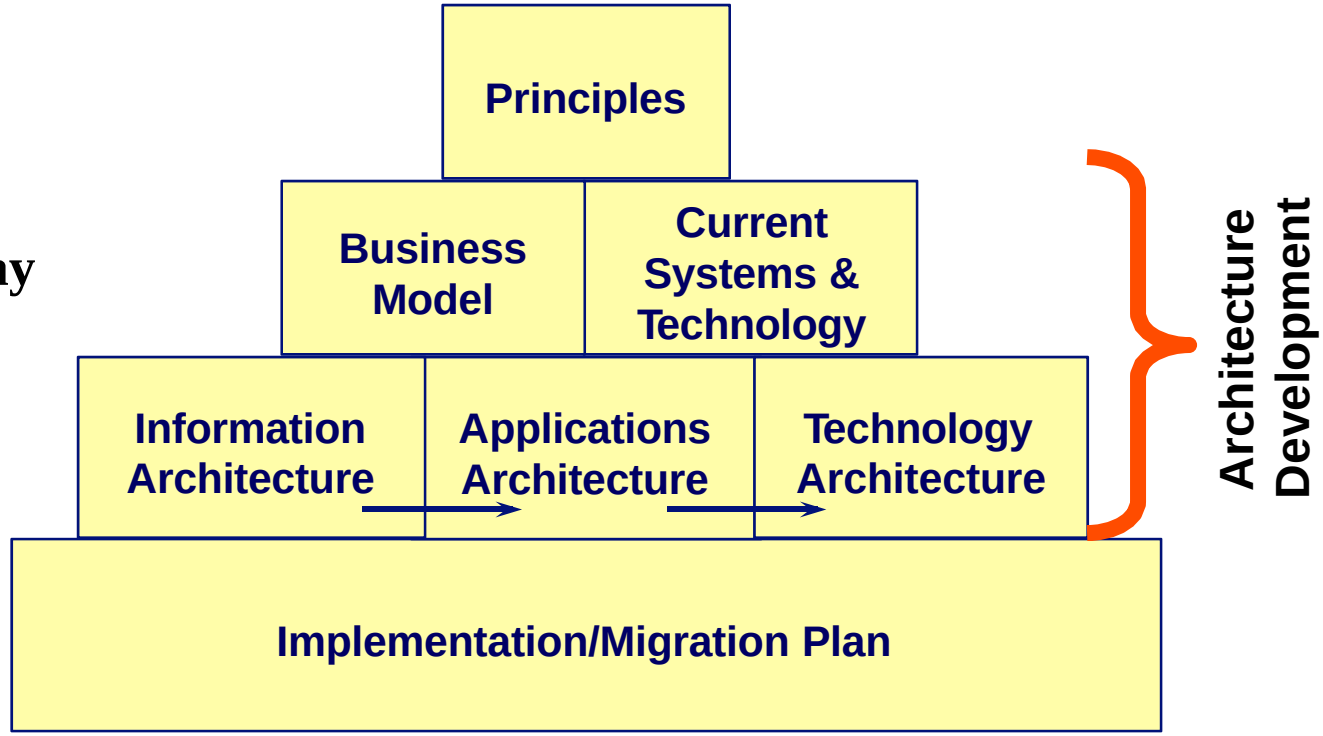
DoD Architecture Framework (DoDAF)



2. Enterprise Architecture Planning (EAP):

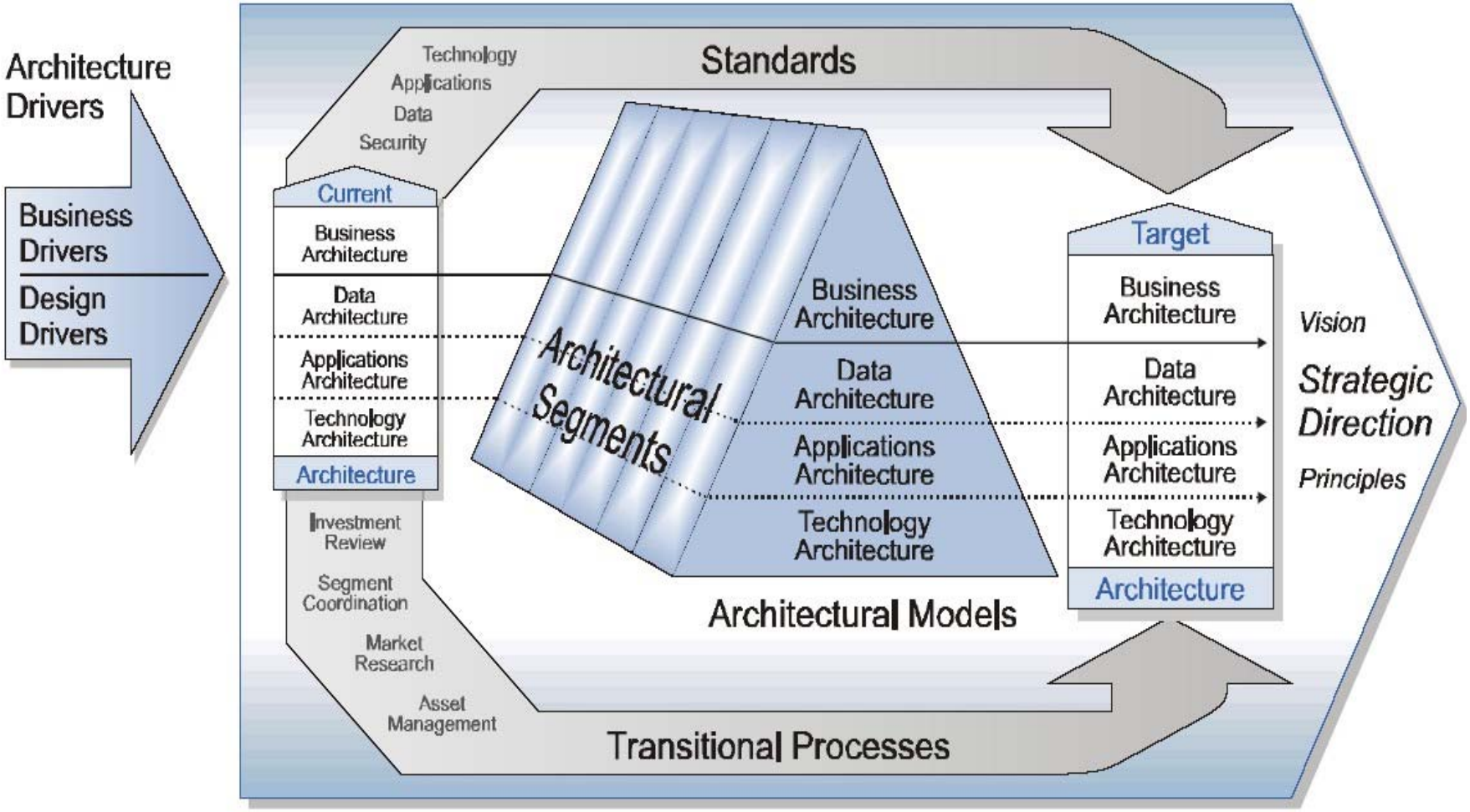
The "Wedding Cake" Framework

1. Where we start
2. Where we are today
3. Where we want to be in the future
4. How to get there



© Steven H. Spewak, Ph.D.

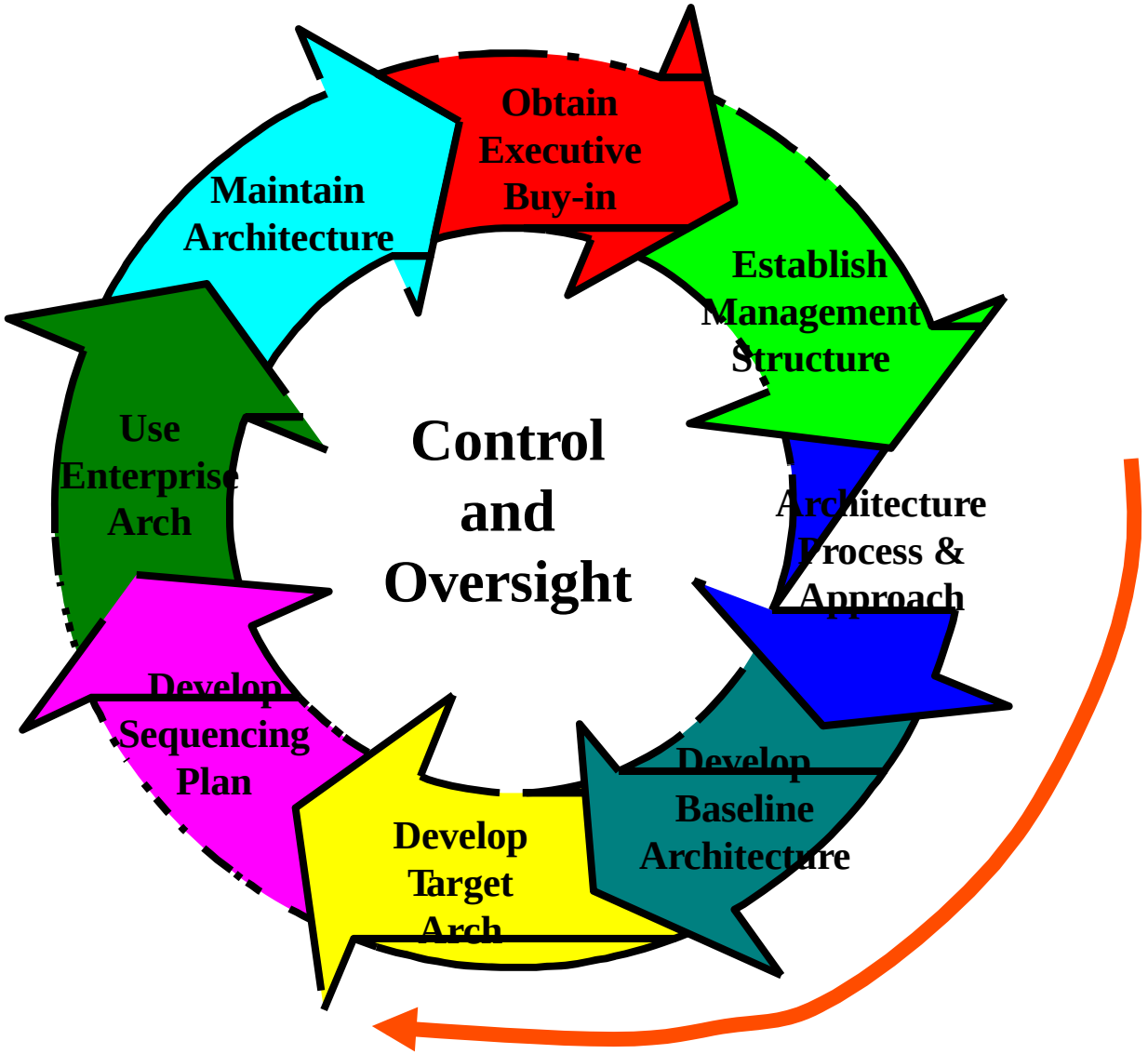
3. Federal Enterprise Architecture Framework



The Federal Enterprise Architecture Framework is an organizing mechanism for managing the development and maintenance of architecture descriptions. The Federal Enterprise Architecture Framework also provides a structure for organizing Federal resources and describing and managing Federal Enterprise Architecture activities.

Source: Federal Enterprise Architecture Framework, Version 1.1, September 1999

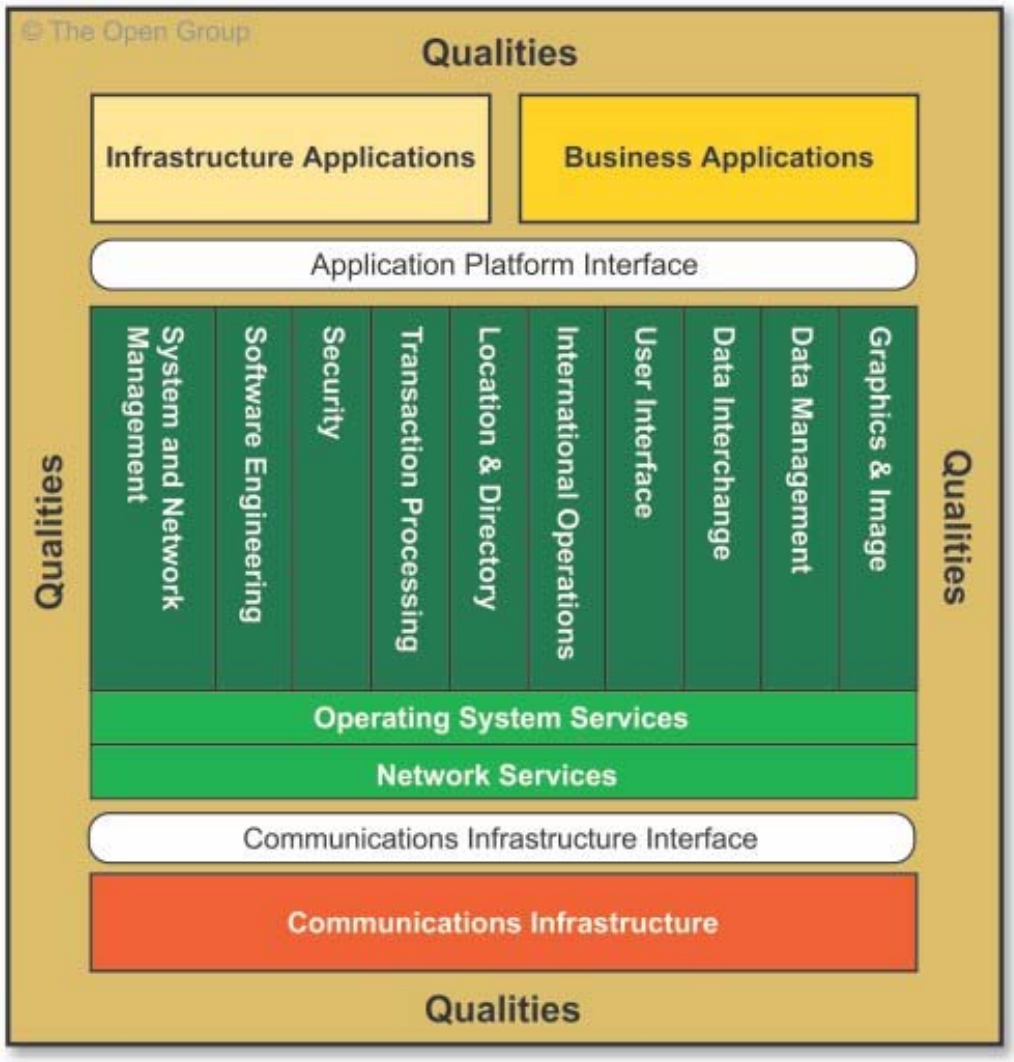
Federal Enterprise Architecture Process



Architecture Development Phases

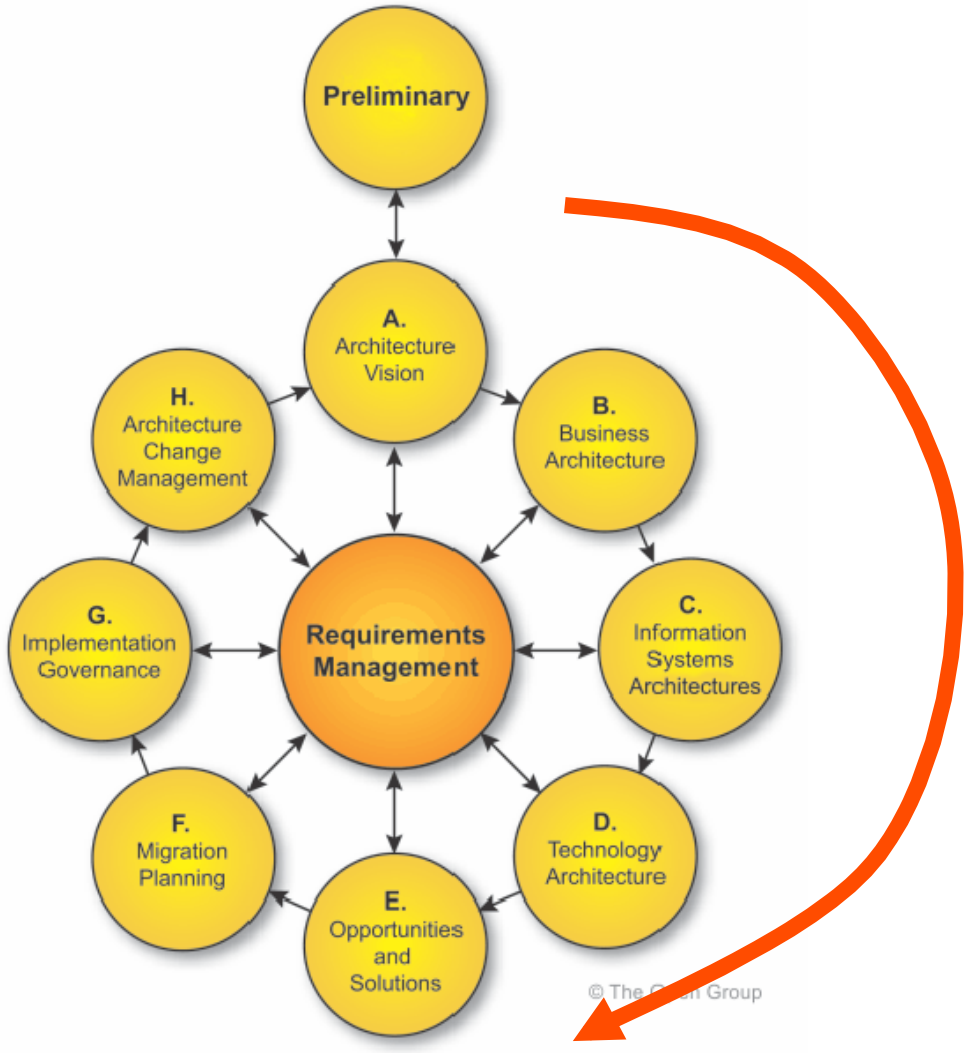
4. The Open Group Architecture Framework (TOGAF)

The Open Group Architecture Framework (TOGAF) is a framework for Enterprise Architecture which provides a comprehensive approach to the design, planning, implementation, and governance of an enterprise information architecture. The architecture is typically modeled at four levels or domains; Business, Application, Data, Technology. A set of foundation architectures are provided to enable the architecture team to envision the current and future state of the architecture.



http://pubs.opengroup.org/architecture/togaf9-doc/arch/

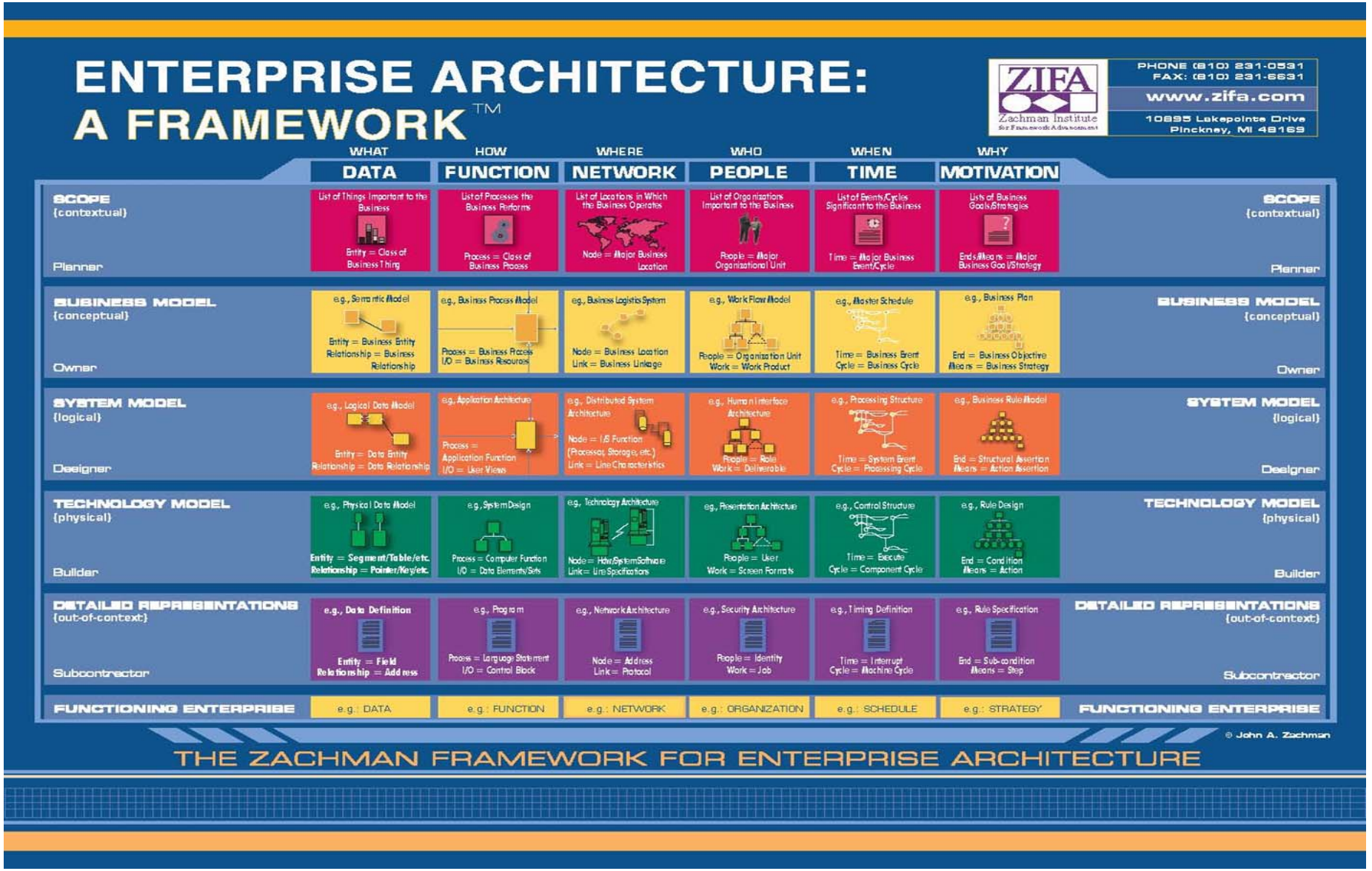
The Open Group Architectural Framework (TOGAF)



Architecture Development Phases

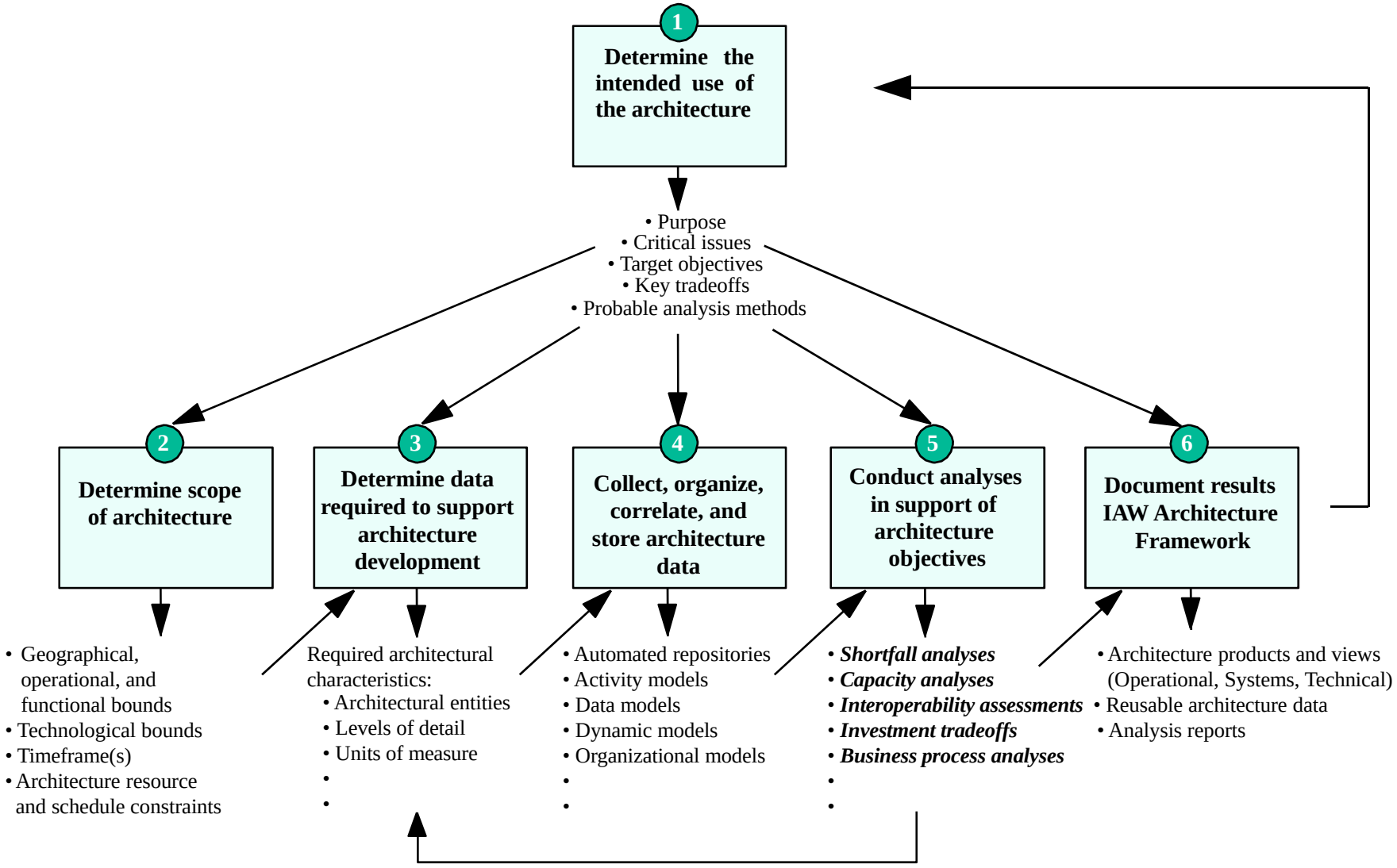
http://pubs.opengroup.org/architecture/togaf9-doc/arch/

5. Zachman Framework



Architecture Development:

A Notional 6-step Process



Architecture Development Guidelines

- **Architectures should be built with a purpose in mind**
  - Having a specific and commonly understood purpose before starting to build an architecture greatly increases the efficiency of the effort and the utility of the resulting architecture. The purpose determines how wide the scope needs to be, which characteristics need to be captured, and what timeframes need to be considered.
  - If groups of architectures built by various organizations are to be compared, it is important that they all be built from the start with the purpose of comparison in mind.
- **Architectures should facilitate, not impede, communication among humans**

Architecture Development Guidelines

(concluded)

- **Architectures should be relatable/comparable across the entire enterprise**
  - **This principle requires the use of *common terms and definitions*.**
  - **This requires that a common set of architectural "building blocks" be used as the basis for architecture descriptions.**

- **Architectures should be modular, reusable, and decomposable**
  - **Architecture representations should consist of *separate but related pieces* that can be recombined with a minimum amount of tailoring, so that they can be used for multiple purposes.**

Architecture Development Lessons Experienced

(but not necessarily learned)

- Architectures are *not the end product*; they are a *means to an end*
- The "end" is to develop and field systems that enable *successful operations*:
  - To implement chosen Operational Concepts (CONOPs) and selected Courses of Action (COA)
  - To inter-operate with customers and business partners
  - To interact with supporting organizations as required
- There is no clear demarcation line between architecture development and system design. Architectures, to be useful, should normally remain at a high level
- Architectures should *facilitate*, not impede, *communication among humans* (showing the computations instead of the results)

Architecture and Systems Engineering

- **Architecting is an essential part of Systems Engineering, especially when complex, unprecedented systems are to be developed and when problems are ill structured**
- **In each case, there is *one framework* with *multiple views*. Consequently, one Chief Architect should be responsible for all architecture views – not an Operational Architect, a Systems Architect, etc.**
- **The Architect's primary responsibility is to *articulate the vision of the user/operator***
- **The System Engineer's role is to implement systems that conform to the architecture (iaw views created) within cost and time constraints**
- **There is *healthy dynamic tension* between the two roles, i.e., pragmatic checks and balances.**

Architecture and Systems Engineering

- Architecting is ONE facet of systems engineering
- It *isn't every* facet of systems engineering
- It *isn't the "most important"* facet either
  - But no facet is more important!
- That's because importance is context sensitive!

HOWEVER

- (verb) ***To DESIGN*** is to progressively refine (formalized and structured additions of detail) the "functional" content of something
- (noun) ***a DESIGN*** is that refinement at any stage in the process.
- Design begins when the architecture has been initially defined and stops when implementation begins
- Designs are specific to disciplines

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		

All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?

Back-up slides

- The following slides provide more context on the use of IDEF0 diagrams.
- These slides are best used right after slide 17.

IDEF0 Diagrams and Text

- Top-Level Context Diagram
- Child Diagram
- Parent Diagram
- Text and Glossary
- For Exposition Only Diagrams

Text and Glossary

- Text
  - Associated textual information used to clarify model
- Glossary
  - Definitions of
    - Processes (activities, functions)
    - Inputs
    - Controls
    - Outputs
    - Mechanisms
  - Examples
    - Get widget parts (process)
      - The process of getting widget parts from the stock areas so that widgets may be assembled.
    - Parts for widgets (output)
      - Parts retrieved from the workstation stock areas and ready to be used in assembly.

For Exposition Only Diagram

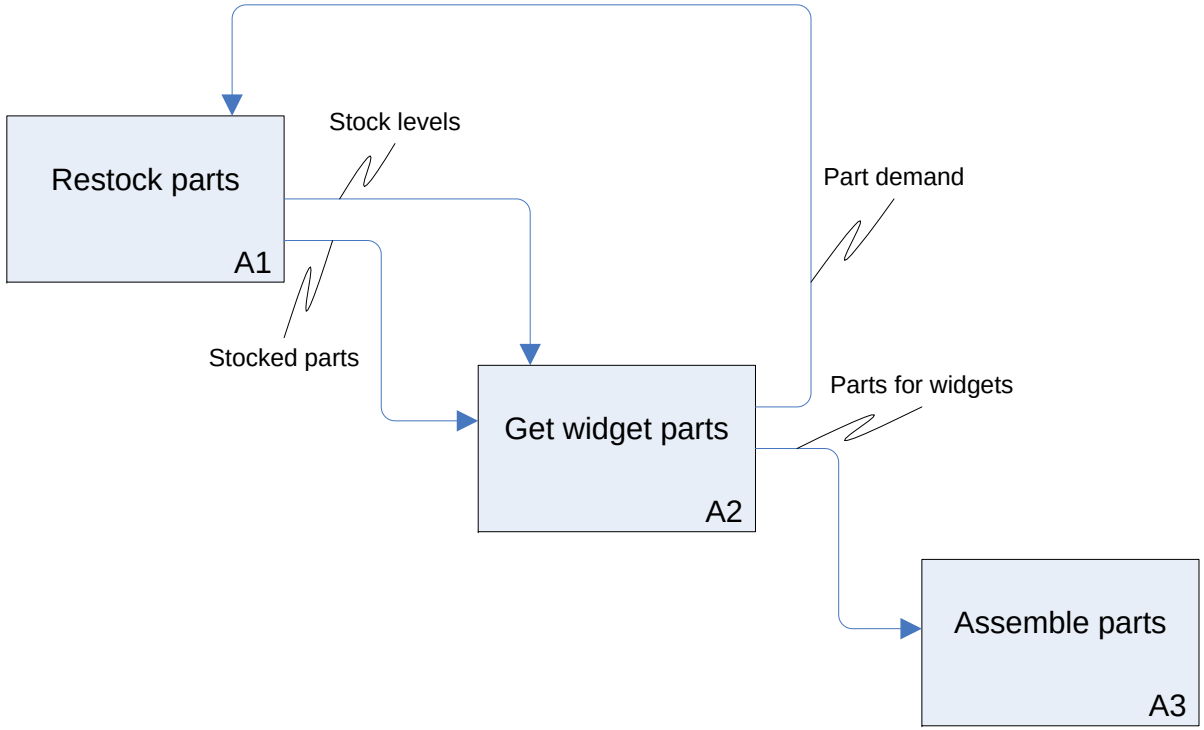
- **FEO (“fee-oh”)**
- **Provides supplementary information to help reader understand model**
- **Need not comply with IDEF0 rules**
- **Example: Flowchart to describe a procedure (action/decision sequence) that can be used to perform the process**

Diagram Features

- Arrows As Constraints
- Concurrent Operation
- Arrows As Pipelines
- Branching Arrows
- Inter-Box Connections
- Boundary Arrows
- Tunneled Arrows
- Call Arrows

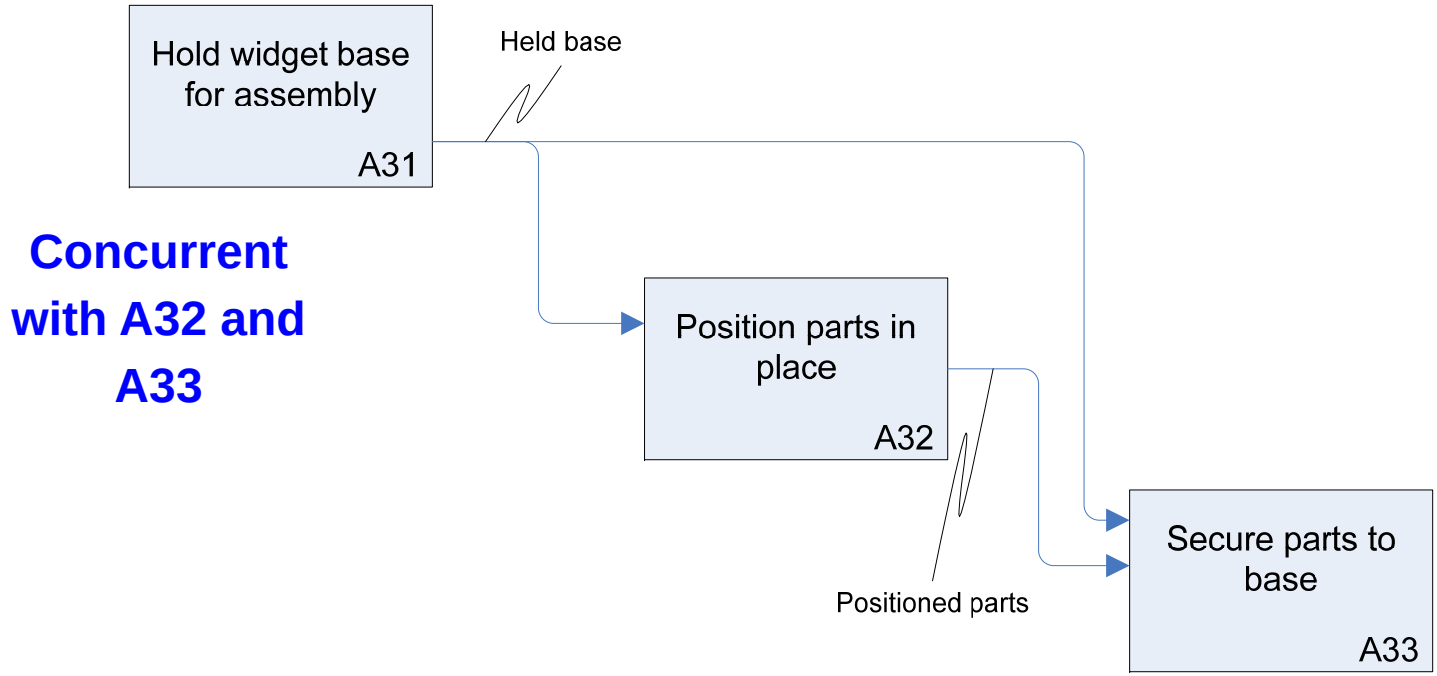
Arrows As Constraints

- **Connecting output of a box representing a process that is input/control/mechanism to another box means that the second process is constrained by the first.**



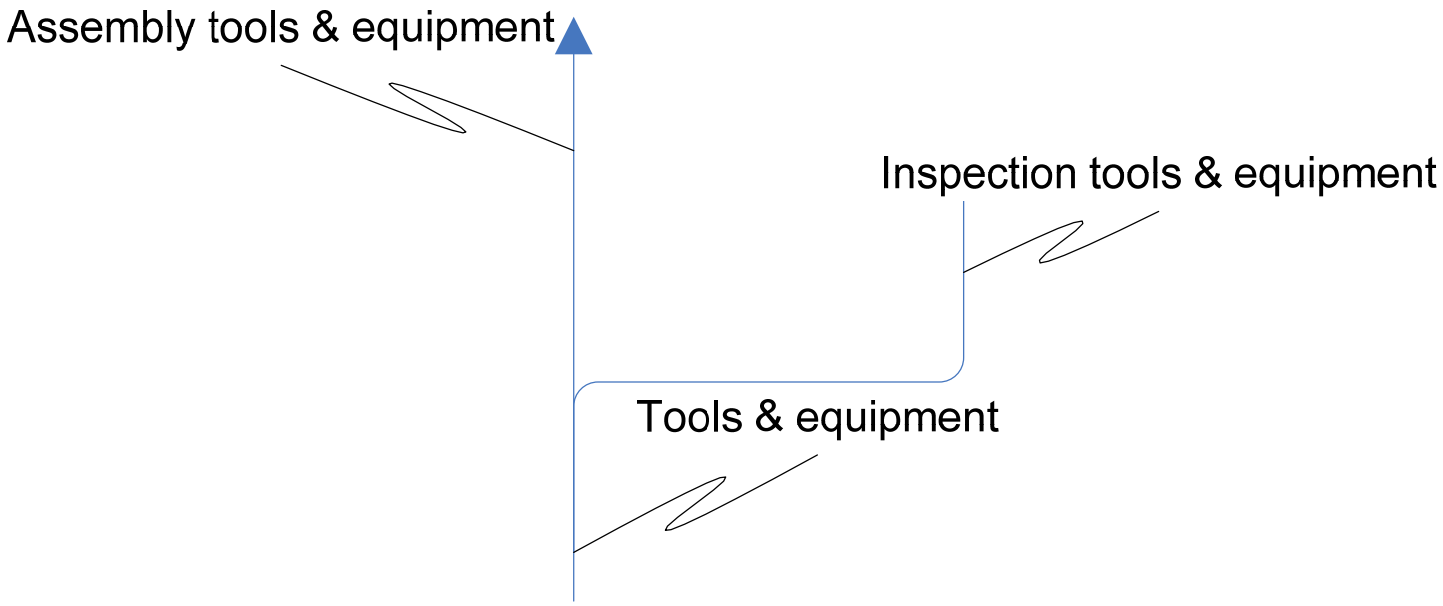
Concurrent Operation

- **Box order and connections do not necessarily imply sequence!**
- **Processes may proceed concurrently**

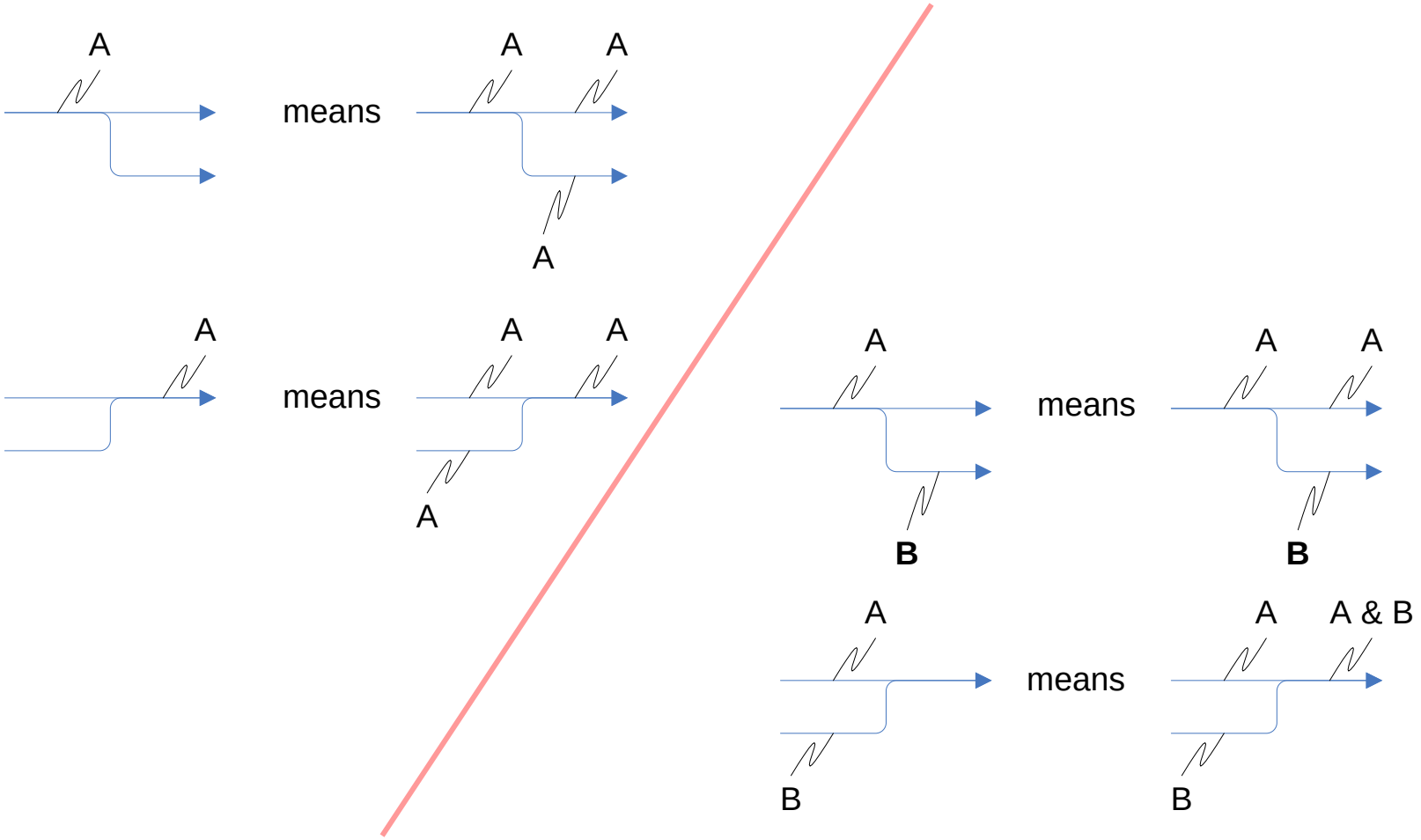


Arrows As Pipelines

- **Think of arrows as pipelines or conduits**
- **High-level arrows have general labels**
- **Low-level arrows have specific labels**
- **If an arrow forks, the branches may have more specific labels**



Forking and Joining Arrows

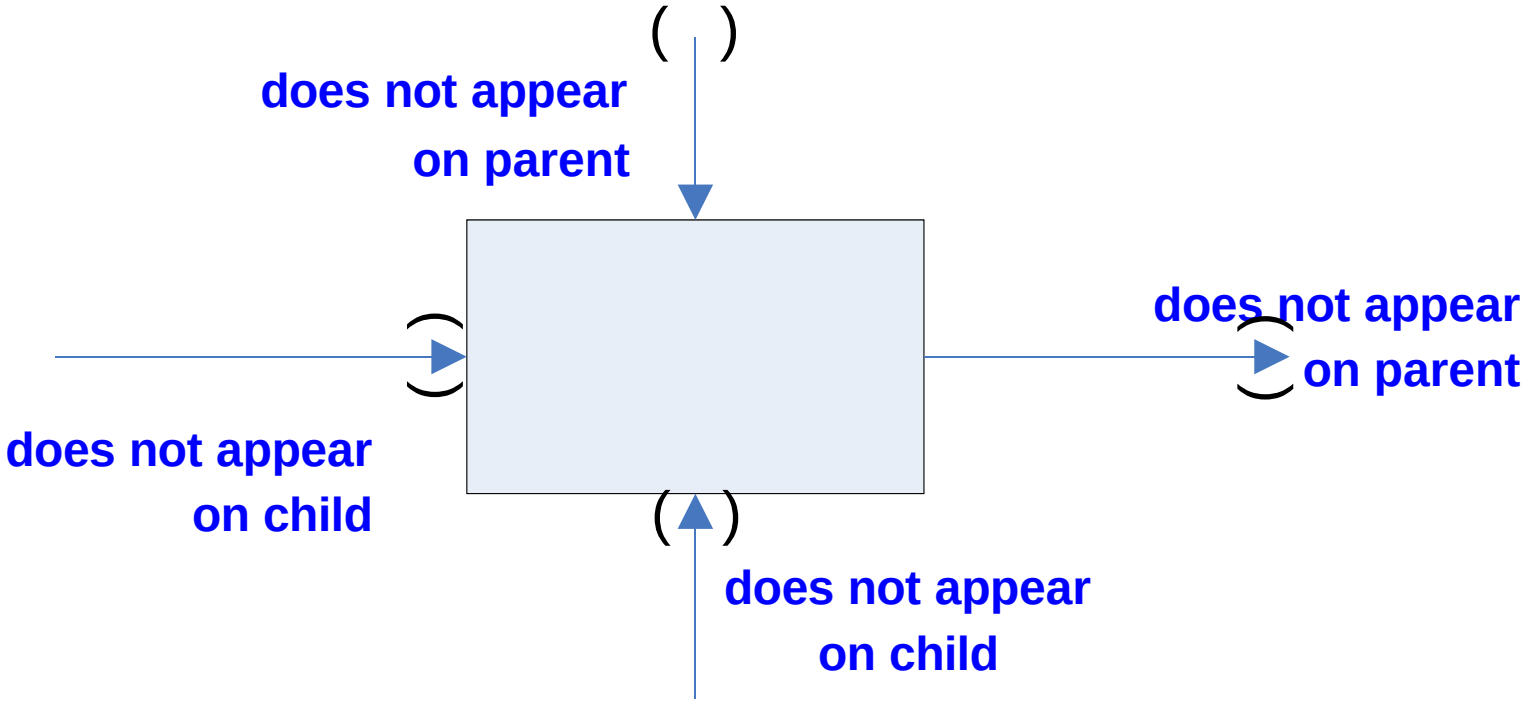


Inter-Box Connections

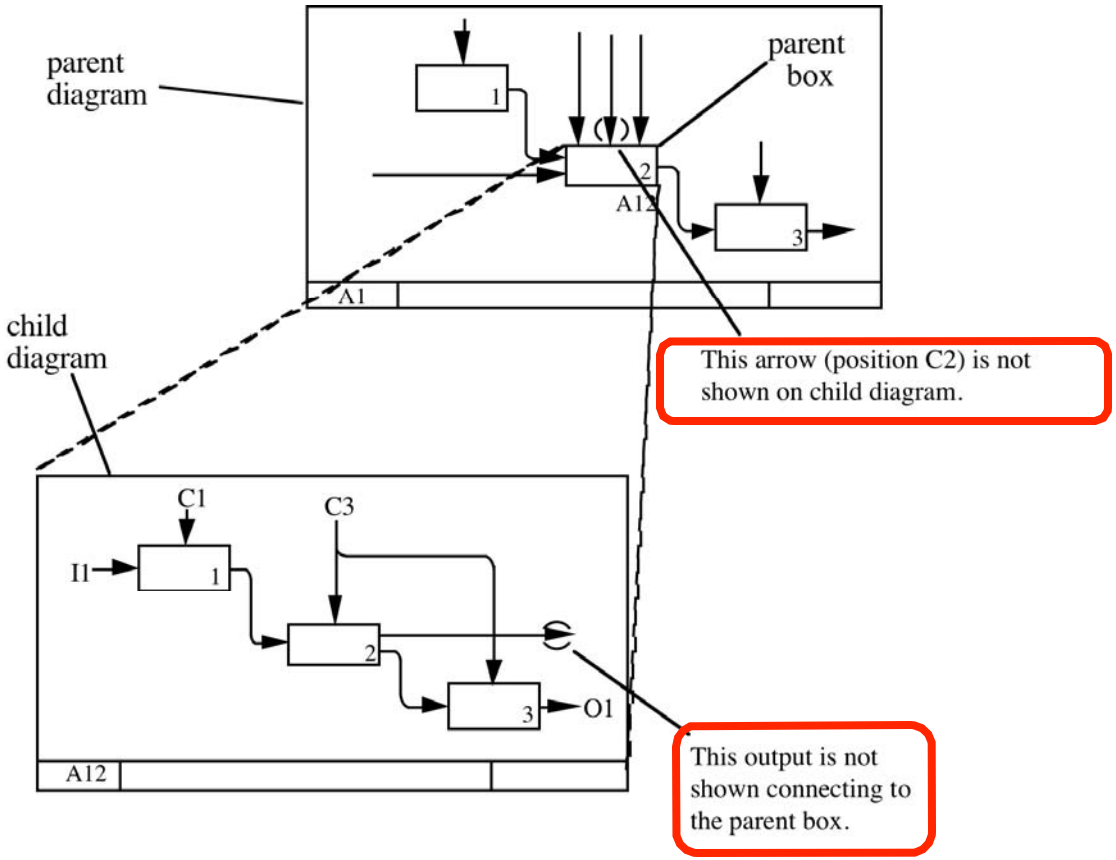
- **Except for A–0, diagrams contain 3 – 6 boxes**
- **Normally organized on diagonal (“staircase”)**
- **Any output of one box may be input, control, or mechanism of another box**
- **If box is detailed on child diagram, every arrow connected to the box appears on the child diagram (unless it is *tunneled*, discussed later)**

Tunneled Arrows

- **Arrows that provide information at one level of decomposition but are not needed at another (parent, child) level.**



Tunneled Arrows



Call Arrows

- Special case of mechanism arrow
- Caller box does not have its own child diagram
- Detailed by another box in same or other model
- Example: same “Fly aircraft” model called by
  - “Fly aircraft in climb”
  - “Fly aircraft in cruise”
  - “Fly aircraft in descent”



Systems Engineering Technical Short Course

(Session 4: System Architecture and Design)

Appendix I: SysML Diagrams and Language Concepts

References include OMG Systems Modeling Language (OMG SysML™) Tutorial, 19 June 2008 revb, Sanford Friedenthal, Alan Moore and Rick Steiner, presented at the INCOSE International Symposium 2008, The Netherlands.

What is SysML?

- A graphical modeling language in response to the UML for Systems Engineering RFP developed by OMG, INCOSE and AP233

- It IS a visual modeling language that provides…
  - Semantics (meaning)
  - Notation (representation of meaning)

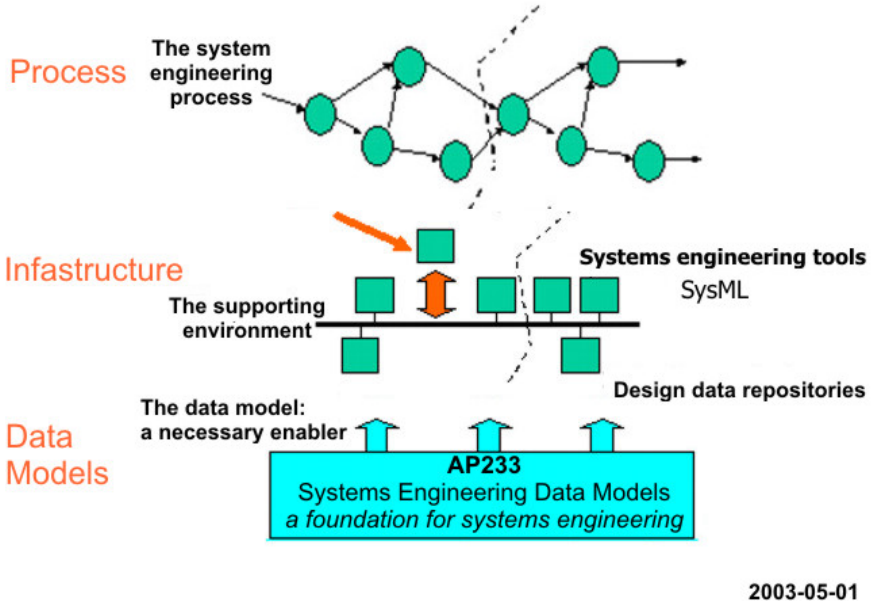
- It IS NOT a methodology or a tool
  - SysML is methodology and tool independent

SysML is Critical Enabler for Model Driven SE

AP233 and ISO 10303 (STEP)

- **AP233 is a data exchange protocol for systems engineering data based on ISO 10303 (STEP)**

- **STEP is a standard to describe, represent and exchange industrial data in a computer interpretable format**
  - **The intent of AP233 is to support the whole system development life cycle ranging from requirements definition to system verification and validation**



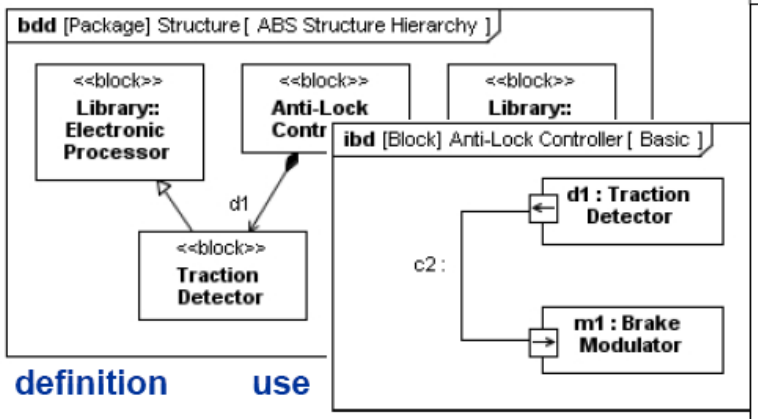
See http://pdesinc.aticorp.org and http://www.ap233.org

SysML Describes System Aspects

- Structural composition, interconnection, and classification
- Function-based, message-based, and state-based behavior
- Constraints of the physical and performance properties
- Allocations between behavior, structure, and constraints (e.g. functions allocated to components)
- Requirements and their relationship to other requirements, design elements, and test cases

Four Pillars of SysML

1. Structure

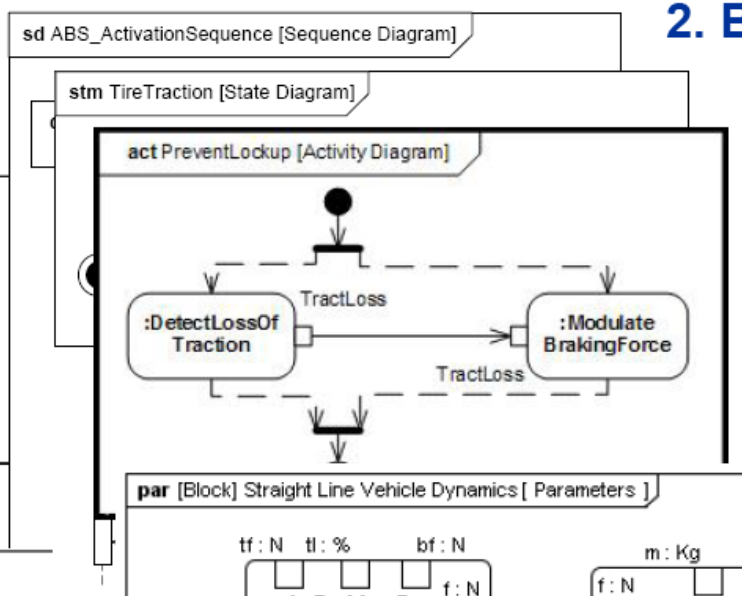


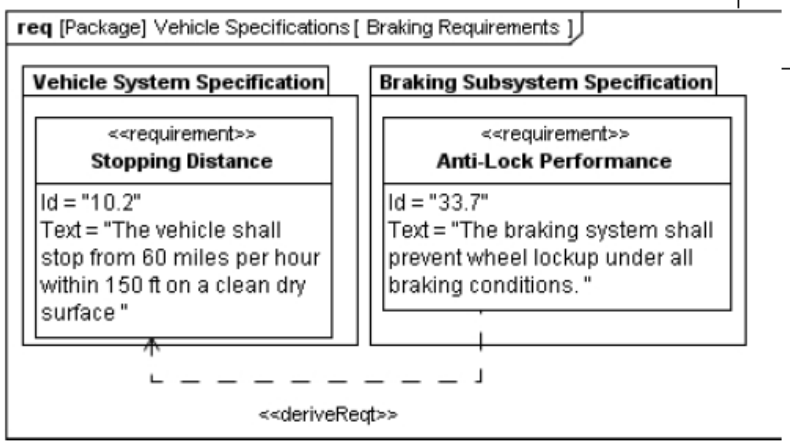
2. Behavior

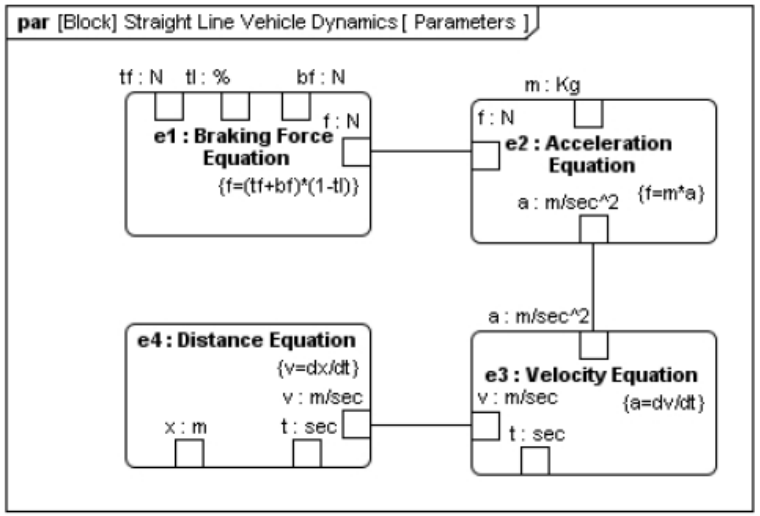
interaction

state machine

activity/ function





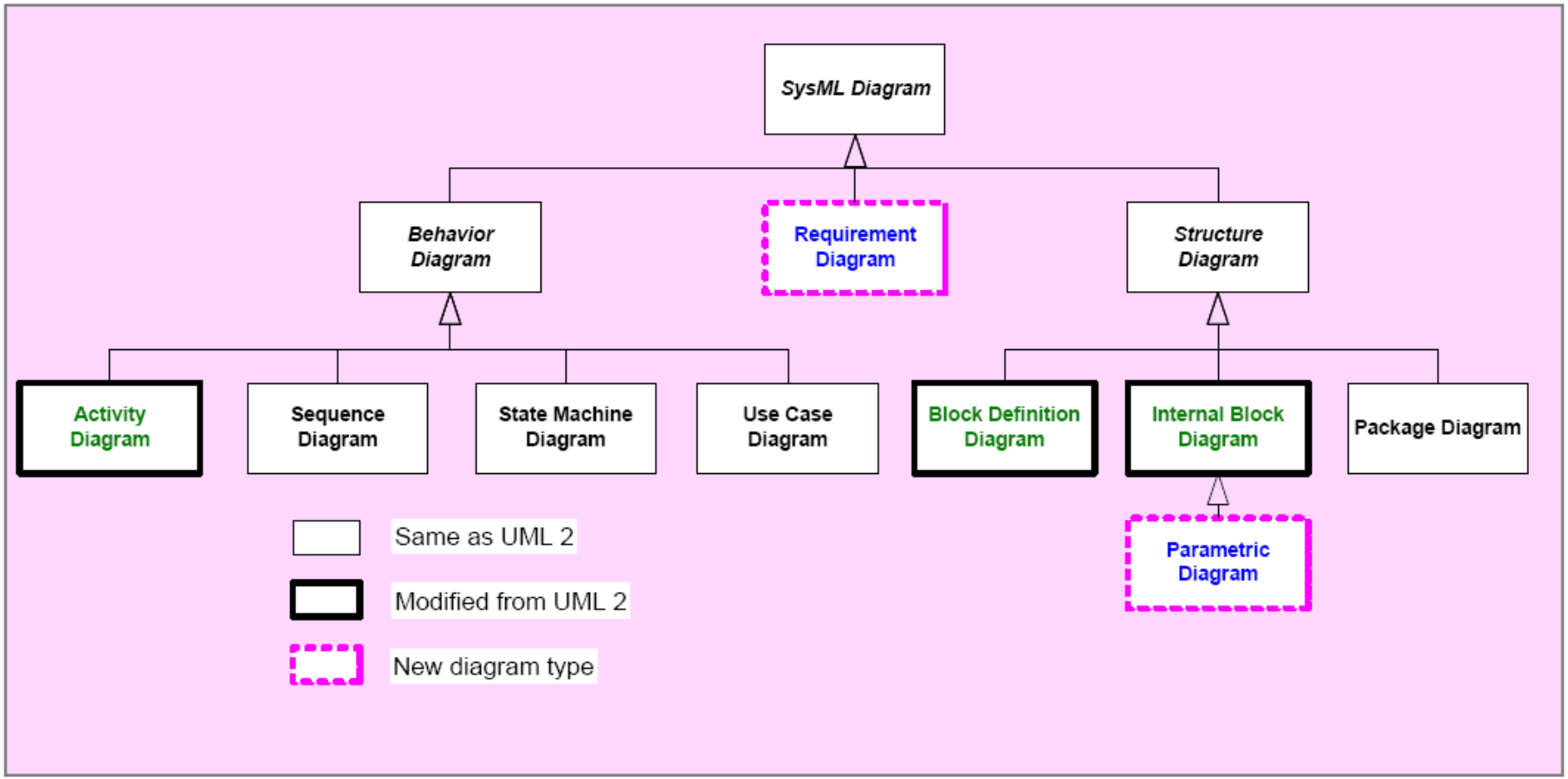


3. Requirements

4. Parametrics

Antilock Braking System

SysML Diagram Taxonomy



SysML Structure Diagrams

- **Block Definition Diagram (bdd)**
  - **Represents structural elements called blocks, and their composition and classification**
- **Internal Block Diagram (ibd)**
  - **Represents interconnections and interfaces between the parts of a block**
- **Parametric Diagram (par)**
  - **Represents constraints on property values, such as *F=ma*, used to support engineering analysis**
- **Package Diagram (pkg)**
  - **Represents the organization of a model in terms of packages that contain model elements**

SysML Behavior Diagrams

- **Activity Diagram (act)**
  - **Represents behavior in terms of ordering actions based on the availability of inputs, outputs, and control, and how the actions transform the inputs to outputs**
- **Sequence Diagram (sd)**
  - **Represents behavior in terms of a sequence of messages exchanged between parts**
- **State Machine Diagram (stm)**
  - **Represents behavior of an entity in terms of its transitions between states triggered by events**
- **Use Case Diagram (uc)**
  - **Represents functionality in terms of how a system or other entity is used by external entities (i.e. actors) to accomplish a set of goals.**

SysML Requirements Diagram

- **Requirements Diagram (req)**
  - **Represents text-based requirements and their relationship to other requirements, design elements, and test cases to support requirements traceability**

SE Activities and SysML

- **Capture and analyze black box system requirements**
  - **Capture text-based requirements in a requirements management tool**
  - **Import requirements into the SysML tool**
  - **Identify top-level functionality in terms of system use cases**
  - **Capture the traceability between the use cases and requirements**
  - **Model the use case scenarios as activity diagrams, sequence diagrams and/or state machine diagrams**
  - **Create a context diagram**
  - **Identify test cases to support system verification**

SE Activities and SysML

- **Develop one or more candidate system architectures to satisfy the requirements**
  - ***Decompose*** **the system using the block definition diagram**
  - **Define the *interaction* among the parts using activity or sequence diagrams**
  - **Define the *interconnection* among the parts using the internal block diagram**

- **Perform engineering and trade-off analysis to evaluate and select the preferred architecture**
  - **Capture the *constraints* on system properties using the parametric diagram to support analysis of performance, reliability, cost, and other critical properties**
  - **Perform the *engineering analysis* to determine the budgeted values to the system properties**

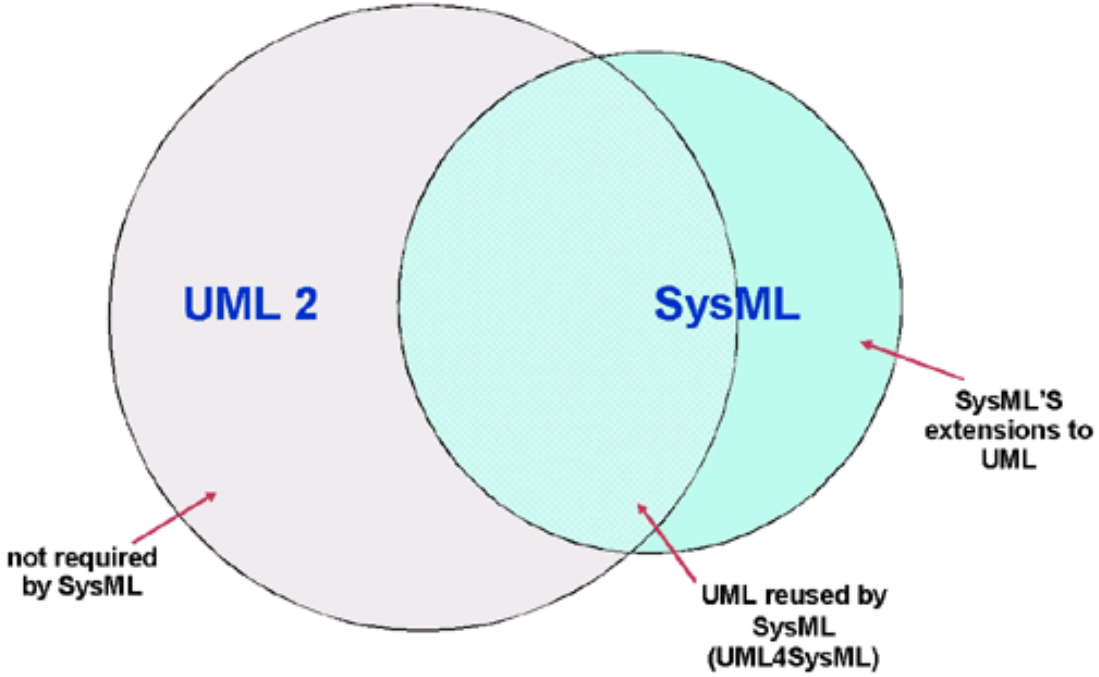
SE Activities and SysML

- **Specify component requirements and their traceability to system requirements**
  - **Capture the functional, interface, and performance requirements for each component (block) in the architecture**
  - **Trace component requirements to the system requirements**

- **Verify that the system design satisfies the requirements by executing system-level test cases**

SysML
Diagrams and Language Concepts

Relationship Between UML and SysML



SysML Extensions

- **Blocks**
- **Item flows**
- **Value properties**
- **Allocations**
- **Requirements**
- **Parametrics**
- **Continuous flows**

Some Fundamental SysML Properties / 1

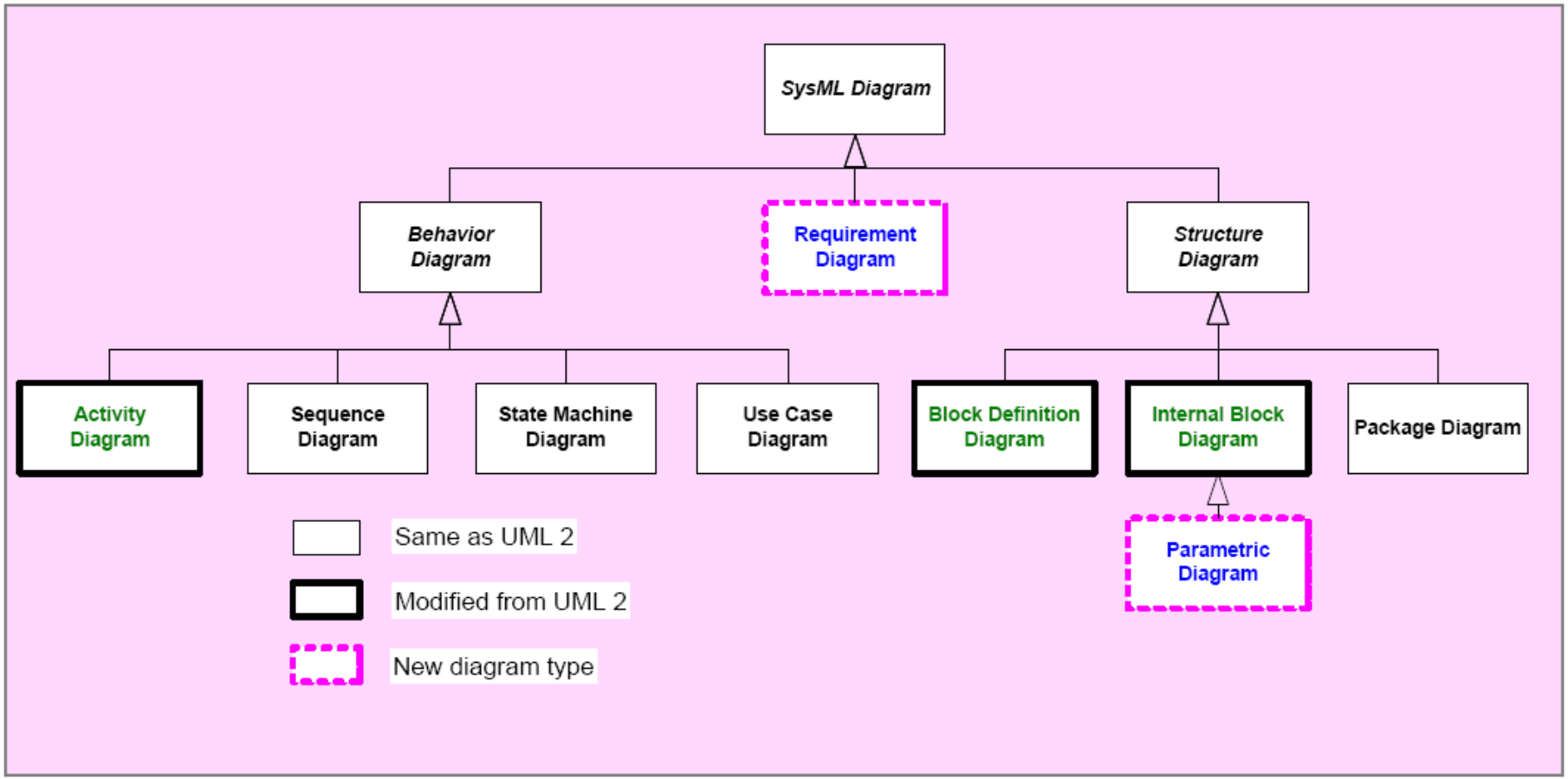
- **The «block» is the basic unit of structure in SysML**
  - **Can represent any level of the system hierarchy including the top level, system, or subsystems**
  - **Can represent *logical* or *physical* components of a system or environment**
  - **Can represent hardware, software, facilities, personnel, etc.**

- **The «block» describes a system as a collection of parts and connections between them that enable communication and other forms of interaction**
  - ***Ports* provide access to the internal structure of a block for use when the object is used within the context of a larger structure**
    - **SysML provides *Service-Oriented Standard Ports***
    - **SysML also provides *Flow Ports* that define flows in or out of a block**

Some Fundamental SysML Properties / 2

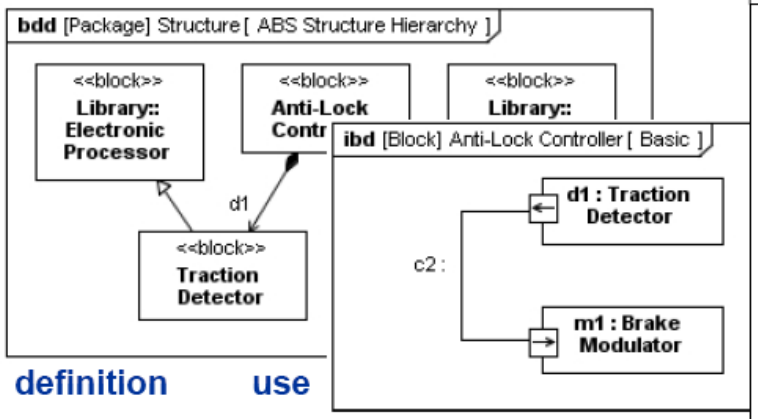
- SysML includes a graphical construct to represent text-based requirements and relate them to other model elements
  - The requirement diagram provides a bridge between the typical requirements management tools and the system models
- The parametric diagram represents constraints on system property values such as performance, reliability, and mass properties, and serves as a means to integrate the specification and design models with engineering analysis models
- SysML also includes an allocation relationship to represent various types of allocation, including allocation of functions to components, logical to physical components, software to hardware, etc.

SysML Diagram Taxonomy



Four Pillars of SysML

1. Structure

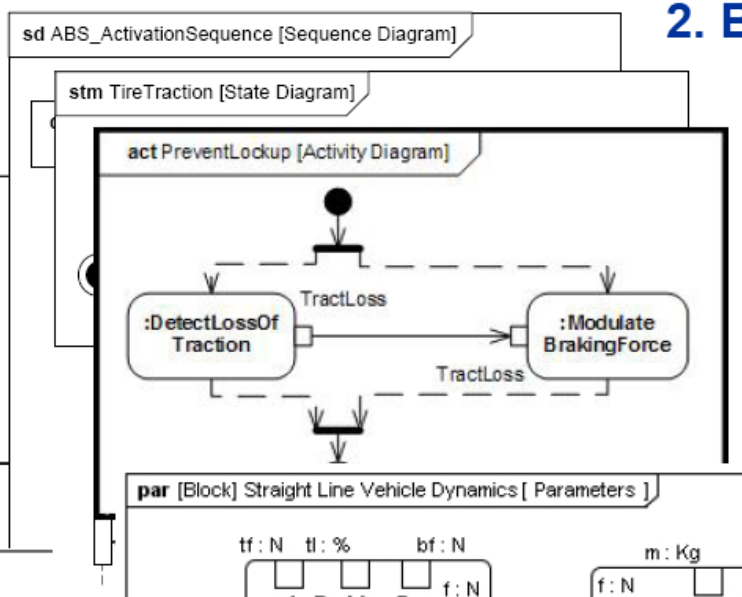


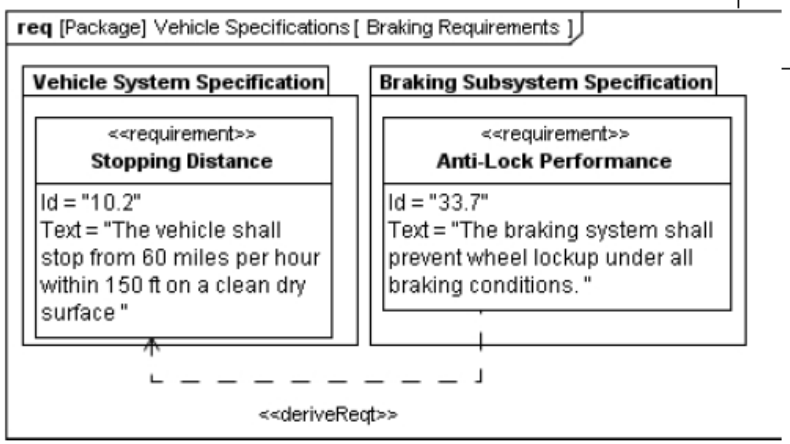
2. Behavior

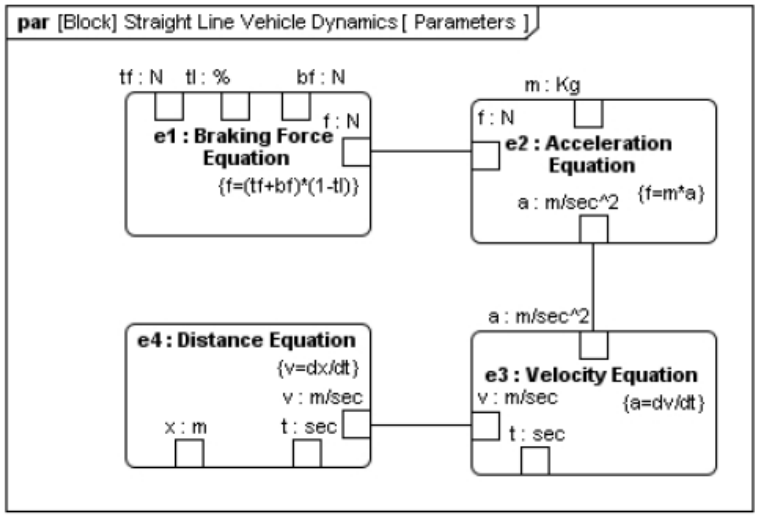
interaction

state machine

activity/ function







3. Requirements

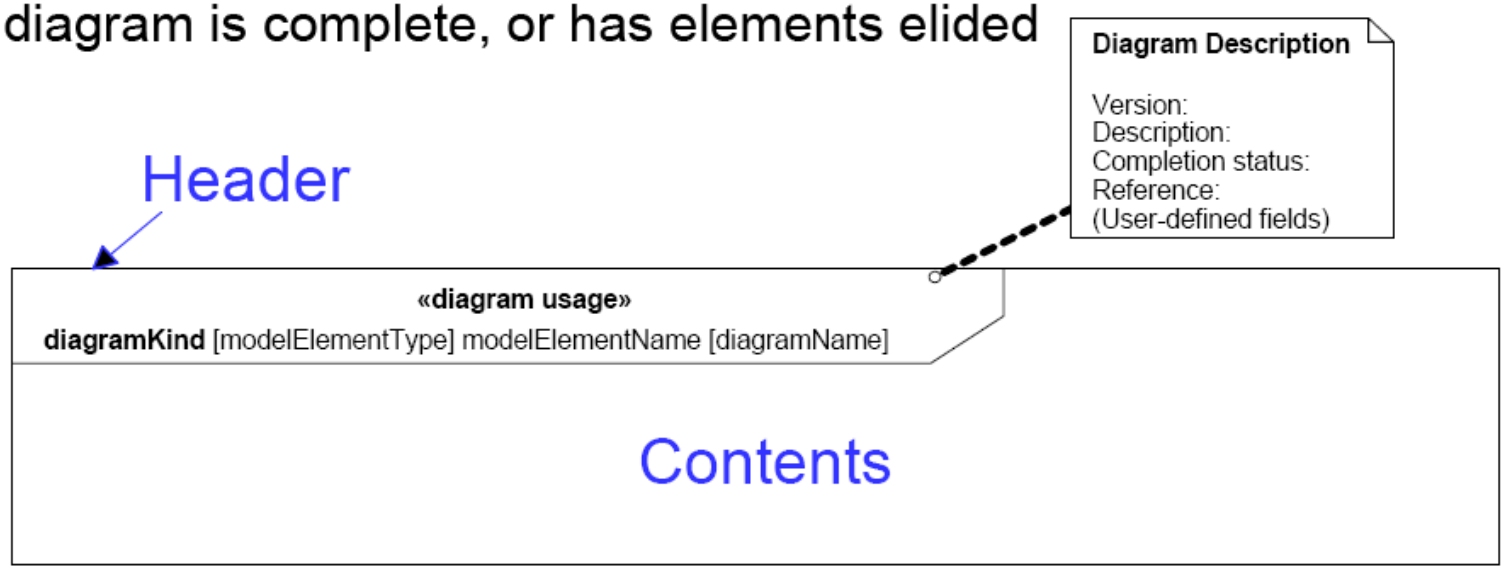
4. Parametrics

18

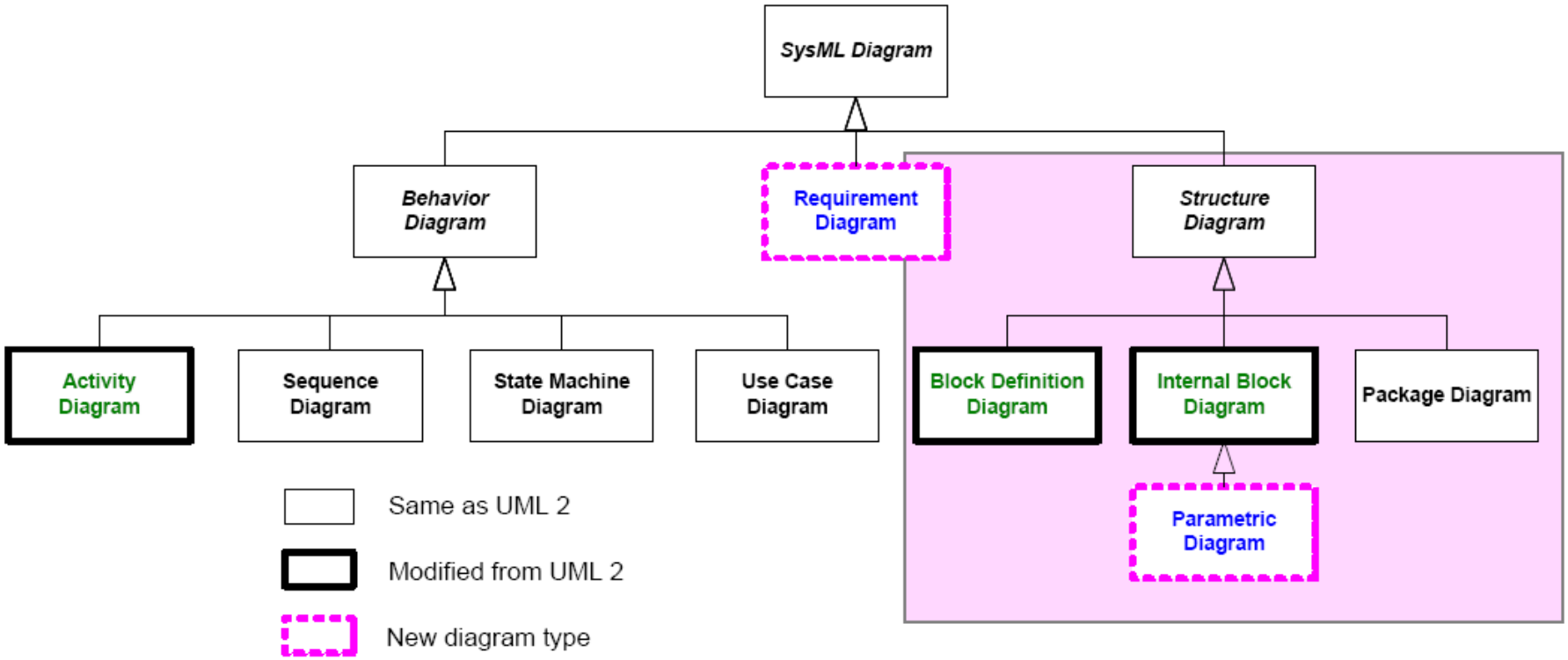
Antilock Braking System

SysML Diagram Features

- Each SysML diagram represents a model element
- Each SysML Diagram must have a Diagram Frame
- Diagram context is indicated in the header:
  - Diagram kind (act, bdd, ibd, sd, etc.)
  - Model element type (package, block, activity, etc.)
  - Model element name
  - User defined diagram name or view name
- A separate diagram description block is used to indicate if the diagram is complete, or has elements elided



Structure Diagrams



SysML Diagram Overview

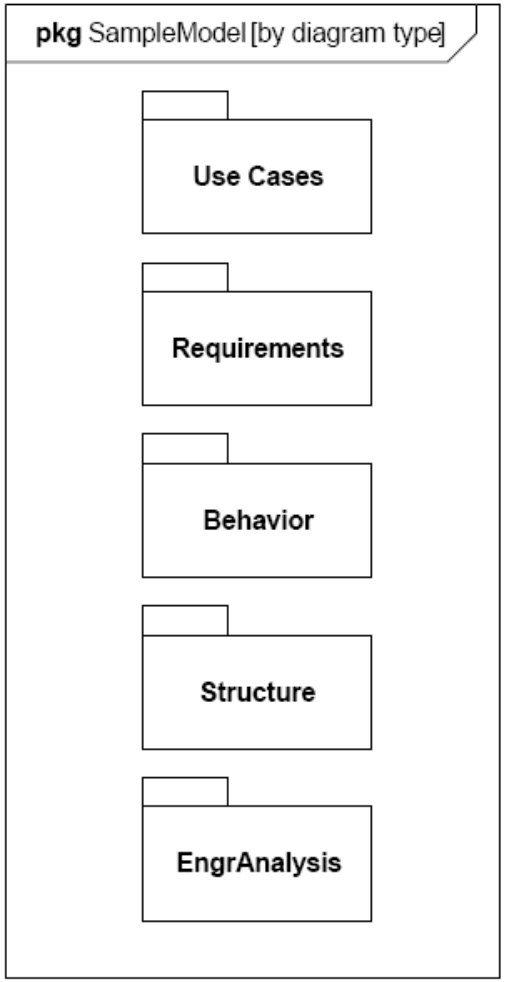
- **Package Diagrams** [pkg] **represent the organization of a model in terms of packages that contain model elements (package, block, activity, ...)**

Package Diagram

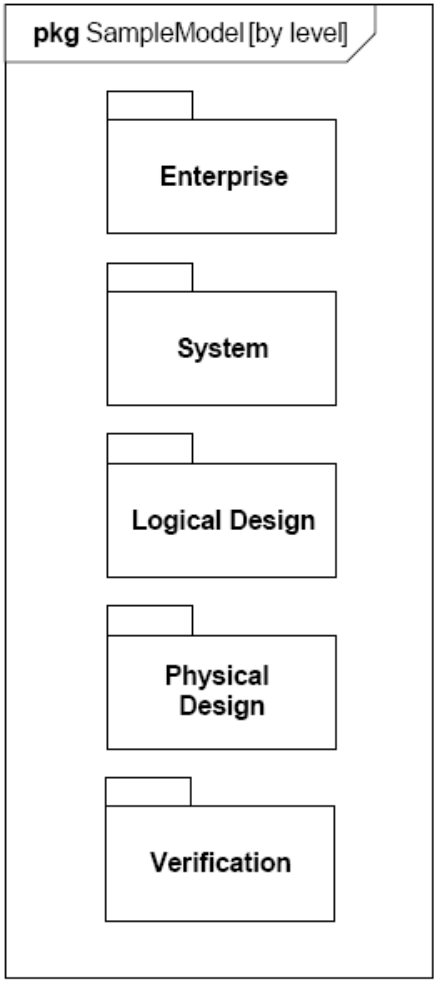
- Package diagram is used to organize the model
  - Groups model elements into a name space
  - Often represented in tool browser
  - Supports model configuration management (check-in/out)
- Model can be organized in multiple ways
  - By System hierarchy (e.g., enterprise, system, component)
  - By diagram kind (e.g., requirements, use cases, behavior)
  - Use viewpoints to augment model organization
- Import relationship reduces need for fully qualified name (package1::class1)

Package Diagram

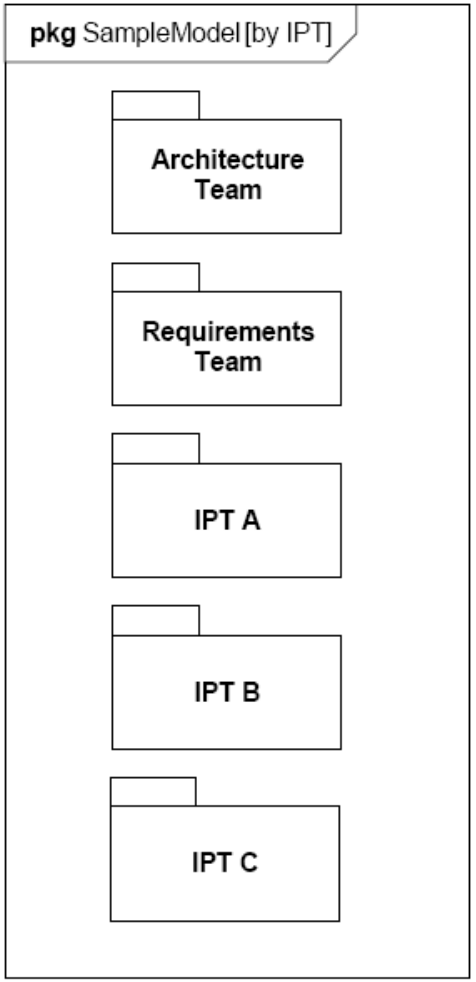
Organizing the Model



By Diagram Type



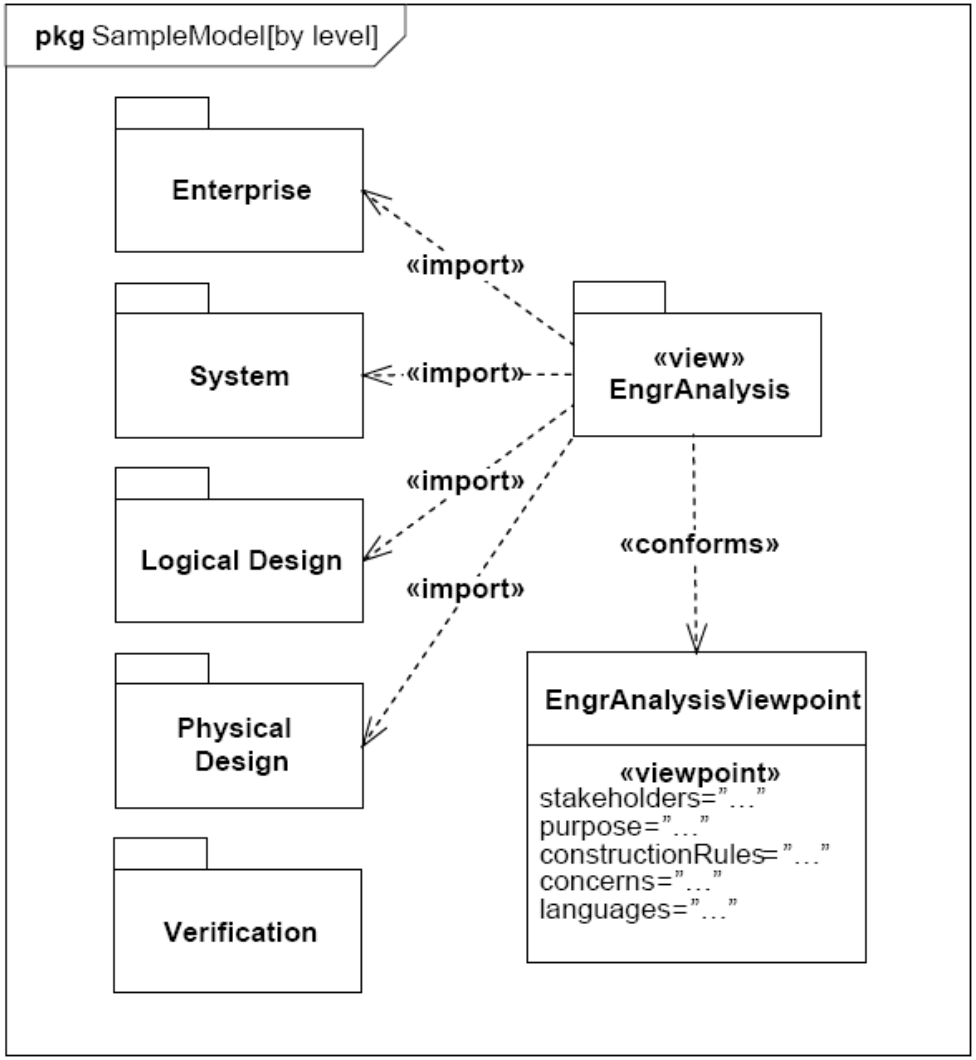
By Hierarchy



By IPT

Package Diagram

Views



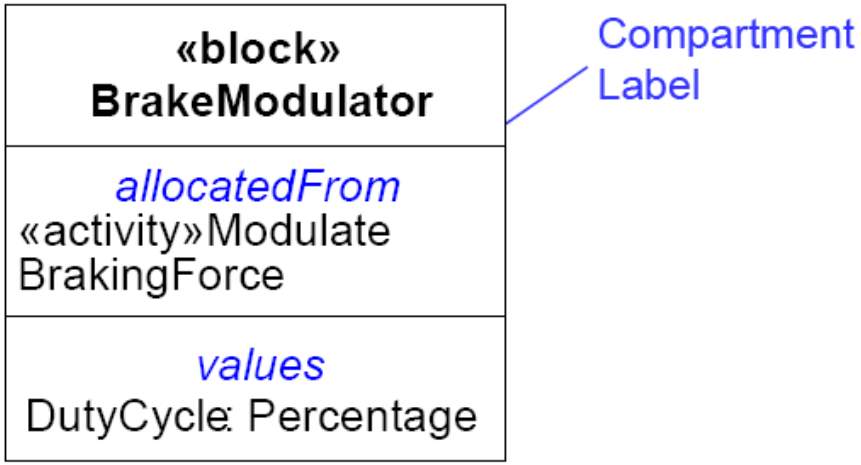
- Viewpoint represents the stakeholder perspective
- View conforms to a particular viewpoint
  - Imports model elements from multiple packages
  - Can represent a model query based on query criteria
- View and Viewpoint consistent with IEEE 1471 definitions

SysML Diagram Overview

- Package Diagrams [pkg] represent the organization of a model in terms of packages that contain model elements (package, block, activity, ...)
- **Block Definition Diagrams** [bdd] **represent structural elements called *blocks* and their composition and classification**
- **Internal Block Diagrams** [ibd] **represent interconnections and interfaces between the parts of a block**

Blocks are Basic Structural Elements

- Provides a unifying concept to describe the structure of an element or system
  - System
  - Hardware
  - Software
  - Data
  - Procedure
  - Facility
  - Person



- Multiple standard compartments can describe the block characteristics
  - Properties (parts, references, values, ports)
  - Operations
  - Constraints
  - Allocations from/to other model elements (e.g. activities)
  - Requirements the block satisfies
  - User defined compartments

Property Types

- Property is a structural feature of a block
  - **Part property** aka. part (typed by a block)
    - Usage of a block in the context of the enclosing (composite) block
    - Example - right-front:wheel
  - **Reference property** (typed by a block)
    - A part that is not owned by the enclosing block (not composition)
    - Example – aggregation of components into logical subsystem
  - **Value property** (typed by value type)
    - A quantifiable property with units, dimensions, and probability distribution
    - Example
      - *Non-distributed value:* tirePressure:psi=30
      - *Distributed value:* «uniform» {min=28,max=32} tirePressure:psi

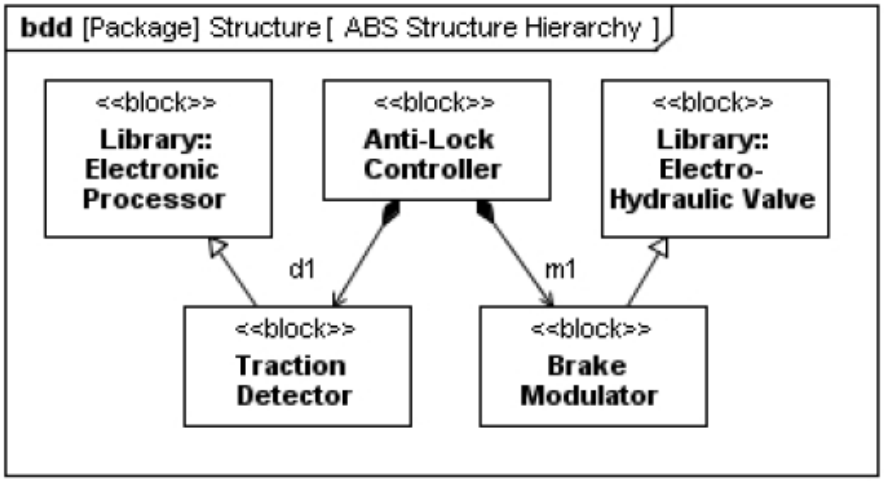
Using Blocks

- Based on UML Class from UML Composite Structure
  - Supports unique features (e.g., flow ports, value properties)
- Block definition diagram describes the relationship among blocks (e.g., composition, association, specialization)
- Internal block diagram describes the internal structure of a block in terms of its properties and connectors
- Behavior can be allocated to blocks

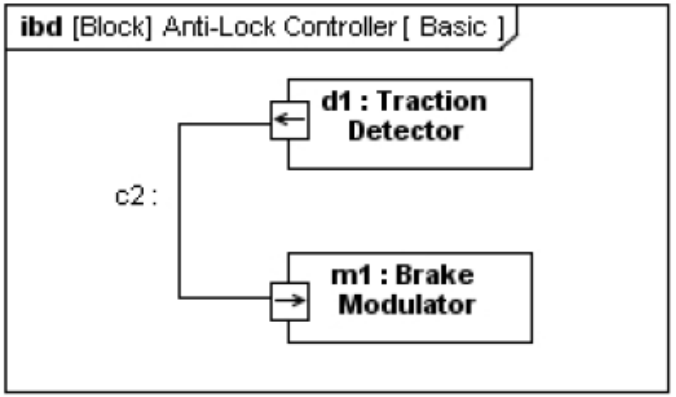
Blocks Used to Specify Hierarchies and Interconnection

Block Definition vs. Usage

Block Definition Diagram



Internal Block Diagram



Definition

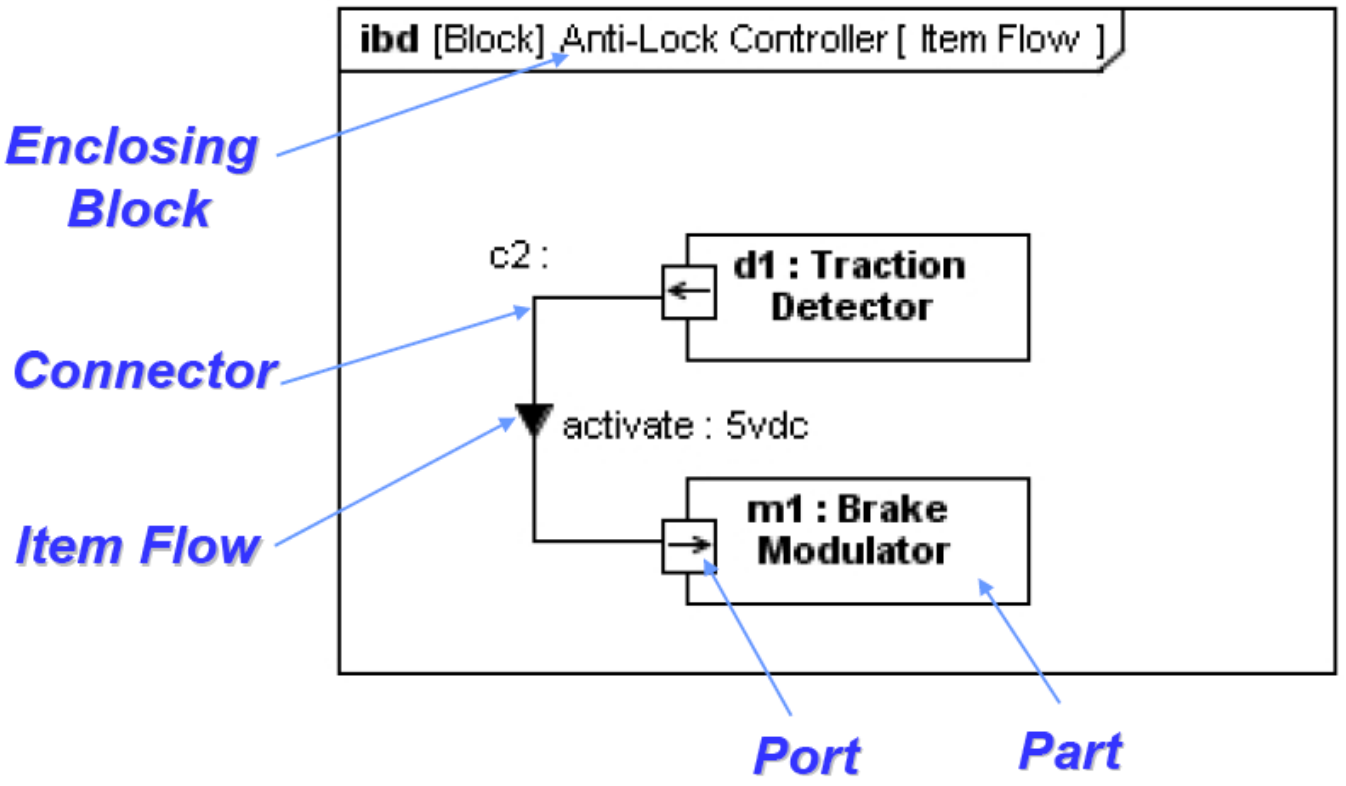
- Block is a definition/type
- Captures properties, etc.
- Reused in multiple contexts

Usage

- Part is the usage in a particular context
- Typed by a block
- Also known as a role

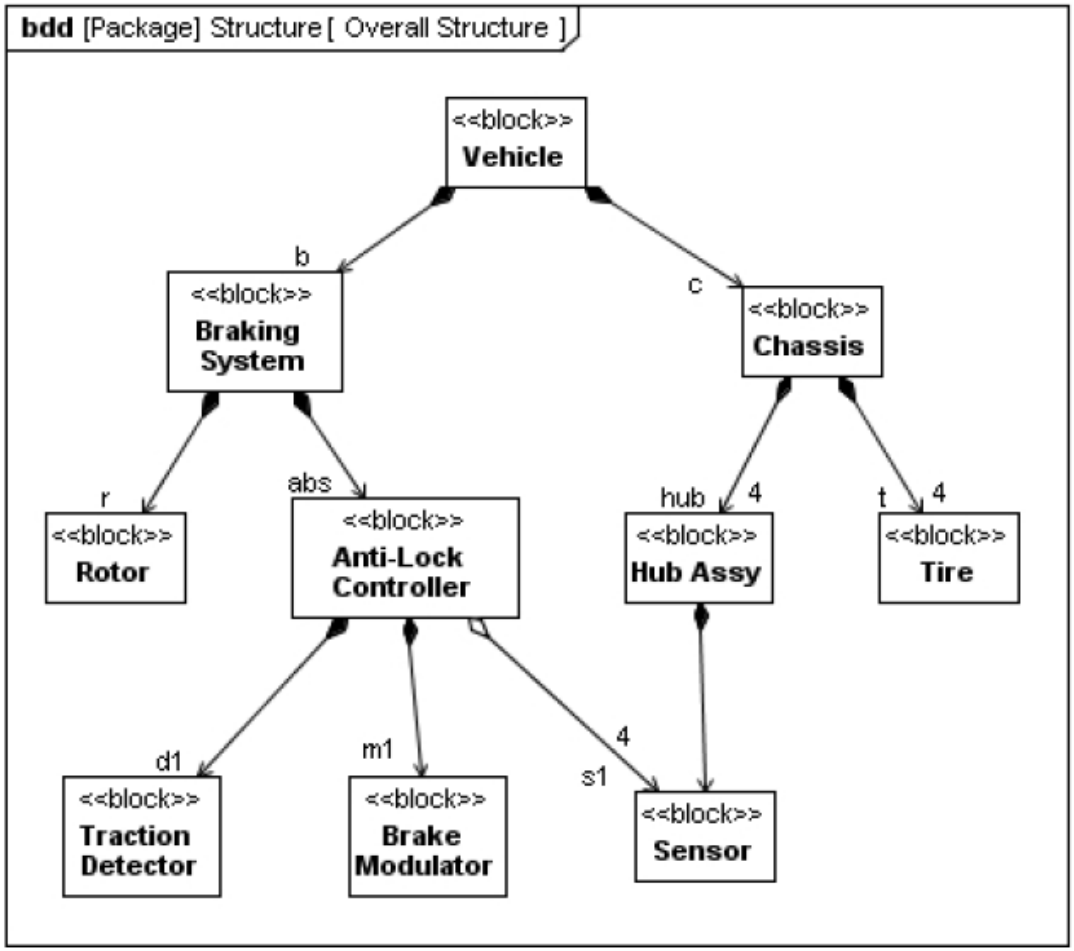
Internal Block Diagrams (ibd)

Blocks, Parts, Ports, Connectors, and Flows

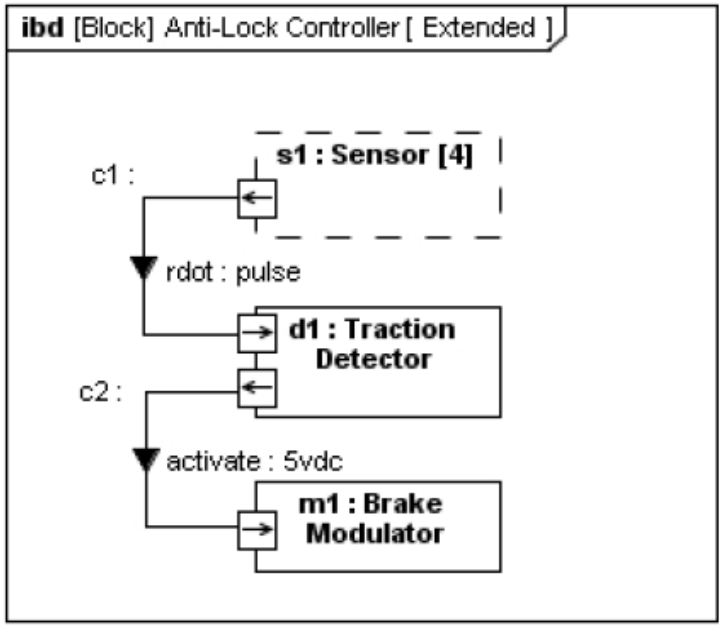


Internal Block Diagram Specifies Interconnection of Parts

Reference Property Explained



- **S1 is a reference part***
- **Shown in dashed outline box**



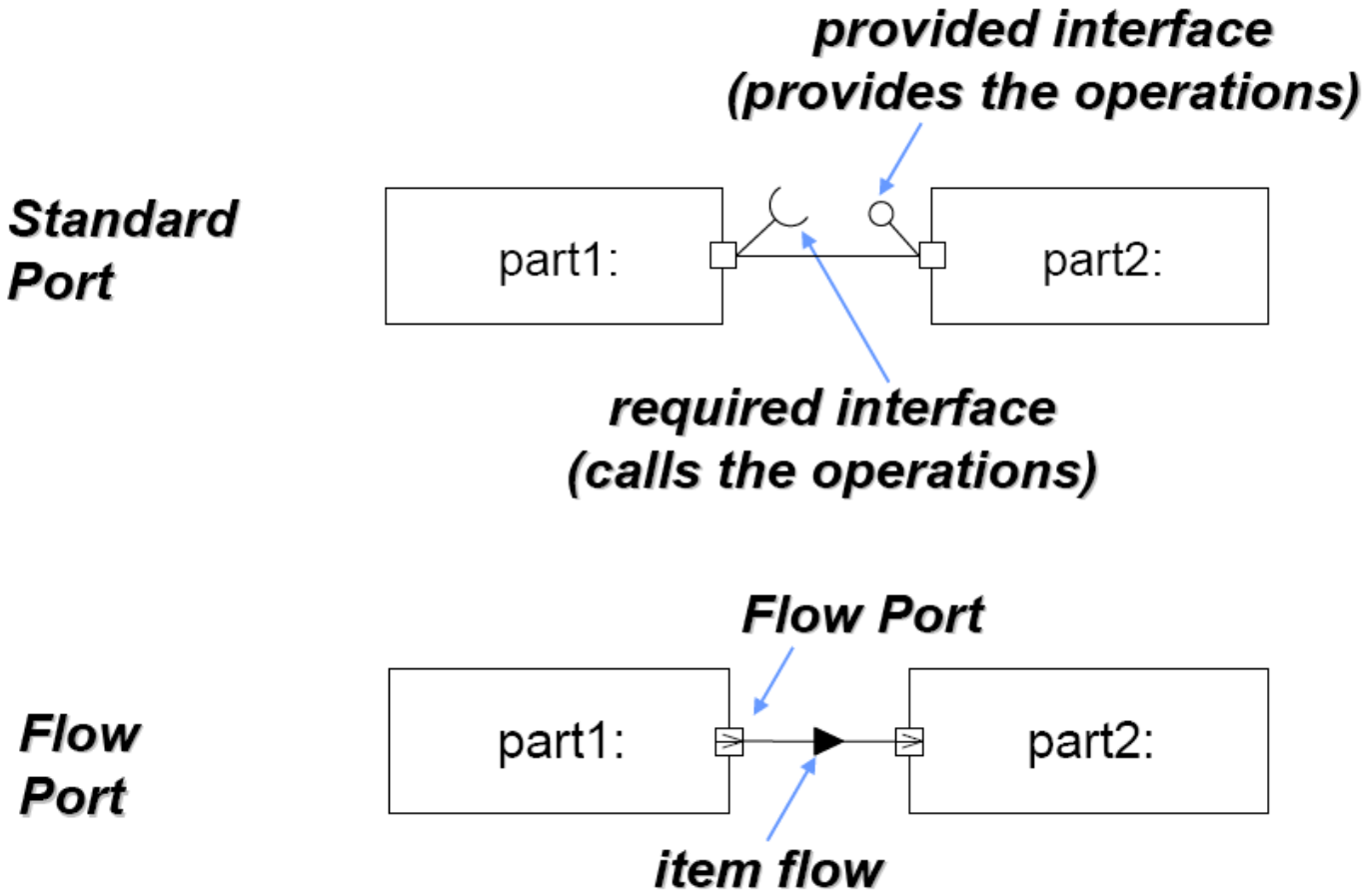
*Actual name is reference property

SysML Ports

- Specifies interaction points on blocks and parts
  - Integrates behavior with structure
  - portName:TypeName
- Kinds of ports
  - Standard (UML) Port
    - Specifies a set of required or provided operations and/or signals
    - Typed by a UML interface
  - Flow Port
    - Specifies what can flow in or out of block/part
    - Typed by a block, value type, or flow specification
    - Atomic, non-atomic, and conjugate variations

Standard Port and Flow Port Support Different Interface Concepts

Port Notation



SysML Diagram Overview

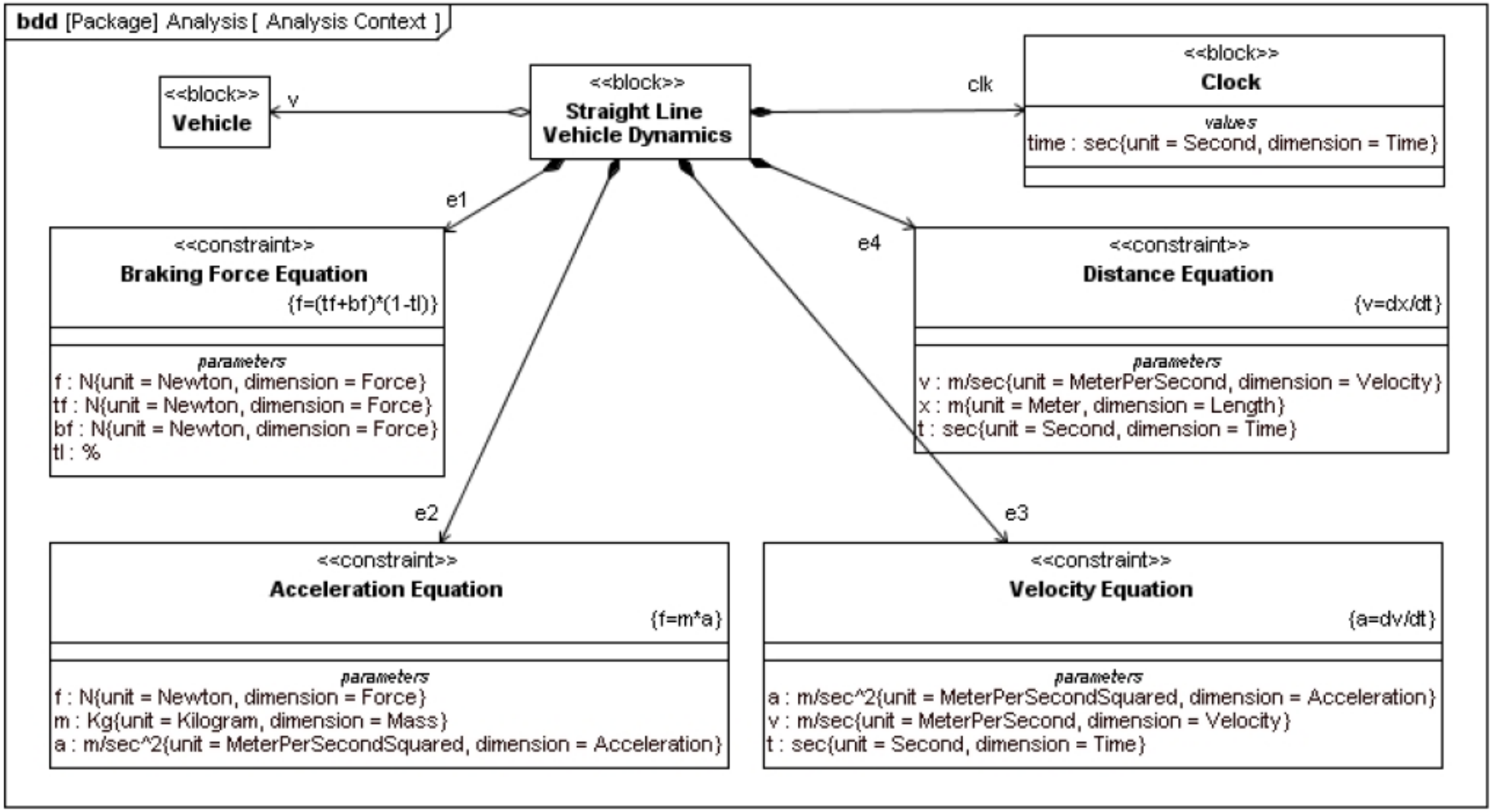
- Package Diagrams [pkg] represent the organization of a model in terms of packages that contain model elements (package, block, activity, ...)
- Block Definition Diagrams [bdd] represent structural elements called *blocks* and their composition and classification
- Internal Block Diagrams [ibd] represent interconnections and interfaces between the parts of a block
- **Parametric Diagrams** [par] **represent constraints on property values to to support engineering analyses**

Parametrics

- Used to express constraints (equations) between value properties
  - Provides support for engineering analysis (e.g., performance, reliability)
  - Facilitates identification of critical performance properties
- Constraint block captures equations
  - Expression language can be formal (e.g., MathML, OCL) or informal
  - Computational engine is provided by applicable analysis tool and not by SysML
- Parametric diagram represents the usage of the constraints in an analysis context
  - Binding of constraint parameters to value properties of blocks (e.g., vehicle mass bound to parameter 'm' in $F = m \times a$)

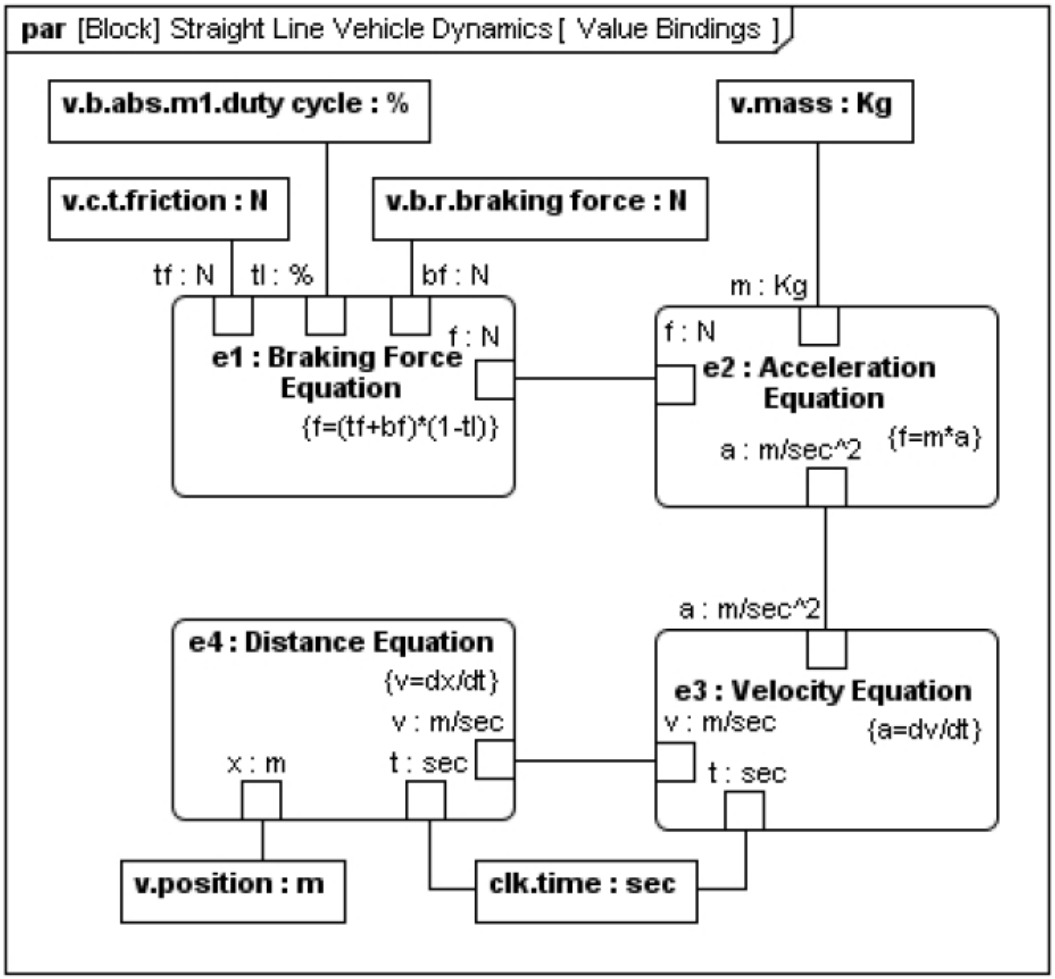
Parametrics Enable Integration of Engineering Analysis with Design Models

Defining Vehicle Dynamics



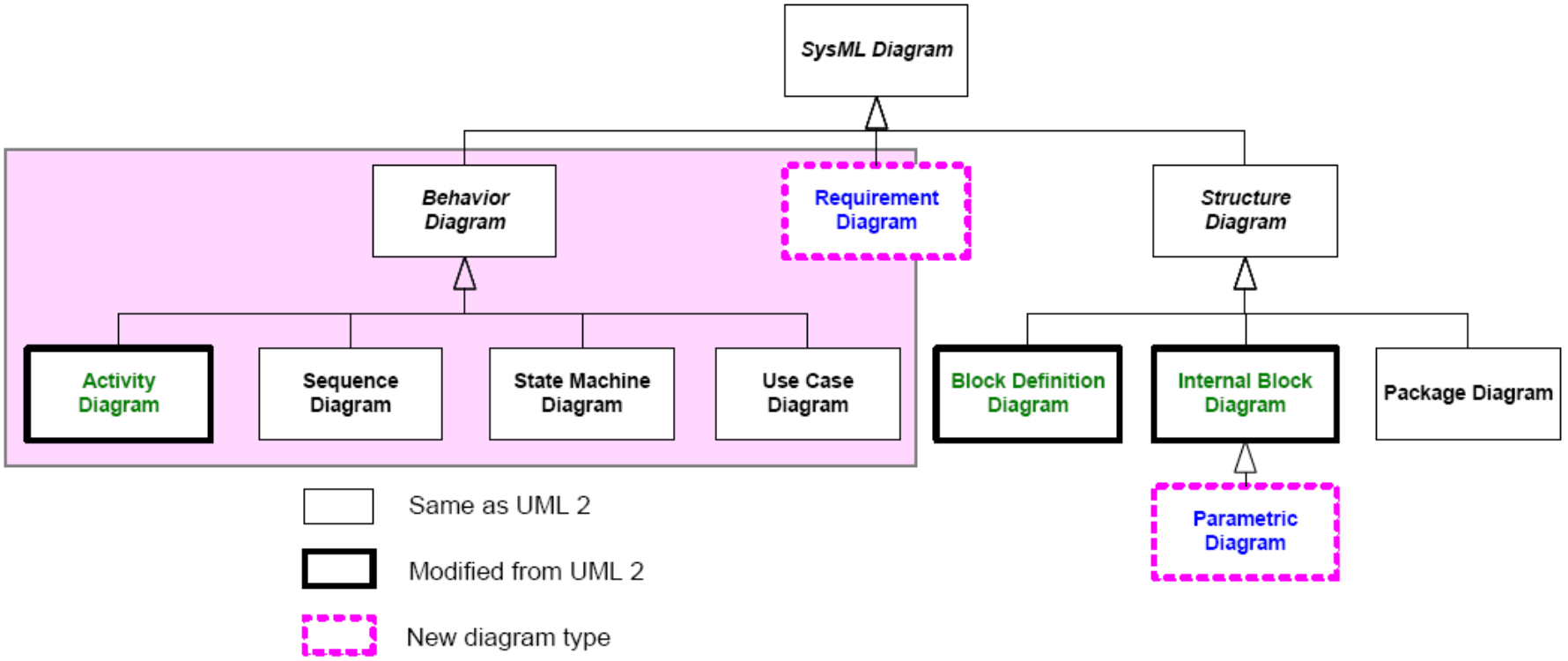
Defining Reusable Equations for Parametrics

Vehicle Dynamics Analysis



Using the Equations in a Parametric Diagram to Constrain Value Properties

Behavior Diagrams



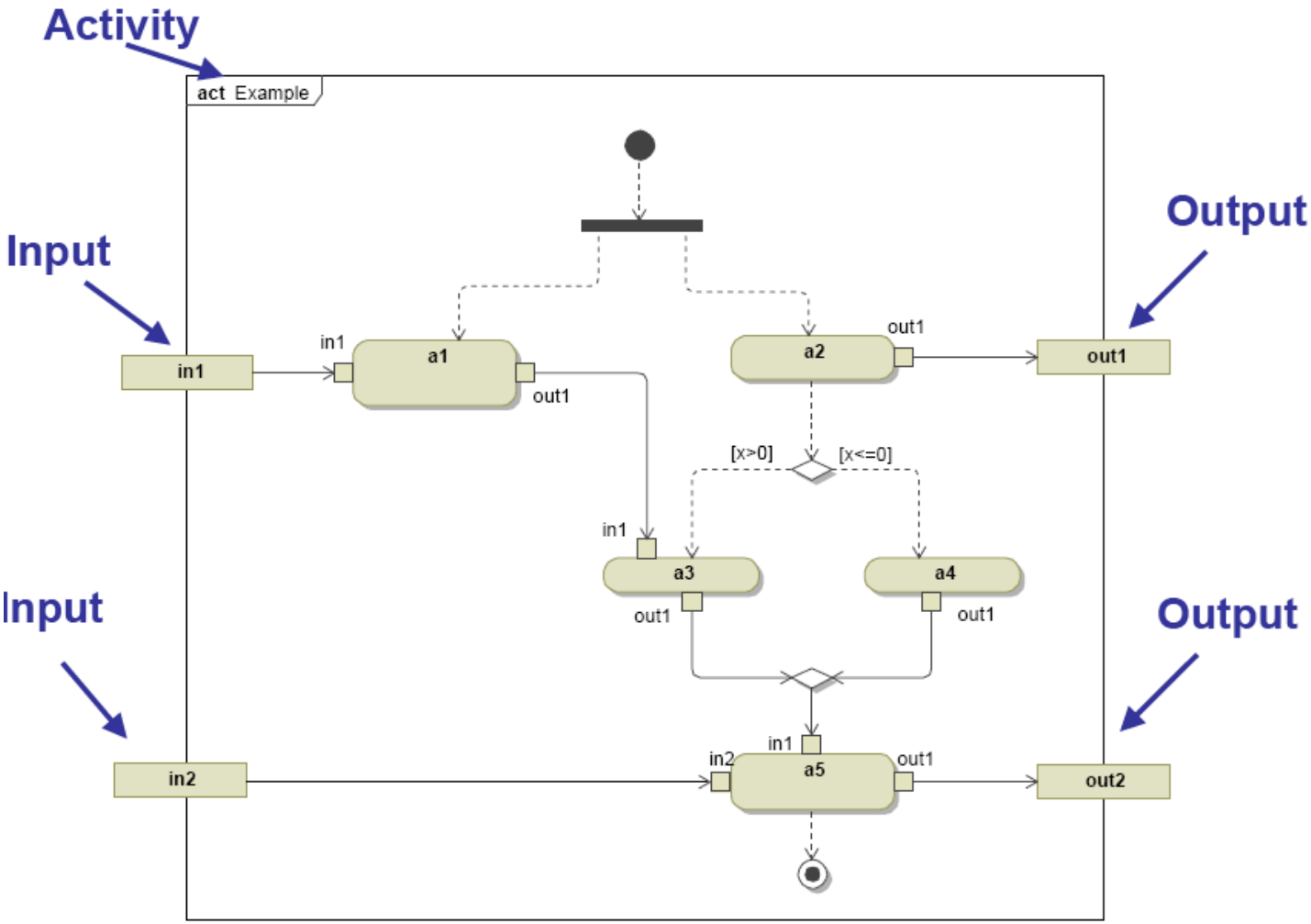
SysML Diagram Overview

- Package Diagrams [pkg] represent the organization of a model in terms of packages that contain model elements (package, block, activity, ...)
- Block Definition Diagrams [bdd] represent structural elements called *blocks* and their composition and classification
- Internal Block Diagrams [ibd] represent interconnections and interfaces between the parts of a block
- Parametric Diagrams [par] represent constraints on property values to to support engineering analyses
- **Activity Diagrams** [act] **represent behavior in terms of the *ordering of actions*, based on the availability of inputs, outputs, and control, and how the actions transform the inputs to outputs**

Activities

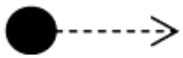
- Activity specifies transformation of inputs to outputs through a controlled sequence of actions
- Secondary constructs show responsibilities for the activities using activity partitions (i.e., swim lanes)

Activity Diagram

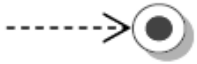


Activity Diagram Specifies Controlled Sequence of Actions

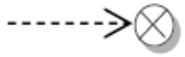
Routing Flows



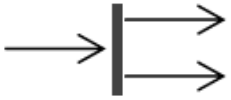
Initial Node – On execution of parent control token placed on outgoing control flows



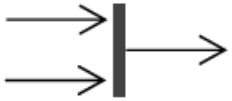
Activity Final Node – Receipt of a control token terminates parent



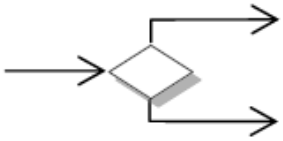
Flow Final Node – Sink for control tokens



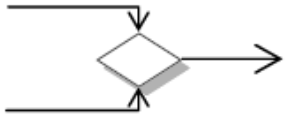
Fork Node – Duplicates input (control or object) tokens from its input flow onto all outgoing flows



Join Node – Waits for an input (control or object) token on all input flows and then places them all on the outgoing flow



Decision Node – Waits for an input (control or object) token on its input flow and places it on one outgoing flow based on guards

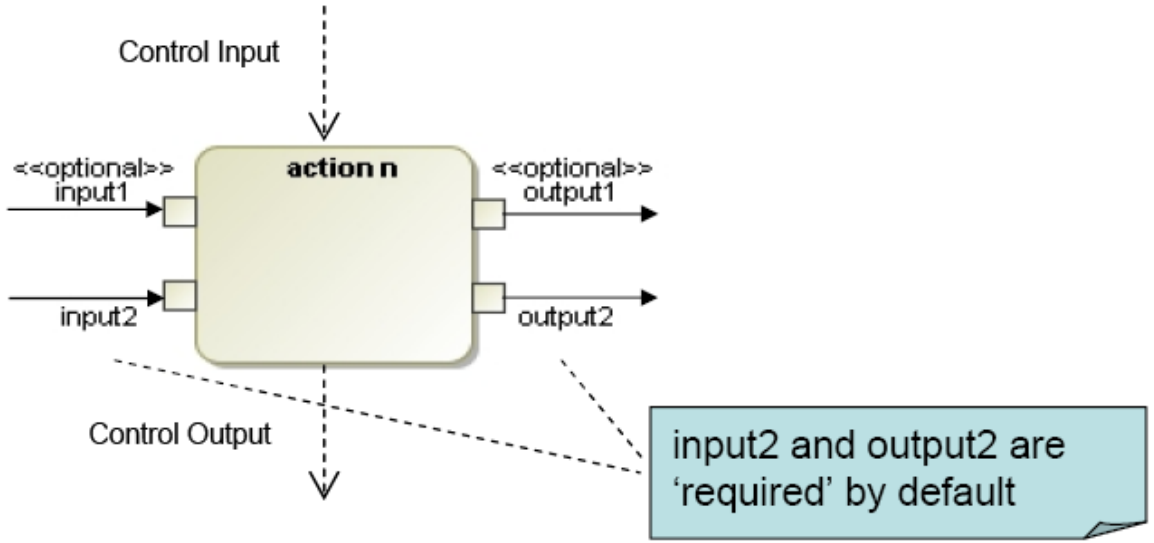


Merge Node – Waits for an input (control or object) token on any input flows and then places it on the outgoing flow

Guard expressions can be applied on all flows

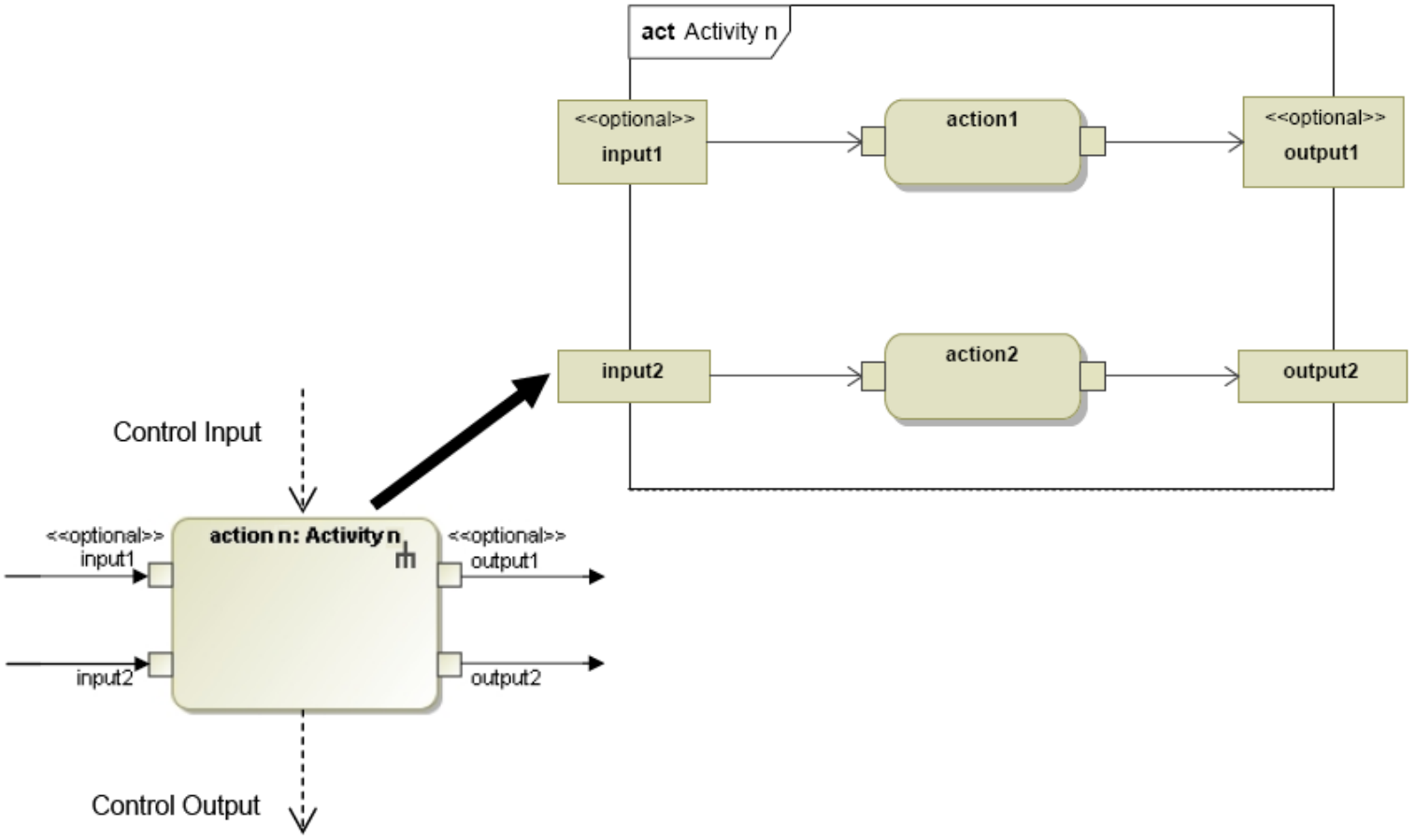
Actions Process Flow of Control & Data

- **Two types of flow**
  - **Object / Data**
  - **Control**
- **Unit of flow is called a "token"**
  (consumed & produced by actions)



Actions Execution Begins When Tokens Are Available on "all" Control Inputs and Required Inputs

An Action Can Invoke Another Activity



Activity is Invoked When an Action Begins to Execute

Semantics for Activity Invocation

A call behavior action can have

- 0..* control inputs & outputs
- 0 ..* optional item inputs & outputs
- 0..* required item inputs & outputs
- 0..* streaming (and continuous) item inputs & outputs

Note: The summary is an approximation of the semantics.
The detailed semantics are specified in the UML and SysML specification.

Starting an action:

- An action starts when a token is placed on all of its control inputs and all of its required inputs (must meet minimum multiplicity of its input pins) and the previous invoked activity has completed
- An action invokes an activity when it starts, and passes the tokens from its input pins to the input parameter nodes of the invoked activity

During an execution:

- An action continues to accept streaming inputs and produce streaming outputs

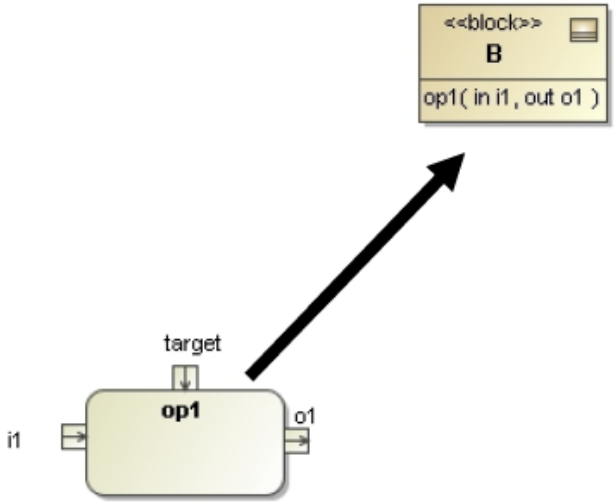
Terminating an action:

- An action terminates when its invoked activity reaches an activity final, or when the action receives a control disable, or as a side affect of other behaviors of the parent activity
- The tokens on the output parameter nodes of the activity are placed on the output pins of the action and a control token is placed on each of the control outputs of the action

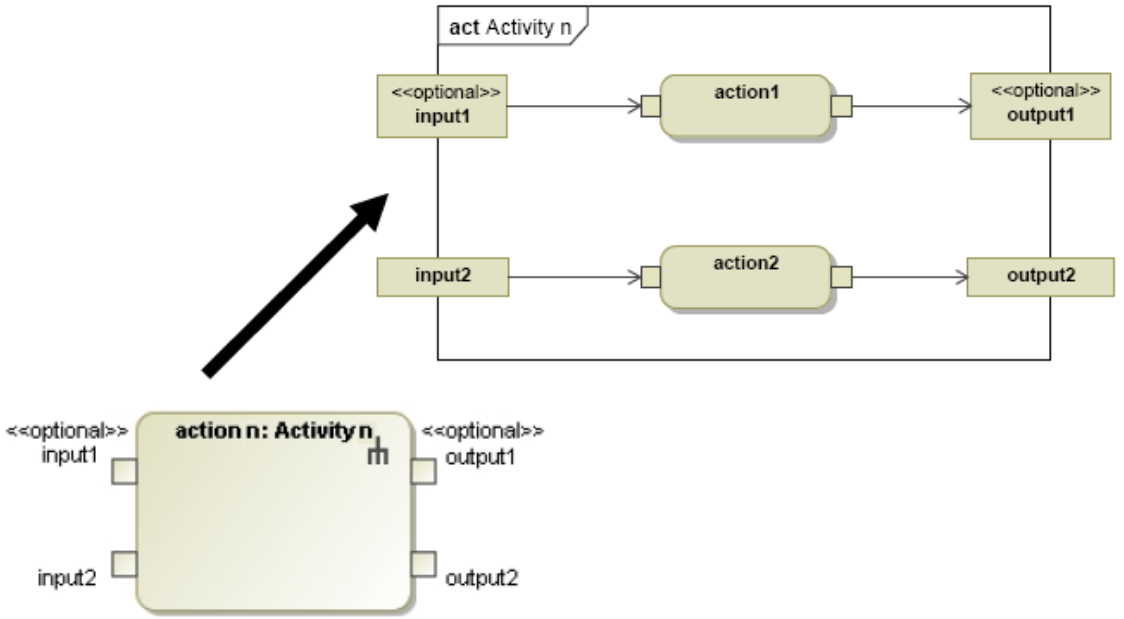
Following action termination:

- The tokens on the output pins and control outputs of the action are moved to the input pins of the next actions when they are ready to start per above
- The action can restart and invoke the activity again when the starting conditions are satisfied per above

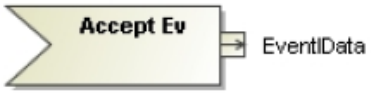
Common Actions



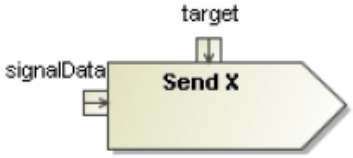
Call Operation Action
(can call leaf level function)



Call Behavior Action

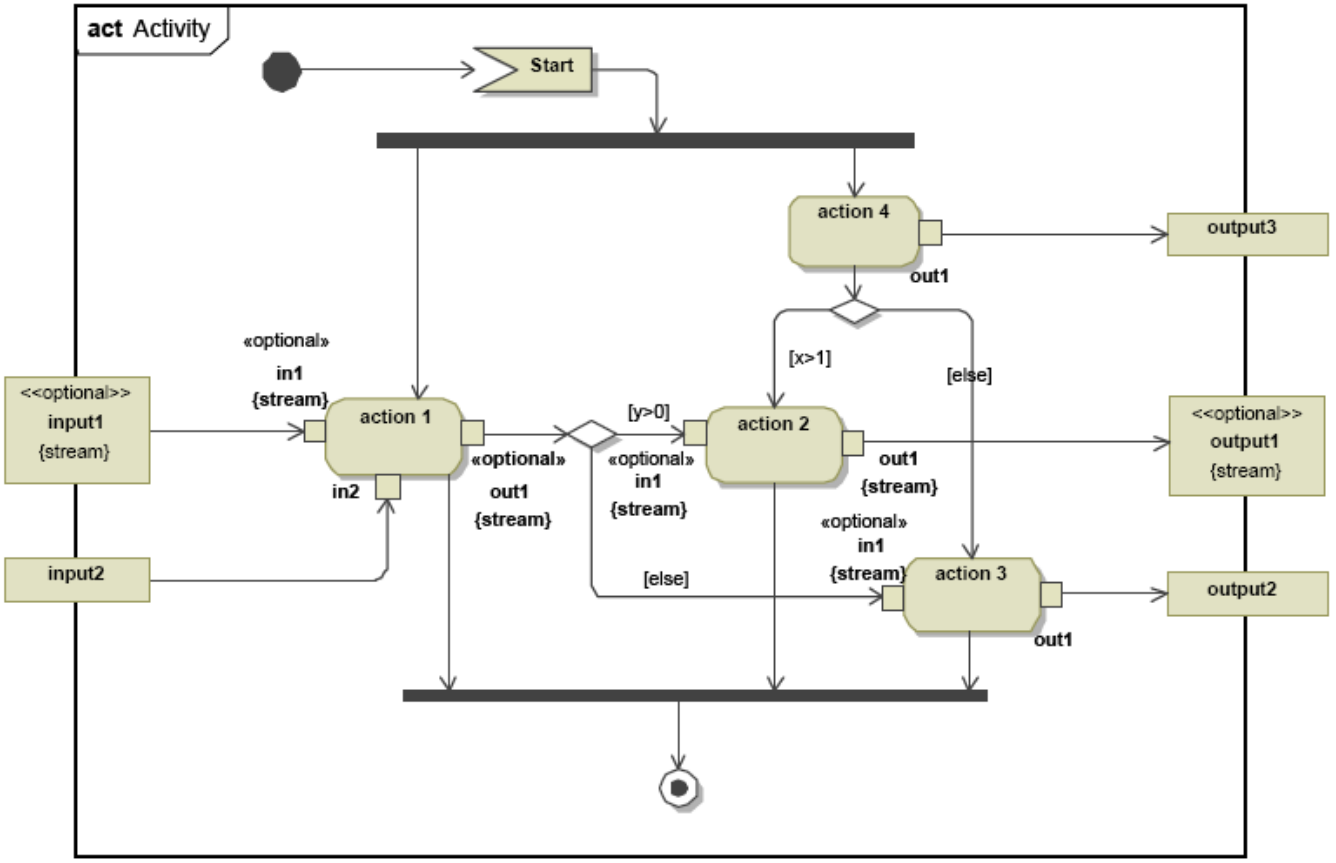


Accept Event Action
(Event Data Pin often elided)



Send Signal Action
(Pins often elided)

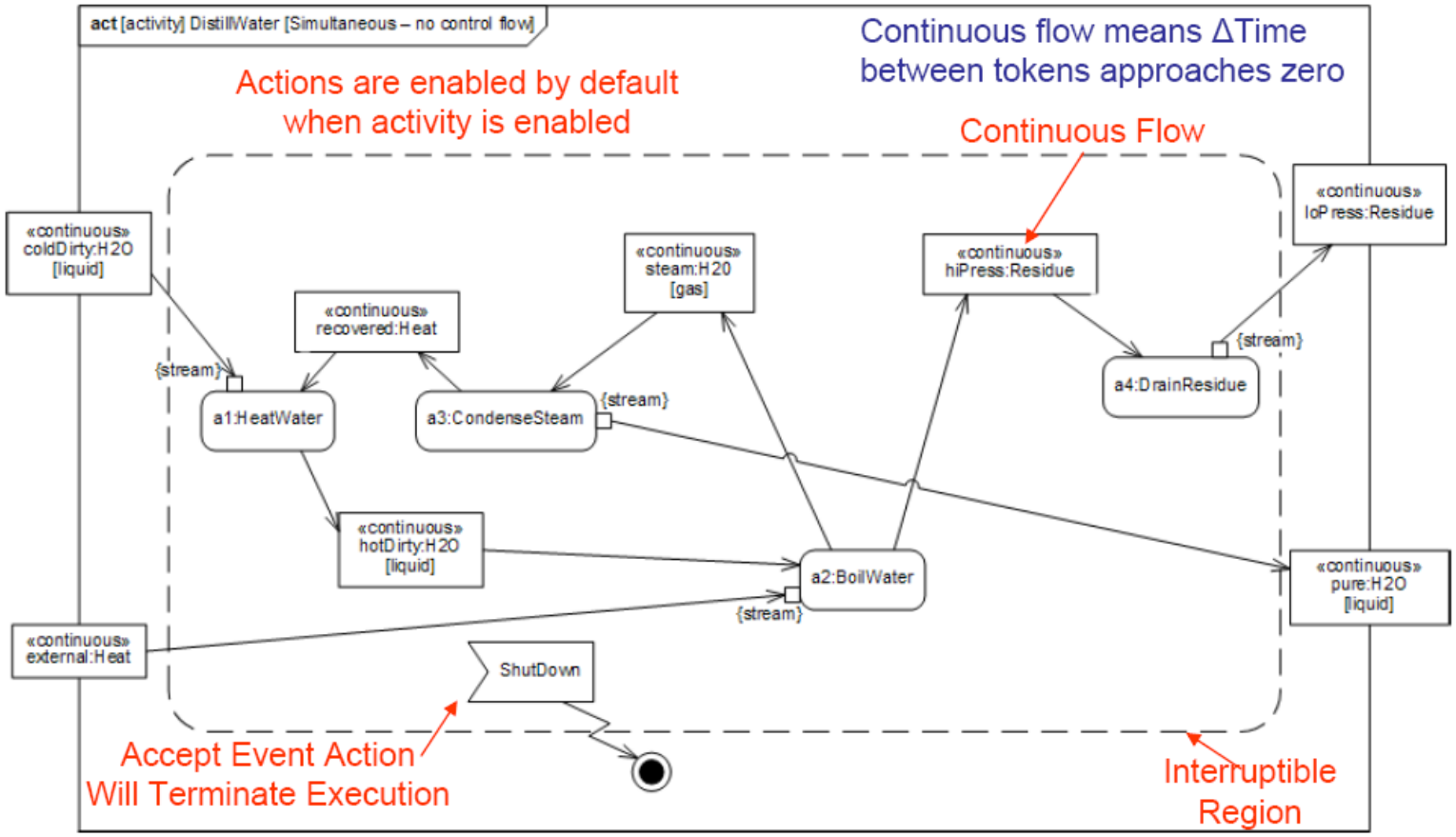
Activity Diagram Example with Streaming Inputs and Outputs



Streaming Inputs and Outputs Continue to Be Consumed and Produced While the Action is Executing

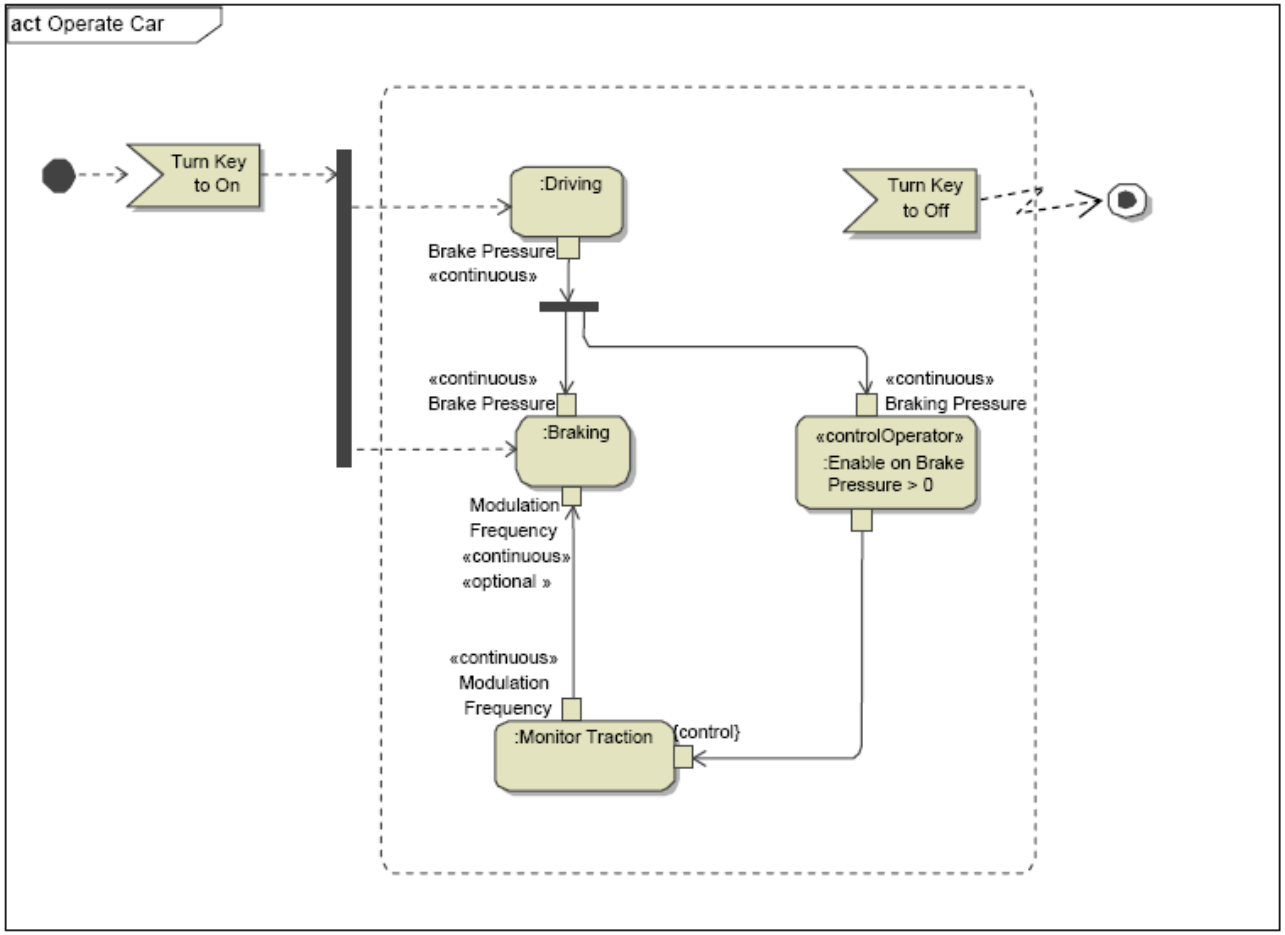
Distilling Water Activity Diagram

Continuous Flow Modeling – Example Continued Later



Continuous Flow Is Representative of Many Physical Processes

Example: Operate Car

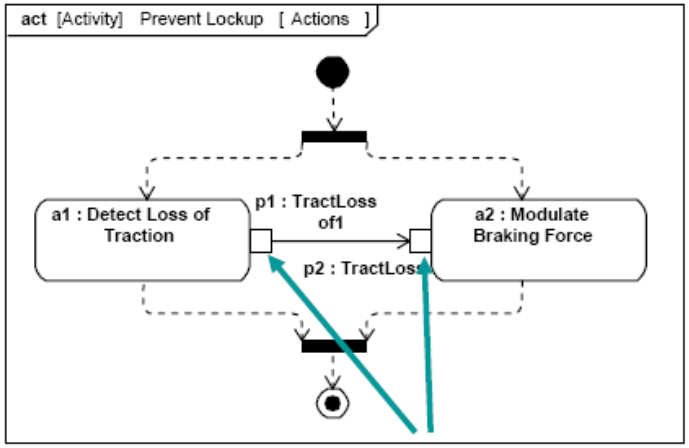


Enabling and Disabling Actions With Control Operators

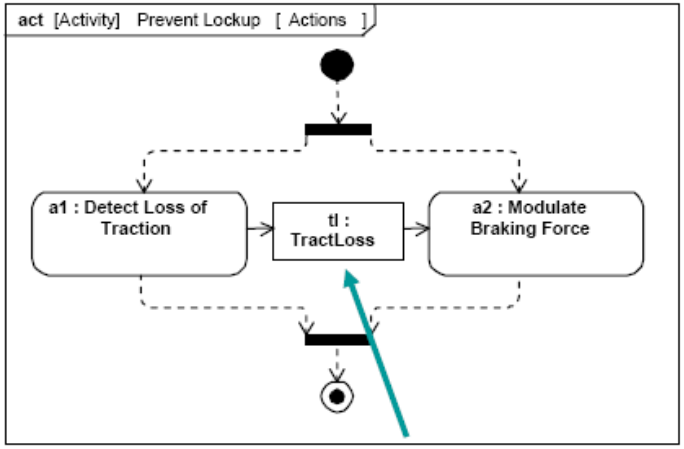
Activity Diagrams

(Pins vs. Object Node Notation)

- Pins are kinds of Object Nodes
  - Used to specify inputs and outputs of actions
  - Typed by a block or value type
  - Object flows connect object nodes
- Object flows between pins have two diagrammatic forms
  - Pins shown with object flow between them
  - Pins elided and object node shown with flow arrows in and out



Pins

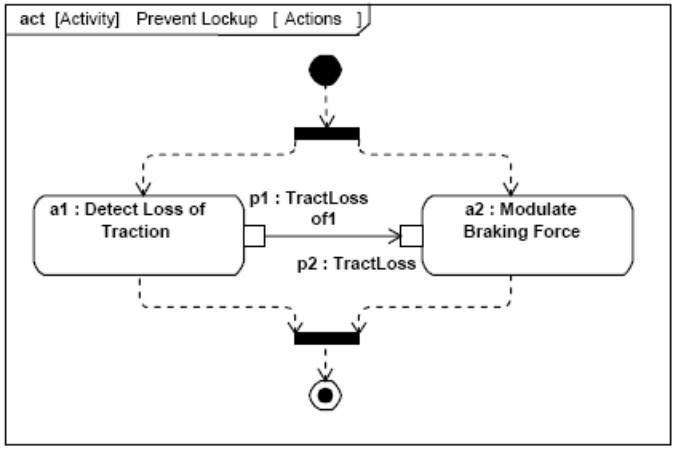


ObjectNode

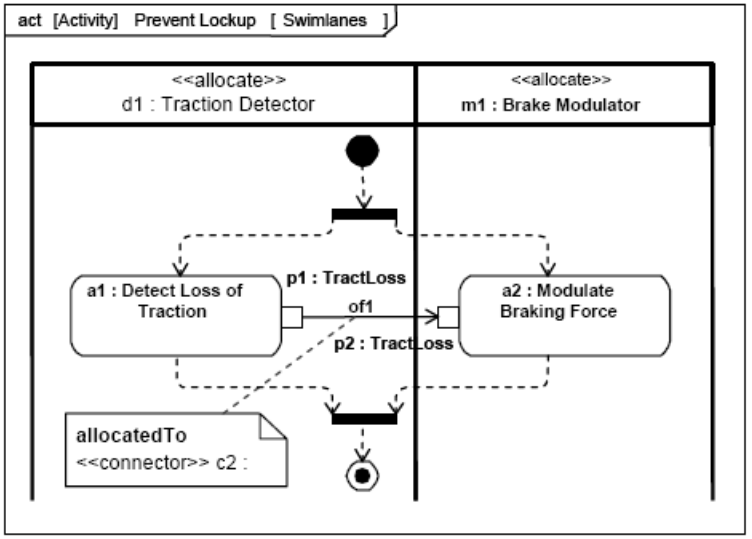
Pins must have same characteristics (name, type etc.)

Explicit Allocation of Behavior to Structure Using Swim Lanes

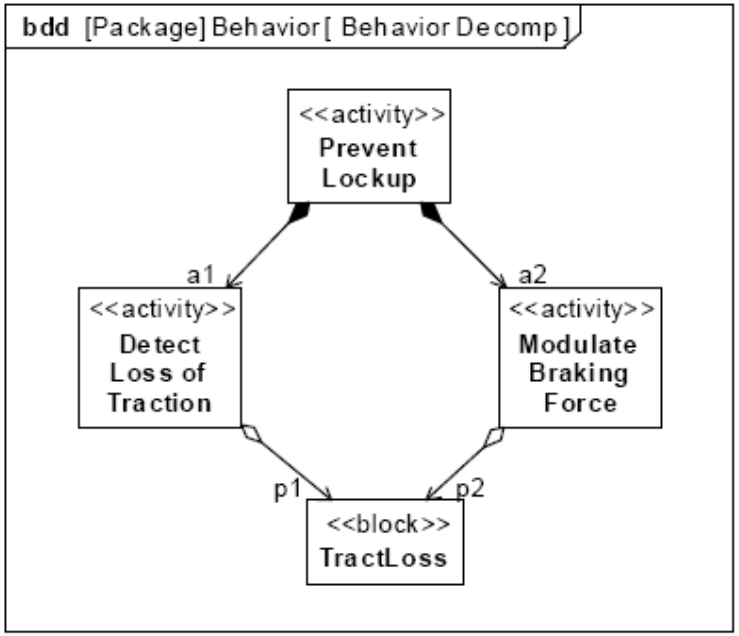
Activity Diagram
(without Swimlanes)

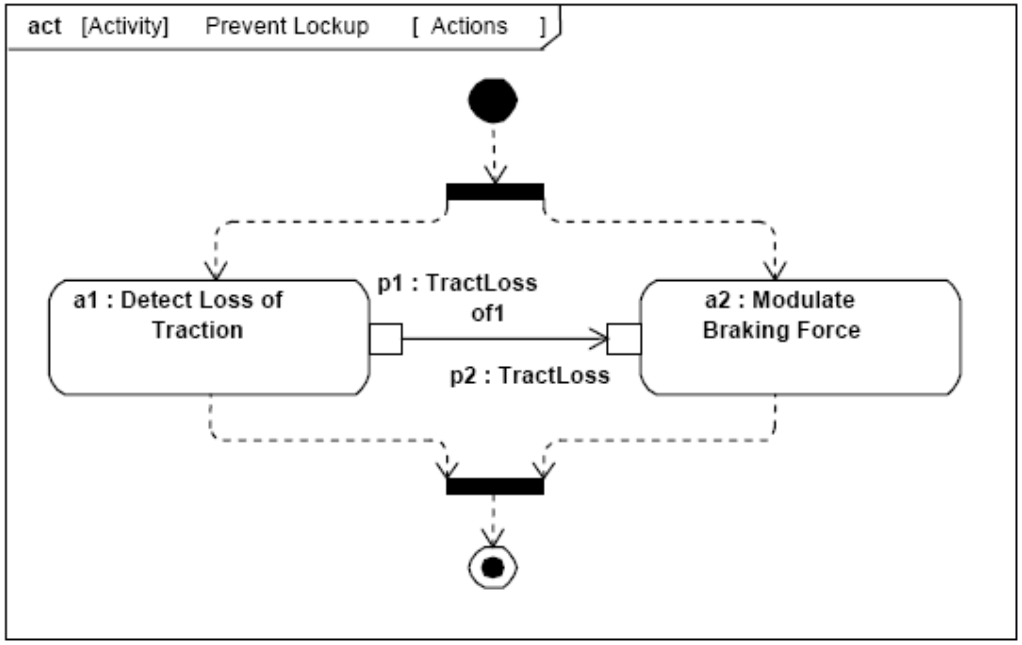


Activity Diagram
(with Swimlanes)



Activity Decomposition





Definition

Use

SysML Diagram Overview

- Package Diagrams [pkg] represent the organization of a model in terms of packages that contain model elements (package, block, activity, ...)
- Block Definition Diagrams [bdd] represent structural elements called *blocks* and their composition and classification
- Internal Block Diagrams [ibd] represent interconnections and interfaces between the parts of a block
- Parametric Diagrams [par] represent constraints on property values to to support engineering analyses
- Activity Diagrams [act] represent behavior in terms of the *ordering of actions*, based on the availability of inputs, outputs, and control, and how the actions transform the inputs to outputs
- **Sequence Diagrams** [sd] **represent behavior in terms of a sequence of messages exchanged between parts**

Interactions

- Sequence diagrams provide representations of message based behavior
  - represent flow of control
  - describe interactions between parts
- Sequence diagrams provide mechanisms for representing complex scenarios
  - reference sequences
  - control logic
  - lifeline decomposition
- SysML does not include timing, interaction overview, and communications diagram

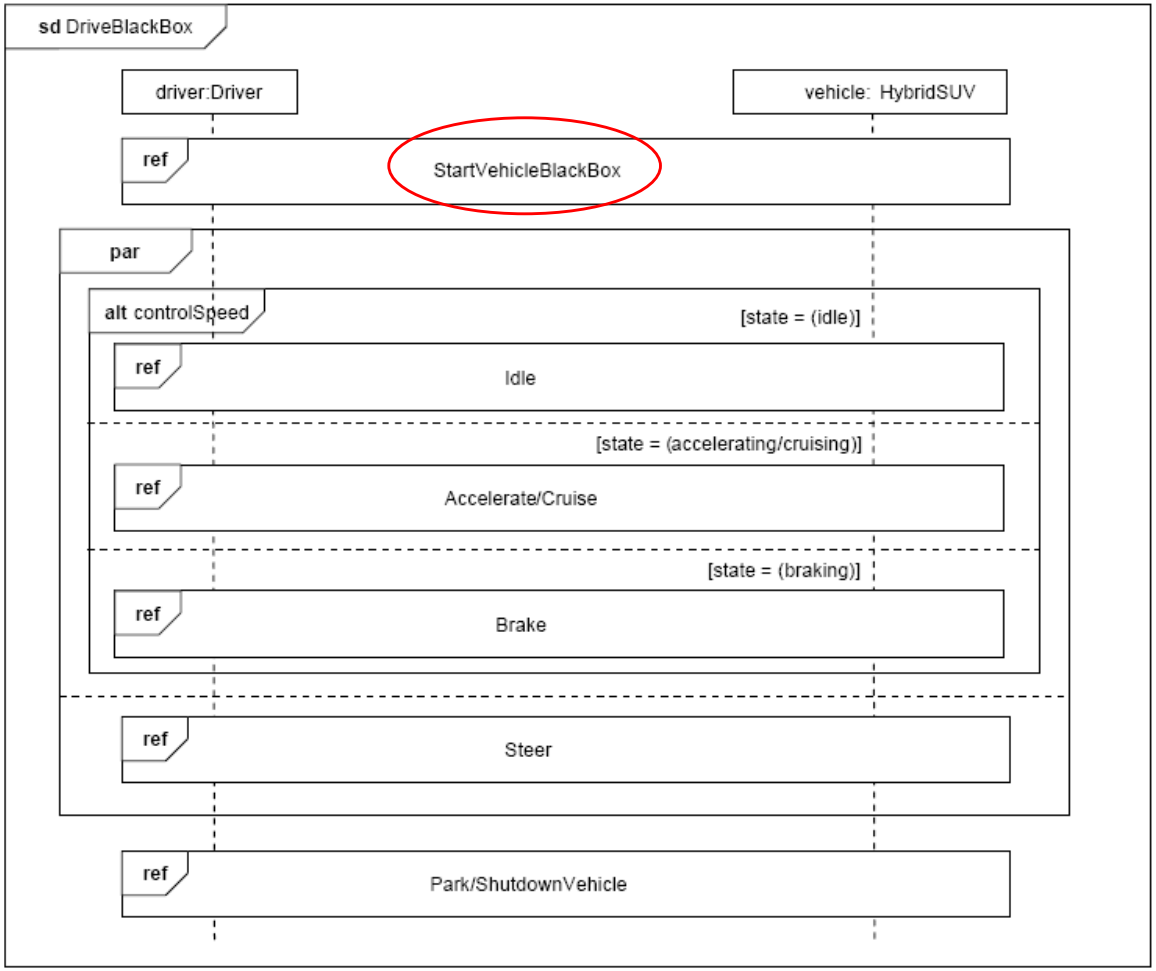
Primary Interaction Operators

- **ref** name
  - reference to a sequence diagram fragment defined elsewhere
- **opt** [condition]
  - has 1 part that may be executed based on a condition/state value
- **alt**
  - has 2 or more parts, but only one executes based on a condition/state
  - an operand fragment labeled [else] is executed if no other condition is true
- **par**
  - has 2 or more parts that execute concurrently
    - Concurrence indicates does not require simultaneous, just that the order is undetermined. If there is only one processor the behavior could be (A then B), (B then A), or (A and B interleaving) …
- **loop** min..max [escape]
  - Has a minimum # of executions, and optional maximum # of executions, and optional escape condition
- **break** [condition]
  - Has an optional guard. If true, the contents (if any) are executed, and the remainder of the enclosing operator is not executed

Other Interaction Operators

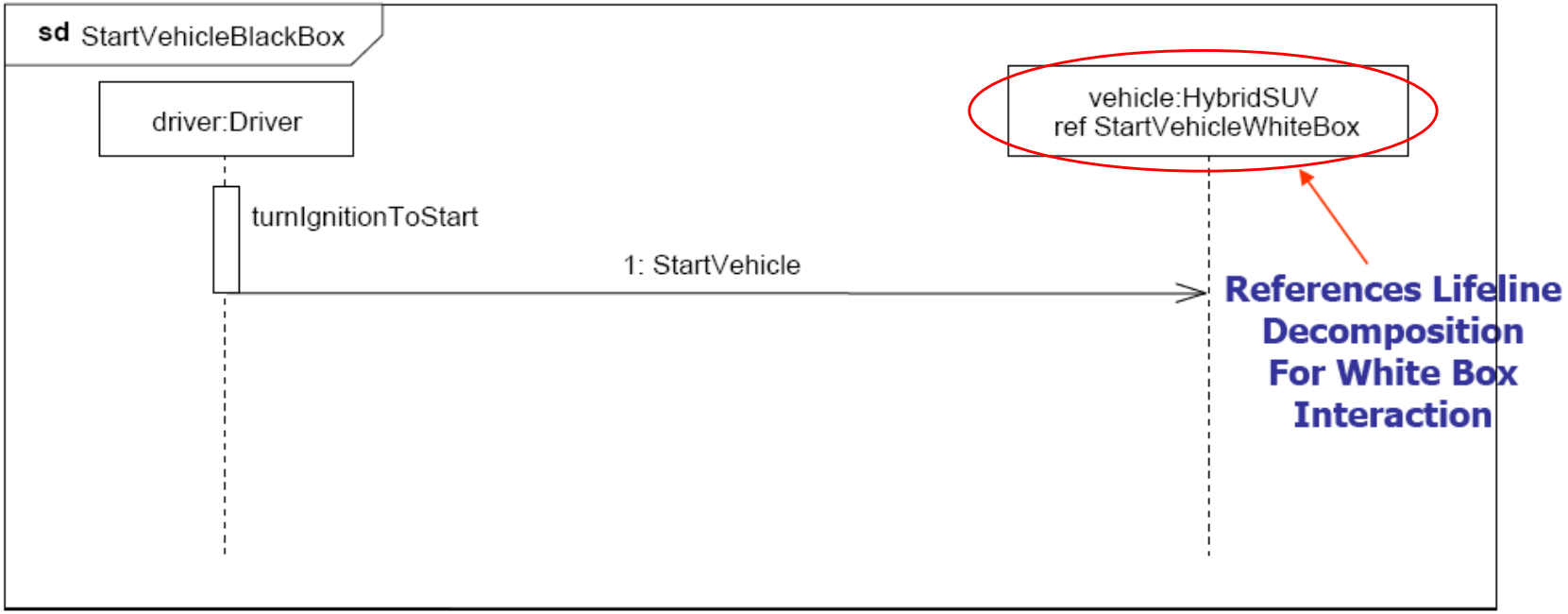
- **critical**
  - The sequence diagram fragment is a critical region. It is treated as atomic – no interleaving with parallel regions
- **neg**
  - The sequence diagram fragment is forbidden. Either it is impossible to occur, or it is the intent of the requirements to prevent it from occurring
- **assert**
  - The sequence diagram fragment is the only one possible (or legal)
- **seq** (weak, the default)
  strict
  - Strict: The message exchange occurs in the order described
  - Weak: Each lifeline may see different orders for the exchange (subject to causality)
- **consider** (list of messages)
  ignore (list of messages)
  - Consider: List the messages that are relevant in this sequence fragment
  - Ignored: List the messages that may arrive, but are not interesting here

Black Box Interaction (Drive)



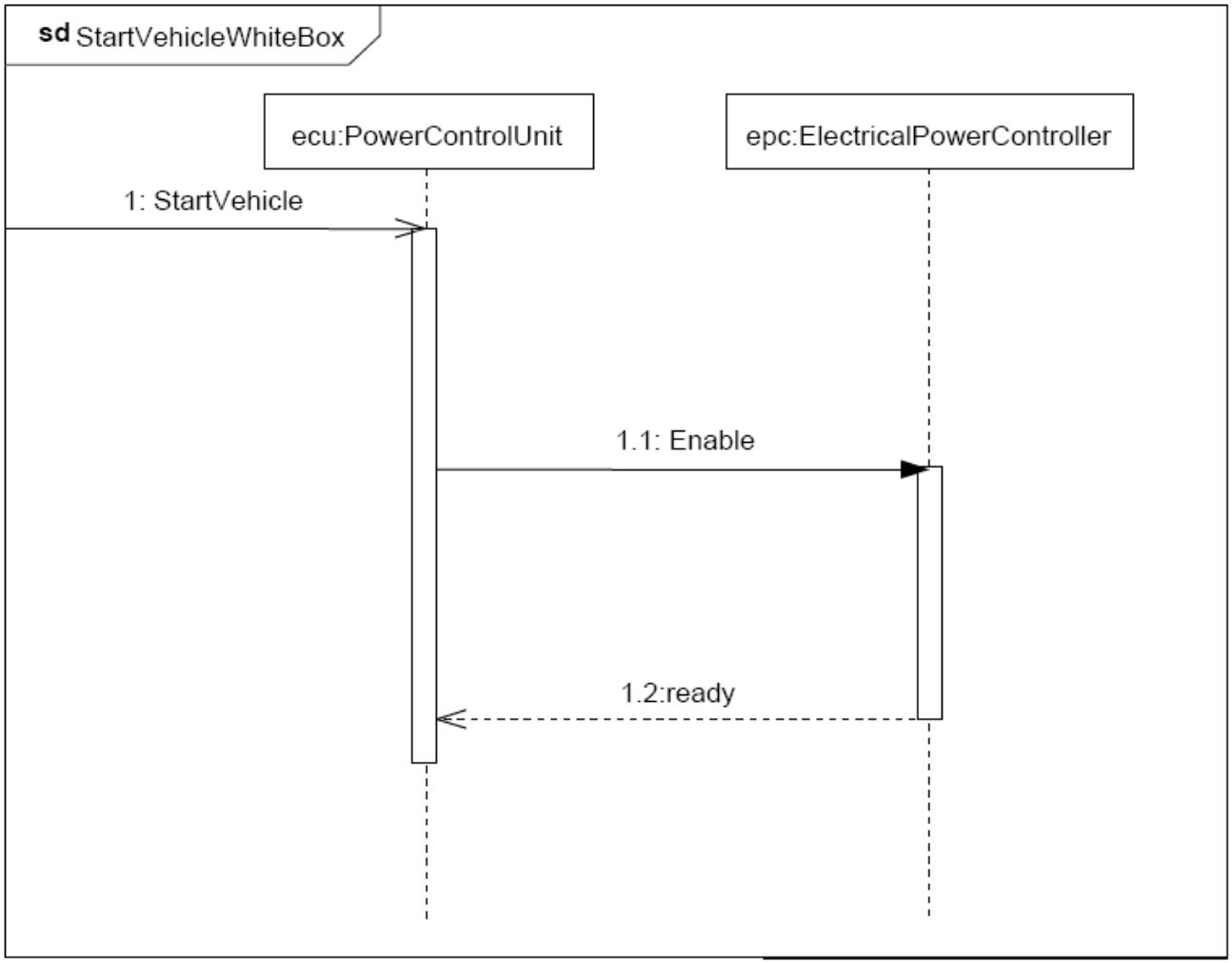
UML 2 Sequence Diagram Scales by Supporting Control Logic and Reference Sequences

Black Box Interaction (Start Vehicle)



Simple Black Box Interaction

White Box Interaction (Start Vehicle)



Decomposition of Black Box Into White Box Interaction

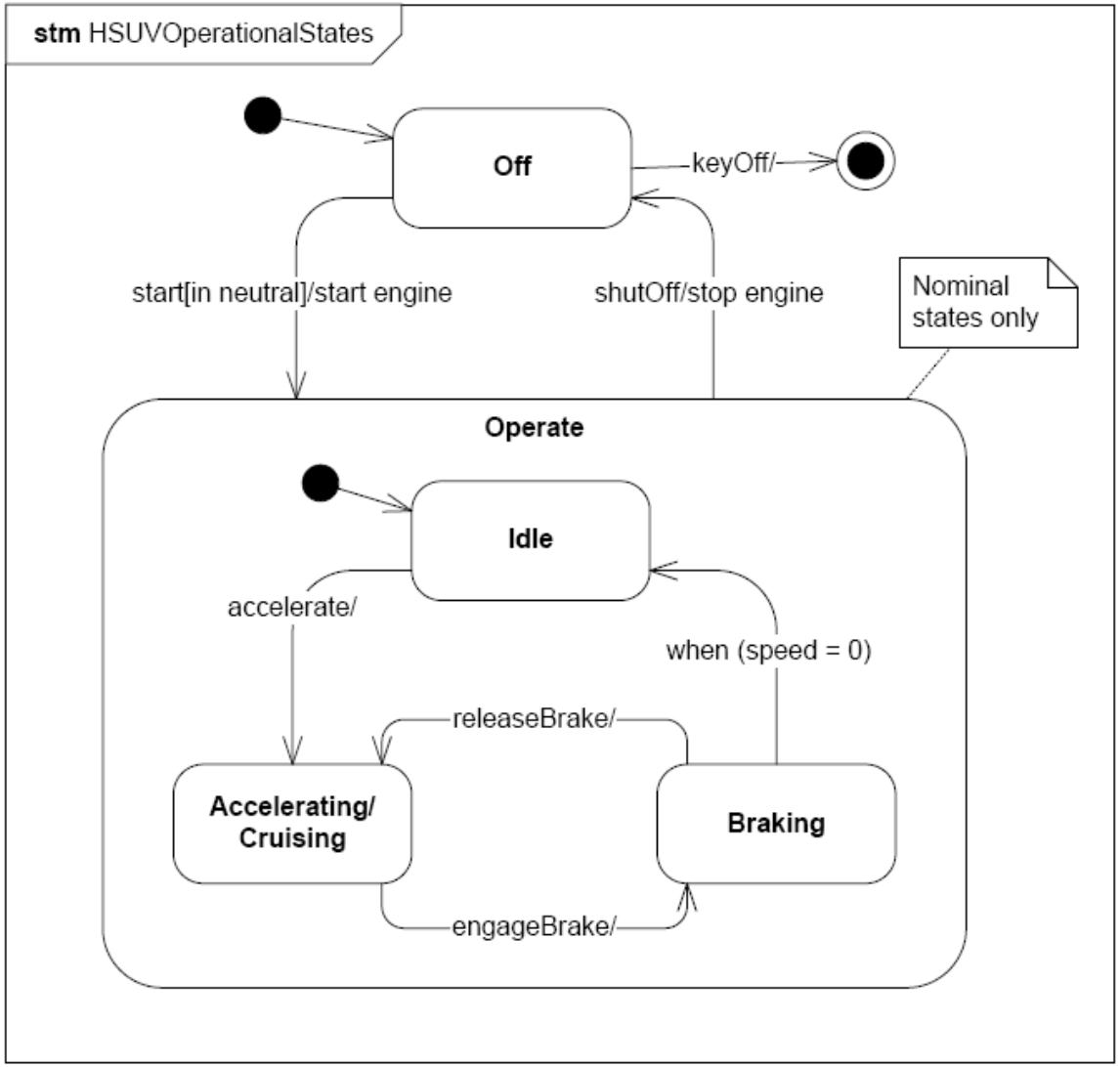
SysML Diagram Overview

- Package Diagrams [pkg] represent the organization of a model in terms of packages that contain model elements (package, block, activity, ...)
- Block Definition Diagrams [bdd] represent structural elements called *blocks* and their composition and classification
- Internal Block Diagrams [ibd] represent interconnections and interfaces between the parts of a block
- Parametric Diagrams [par] represent constraints on property values to to support engineering analyses
- Activity Diagrams [act] represent behavior in terms of the *ordering of actions*, based on the availability of inputs, outputs, and control, and how the actions transform the inputs to outputs
- Sequence Diagrams [sd] represent behavior in terms of a sequence of messages exchanged between parts
- **State Machine Diagrams** [stm] **represent behavior of an entity in terms of the transitions between states triggered by events**

State Machines

- Typically used to represent the life cycle of a block
- Support event-based behavior (generally asynchronous)
  - Transition with trigger, guard, action
  - State with entry, exit, and do-activity
  - Can include nested sequential or concurrent states
  - Can send/receive signals to communicate between blocks during state transitions, etc.
- Event types
  - Change event
  - Time event
  - Signal event

Operational States (Drive)



Transition notation:
trigger[guard]/action

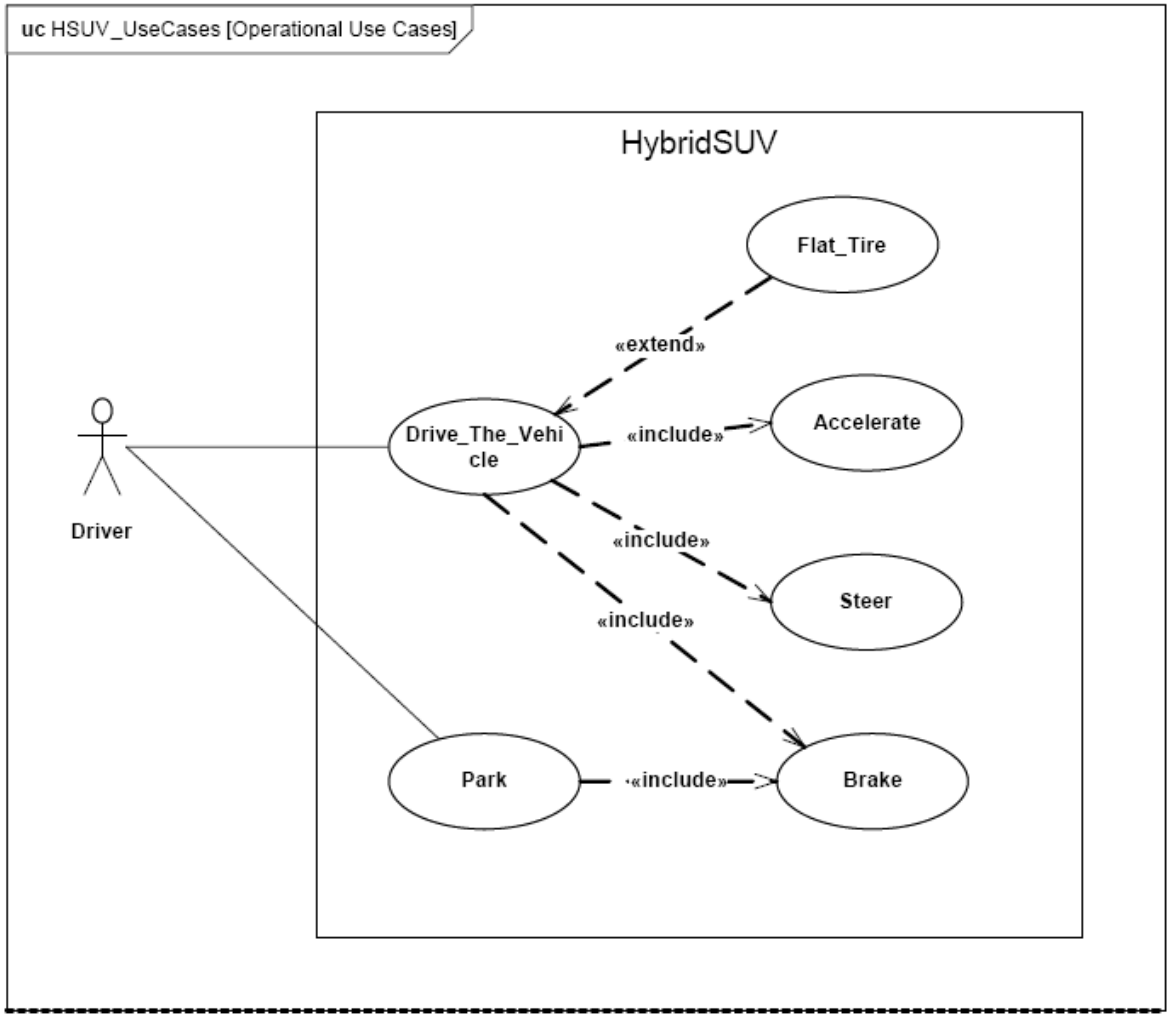
SysML Diagram Overview

- Package Diagrams [pkg] represent the organization of a model in terms of packages that contain model elements (package, block, activity, ...)
- Block Definition Diagrams [bdd] represent structural elements called *blocks* and their composition and classification
- Internal Block Diagrams [ibd] represent interconnections and interfaces between the parts of a block
- Parametric Diagrams [par] represent constraints on property values to to support engineering analyses
- Activity Diagrams [act] represent behavior in terms of the *ordering of actions*, based on the availability of inputs, outputs, and control, and how the actions transform the inputs to outputs
- Sequence Diagrams [sd] represent behavior in terms of a sequence of messages exchanged between parts
- State Machine Diagrams [stm] represent behavior of an entity in terms of the transitions between states triggered by events
- **Use Case Diagrams** [uc] **represent functionality in terms of how a system or other entity is used by external entities (i.e., *actors*) to accomplish a set of goals**

Use Cases

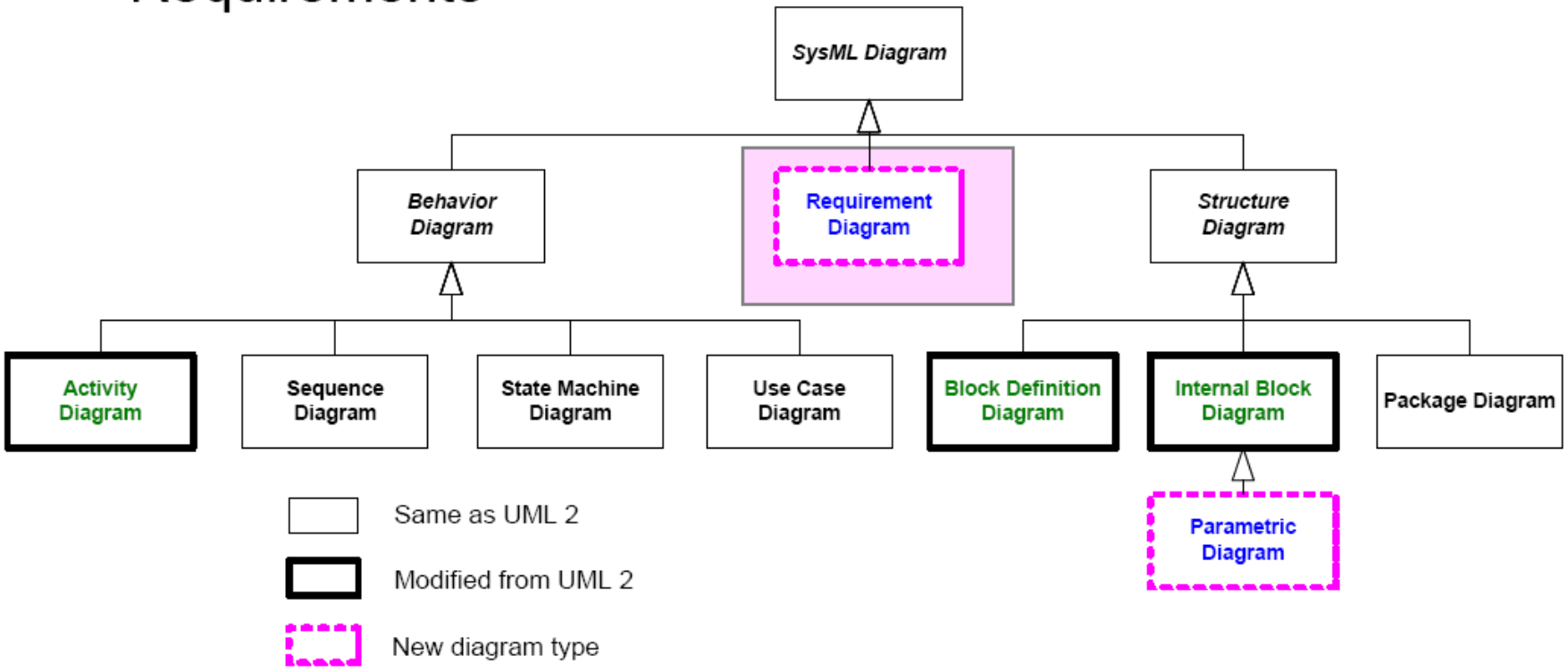
- Provide means for describing basic functionality in terms of usages/goals of the system by actors
  - Use is methodology dependent
  - Often accompanied by use case descriptions
- Common functionality can be factored out via «include» and «extend» relationships
- Elaborated via other behavioral representations to describe detailed scenarios
- No change to UML

Operational Use Cases



Cross-Cutting Constructs

- Allocations
- Requirements

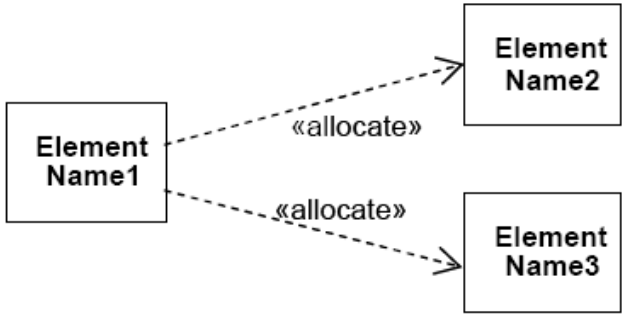


Allocations

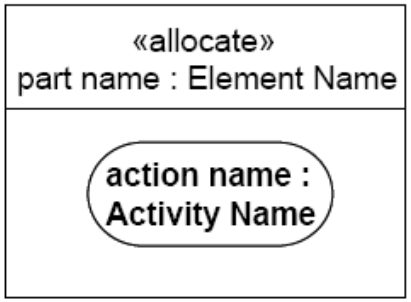
- Represent general relationships that map one model element to another
- Different types of allocation are:
  - Behavioral (i.e., function to component)
  - Structural (i.e., logical to physical)
  - Software to Hardware
  - ….
- Explicit allocation of activities to structure via swim lanes (i.e., activity partitions)
- Both graphical and tabular representations are specified

Various Allocation Representations

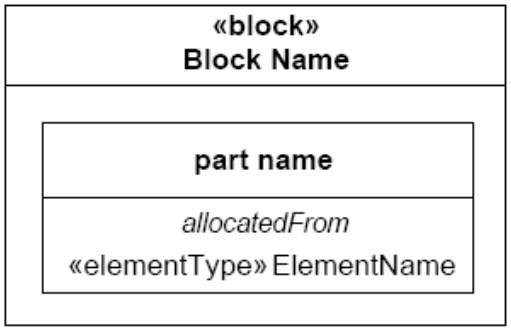
(Tabular Not Shown)



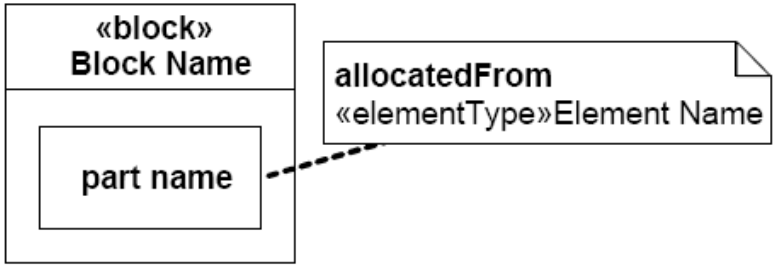
Allocate Relationship



Explicit Allocation of Action to Part Property



Compartment Notation



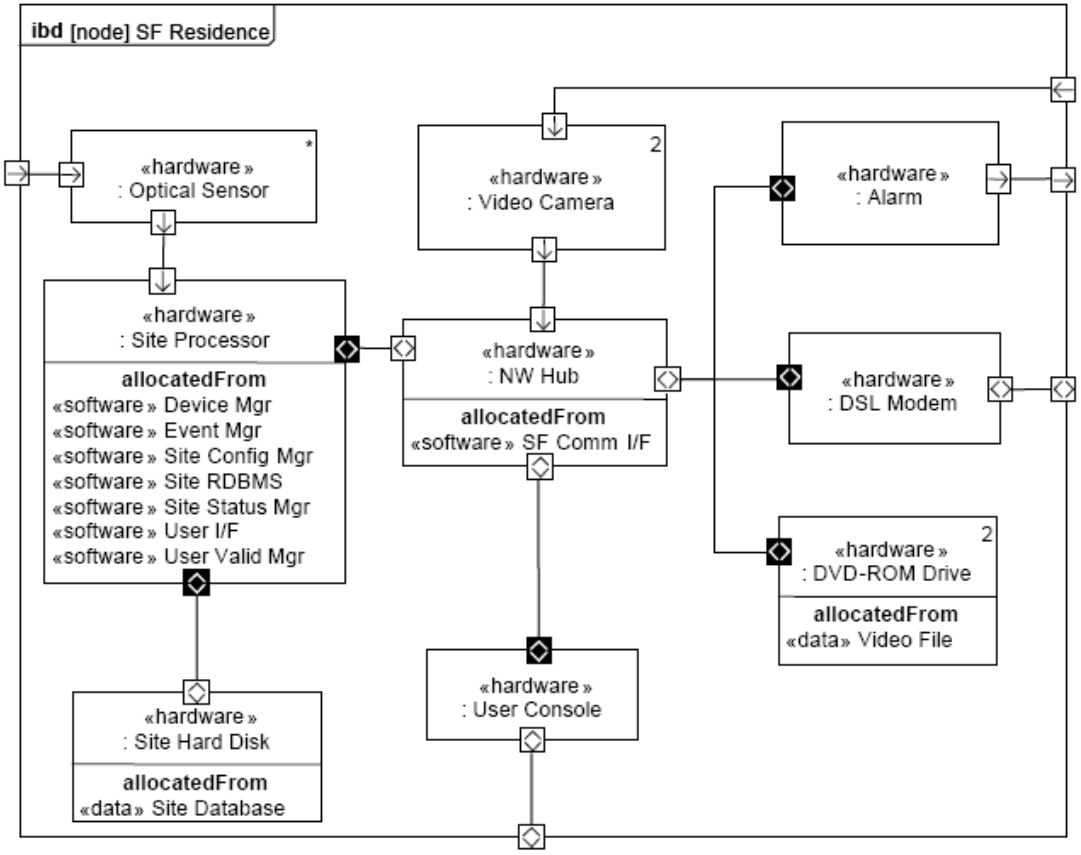
Callout Notation

Read as follows

"part name has constraints that are allocated to/from an <<element type>> Element Name" OR
"part name has an <<element type>> allocated from Element Name"
"Element Name is allocated to a part called part name"

Allocation of Software to Hardware

- In UML, the deployment diagram is used to deploy artifacts to nodes
- In SysML, «allocation» on an **ibd** and **bdd** is used to deploy software/data to hardware



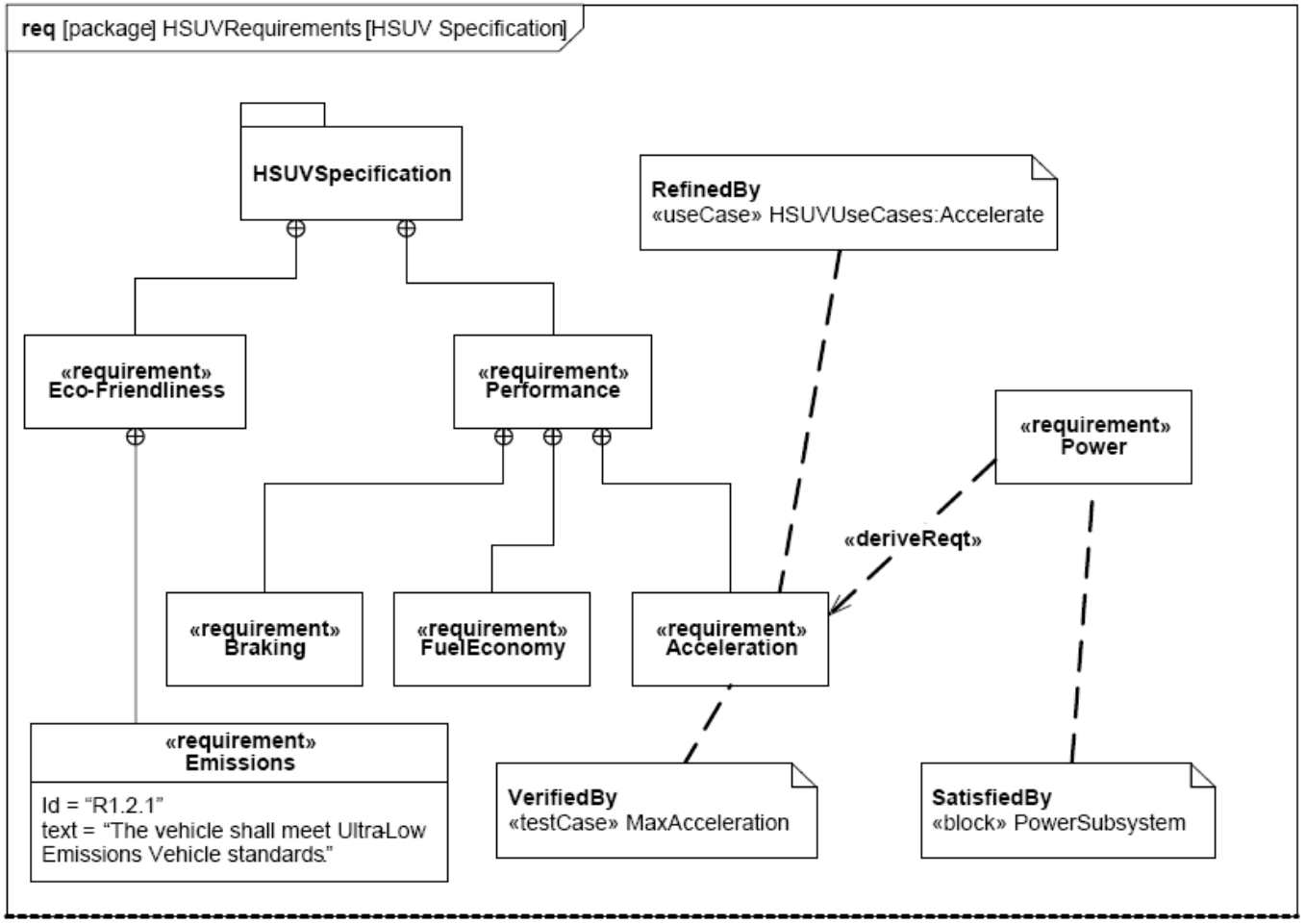
SysML Diagram Overview

- Package Diagrams [pkg] represent the organization of a model in terms of packages that contain model elements (package, block, activity, ...)
- Block Definition Diagrams [bdd] represent structural elements called *blocks* and their composition and classification
- Internal Block Diagrams [ibd] represent interconnections and interfaces between the parts of a block
- Parametric Diagrams [par] represent constraints on property values to to support engineering analyses
- Activity Diagrams [act] represent behavior in terms of the *ordering of actions*, based on the availability of inputs, outputs, and control, and how the actions transform the inputs to outputs
- Sequence Diagrams [sd] represent behavior in terms of a sequence of messages exchanged between parts
- State Machine Diagrams [stm] represent behavior of an entity in terms of the transitions between states triggered by events
- Use Case Diagrams [uc] represent functionality in terms of how a system or other entity is used by external entities (i.e., *actors*) to accomplish a set of goals
- **Requirement Diagrams** [req] **represent text-based requirements and their relationship with other requirements, design elements, and test cases to support requirements traceability**

Requirements

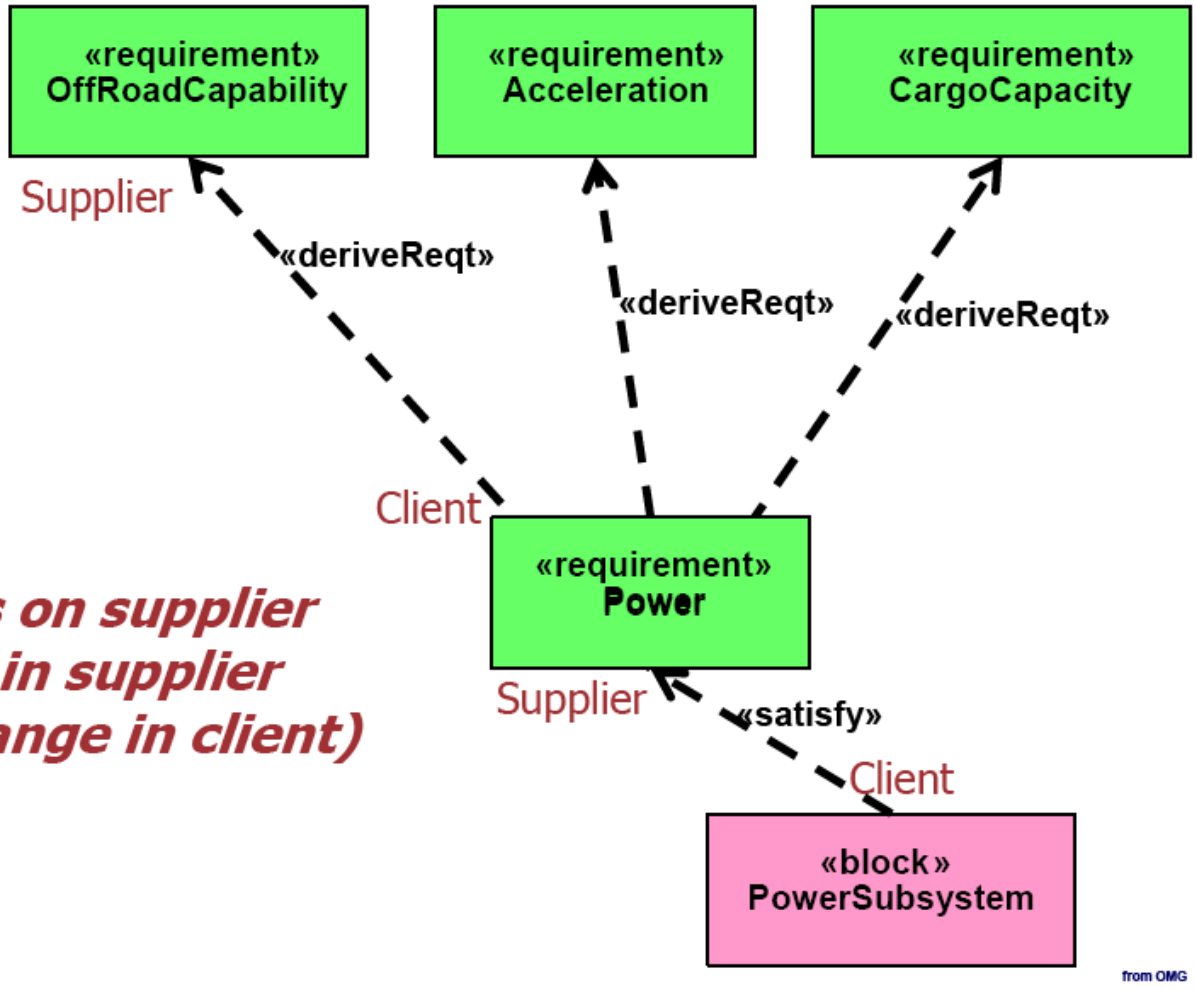
- The «requirement» stereotype represents a text based requirement
  - Includes id and text properties
  - Can add user defined properties such as verification method
  - Can add user defined requirements categories (e.g., functional, interface, performance)
- Requirements hierarchy describes requirements contained in a specification
- Requirements relationships include DeriveReqt, Satisfy, Verify, Refine, Trace, Copy

Requirements Breakdown



Requirement Relationships Model the Content of a Specification

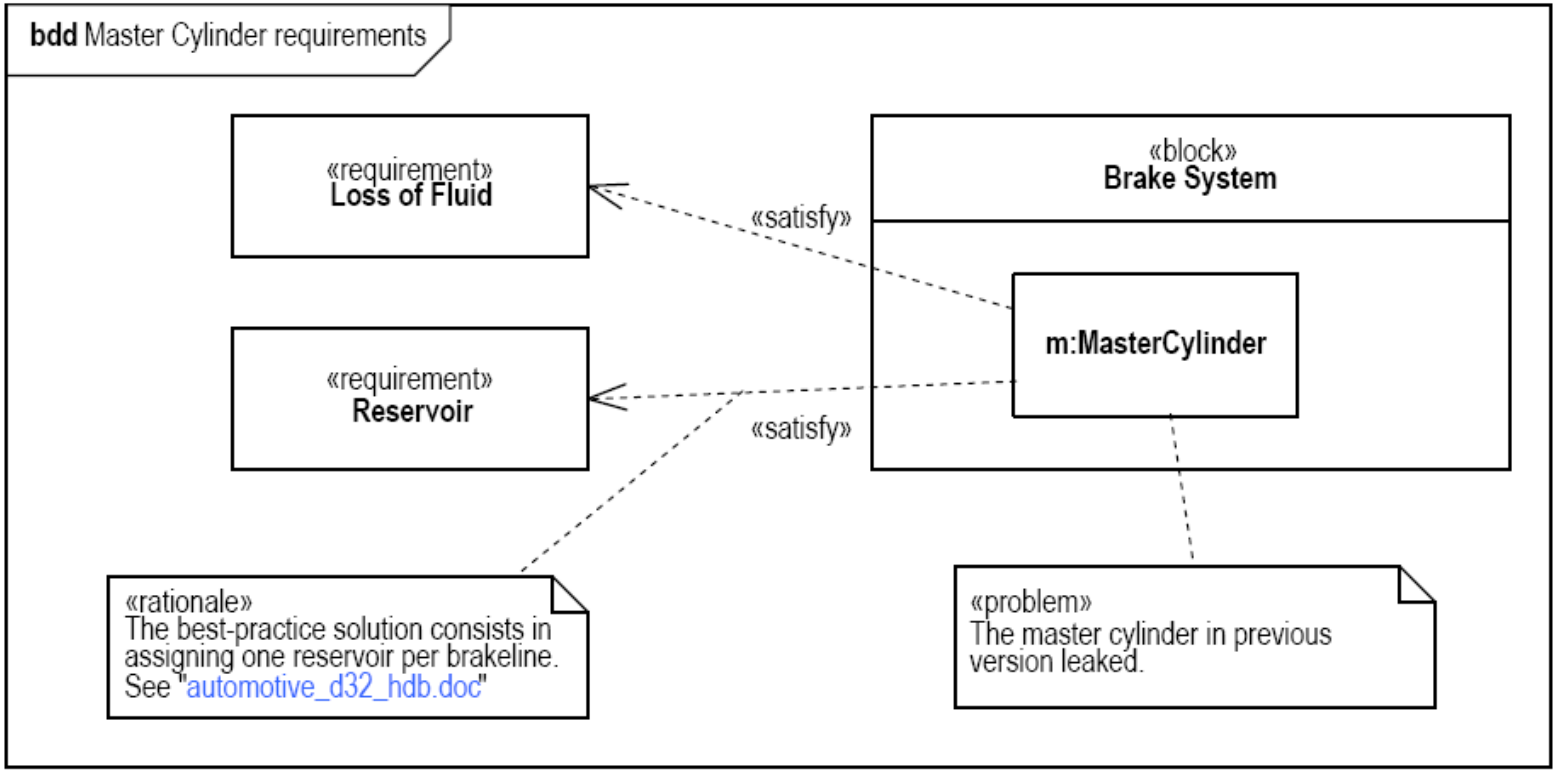
Derive/Satisfy Example



Client depends on supplier (i.e., a change in supplier results in a change in client)

Arrow Direction Opposite Typical Requirements Flow-Down

Problem and Rationale

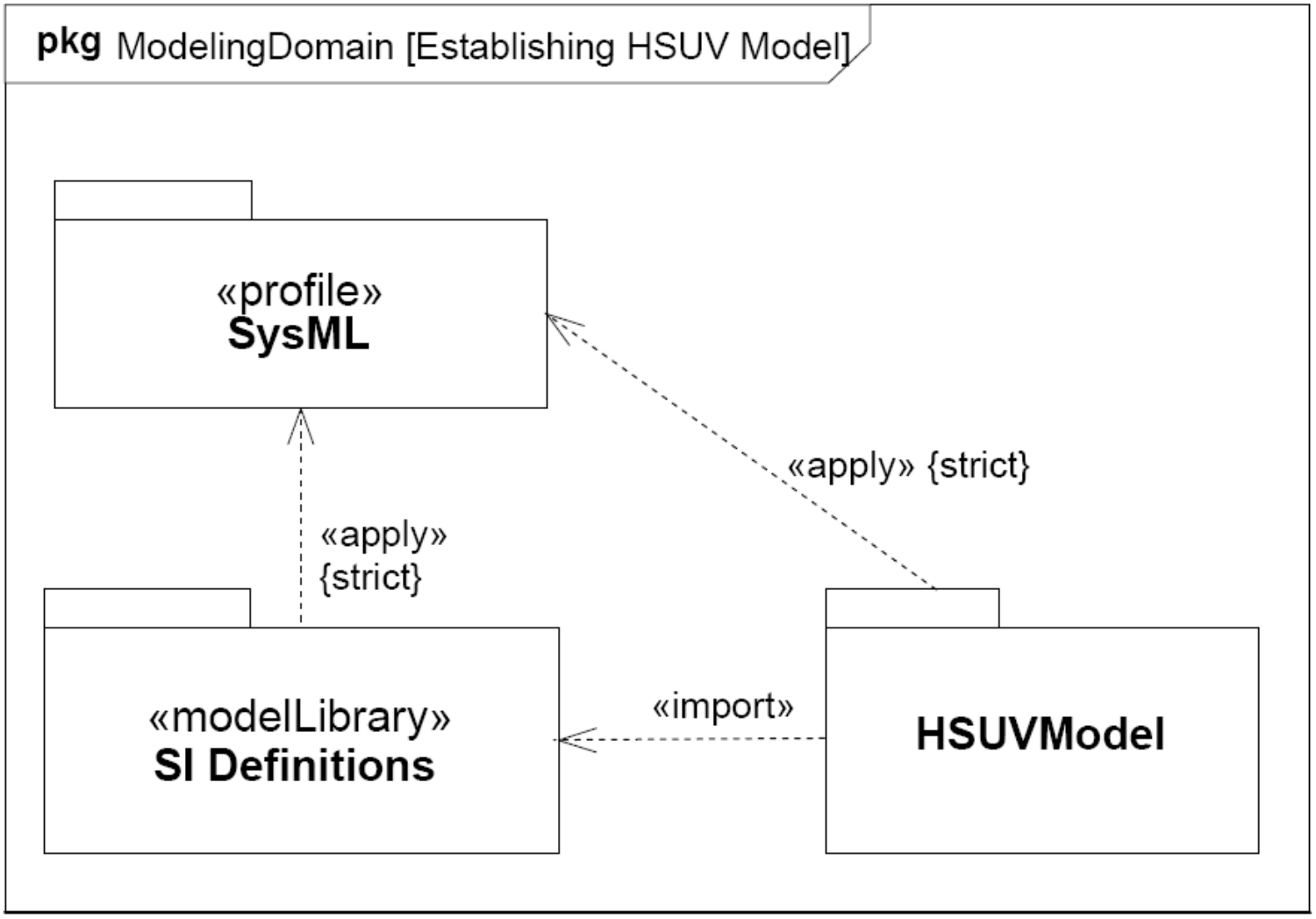


Problem and Rationale can be attached to any Model Element to Capture Issues and Decisions

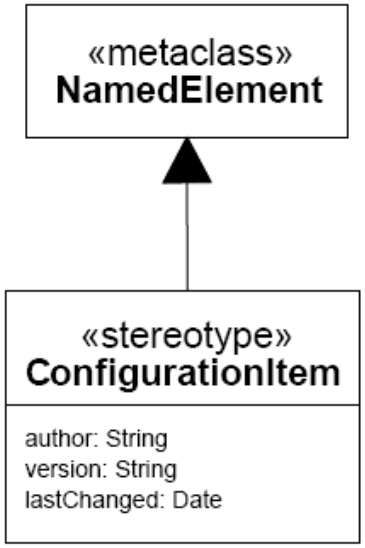
Profiles and Model Libraries

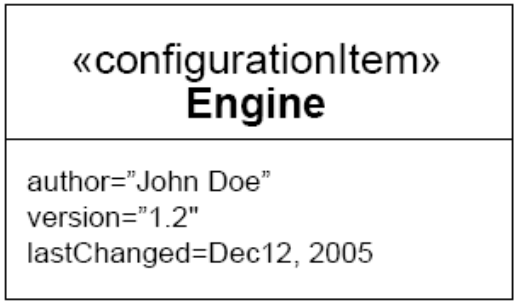
- Mechanisms for further customizing SysML
- Profiles represent extensions to the language
  - Stereotypes extend meta-classes with properties and constraints
    - Stereotype properties capture metadata about the model element
  - Profile is applied to user model
  - Profile can also restrict the subset of the meta-model used when the profile is applied
- Model Libraries represent reusable libraries of model elements

Applying a Profile and Importing a Model Library



Stereotypes

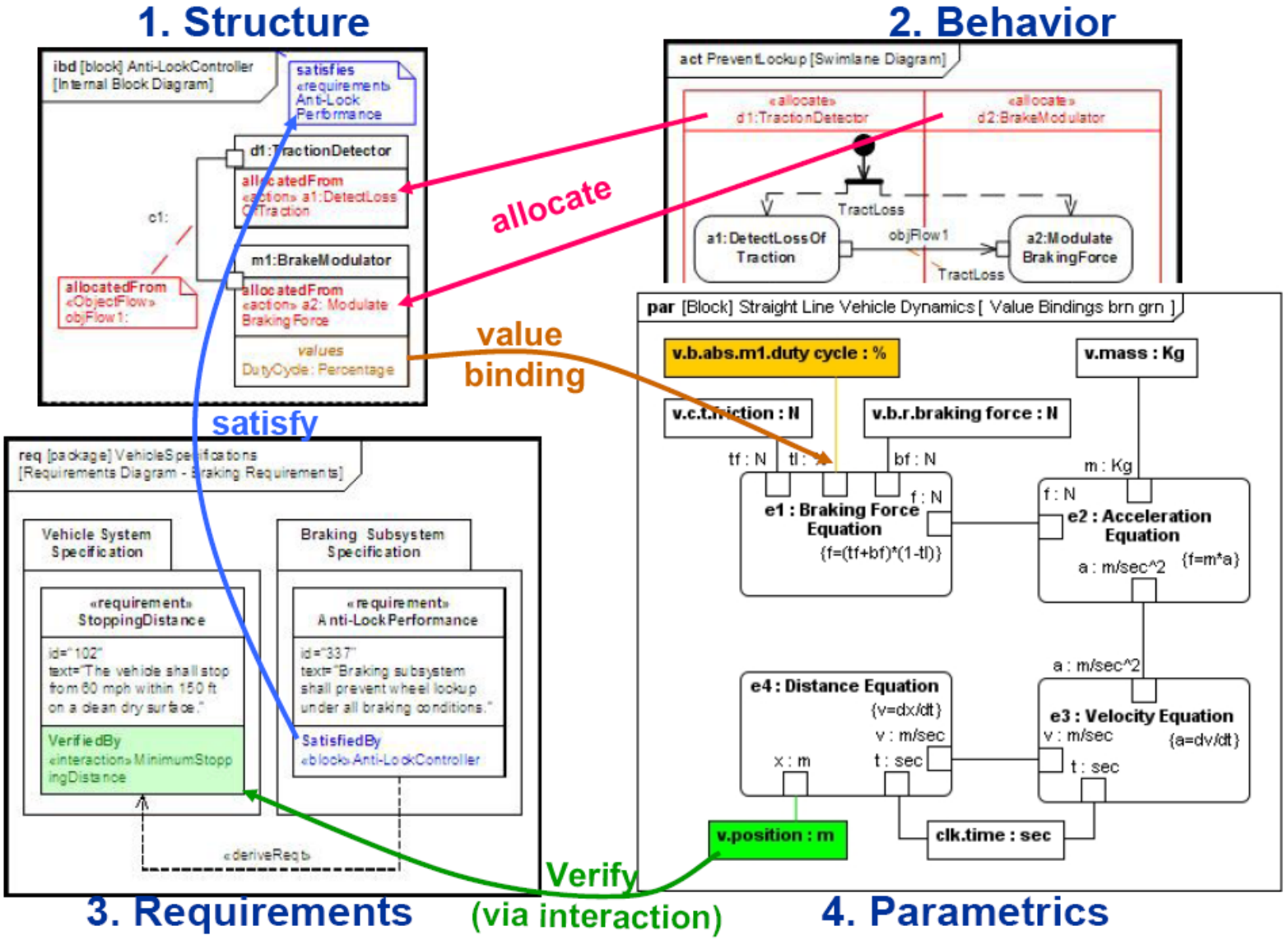




Defining the Stereotype

Applying the Stereotype

Cross-Connecting Model Elements



SysML Diagram Review

- **Package Diagrams** [pkg] **represent the organization of a model in terms of packages that contain model elements (package, block, activity, ...)**
- **Block Definition Diagrams** [bdd] **represent structural elements called *blocks* and their composition and classification**
- **Internal Block Diagrams** [ibd] **represent interconnections and interfaces between the parts of a block**
- **Parametric Diagrams** [par] **represent constraints on property values to to support engineering analyses**
- **Activity Diagrams** [act] **represent behavior in terms of the *ordering of actions*, based on the availability of inputs, outputs, and control, and how the actions transform the inputs to outputs**
- **Sequence Diagrams** [sd] **represent behavior in terms of a sequence of messages exchanged between parts**
- **State Machine Diagrams** [stm] **represent behavior of an entity in terms of the transitions between states triggered by events**
- **Use Case Diagrams** [uc] **represent functionality in terms of how a system or other entity is used by external entities (i.e., *actors*) to accomplish a set of goals**
- **Requirement Diagrams** [req] **represent text-based requirements and their relationship with other requirements, design elements, and test cases to support requirements traceability**

References

Reference Papers

- **Exploring Model Based Engineering for Large Telescopes**
  Karban, Robert, M Zamparelli, B Bauvir, B Koehler, L Noethe, A Balestra, SPIE Astronomical Telescopes and Instrumentation, 2008
- **Building Bridges Between Systems and Software With SysML and UML**
  Matthew Hause, Francis Thom, INCOSE Intl. Symposium, June ' 2008
- **Integrating Models and Simulations of Continuous Dynamics into SysML**
  Thomas Johnson, Christiaan Paredis, Roger Burkhart, Jan ' 2008
- **Modeling Continuous System Dynamics in SysML**
  Thomas Johnson, Jonathan Jobe, Christiaan Paredis, Roger Burkhart, Proceedings of the IMECE 2007, Nov ' 2007
- **Survey of Model Based Systems Engineering Methodologies (MBSE) revB**
  Jeff A Estefan, May 23 ' 2008
- **Simulation-Based Design Using SysML—Part 1: A Parametrics Primer,**
  RS Peak, RM Burkhart, SA Friedenthal, MW Wilson, M Bajaj, I Kim, INCOSE Intl. Symposium, June ' 2007
- **Simulation-Based Design Using SysML--Part 2: Celebrating Diversity by Example,**
  RS Peak, RM Burkhart, SA Friedenthal, MW Wilson, M Bajaj, I Kim, INCOSE Intl. Symposium, June ' 2007
- **An Overview of the Systems Modeling Language for Products and Systems Development,**
  Laurent Balmelli, Oct ' 2006.
- **Model-driven systems development,**
  L. Balmelli, D. Brown, M. Cantor, M. Mott, July ' 2006.
- **The SysML Modelling Language,**
  Matthew Hause, Fifteenth European Systems Engineering Conference, Sept ' 2006.
- **SysML and UML 2.0 Support for Activity Modeling**
  Bock. C., vol. 9 no.2, pp. 160-186, Journal of International Council of Systems Engineering, 2006.
- **Formal Performance Analysis and Simulation of UML/SysML Models for ESL Design**
  Alexander Viehl , Timo Schönwald, Oliver Bringmann, Wolfgang Rosenstiel
- **A SysML Extension for Bond Graphs Support**
  Turki S, Soriano T
- **SysML and Systems Engineering Applied to UML-Based SoC Design**
  Yves Vanderperren, Wim Dehaene
- **From UML/SysML to Matlab/Simulink: Current State and Future Perspectives**
  Yves Vanderperren, Wim Dehaene

Reference Books

- Systems Engineering with SysML/UML by Tim Weilkiens.
- A Practical Guide to SysML by Sanford Friedenthal, Alan Moore, and Rick Steiner.
- SysML for Systems Engineering: The Emperor's New Modelling Language by John Holt and Simon Perry.

SysML Vendors

- **See http://sysml-directory.omg.org/vendor/list.htm for a current complete list**

- Armstrong Process Group, Inc.
- **Artisan Software Tools**
- **EmbeddedPlus Engineering (3rd party to IBM Rational)**
- **IBM Telelogic (Rhapsody & Tau)**
- **No Magic, Inc.**
- North State Software
- oose Innovative Informatik GmbH
- Pathfinder Solutions
- SODIUS
- **Software Stencils** (Visio, SysML, and UML Templates)
- **Sparx Systems**



Systems Engineering Technical Short Course

(Session 4: System Architecture and Design)

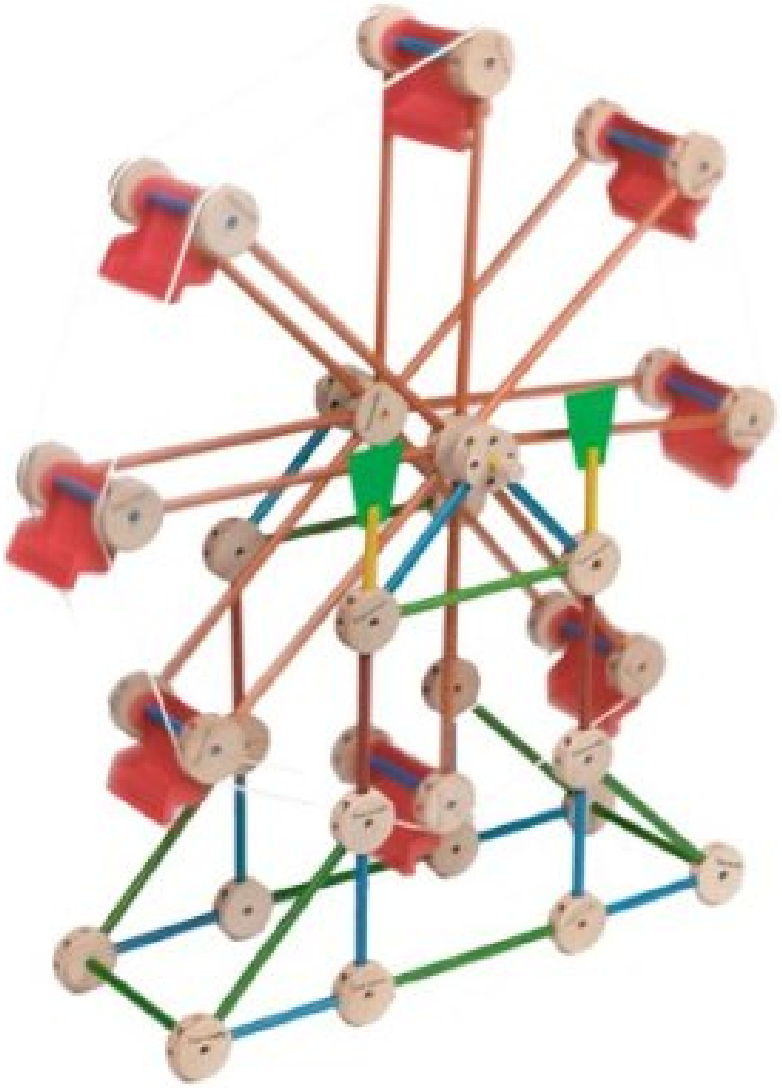
Appendix J:
Architecture Frameworks In-depth

Outline

- **DoDAF (1.0)**
- TOGAF
- Zachman
- FEAF

TinkerToys

- Views
  - Operational
    - Give [dolls?] a ride…
  - System
    - How implemented ☞
  - Technical
    - Sticks
      - Lengths
      - Colors
      - Strengths
      - …
    - Connectors
      - Round
      - Straight
      - …



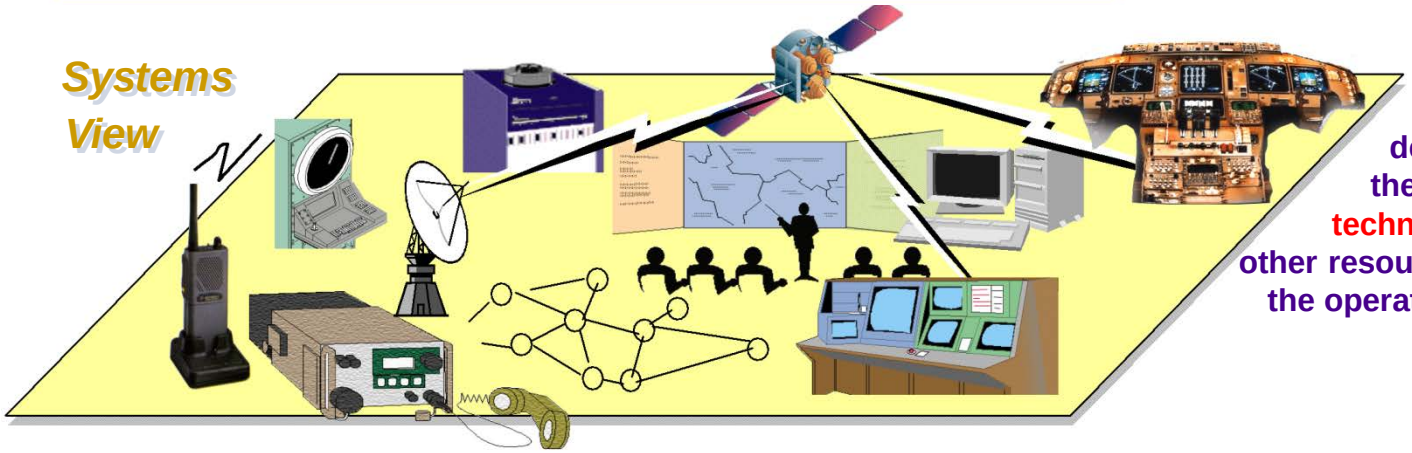
DoD Architecture Framework
One Architecture – Three Views

Operational View



The *Operational View* describes and interrelates the operational elements, tasks and activities, and information flows required to accomplish mission operations.

Systems View



The *Systems View* describes and interrelates the existing or postulated technologies, systems, and other resources intended to support the operational requirements.

Technical View



The *Technical View* describes the profile of rules, standards, and conventions governing systems implementation and forecasts their future direction.

DoD Architecture Framework Products

OPERATIONAL (OV)

- *1: High-Level Operational Concept Graphic**
- *2: Operational Node Connectivity Description**
- *3: Operational Information Exchange Matrix**
- 4: Organizational Relationships Chart
- *5: Operational Activity Model**
- 6a: Operational Rules Model
- 6b: Operational State Transition Description
- 6c: Operational Event/Trace Description
- 7: Logical Data Model

SYSTEMS (SV)

- *1: Systems Interface Description**
- *2: Systems Communications Desc.**
- 3: Systems-Systems Matrix
- 4: Systems Functionality Description
- 5: Operational Activity to System Function Traceability Matrix
- *6: Sys Data Exchange Matrix**
- 7: Sys Performance Parameters Matrix
- 8: Systems Evolution Description
- 9: Systems Technology Forecast
- 10a: Systems Rules Model
- 10b: Systems State Transition Desc.
- 10c: Systems Event/Trace Desc.
- 11: Physical Data Model

TECHNICAL (TV)

- *1: Technical Standards Profile**
- 2: Technical Standards Forecast

ALL (AV)

- *Overview & Summary**
- *Integrated Dictionary**

**** Denotes critical products***

CADM: Core Architecture Data Model
(common way of representing architectures)

Spreadsheets | Static Models & Graphics | Text | Dynamic Models

Scope

- The Department of Defense Architecture Framework (DoDAF) provides rules and guidance for developing and presenting architecture descriptions
  - The products defined by the framework are the work products of architecture development (the descriptive artifacts that communicate the architecture)
- DoDAF provides direction on how to describe architectures; it does not provide guidance in how to construct or implement a specific architecture or how to develop and acquire systems or systems-of-systems
- This framework covers the military domain and is used mainly by the DoD

Principles

- **Complete architectural descriptions require the use of multiple views, each of which conveys different aspects of the architecture in several products (descriptive artifacts or models)**

- **DoDAF defines the following views:**
  - ***Operational Views*** **depict what is going on in the real world that is to be supported or enabled by systems represented in the architecture. Activities performed as parts of DoD missions and the associated information exchanges among personnel or organizations are the primary items modeled in operational views. The operational view reveals requirements for capabilities and interoperability.**
  - ***Systems Views*** **describe existing and future systems and physical interconnections that support the DoD needs documented in the operational view.**
  - ***Technical (Standards) Views*** **catalog standard (COTS, GOTS) system parts or components and their interconnections. This view augments the systems view with technical detail and forecasts of standard technology evolution.**
  - ***All Views*** **augment the other views by providing context, summary or overview-level information, and an integrated dictionary to define terms.**

Structure

- **DoDAF contains four main types of guidance for architecture development:**
  - ***Guidelines*, which include a set of guiding principles and guidance for building architectures that are compliant with the framework**
  - **High-level *processes*, for using the framework to develop architecture descriptions that fulfill a purpose**
  - ***Discussion* of architecture data and tools that can serve as facilitators of the architecture-description process**
  - **Detailed *descriptions* of the products**

Guidelines

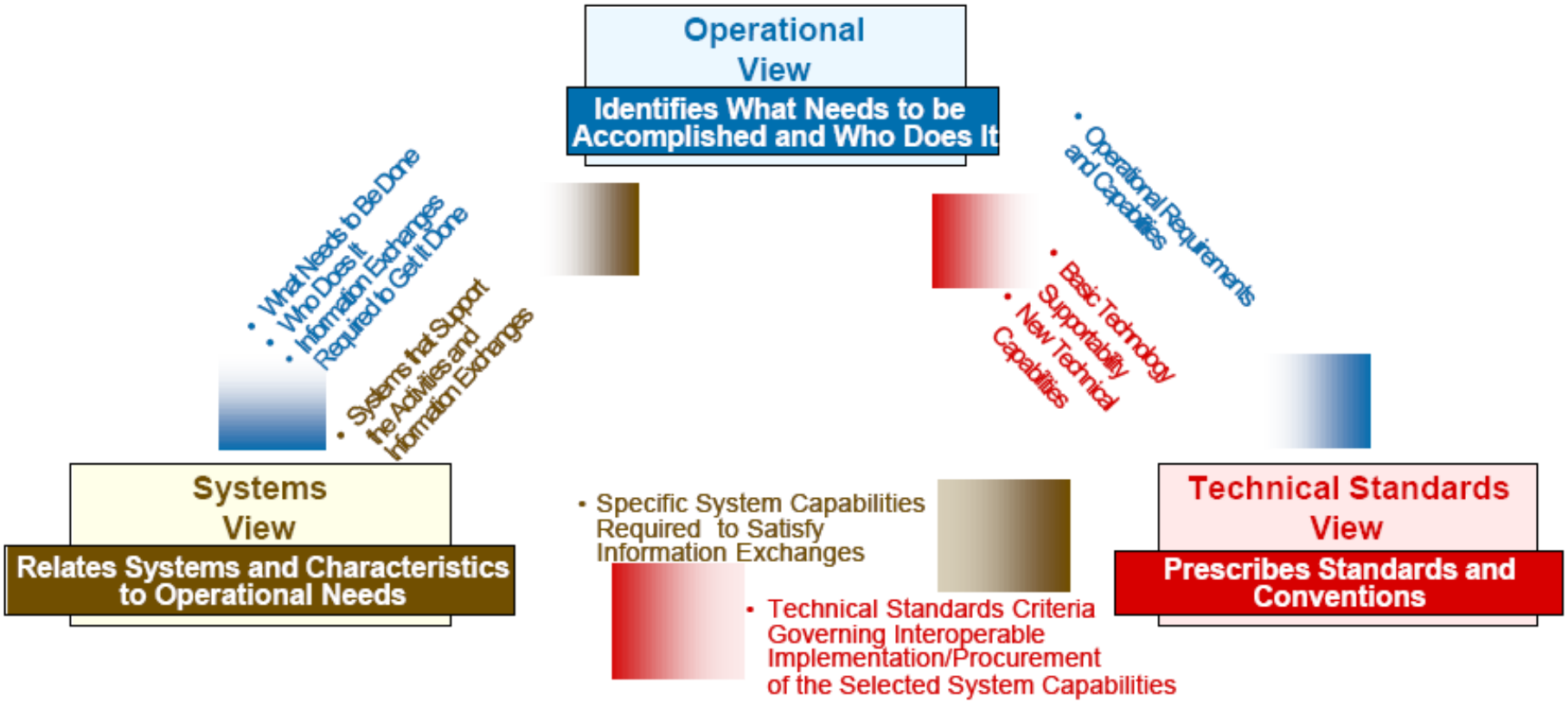
- **DoDAF is organized into three volumes.**
  - **Volume I - provides general guidance on the need for and use of architecture descriptions in the DoD context.**
  - **Volume II - provides detailed definitions of the 26 products contained in the 3 views. (See next page.)**
  - **Volume III is a deskbook that provides examples of compliant architectures, approaches to architecture development, and information on reference resources.**

Products

- **DoDAF defines 26 different architecture products, which are organized into the All, Operational, Systems, and Technical Standards views**

- **DoDAF emphasizes the need for developing or presenting the products *that are appropriate for a given audience***

The Three Views and their Major Linkages



Operational View (OV)

- **The OV is a description of the tasks and activities, operational elements, and information exchanges required to accomplish DoD missions**
- **DoD missions include both warfighting missions and business processes**
- **The OV contains graphical and textual products that comprise an identification of the operational nodes and elements, assigned tasks and activities, and information flows required between nodes**
- **It defines the types of information exchanged, the frequency of exchange, which tasks and activities are supported by the information exchanges, and the nature of information exchanges**

Technical Standards View (TV)

- **The TV is the minimal set of rules governing the arrangement, interaction, and interdependence of system parts or elements.**
- **Its purpose is to ensure that a system satisfies a specified set of operational requirements.**
- **The TV provides the technical systems implementation guidelines upon which engineering specifications are based, common building blocks are established, and product lines are developed.**
- **The TV includes a collection of the technical standards, implementation conventions, standards options, rules, and criteria organized into profile(s) that govern systems and system elements for a given architecture.**

Systems View (SV)

- **The SV is a set of graphical and textual products that describes systems and interconnections providing for, or supporting, DoD functions.**
- **DoD functions include both warfighting and business functions.**
- **The SV associates systems resources to the OV.**
- **These systems resources support the operational activities and facilitate the exchange of information among operational nodes.**

Requirements Drivers

- **The high-level operational concept should drive the Operational View (OV).**
- **The OV drives the Systems View (SV) to identify shortfalls and systems requirements.**
- **The SV requirements drive the Technical View (TV) to address a common set of applicable standards**

Architectural Aspects that Concern All Views (AVs)

- There are some overarching aspects of architecture that relate to all three of the views.
- These overarching aspects are captured in the All-Views (AV) products. The AV products provide information pertinent to the entire architecture but do not represent a distinct view of the architecture. AV products set the scope and context of the architecture.
- The scope includes the subject area and timeframe for the architecture.
- The setting in which the architecture exists comprises the interrelated conditions that compose the context for the architecture.
- These conditions include doctrine; tactics, techniques, and procedures; relevant goals and vision statements; concepts of operations; scenarios; and environmental conditions.

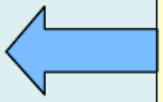
All Views

- **AV-1 Overview and Summary Information**
  - **Scope, purpose, intended users, environment depicted, analytical findings**

- **AV-2 Integrated Dictionary**
  - **Architecture data repository with definitions of all terms used in all products**

AV-1 Overview and Summary Information

- **Architecture Project Identification**
  - Name: **Combat Assessment**
  - Architect: **Contractor ABC**
  - Organization Developing the Architecture: **ASD(C3I)/CISA**
  - Assumptions and Constraints: **None**
  - Approval Authority: **USCENTCOM**
  - Date Completed: **12/10/98**
  - Level of Effort and Projected Costs to Develop the Architecture
- **Scope: Architecture View(s) and Products Identification**
  - Views and Products Developed: **All**
  - Time Frames Addressed: **Current**
  - Organizations Involved: **USCENTCOM J2 and J3**
- **Purpose and Viewpoint**
  - Purpose, Analysis, Questions to be Answered by Analysis of the Architecture: **Are information needs at operational nodes met by systems available?**
  - From Whose Viewpoint the Architecture is Developed: **Targeteer**
- **Context**
  - Mission: **Assess combat results**
  - Doctrine, Goals, and Vision
  - Rules, Criteria, and Conventions Followed: **War time conventions**
  - Tasking for Architecture Project, and Linkages to Other Architectures
- **Tools and File Formats Used: Combination**
- **Findings**
  - Analysis Results
  - Recommendations



Section to be completed after architecture description and analysis is completed

AV-2 Integrated Dictionary

- **The Integrated Dictionary contains definitions of terms used in the given architecture**

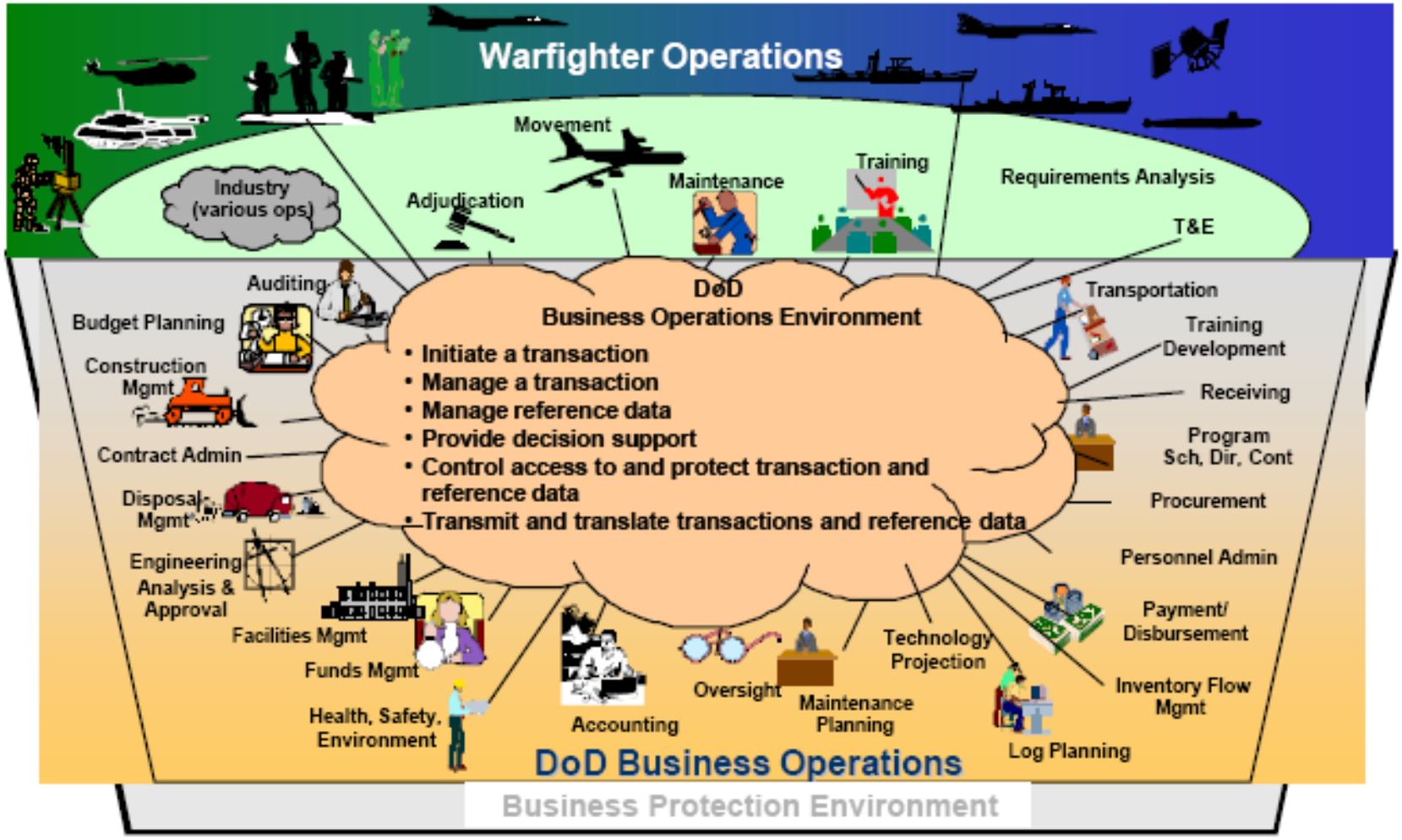
- **It consists of textual definitions in the form of a glossary, a repository of architecture data, their taxonomies, and their metadata (i.e., data about architecture data), including metadata for tailored products associated with the architecture products developed**
  - **Metadata are the architecture data types, possibly expressed in the form of a physical schema**

Operational Views

- OV-1 High-Level Operational Concept Graphic
  - High-level graphical/textual description of operational concept
- OV-2 Operational Node Connectivity Description
  - Operational nodes, connectivity, and information exchange between nodes
- OV-3 Operational Information Exchange Matrix
  - Information exchanged between nodes and the relevant attributes of that exchange
- OV-4 Organizational Relationships Chart
  - Organizational, role, or other relationships among organizations
- OV-5 Operational Activity Model
  - Capabilities, operational activities, relationships among activities, inputs, and outputs; overlays can show cost, performing nodes, or other pertinent information
- OV-6a Operational Rules Model
  - One of three products used to describe operational activity—identifies business rules that constrain operation
- OV-6b Operational State Transition Description
  - One of three products used to describe operational activity—identifies business process responses to events
- OV-6c Operational Event-Trace Description
  - One of three products used to describe operational activity—traces actions in a scenario or sequence of events
- OV-7 Logical Data Model
  - Documentation of the system data requirements and structural business process rules of the Operational View

OV-1 High-Level Operational Concept (1/3)

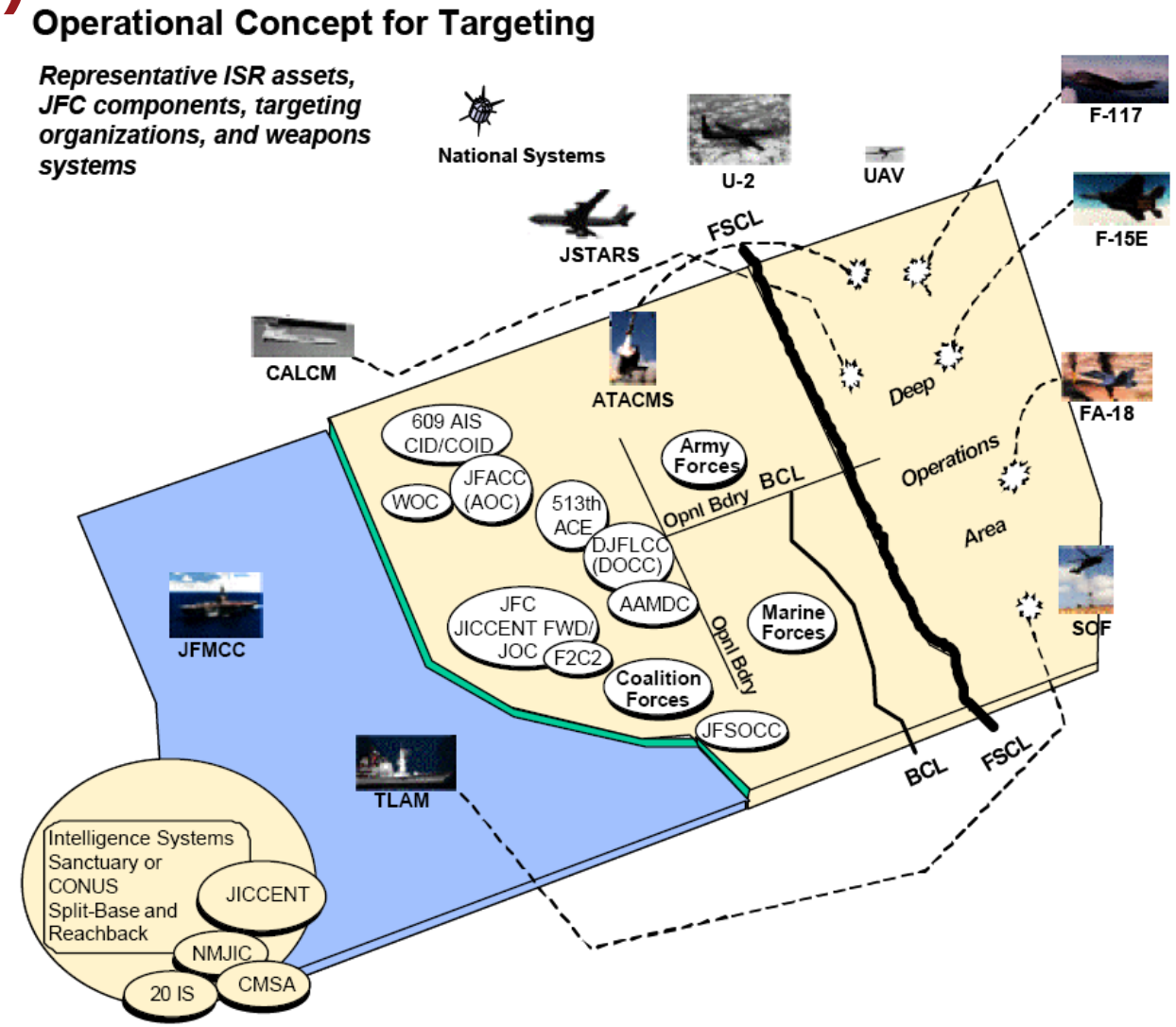
High-level graphical/textual description of operational concept



OV-1 High-Level Operational Concept (2/3)

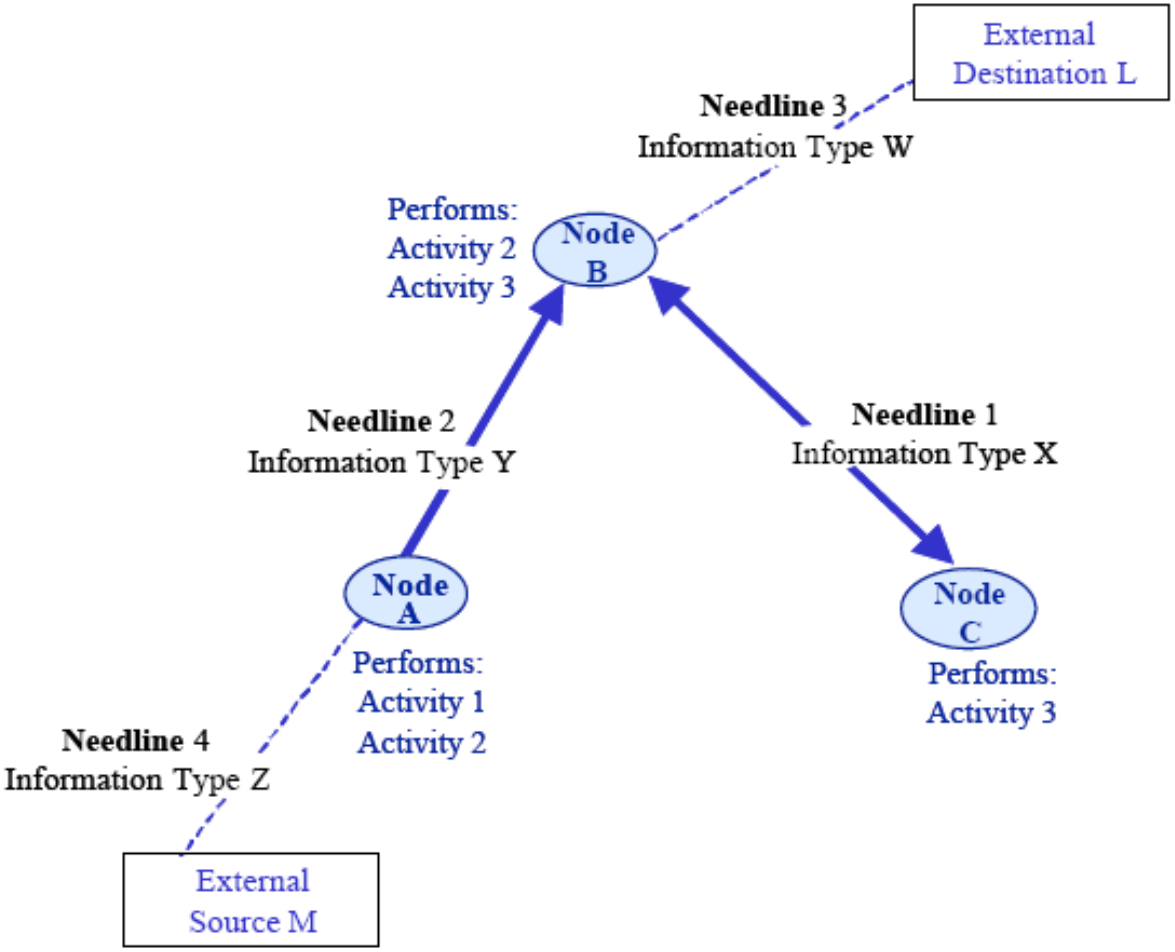


OV-1 High-Level Operational Concept (3/3)

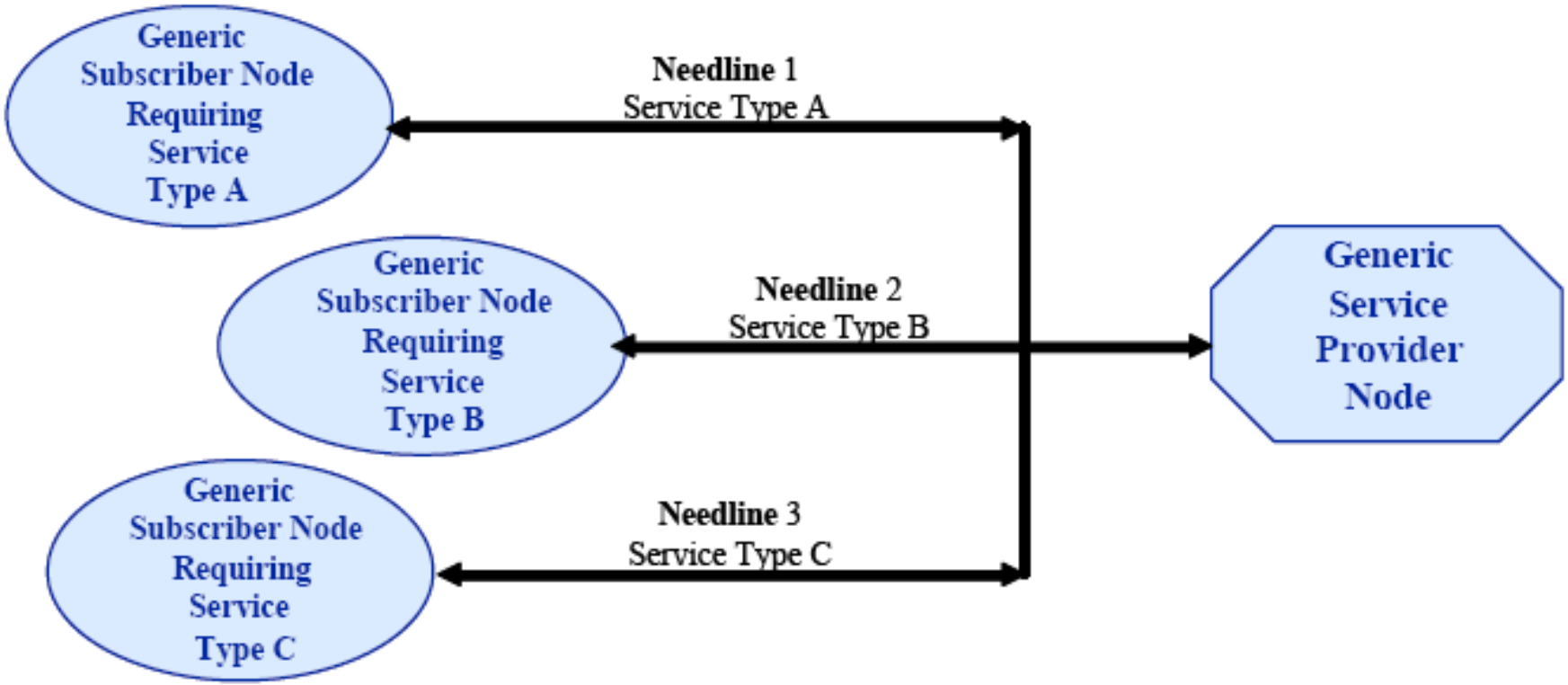


OV-2 Operational Node Connectivity (1/3)

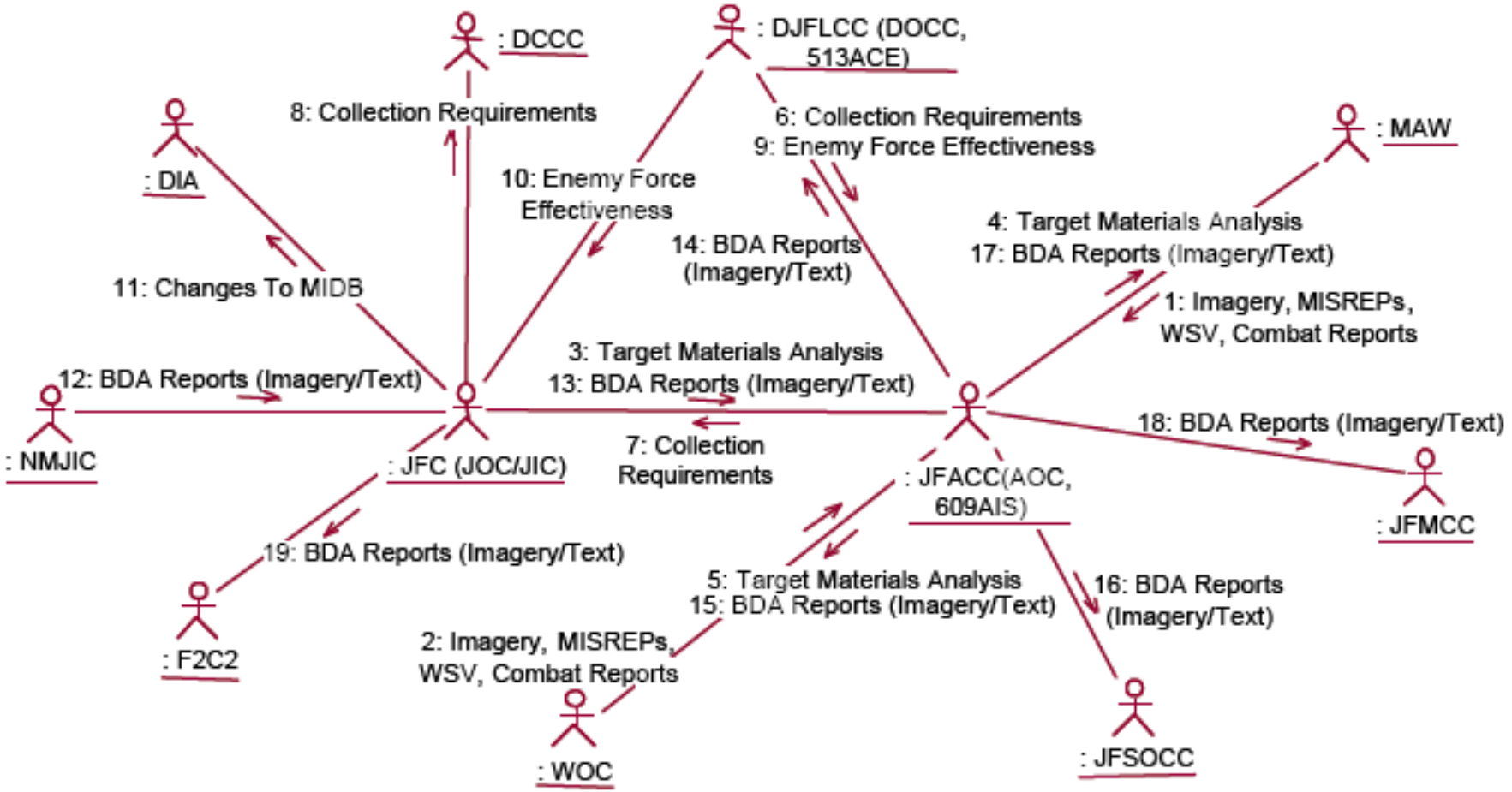
Operational nodes, connectivity, & internodal info. exchange



OV-2 Operational Node Connectivity (2/3)



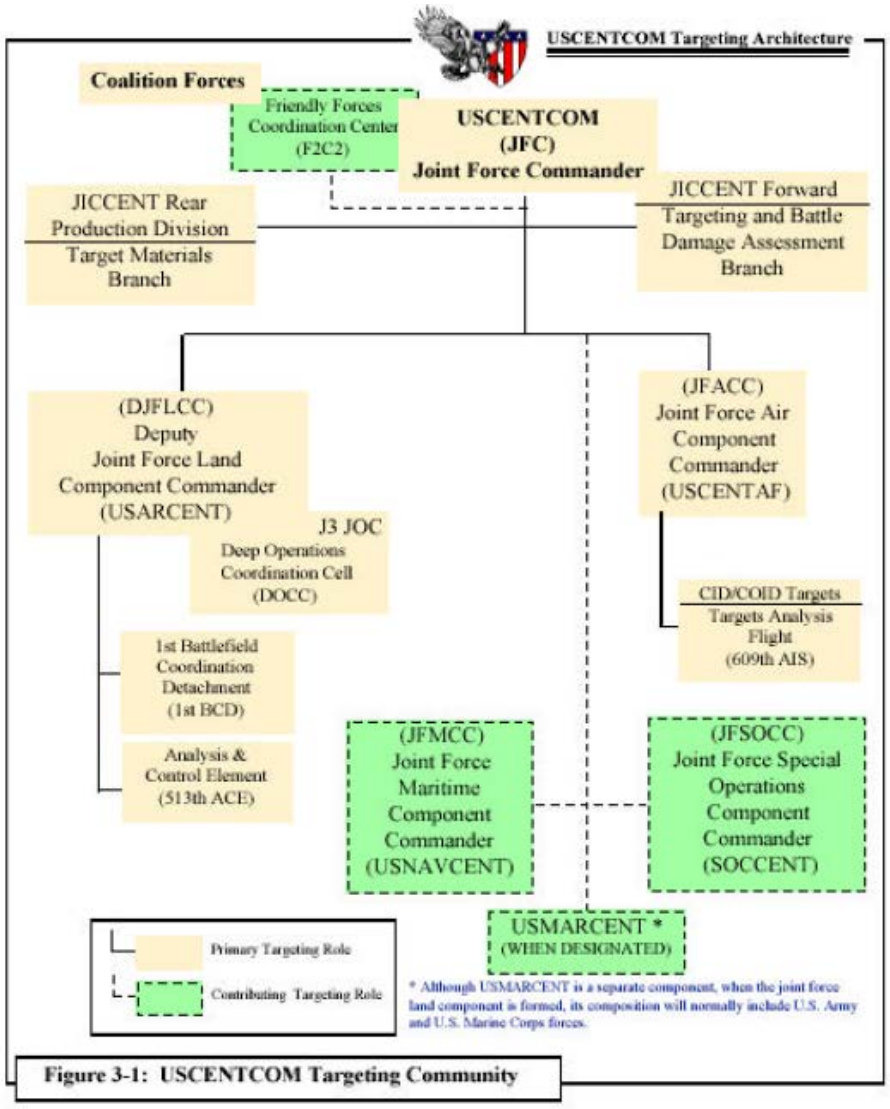
OV-2 Operational Node Connectivity (3/3)



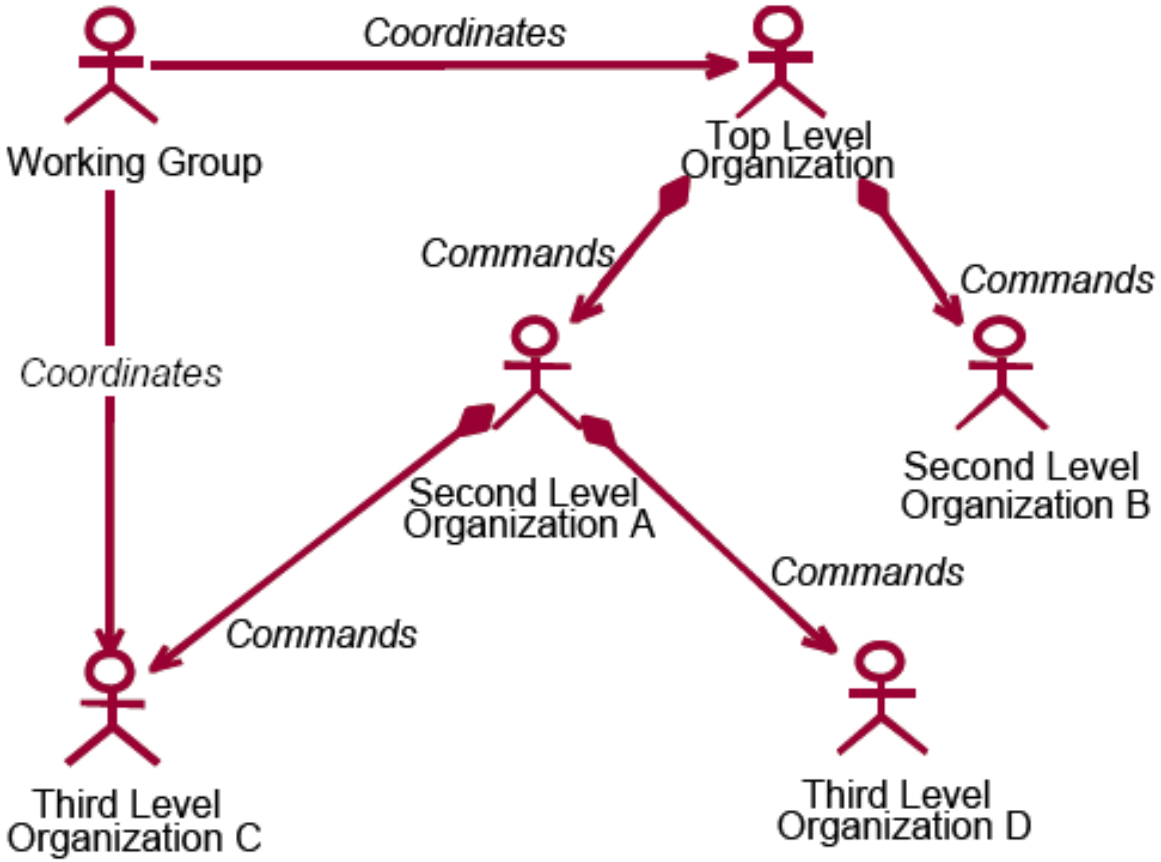
OV-3 Operational Information Exchange Matrix

Needline Identifier	Information Exchange Identifier	Information Element Description					Producer		Consumer		Nature of Transaction				
		Information Element Name and Identifier	Content	Scope	Accuracy	Language	Sending Op Node Name and Identifier	Sending Op Activity Name and Identifier	Receiving Op Node Name and Identifier	Receiving Op Activity Name and Identifier	Mission / Scenario UJTL or METL	Transaction Type	Triggering Event	Interoperability Level Required	Criticality
1	WOC-JFAC C1	BDA Report	Report on Battle Damage	Theater	1 Day	English	WOC	Conduct Battle Damage Assess-ment	JFACC	Conduct Munitions Effects Assess-ment	Combat Assess-ment	Collab-orate	Air Strike 072200, 0615am	2A	High
1	WOC-JFAC C2	Target Nomin-ations	Report on Possible Targets	Theater	2 Hours	English	WOC	Recom-mend Restrike	JFACC	Request Target Materials	Combat Assess-ment	Direct	AitTO XX, 072300	1B	High
2															
…															
11	MAW-JFAC C1	BDA Report	Report on Battle Damage	Theater	1 Day	English	MA W	Conduct Battle Damage Assess-ment	JFACC	Conduct Munitions Effects Assess-ment	Combat Assess-ment	Collab-orate	Air Strike 072200, 0615am	2A	High

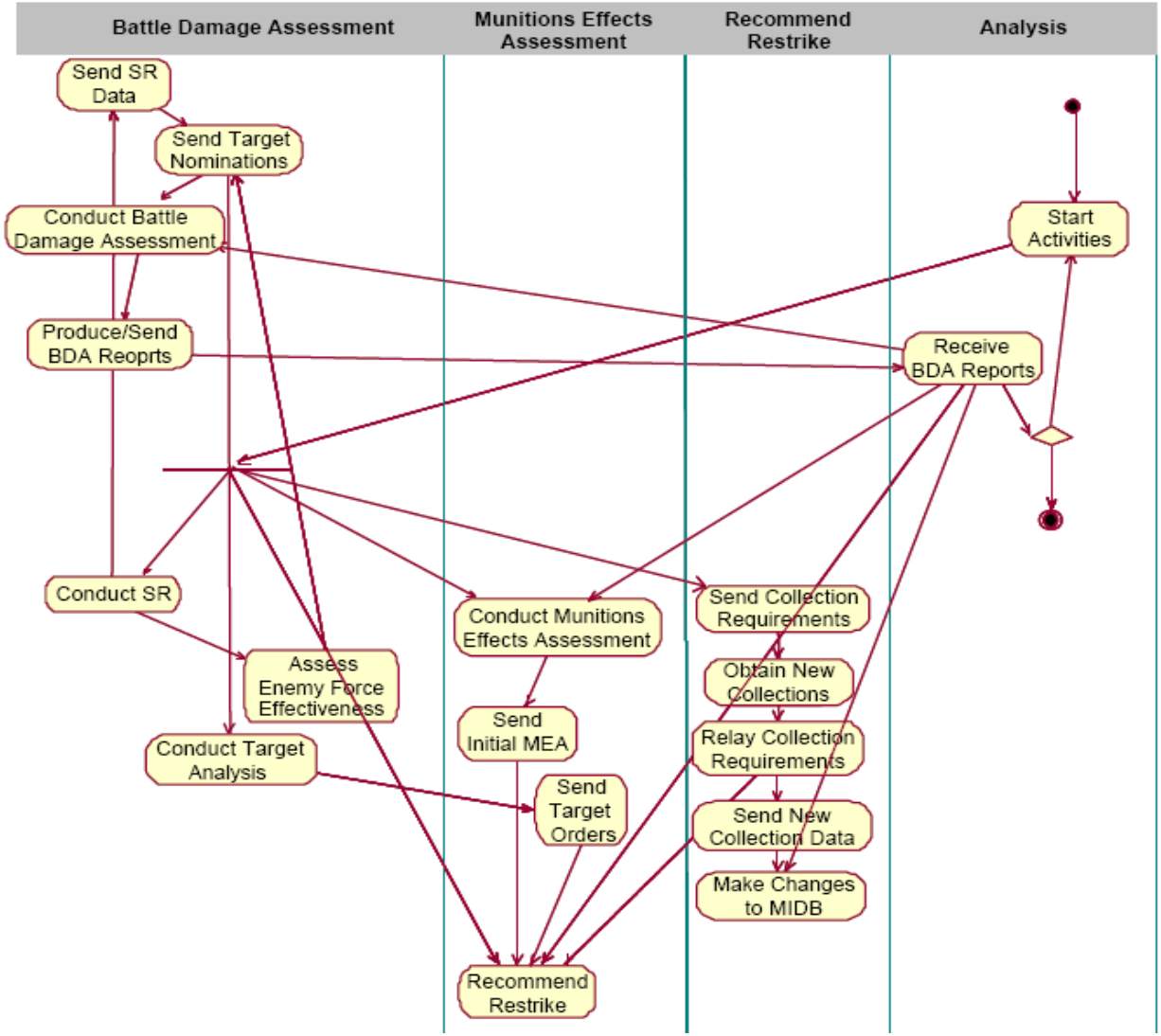
OV-4 Organizational Relationships Chart (1/2)



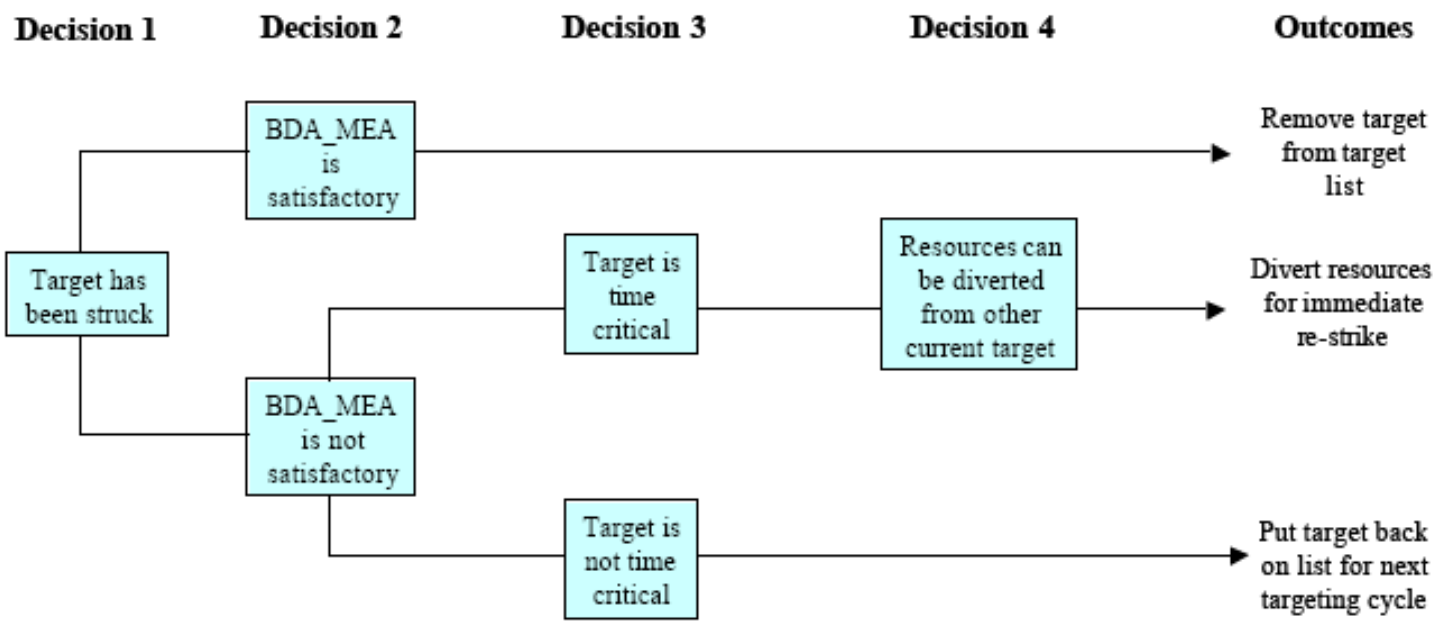
OV-4 Organizational Relationships Chart (2/2)



OV-5 Operational Activity Model



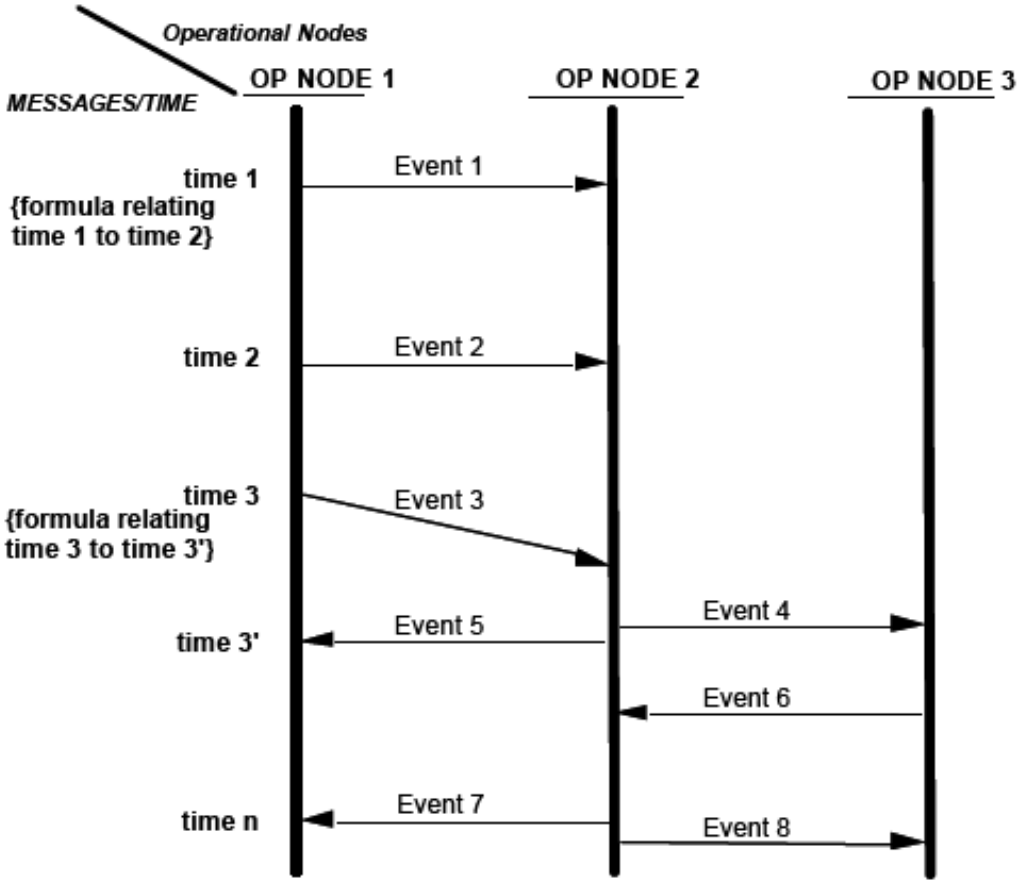
OV-6a Operational Rules Model



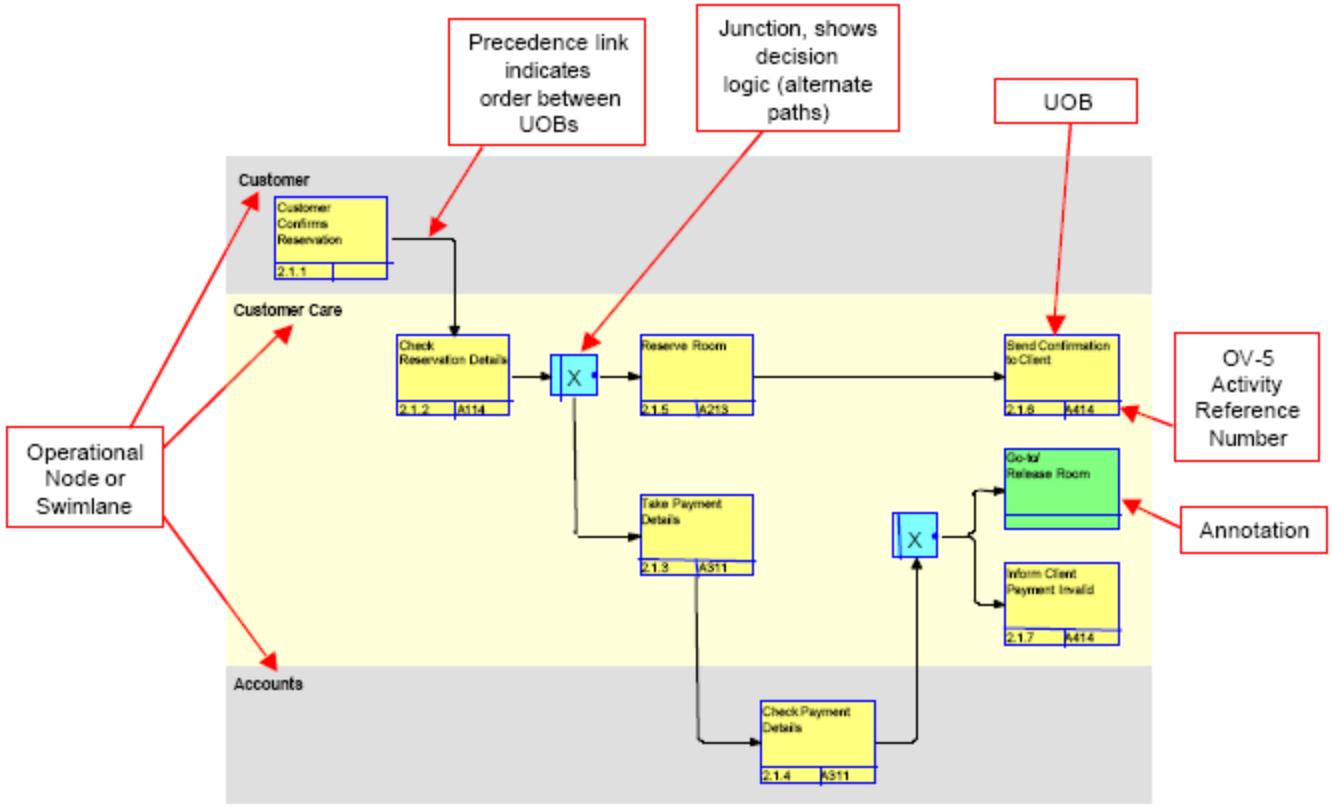
OV-6b Operational State Transition Description



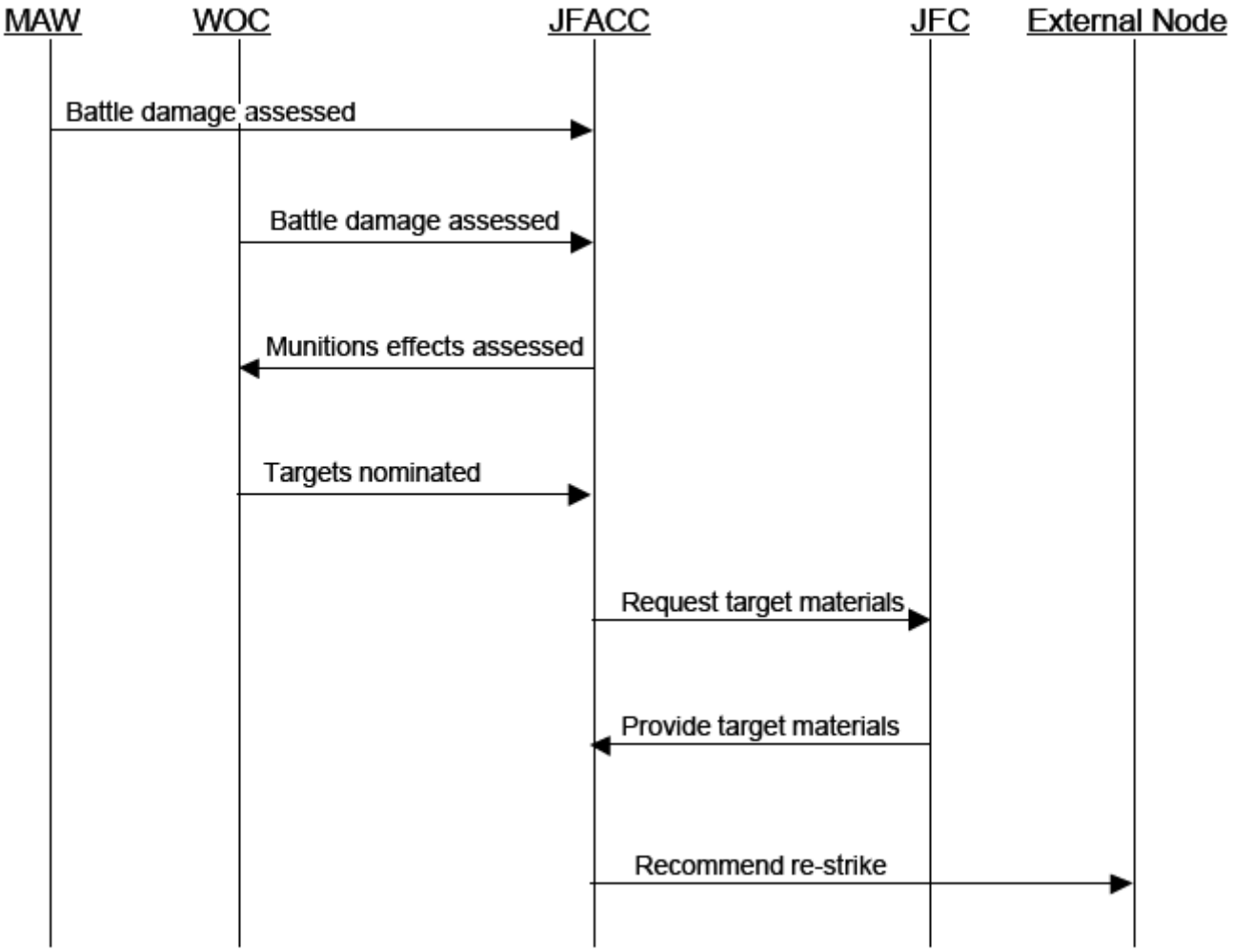
OV-6c Operational Event-Trace Description (1/3)



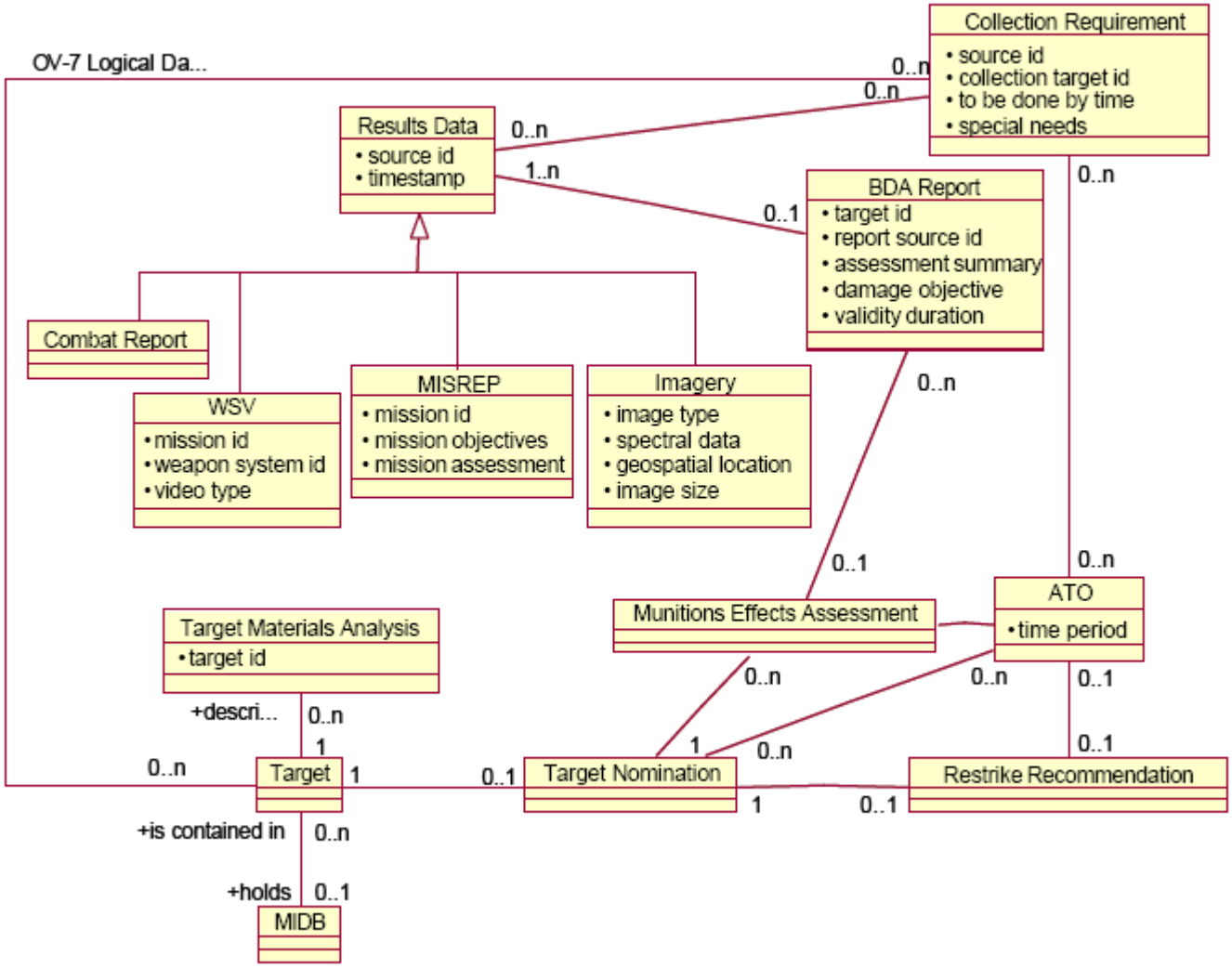
OV-6c Operational Event-Trace Description (2/3)



OV-6c Operational Event-Trace Description (3/3)



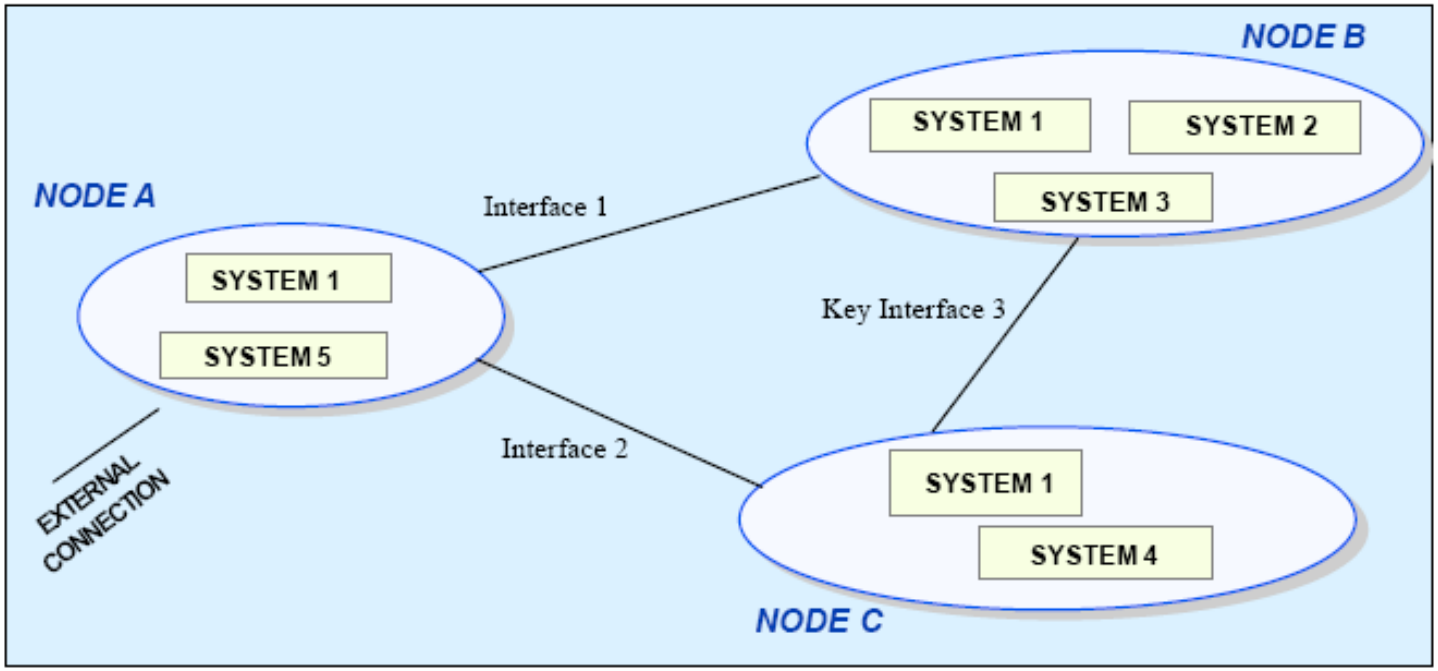
OV-7 Logical Data Model



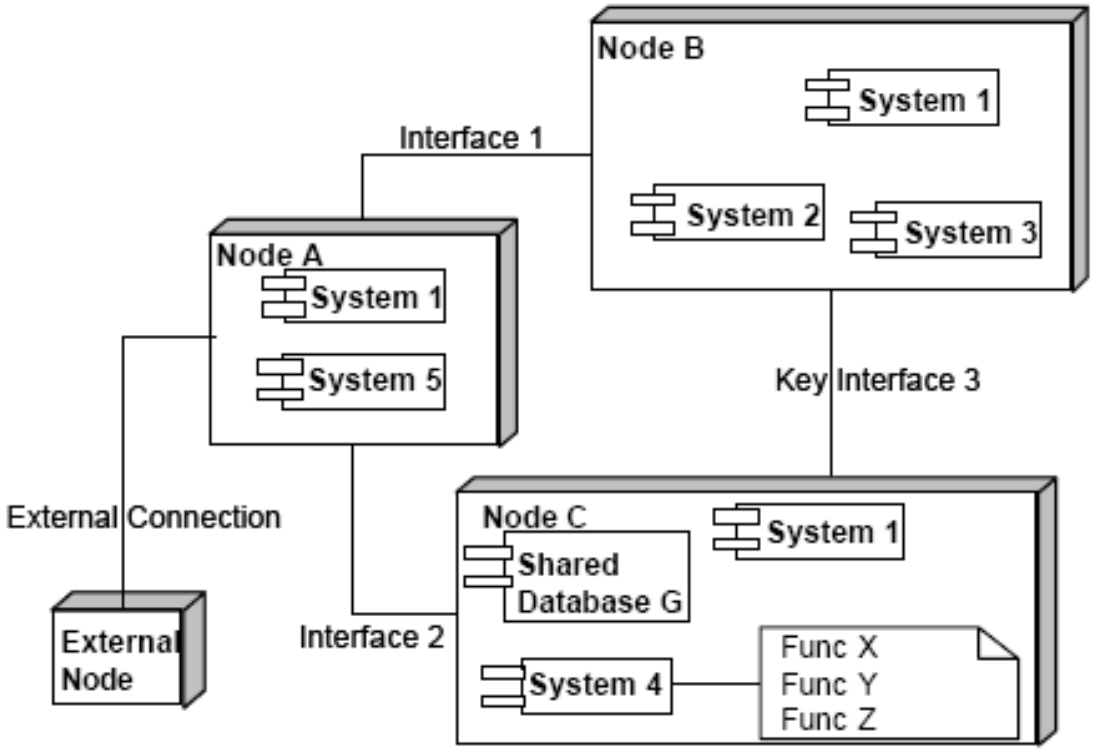
System Views

- SV-1 Systems Interface Description
  - Identification of systems nodes, systems, and system items and their interconnections, within and between nodes
- SV-2 Systems Communications Description
  - Systems nodes, systems, and system items, and their related communications lay-downs
- SV-3 Systems-Systems Matrix
  - Relationships among systems in a given architecture; can be designed to show relationships of interest, e.g., system-type interfaces, planned vs. existing interfaces, etc.
- SV-4 Systems Functionality Description
  - Functions performed by systems and the system data flows among system functions
- SV-5 Operational Activity to Systems Function Traceability Matrix
  - Mapping of systems back to capabilities or of system functions back to operational activities
- SV-6 Systems Data Exchange Matrix
  - Provides details of system data elements being exchanged between systems and the attributes of that exchange
- SV-7 Systems Performance Parameters Matrix
  - Performance characteristics of Systems View elements for the appropriate time frame(s
- SV-8 Systems Evolution Description
  - Planned incremental steps toward migrating a suite of systems to a more efficient suite, or toward evolving a current system to a future implementation
- SV-9 Systems Technology Forecast
  - Emerging technologies and software/hardware products that are expected to be available in a given set of time frames and that will affect future development of the architecture
- SV-10a Systems Rules Model
  - One of three products used to describe system functionality—identifies constraints that are imposed on systems functionality due to some aspect of systems design or implementation
- SV-10b Systems State Transition Description
  - One of three products used to describe system functionality—identifies responses of a system to events
- SV-10c Systems Event-Trace Description
  - One of three products used to describe system functionality—identifies system-specific refinements of critical sequences of events described in the Operational View
- SV-11 Physical Schema
  - Physical implementation of the Logical Data Model entities, e.g., message formats, file structures, physical schema

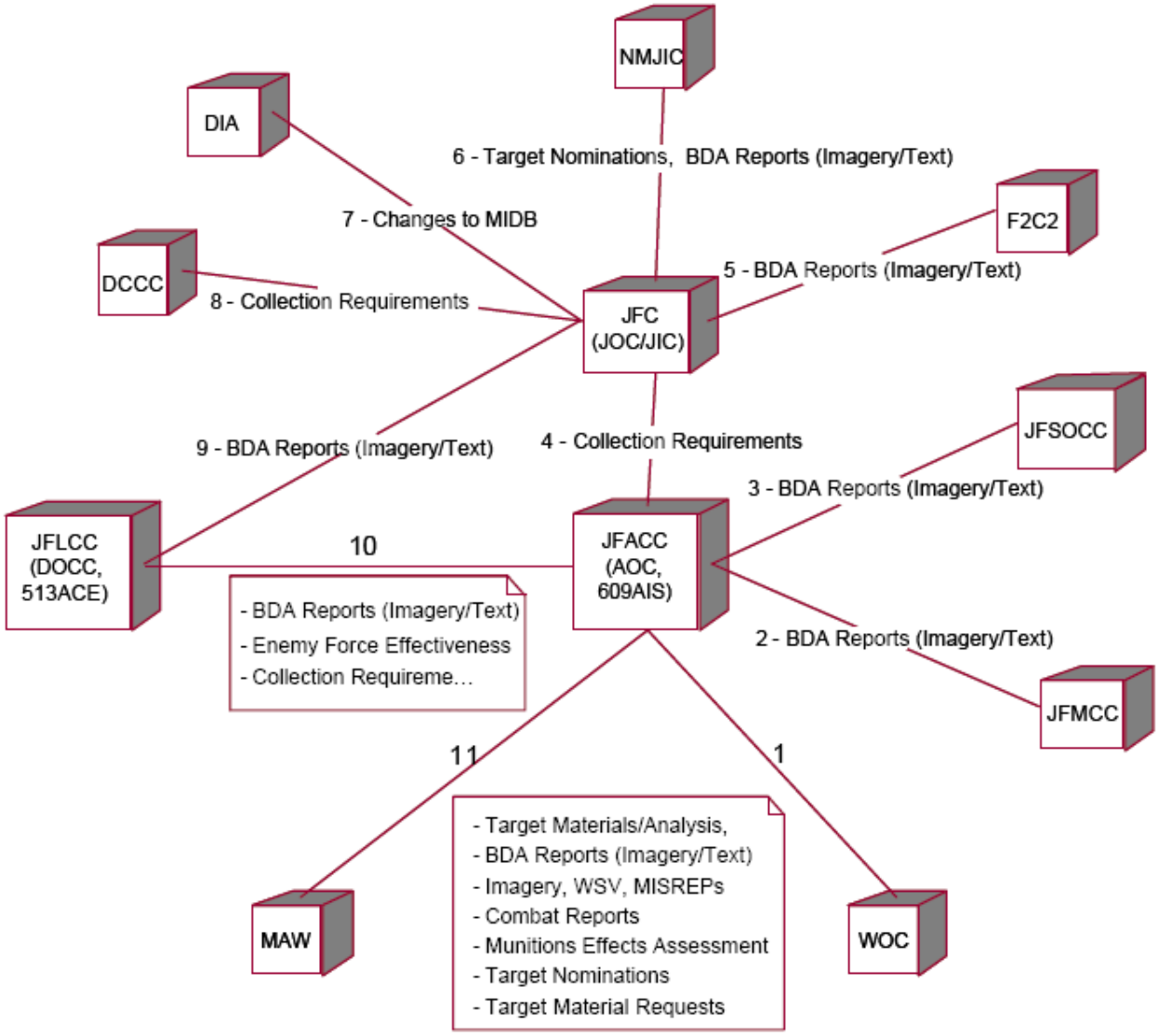
SV-1 Systems Interface Description (1/3)



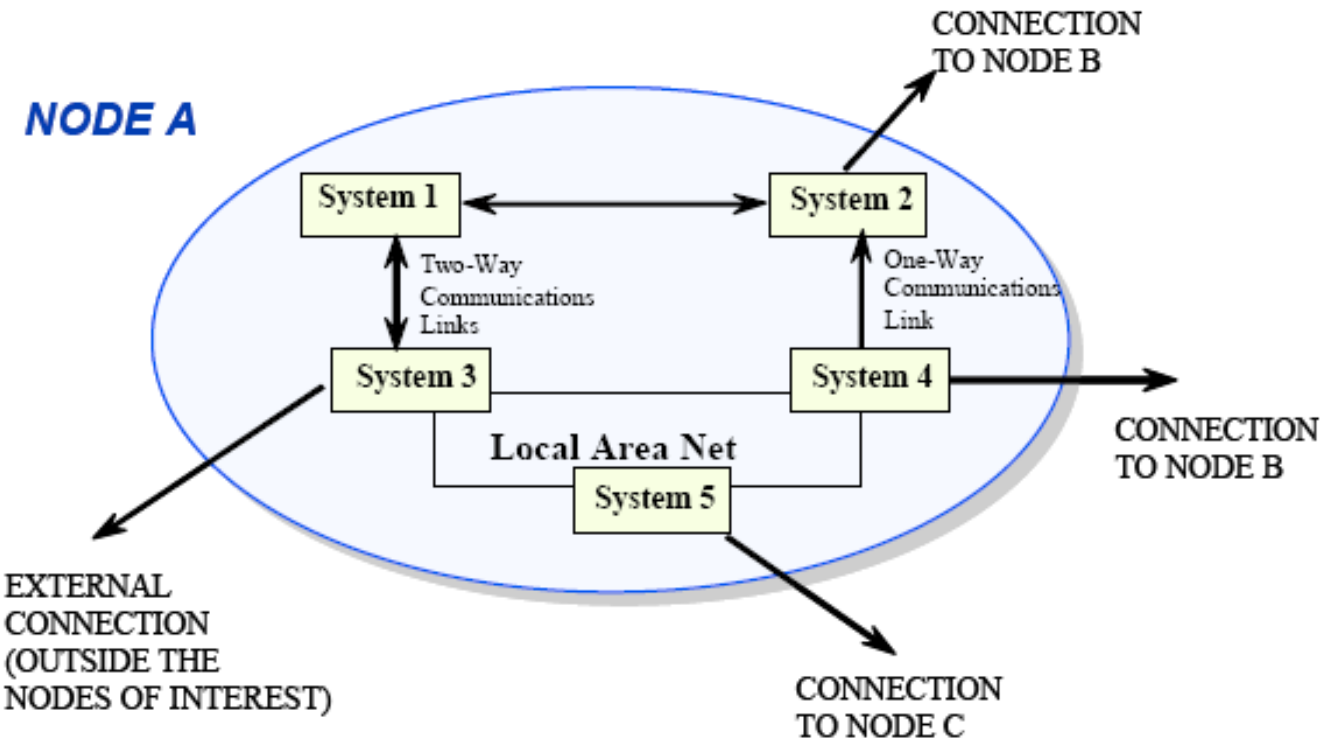
SV-1 Systems Interface Description (2/3)



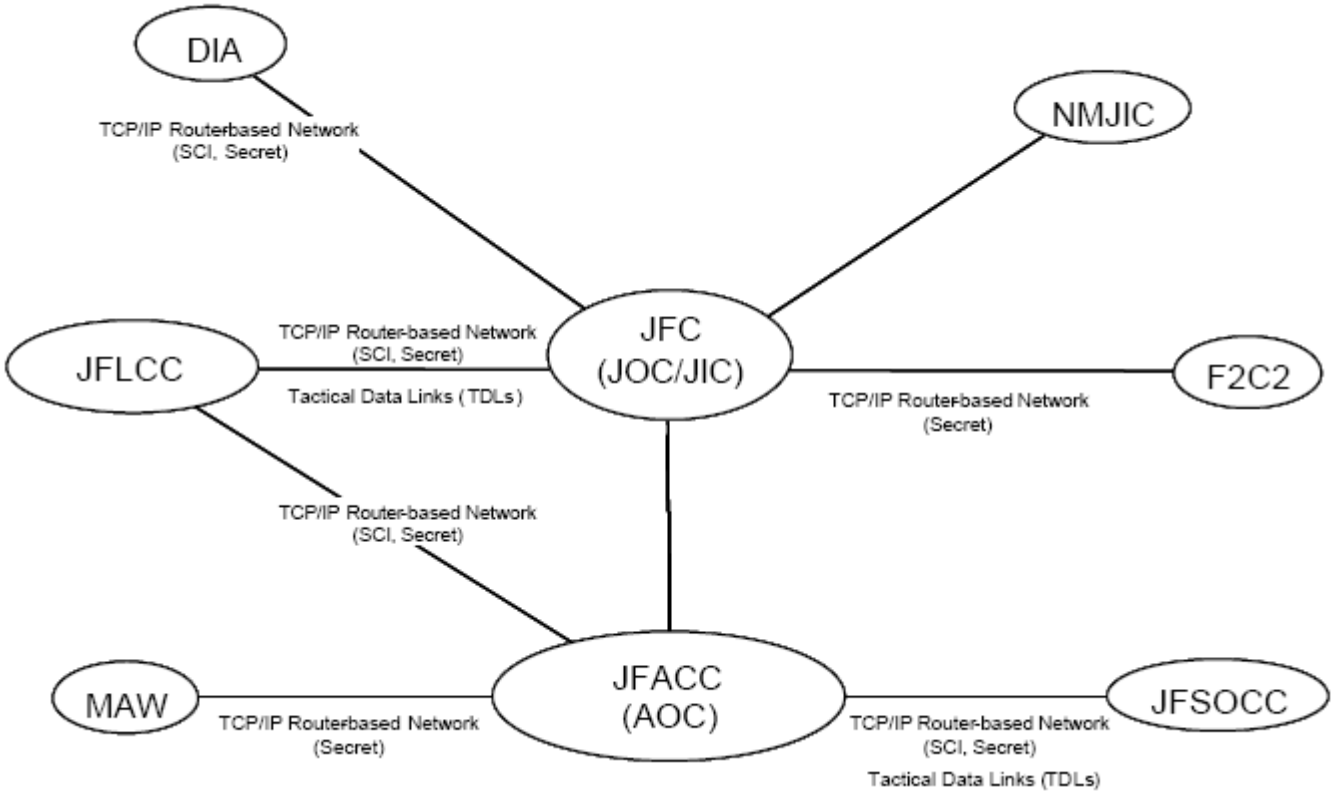
SV-1 Systems Interface Description (3/3)



SV-2 Systems Communications Description (1/2)



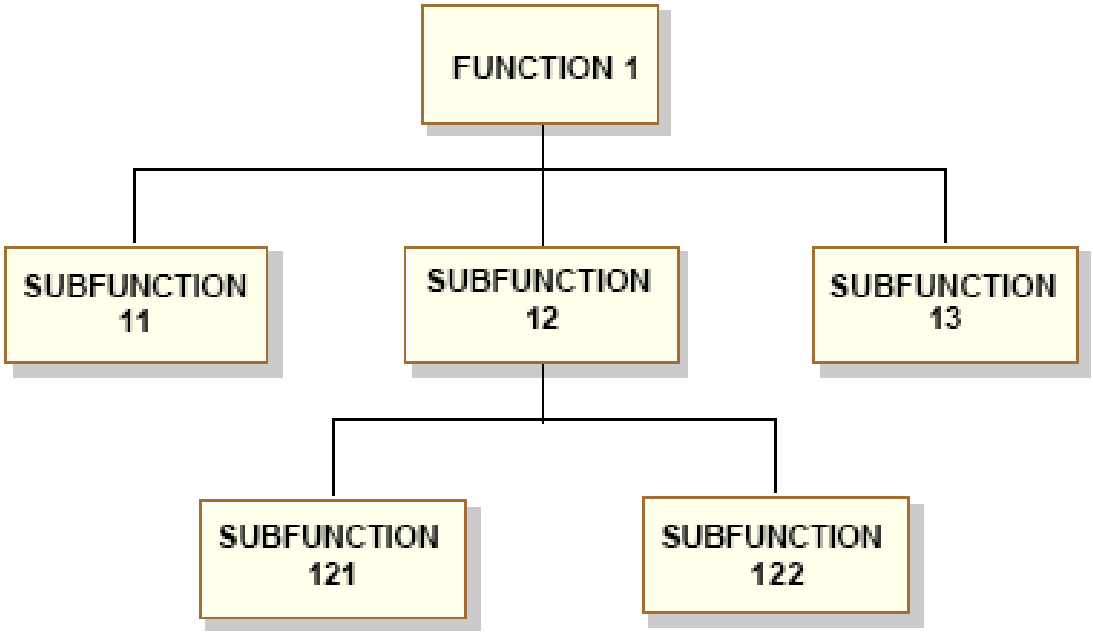
SV-2 Systems Communications Description (2/2)



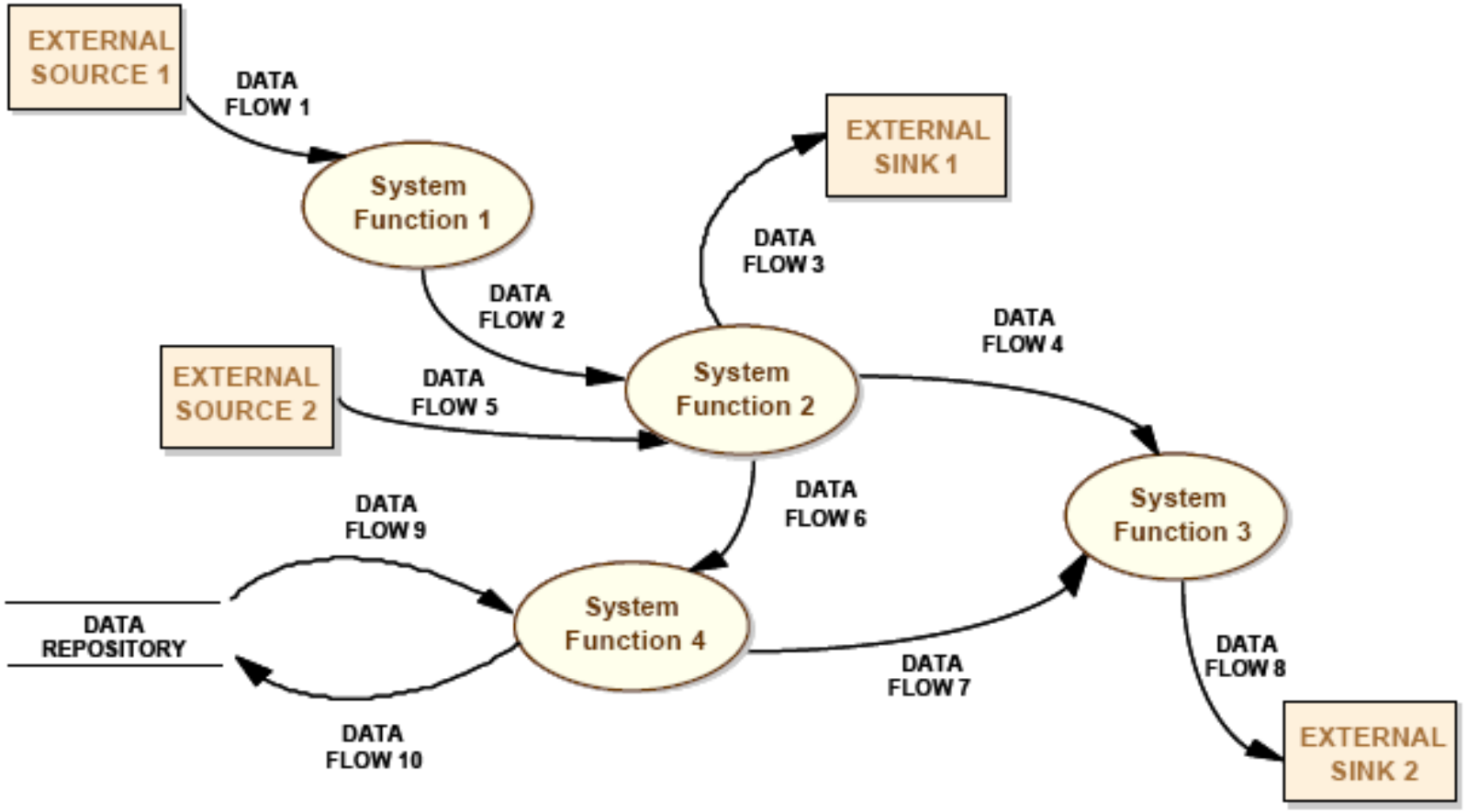
SV-3 Systems-Systems Matrix

	SYSTEM 1	SYSTEM 2	SYSTEM 3	SYSTEM 4	SYSTEM 5	SYSTEM 6	SYSTEM 7	SYSTEM 8	SYSTEM 9	SYSTEM 10
SYSTEM 1		●								
SYSTEM 2	●		●	●	●	●				
SYSTEM 3		●		●	●	●				
SYSTEM 4		●	●		●	●				
SYSTEM 5		●	●	●		●				
SYSTEM 6	●	●	●	●	●		●	●	●	●
SYSTEM 7						●				
SYSTEM 8						●				
SYSTEM 9						●				
SYSTEM 10						●				

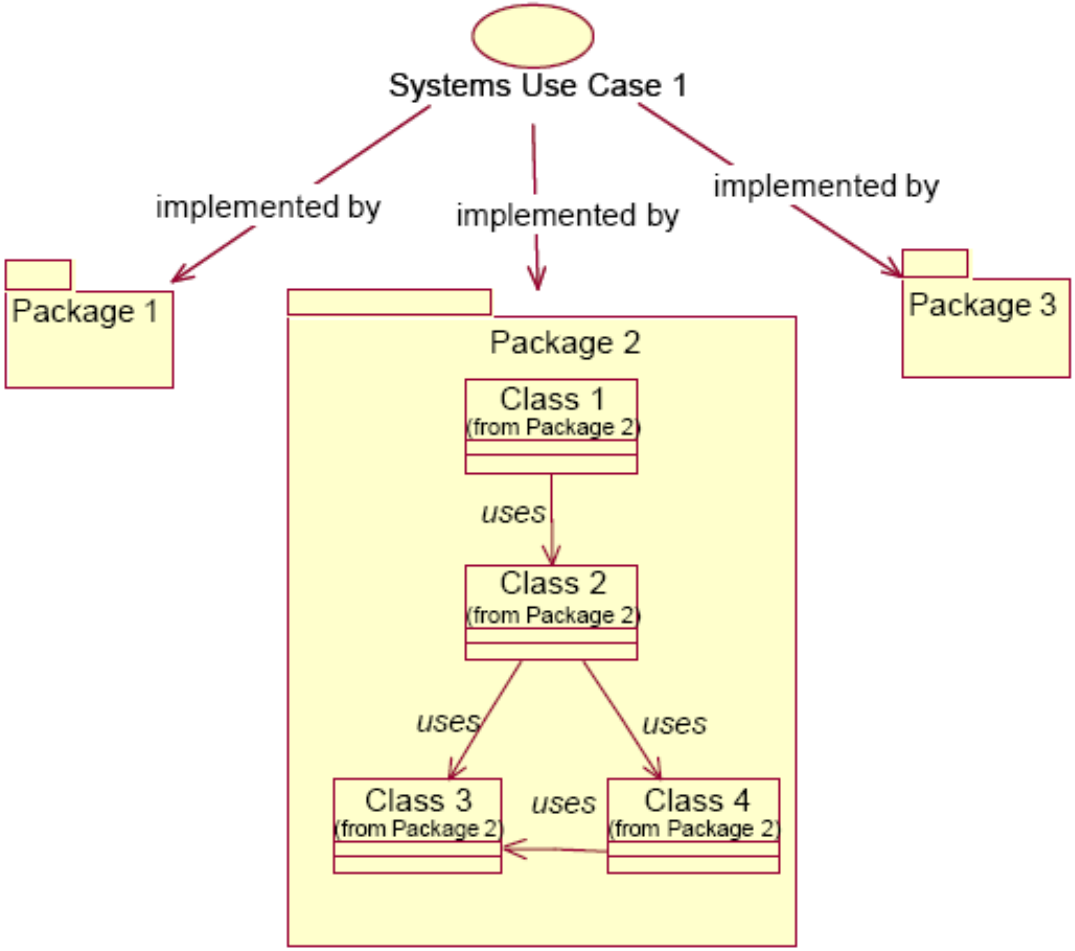
SV-4 Systems Functionality Description (1/3)



SV-4 Systems Functionality Description (2/3)



SV-4 Systems Functionality Description (3/3)



SV-5 Operational Activity to Systems Function Traceability Matrix (1/2)

System Functions	Operational Activities															
	3.11	3.11.3	3.12	3.12.1	3.12.2	3.12.3	3.13	3.14	3.14.1	3.14.2	3.14.3	3.14.4	3.15	3.16	3.17	3.17.1
1	X															
1.1		X														
1.1.1			X													
1.1.1.1	X															
1.1.1.2					X											
1.1.1.3							X									
1.1.2										X						
1.1.2.1				X												
1.1.2.2						X										
1.1.2.3								X								
1.1.3											X					
1.1.3.1													X			
1.1.3.2									X							
1.1.3.3														X		
1.1.3.4														X		

SV-5 Operational Activity to Systems Function Traceability Matrix (2/2)

		Capability 1			Capability 2			Capability 3			
		Operational Activity A	Operational Activity B	Operational Activity C	Operational Activity D	Operational Activity E	Operational Activity F	Operational Activity A	Operational Activity E	Operational Activity G	Operational Activity H
System 1	System Function A	●		●				●		●	
	System Function B					●			●		
	System Function C		●								●
System 2	System Function B				●						
	System Function D						●			●	
	System Function E				●						
	System Function F										●
System 3	System Function G			●							
	System Function H		●				●				
	System Function I			●							

SV-6 Systems Data Exchange Matrix

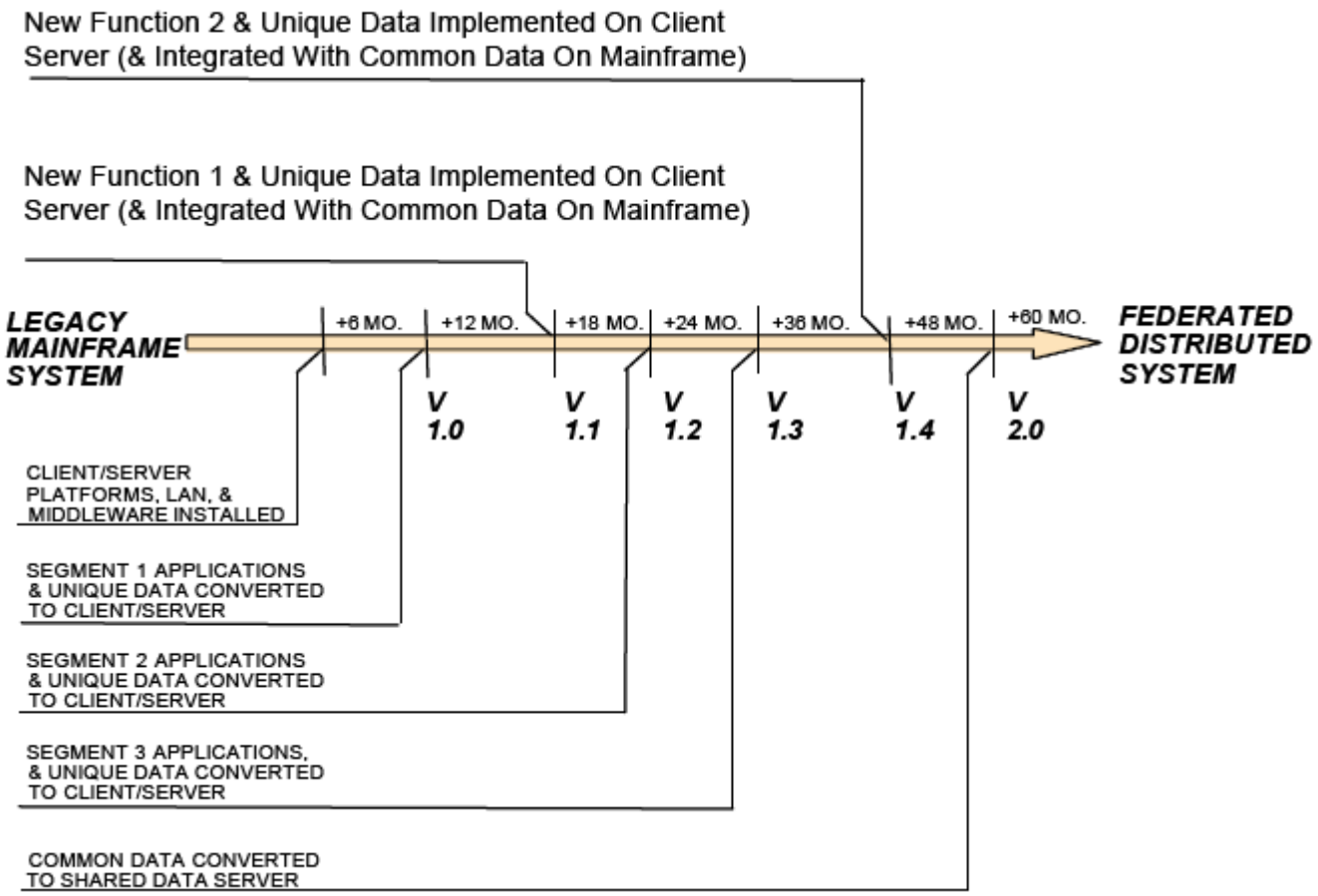
Interface Identifier	Data Exchange Identifier	Data Description						Producer		Consumer		Nature of Transaction			
System Interface Name and Identifier	System Data Exchange Name and Identifier	Data Element Name and Identifier	Content	Format Type	Media Type	Accuracy	Units of Measurement	Data Standard	Sending System Name and Identifier	Sending System Function Name and Identifier	Receiving System Name and Identifier	Receiving System Function Name and Identifier	Transaction Type	Triggering Event	Interoperability Level Achieved	Criticality

Interface Identifier	Data Exchange Identifier	Performance Attributes				Information Assurance						Security				
System Interface Name and Identifier	System Data Exchange Name and Identifier	Periodicity	Timeliness	Throughput	Size	Access Control	Availability	Confidentiality	Dissemination Control	Integrity	Non-Repudiation Producer	Non-Repudiation Consumer	Protection (Type Name, Duration, Date)	Classification	Classification Caveat	Releasability	Security Standard

SV-7 Systems Performance Parameters Matrix

	Performance Range (Threshold and Objective) Measures		
	$Time_0$ (Baseline Architecture Time Period)	**$Time_1$**	**$Time_n$ (Target Architecture Time Period)**
System Name			
Hardware Element 1			
Maintainability			
Availability			
System Initialization Time			
Architecture data Transfer Rate			
Program Restart Time			
S/W Element 1 / H/W Element 1			
Architecture Data Capacity (e.g., throughput or # of input types)			
Automatic Processing Responses (by input type, # processed/unit time)			
Operator Interaction Response Times (by type)			
Availability			
Effectiveness			
Mean Time Between S/W Failures			
Organic Training			
S/W Element 2 / H/W Element 1			
Hardware Element 2			

SV-8 Systems Evolution Description



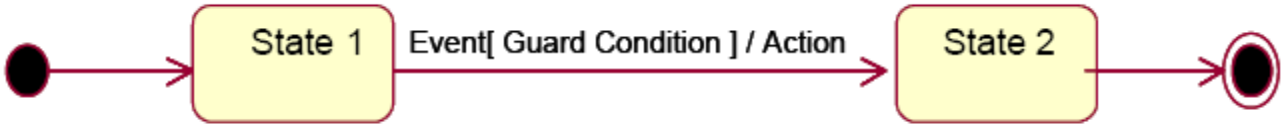
SV-9 Systems Technology Forecast

JTA Service	TECHNOLOGY FORECASTS		
	SHORT TERM (0-6 Months)	MID TERM (6-12 Months)	LONG TERM (12-18 Months)
Application Software			
Support Applications	Microsoft Office 2000 available (for Windows 2000)	Microsoft Office 2000 stable enough for full-scale implementation	Microsoft Office available for Linux E-mail on wireless PDAs commonplace
Application Platform			
Data Management	Oracle 9i available MySQL (Open Source DBMS) available		
Operating System		Next MS Windows desktop upgrade expected Next Red Hat Linux major release expected	Next MS Windows server upgrade expected
Physical Environment			Intel IA-64 becomes standard processor for desktops Initial use of quantum computing technologies
External Environment			
User Interface		Thin screen CRT monitors for PC desktops become price competitive	Thin screen LED monitors become price competitive for desktops Conventional CRT technology monitors for desktops become obsolete
Persistent Storage	5G PCMCIA type 2 card available		Disk storage capacity doubles again
Communications Networks		Cable modem service available for most telecommuting staff	Fiber optic connections available for most telecommuting staff

SV-10a Systems Rules Model

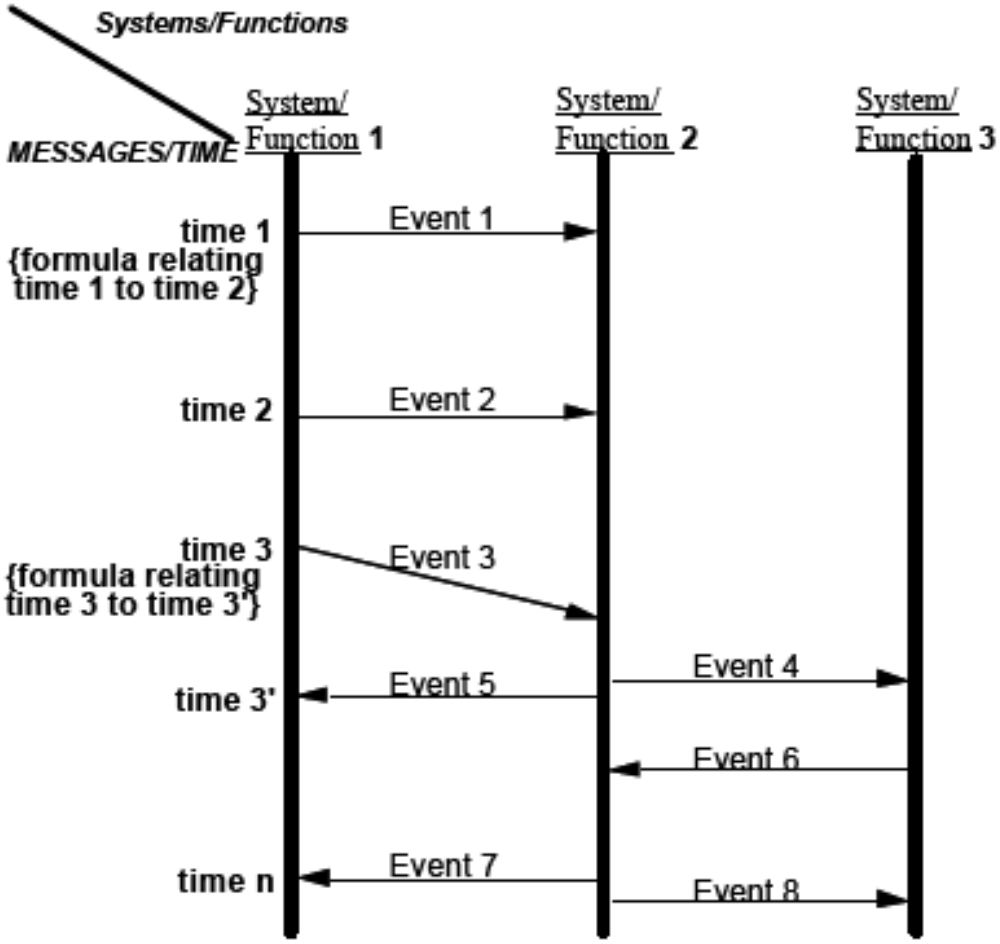
If field A *in* FORM-X *is set to* value T,
  Then field B *in* FORM-Y *must be set to* value T
  And field C *in* FORM-Z *must be set to* value T
End If

SV-10b Systems State Transition Description



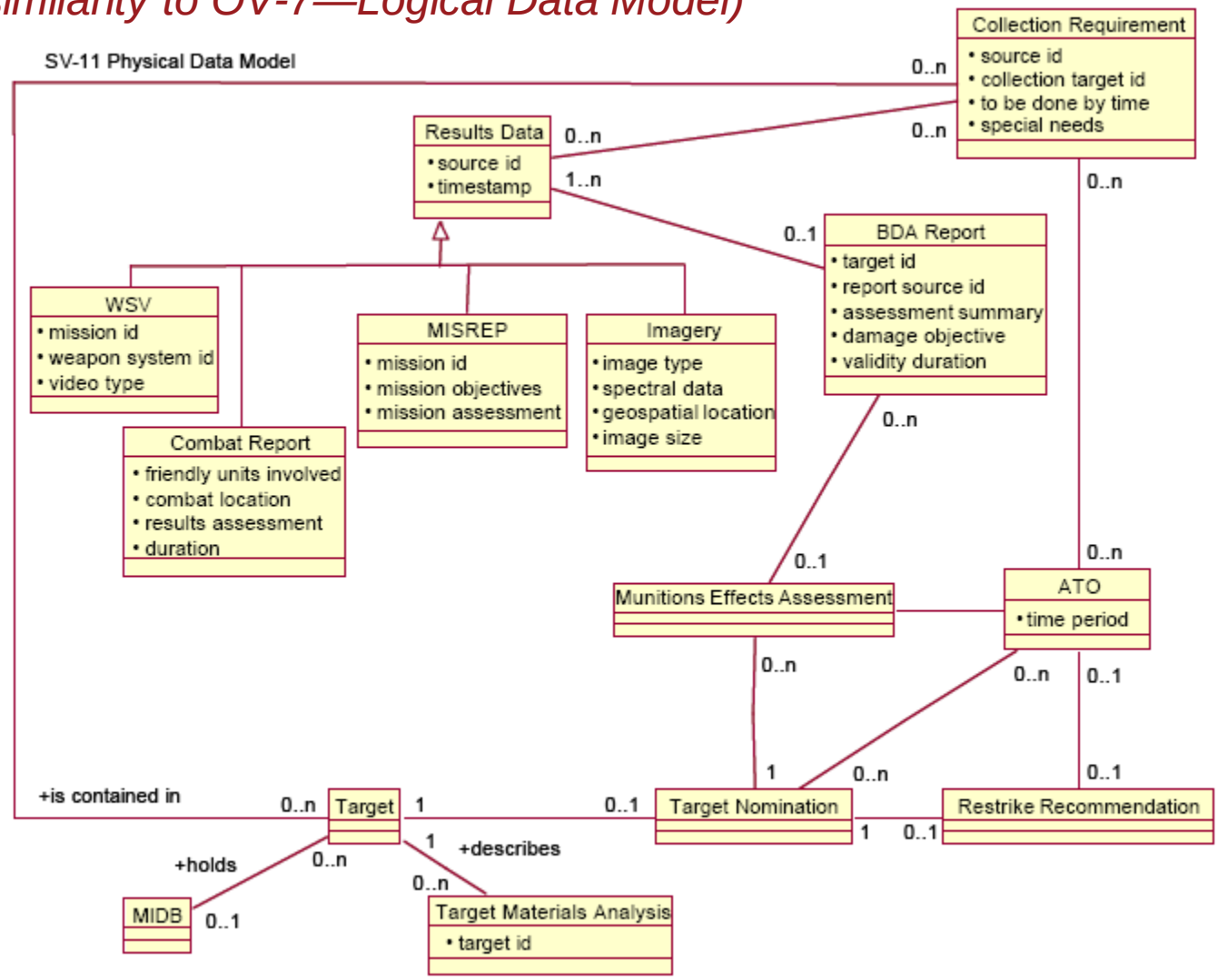
SV-10c Systems Event-Trace Description

(note similarity to OV-6c—Operational Event-Trace Description)



SV-11 Physical Schema

(note similarity to OV-7—Logical Data Model)



Technical Views

- **TV-1 Technical Standards Profile**
  - **Listing of standards that apply to Systems View elements in a given architecture**
- **TV-2 Technical Standards Forecast**
  - **Description of emerging standards and potential impact on current Systems View elements, within a set of time frames**

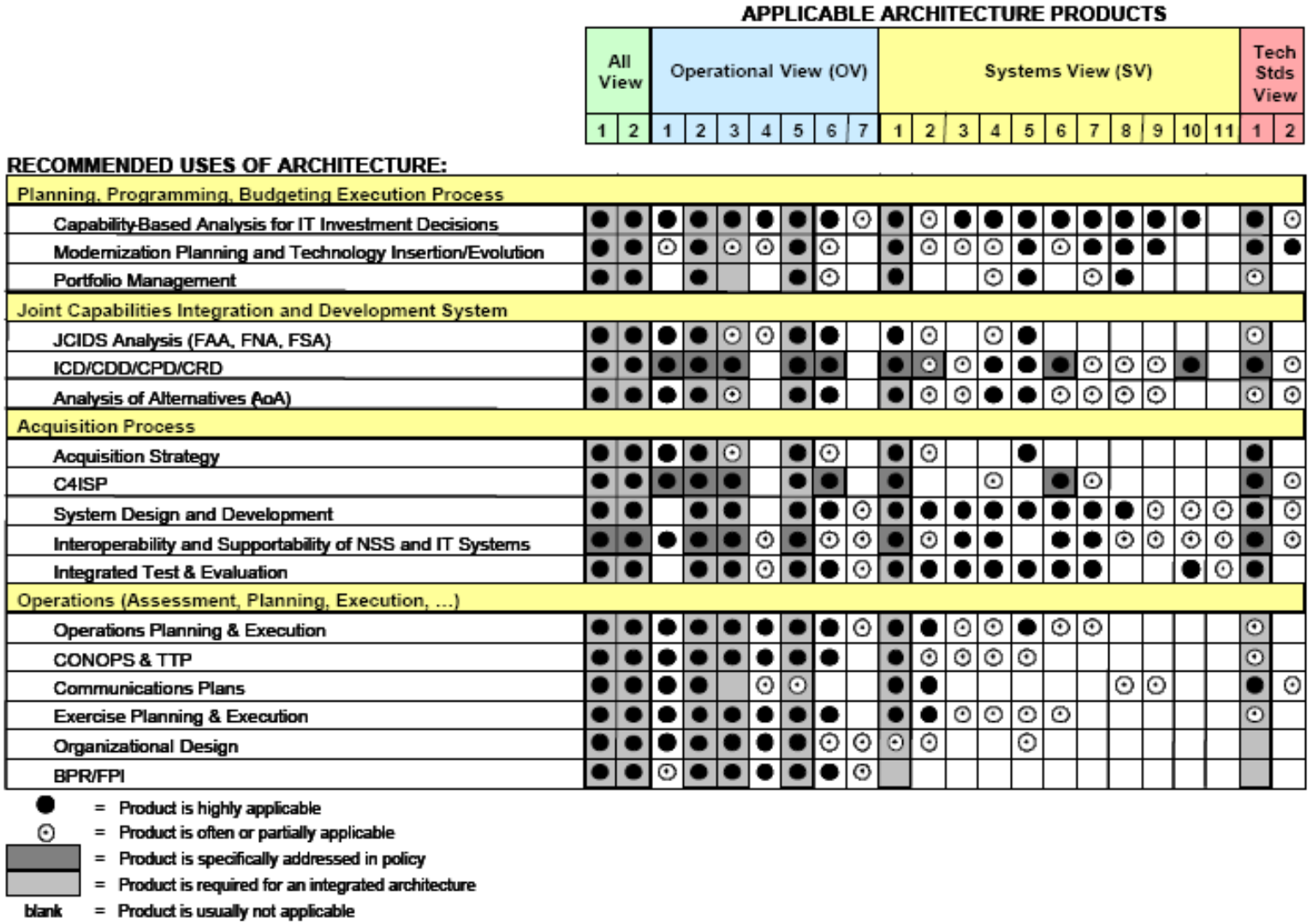
TV-1 Technical Standards Profile

JTA Service Area	Service	JTA Standard and Source Document
Information-Processing Standards	Higher Order Languages	
	Software Life-Cycle Process	
	Geospatial Data Interchange	
	Motion Imagery Data Interchange - Video	
	Distributed-Object Computing	
Information-Transfer Standards	Data Flow Network	
	Command and Control Information (C2I) Network	
	Physical Layer	
	Network Interface	
	Layer Management	
	File Transfer Standards	
	Remote Terminal Standards	
	Network Time Synchronization Standards	
	Web Services Standards	
	Connectionless Data Transfer	
	Transport Services Standards	
Information Modeling, Metadata, and Information Exchange Standards	Activity Modeling	
	Data Modeling	
	Object-Oriented Modeling	
Human Computer Interface	Mandates	
Information Security / Information Infrastructure Standards	Password Security	
	Application Software Entity Security Standards	
	Virtual Private Network Service	
	Intrusion Detection Service	
	Human-Computer Interface Security Standards	

TV-2 Technical Standards Forecast

- **The Technical Standards Forecast contains expected changes in technology-related standards and conventions, which are documented in the TV-1 product**
- **The forecast for evolutionary changes in the standards should be correlated against the time periods as mentioned in the SV-8 and SV-9 products.**

Recommended Uses



RECOMMENDED USES OF ARCHITECTURE:	APPLICABLE ARCHITECTURE PRODUCTS																							
	All View		Operational View (OV)							Systems View (SV)												Tech Stds View	
	1	2	1	2	3	4	5	6	7	1	2	3	4	5	6	7	8	9	10	11		1	2
Planning, Programming, Budgeting Execution Process																								
Capability-Based Analysis for IT Investment Decisions	●	●	●	●	●	●	●	●	⊙	●	⊙	●	●	●	●	●	●	●	●		●	⊙	
Modernization Planning and Technology Insertion/Evolution	●	●	⊙	●	⊙	⊙	●	⊙		●	⊙	⊙	⊙	●	⊙	●	●	●			●	●	
Portfolio Management	●	●		●			●	⊙		●			⊙	●		⊙	●				⊙		
Joint Capabilities Integration and Development System																								
JCIDS Analysis (FAA, FNA, FSA)	●	●	●	●	⊙	⊙	●	●		●	⊙		⊙	●								⊙		
ICD/CDD/CPD/CRD	●	●	●	●	●		●	●		●	⊙	⊙	●	●	●	⊙	⊙	⊙	●		●	⊙	
Analysis of Alternatives (AoA)	●	●	●	●	⊙		●	●		●	⊙	⊙	●	●	⊙	⊙	⊙	⊙			⊙	⊙	
Acquisition Process																								
Acquisition Strategy	●	●	●	●	⊙		●	⊙		●	⊙			●								●		
C4ISP	●	●	●	●	●		●	●		●			⊙		●	⊙					●	⊙	
System Design and Development	●	●		●	●		●	●	⊙	●	●	●	●	●	●	●	●	⊙	⊙	⊙	●	⊙	
Interoperability and Supportability of NSS and IT Systems	●	●	●	●	●	⊙	●	⊙	⊙	●	⊙	●	●		●	●	⊙	⊙	⊙	⊙	●	⊙	
Integrated Test & Evaluation	●	●		●	●	⊙	●	●	⊙	●	●	●	●	●	●	●			●	⊙	●		
Operations (Assessment, Planning, Execution, ...)																								
Operations Planning & Execution	●	●	●	●	●	●	●	●	⊙	●	●	⊙	⊙	●	⊙	⊙						⊙		
CONOPS & TTP	●	●	●	●	●	●	●	●		●	⊙	⊙	⊙	⊙								⊙		
Communications Plans	●	●	●	●		⊙	⊙			●	●						⊙	⊙			●	⊙		
Exercise Planning & Execution	●	●	●	●	●	●	●	●		●	●	⊙	⊙	⊙	⊙						⊙			
Organizational Design	●	●	●	●	●	●	●	⊙	⊙	⊙	⊙			⊙										
BPR/FPI	●	●	⊙	●	●	●	●	●	⊙															

● = Product is highly applicable

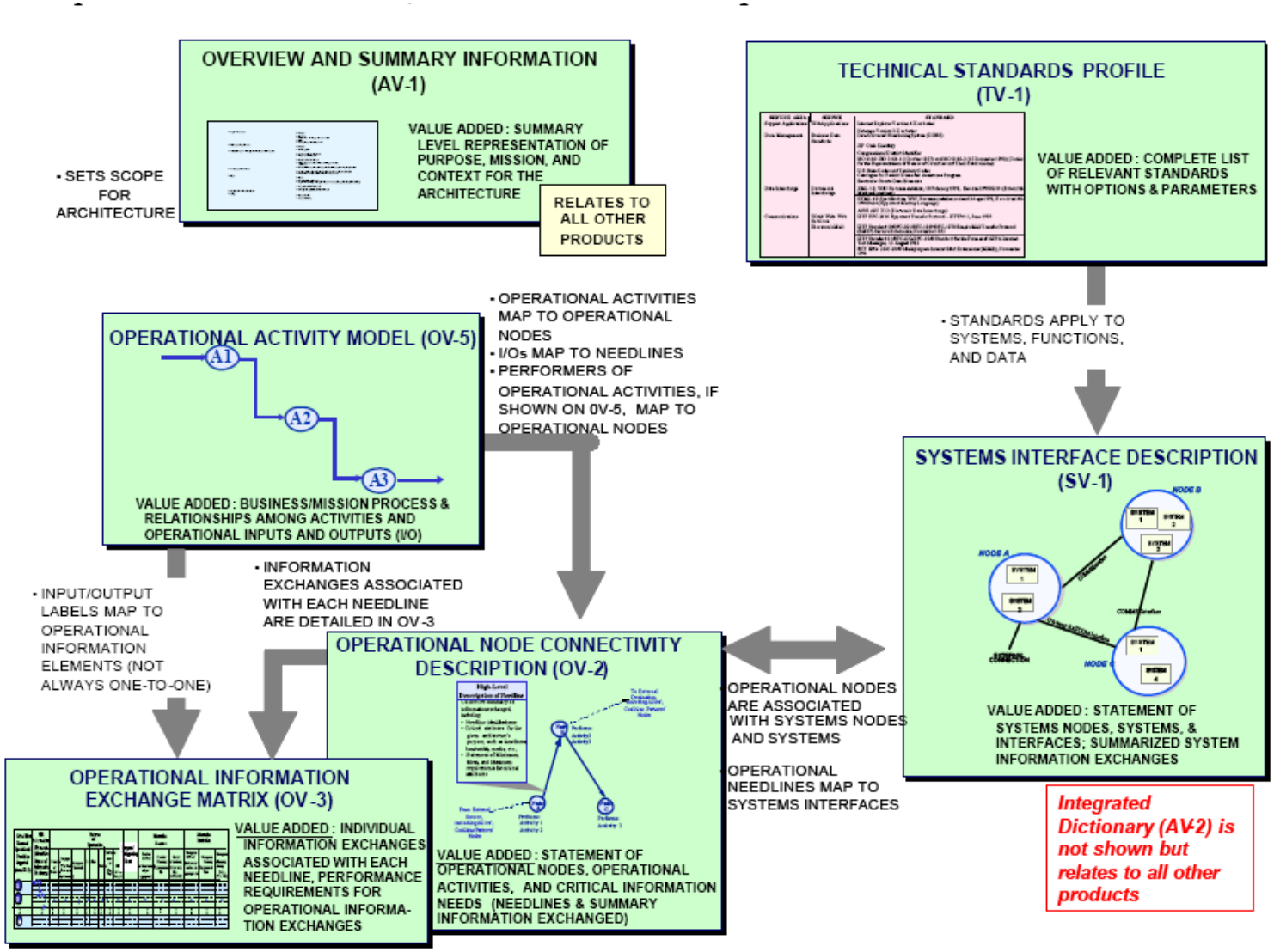
⊙ = Product is often or partially applicable

= Product is specifically addressed in policy

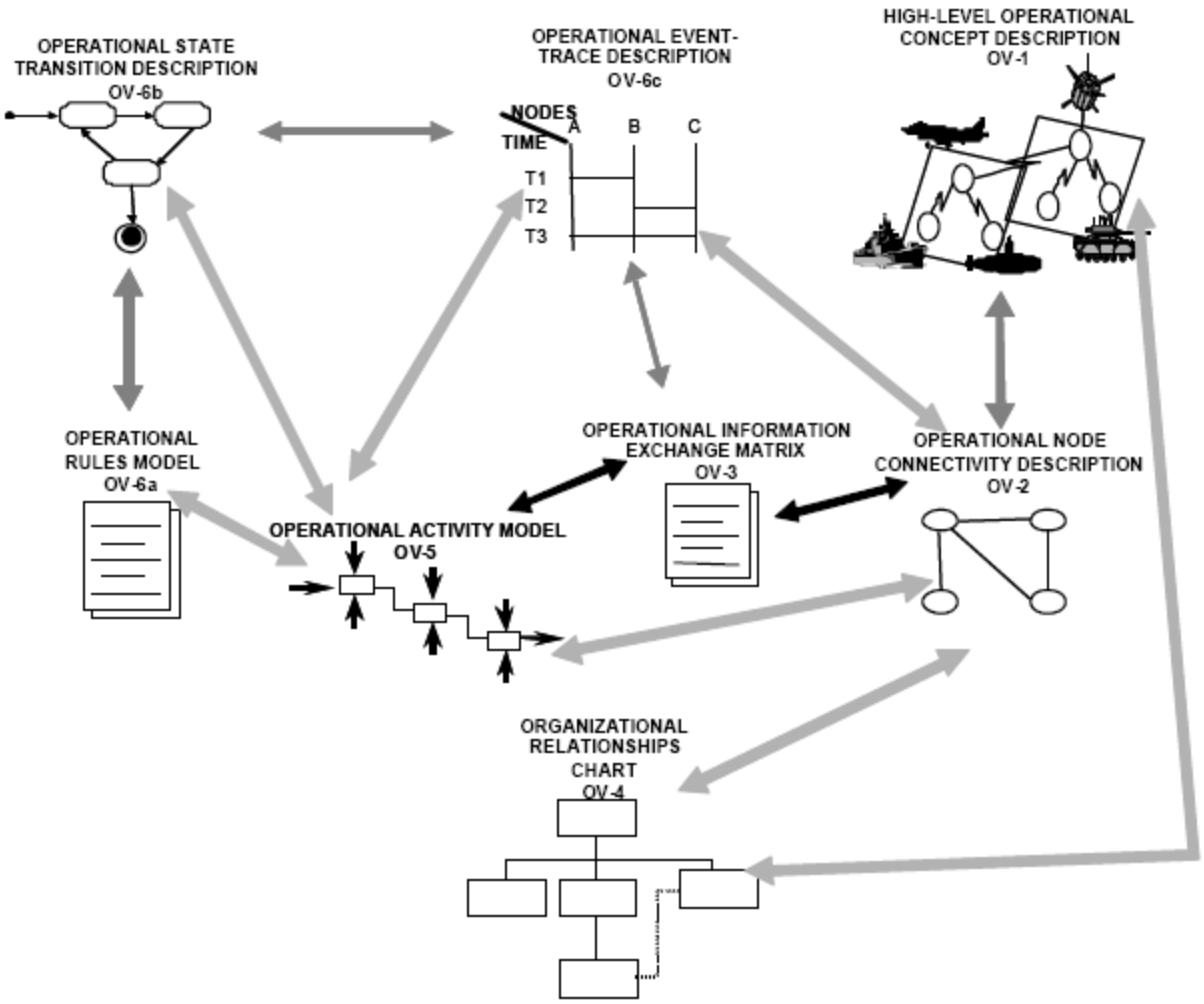
= Product is required for an integrated architecture

blank = Product is usually not applicable

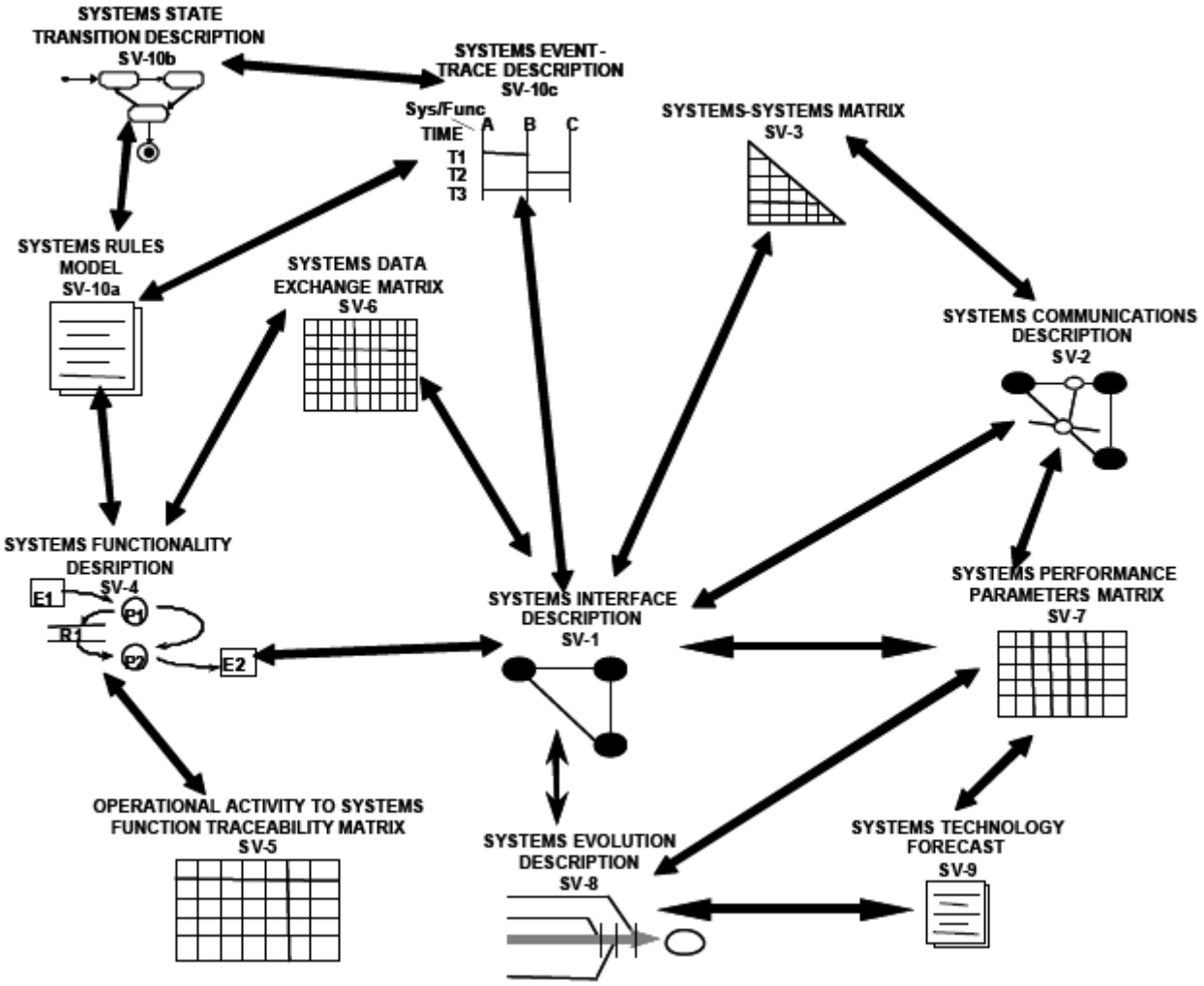
Relationship Among Products



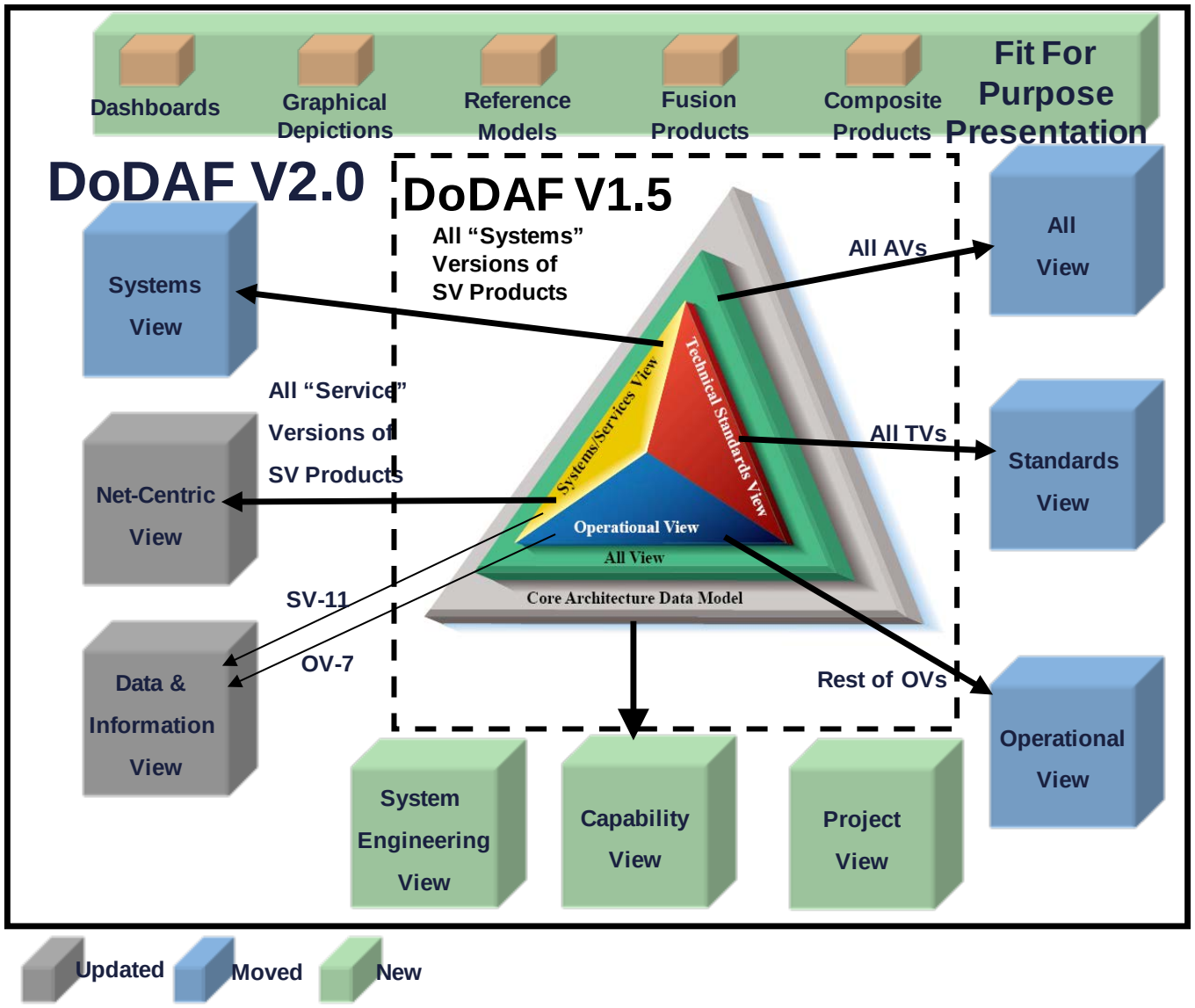
Operational Cross-View Relationships



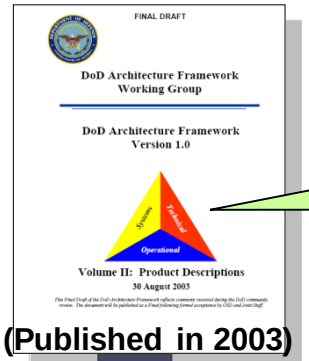
System Cross-View Relationships



Evolution of DoDAF

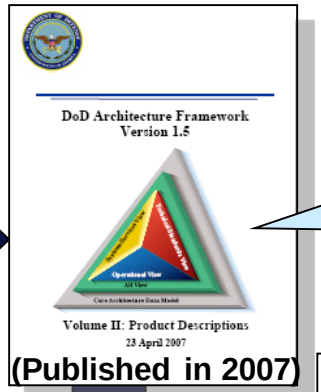


DoDAF Evolution To Support Agile Architecture Environment



(Published in 2003)

DoDAF 1.0
- Program – Level Focus
- Volume III Was A Desk book
- CADM Was Separate
- Baseline For DoDAF 1.5



(Published in 2007)

DoDAF 1.5
- Began to address Net-Centricity
- Program – Level Focus
- Volume III is CADM & Architecture Data Strategy
- Addressed Architecture Federation

Potential Interim Release Items

- Conceptual Data Model
- Guidance on Notation
- Best Practices
- Templates
- Capability Description

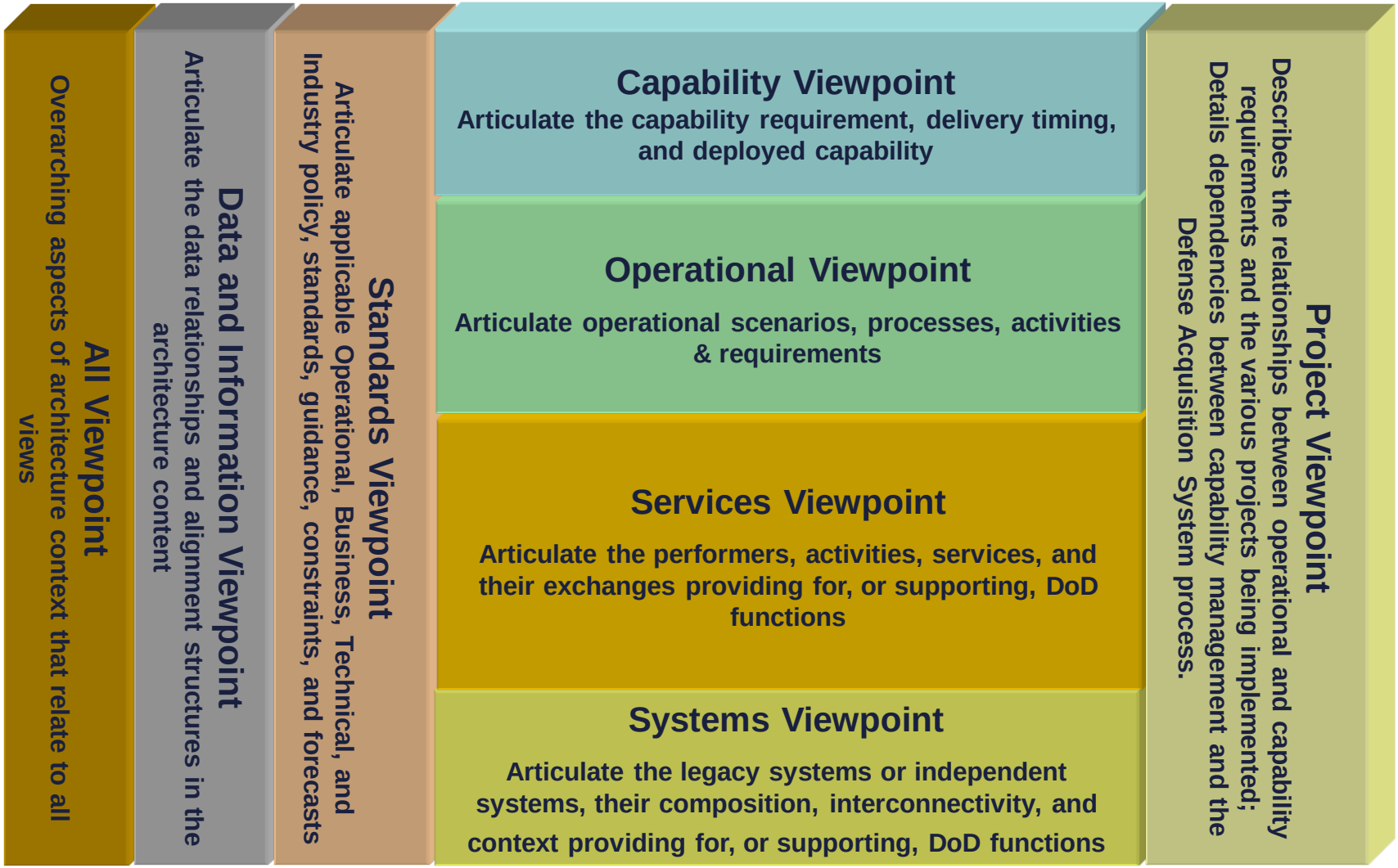
DoDAF



2.0
(Late 2008)

DoDAF 2.0
- Covers Enterprise through Program Architecture Spectrum
- Documents High Level Guidance
- Web-based
- On-Line Journal
  - Best Practices
  - Errata Sheets
  - Interim Releases
  - New Requirements
  - Usage Examples

DoDAF 2.0 Viewpoint



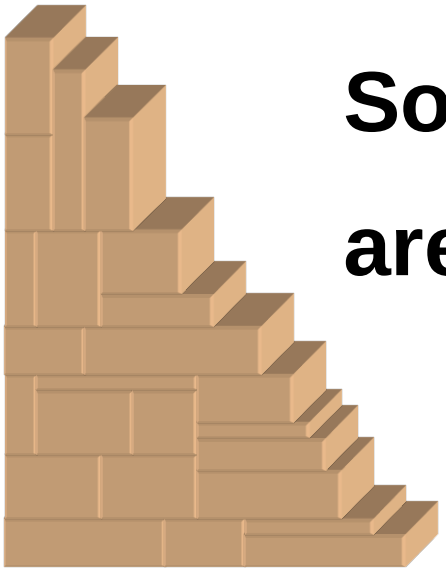
Outline

- DoDAF (1.0)
- **TOGAF**
- Zachman
- FEAF

The Open Group Architecture Forum

- **The forum in which TOGAF has evolved**
- **A worldwide forum for: customers, tool vendors, solution vendors, integrators, academic & research organizations involved with architecture**
- **Exchanging information, experience, and requirements**
- **Access to current work in progress**
- **Shaping the market for architecture methods, tools and services**

Every enterprise has an IT architecture

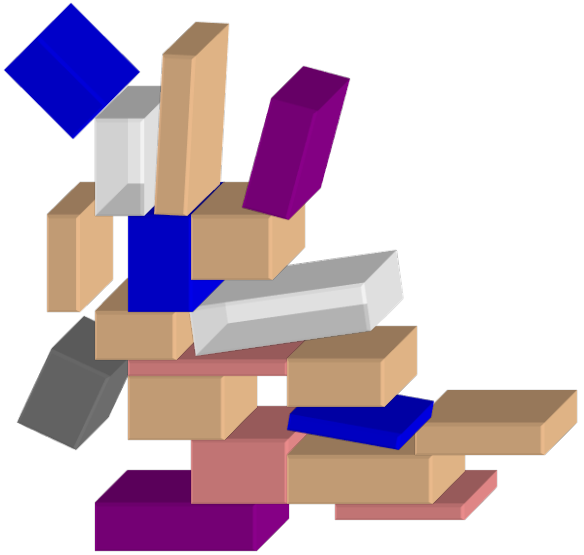


Some

are designed

and some

just happen



A Good IT Architecture Will

- Directly address the needs of the enterprise
- React to change at the rate dictated by the enterprise's markets
- Be understood and supported by senior management
- Clearly define the structure of the existing system
- Provide a roadmap and migration strategy for future purchases / developments
- Reduce the number and complexity of the interfaces between components, improving the ease of:
  - Application portability
  - Component upgrade
  - Component exchange
  - Component development and maintenance

Where Do You Start?

- There are many successful IT architectures in use
- Can you benefit from that accumulation of knowledge?
- Can you examine in detail the IT architecture of another organization?
- Do you have to reinvent the wheel?
- What reference material is available?

- This is why TOGAF exists and is freely available

TOGAF Origins and Motivations

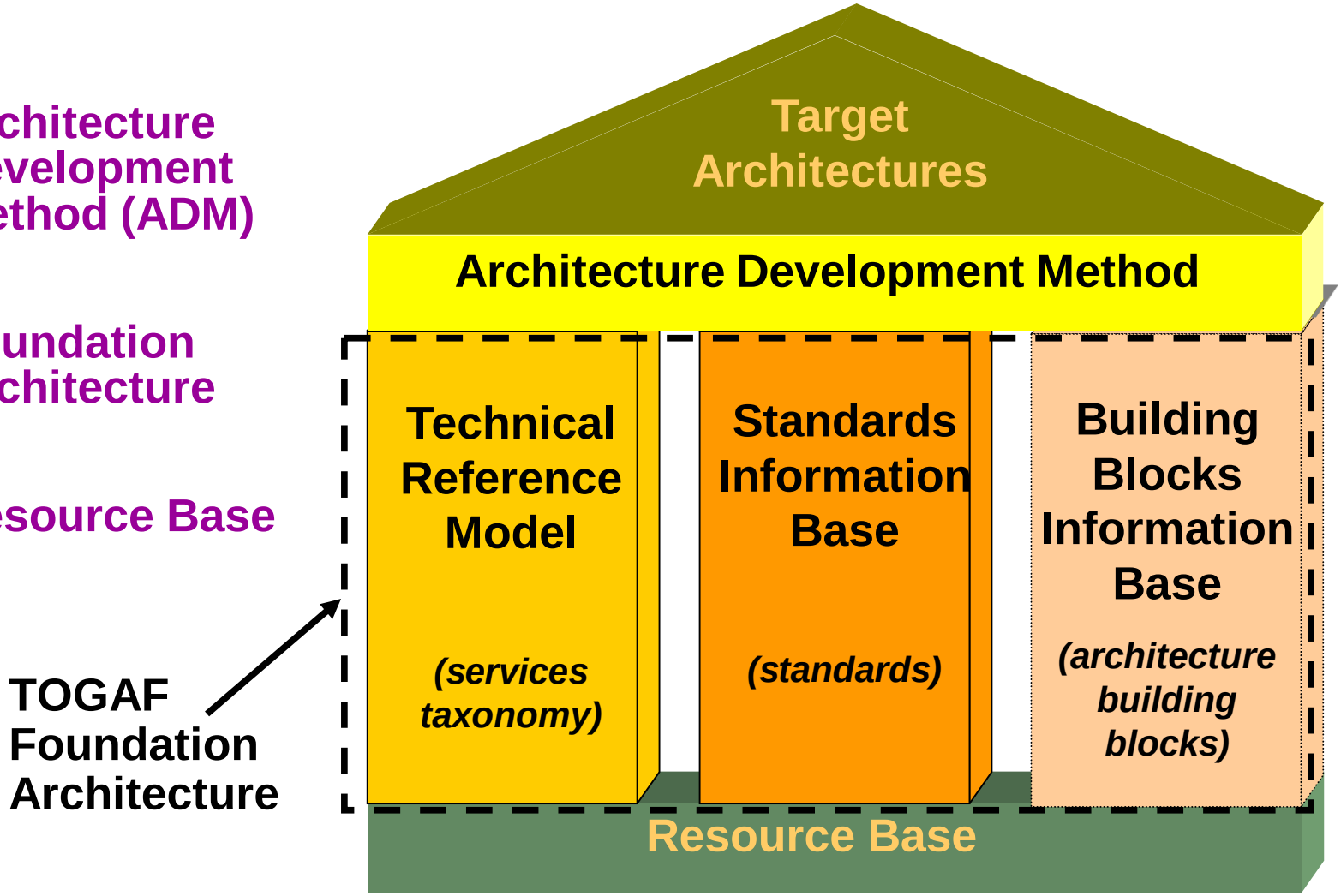
- **A customer initiative:**
  - **Formal user requirement developed 1994**
  - **Main themes:**
    - **A single, unifying Architectural Framework for the IT industry**
    - **A framework for developing architectures to meet specific business needs**
      - **Not a "one-size-fits-all" architecture**

TOGAF Today

- **Industry consensus**
- **Technology- and tool-neutral**
- **>10 years continuous development**
- **Proven in practice**
- **Publicly available:**
  - **http://www.opengroup.org**

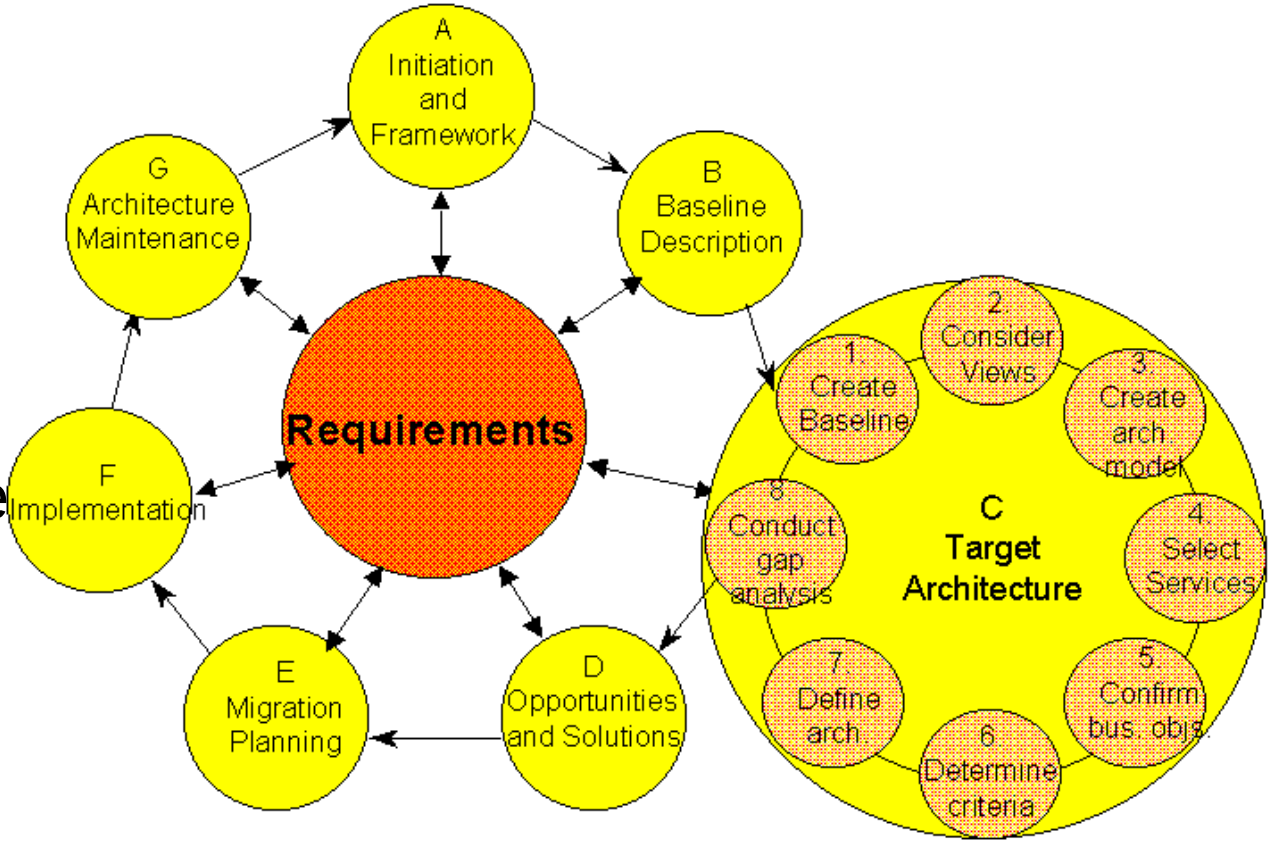
TOGAF Structure and Components

- Architecture Development Method (ADM)
- Foundation Architecture
- Resource Base

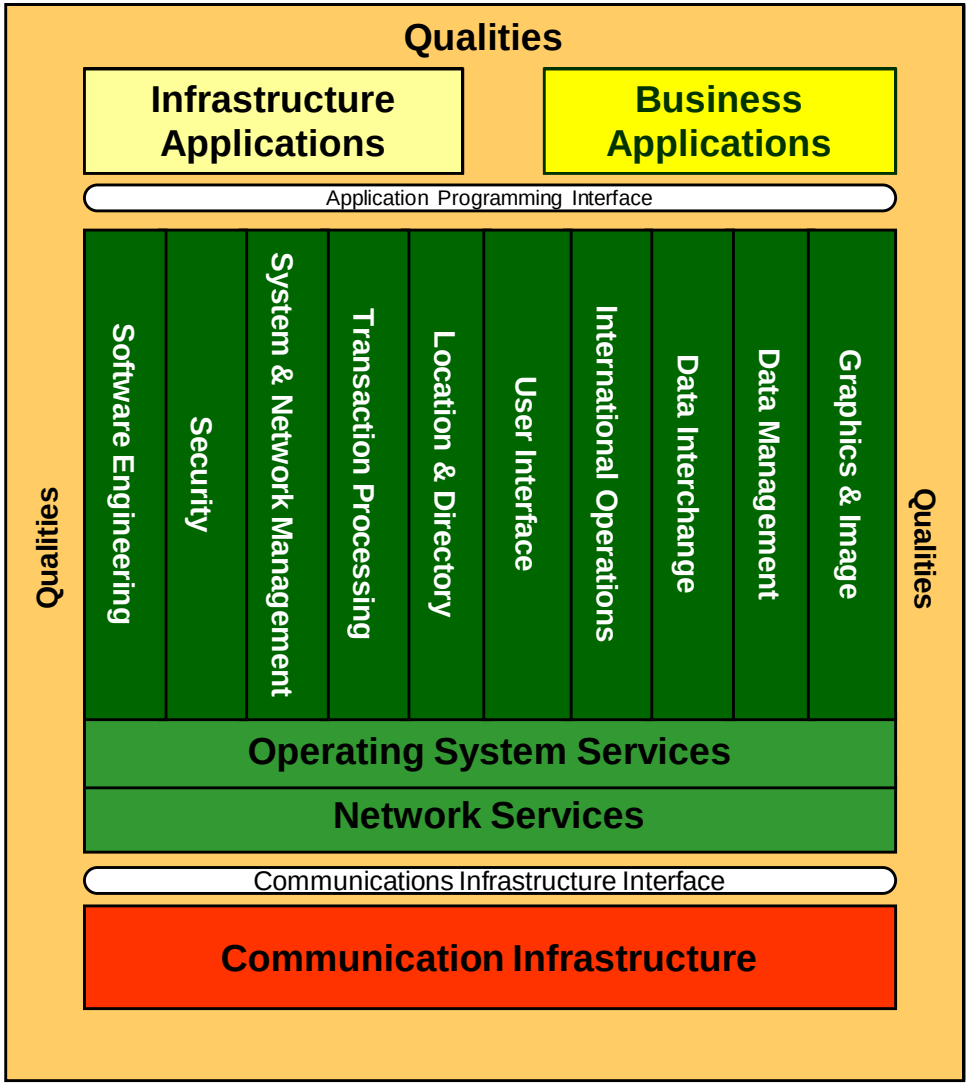


Architecture Development Method (ADM)

- Open, industry consensus method for IT architecture
- Quick-start foundation
- Practical, experience based guidance
- Requires continual validation against requirements



Foundation Architecture: Technical Reference Model (TRM)



- Associated with detailed taxonomy of services
  - Defines scope of each service category
- Identifies system-wide capabilities ("qualities"), e.g.:
  - Internationalization
  - Security
  - Management

Foundation Architecture: Standards Information Base (SIB)

- A database of open industry standards
  - The complete set of Open Group endorsed standards
  - Content determined by Open Group consensus process
- Structured according to TOGAF Technical Reference Model taxonomy
- Regularly updated
- Available for public web access
- Gateway to many linked resources

Resource Base

- **Resources available in applying the TOGAF Architecture Development Method; e.g.**
  - **Architecture Development Markup Language (ADML)**
  - **Architecture Compliance Reviews**
  - **Architecture Principles**
  - **Architecture Views**
  - **Business Scenarios (requirements method)**
  - **Case Studies**
  - **IT Governance Strategies**

Now Included in TOGAF

- **ANSI/IEEE Std 1471-2000, Recommended Practice for Architectural Description**
  - **Extensive rewrite of Architecture Views section, integrating ANSI/IEEE Std 1471 concepts / terminology**
  - **Integration with TOGAF Architecture Development Method**
- **Architecture Compliance Reviews**
  - **Ensuring conformance with architecture in projects - new section based on Boeing work**
- **Architecture Patterns**
  - **New section based on IBM, US Treasury work**
  - **Distinguish from design patterns**

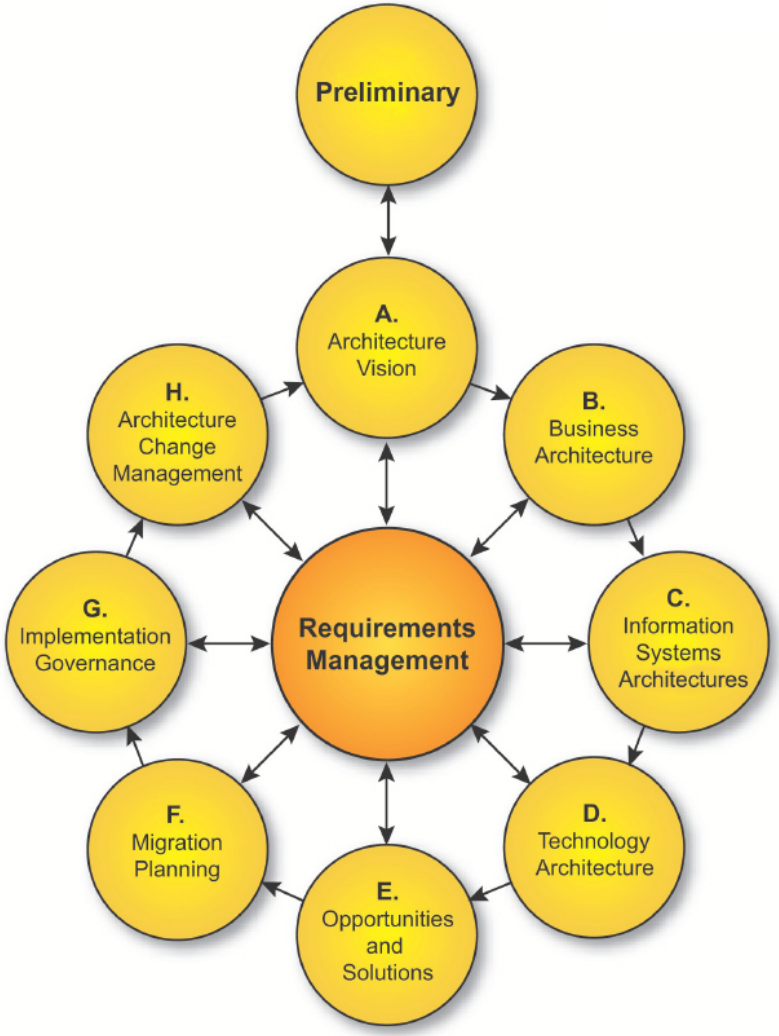
Now Included in TOGAF (concluded)

- **Guidelines for tailoring the TOGAF Architecture Development Method**

- **TOGAF comparison of with other frameworks:**
  - **Federal Enterprise Architecture Framework (FEAF)**
  - **Treasury Enterprise Architecture Framework (TEAF)**
  - **Spewak Enterprise Architecture Planning (EAP)**
  - **Zachman Framework**

- **TOGAF in the Enterprise**
  - **TOGAF positioning relative to enterprise architecture**

TOGAF Version 9 – Enterprise Edition



Preliminary Phase	Prepare the organization for successful TOGAF architecture projects. Undertake the preparation and initiation activities required to meet the Business directive for a new enterprise architecture, including the definition of an Organization-Specific Architecture framework and tools, and the definition of architecture principles.
Requirements Management	Ensure that every stage of a TOGAF project is based on and validates business requirements.
Phase A: **Architecture Vision**	Set the scope, constraints, and expectations for a TOGAF project. Create the Architecture Vision. Define stakeholders. Validate the business context and create the Statement of Architecture Work. Obtain approvals.
Phase B: **Business Architecture** **Phase C:** **Information Systems Architectures** **Phase D:** **Technology Architecture**	Develop architectures at three levels: 1. Business 2. Information Systems 3. Technology In each case, develop the Baseline and Target Architecture and analyze gaps.
Phase E: **Opportunities and Solutions**	Perform initial implementation planning and the identification of delivery vehicles for the building blocks identified in the previous phases. Identify major implementation projects and group them into Transition Architectures.
Phase F: **Migration Planning**	Analyze cost benefits and risk. Develop detailed Implementation and Migration Plan.
Phase G: **Implementation Governance**	Provide architectural oversight for the implementation. Prepare and issue Architecture Contracts (Implementation Governance Board). Ensure that the implementation project conforms to the architecture.
Phase H: **Architecture Change Management**	Provide continual monitoring and a change management process to ensure that the architecture responds to the needs of the enterprise, and maximizes the business value.

http://www.opengroup.org/togaf/

TOGAF Version 9 – Enterprise Edition

TOGAF 9 is divided into seven sections as follows:

PART I: Introduction
A high-level introduction to the key concepts behind enterprise architecture and in particular the TOGAF approach. It contains the definitions of terms used throughout TOGAF and release notes detailing the changes between this version and the previous version of TOGAF.

PART II: Architecture Development Method
The core of TOGAF. It describes the TOGAF Architecture Development Method (ADM) – a step-by-step approach to developing an enterprise architecture.

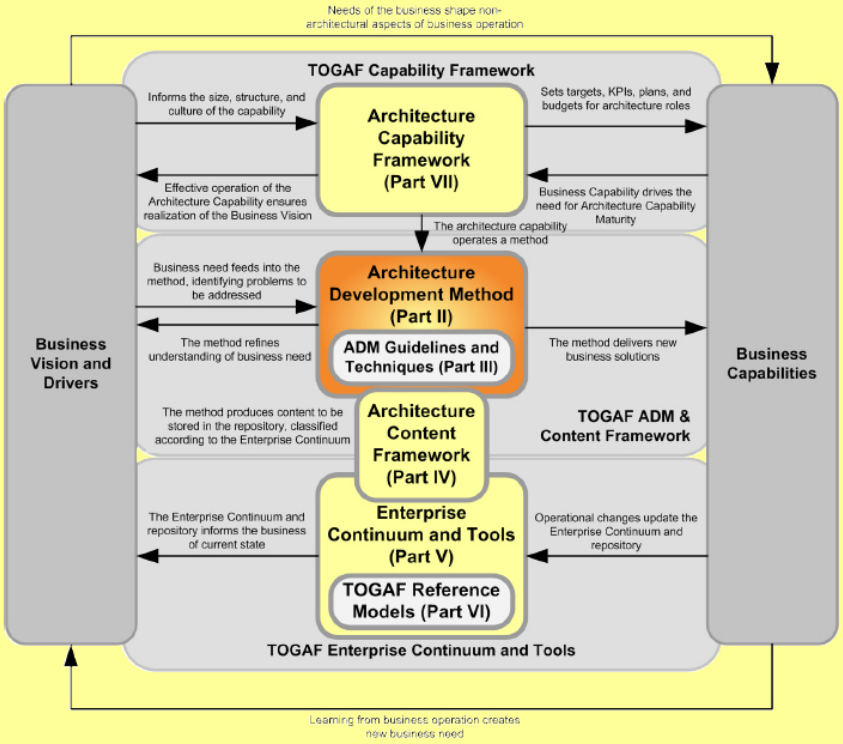
PART III: ADM Guidelines and Techniques
A collection of guidelines and techniques available for use in applying TOGAF and the TOGAF ADM.

PART IV: Architecture Content Framework
The TOGAF content framework, including a structured metamodel for architectural artifacts, the use of re-usable architecture building blocks, and an overview of typical architecture deliverables.

PART V: Enterprise Continuum & Tools
Taxonomies and tools to categorize and store the outputs of architecture activity within an enterprise.

PART VI: TOGAF Reference Models
A selection of architectural reference models, which includes the TOGAF Foundation Architecture, and the Integrated Information Infrastructure Reference Model (III-RM).

PART VII: Architecture Capability Framework
The organization, processes, skills, roles, and responsibilities required to establish and operate an architecture function within an enterprise.



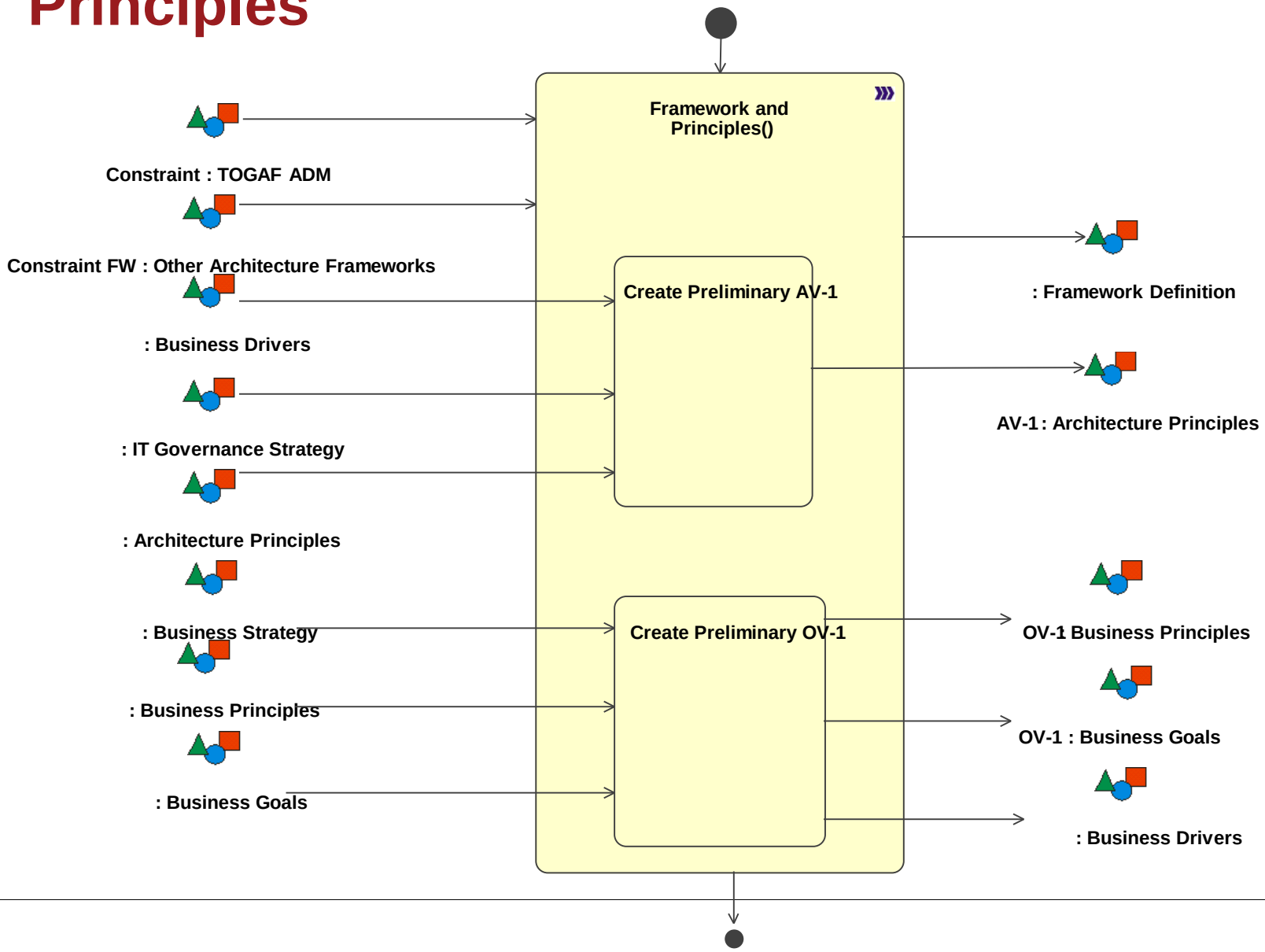
TOGAF Architecture Development Method (ADM) with DoDAF Views

ADM Architecture Requirements Management

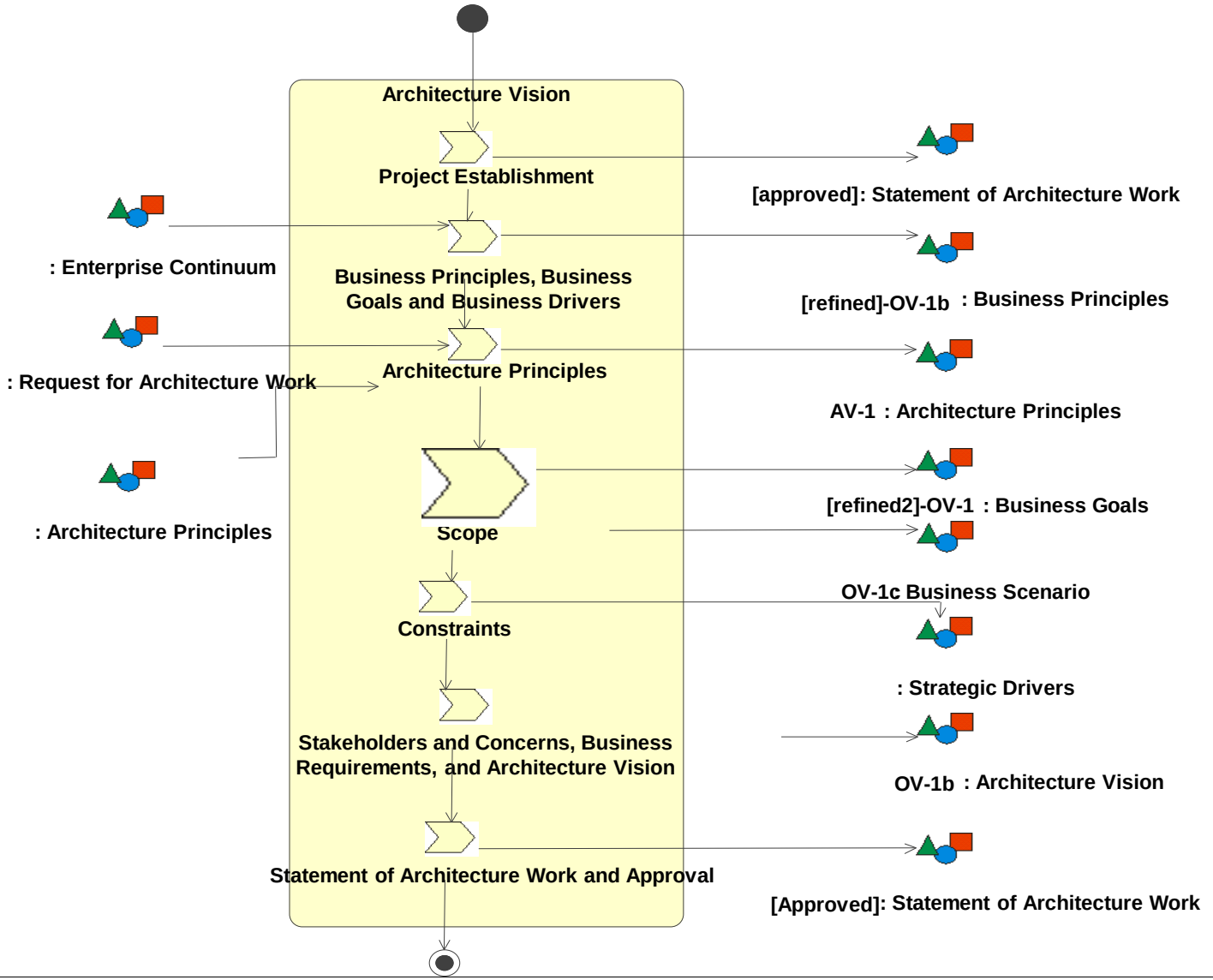
- **Objectives: To define a process whereby requirements for enterprise architecture are identified, stored, and fed into and out of the relevant ADM phases**

- **There are no constructs in DoDAF v1.0 that permit the modeling of requirements**

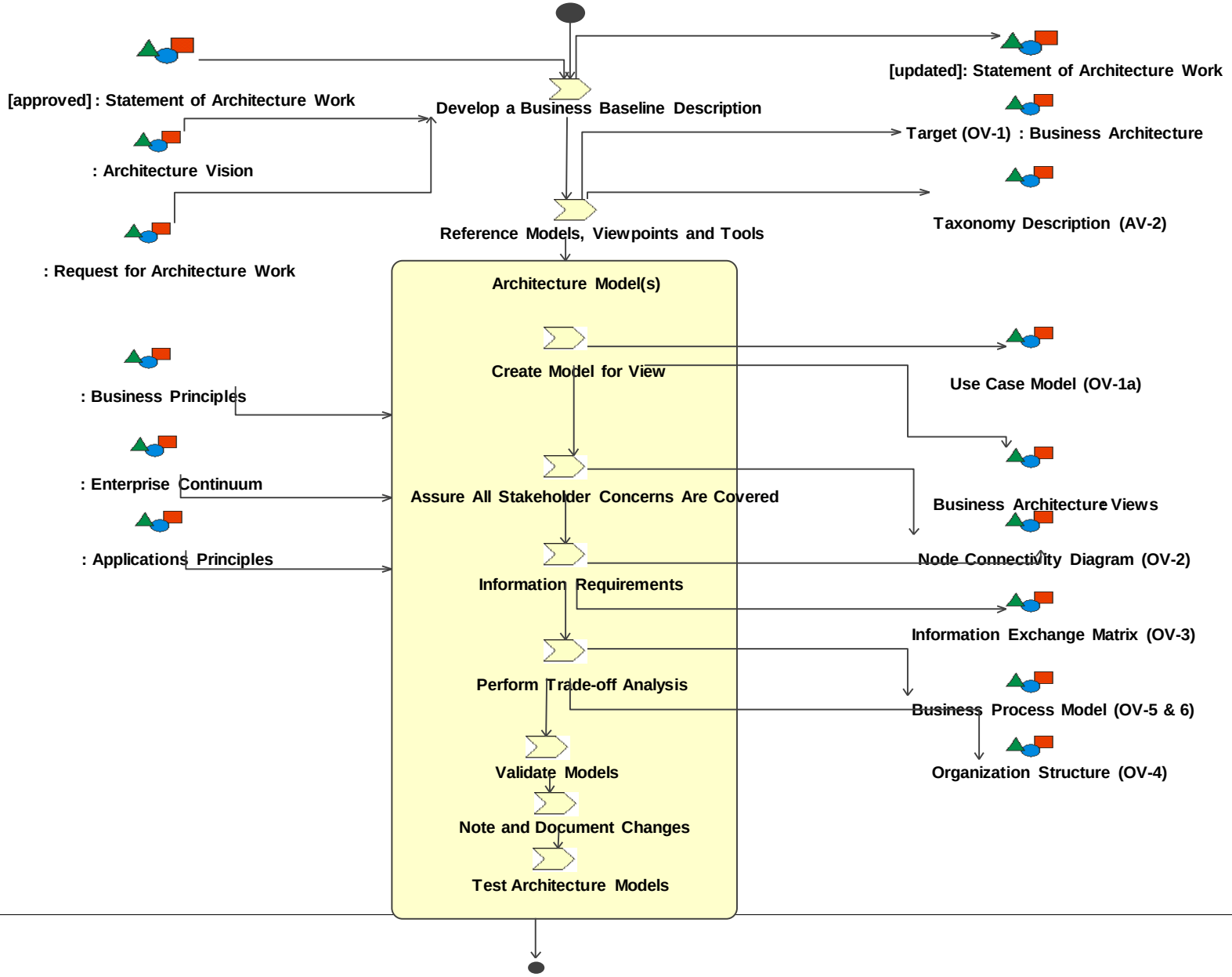
Preliminary Phase: Framework and Principles



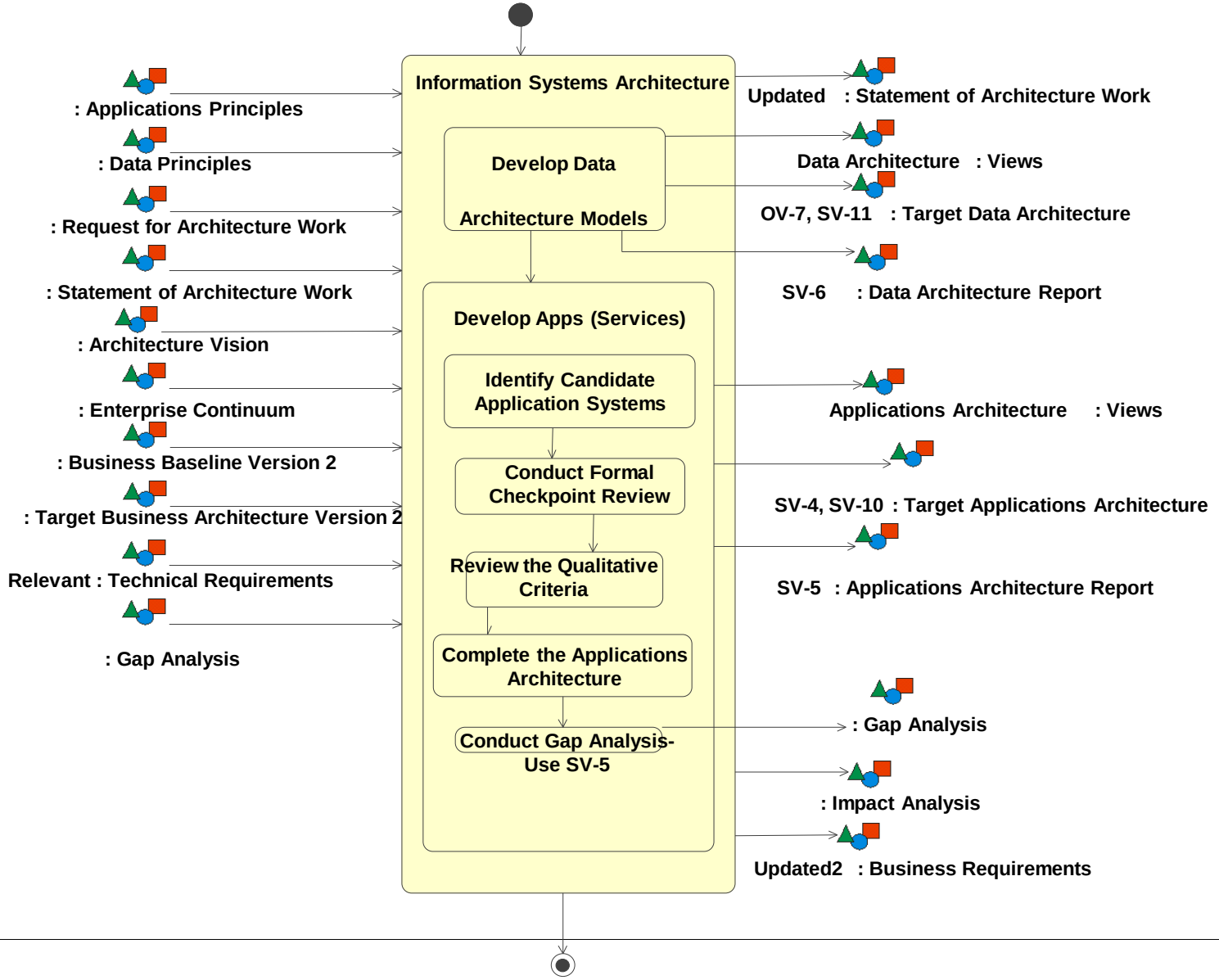
Phase A: Architecture Vision



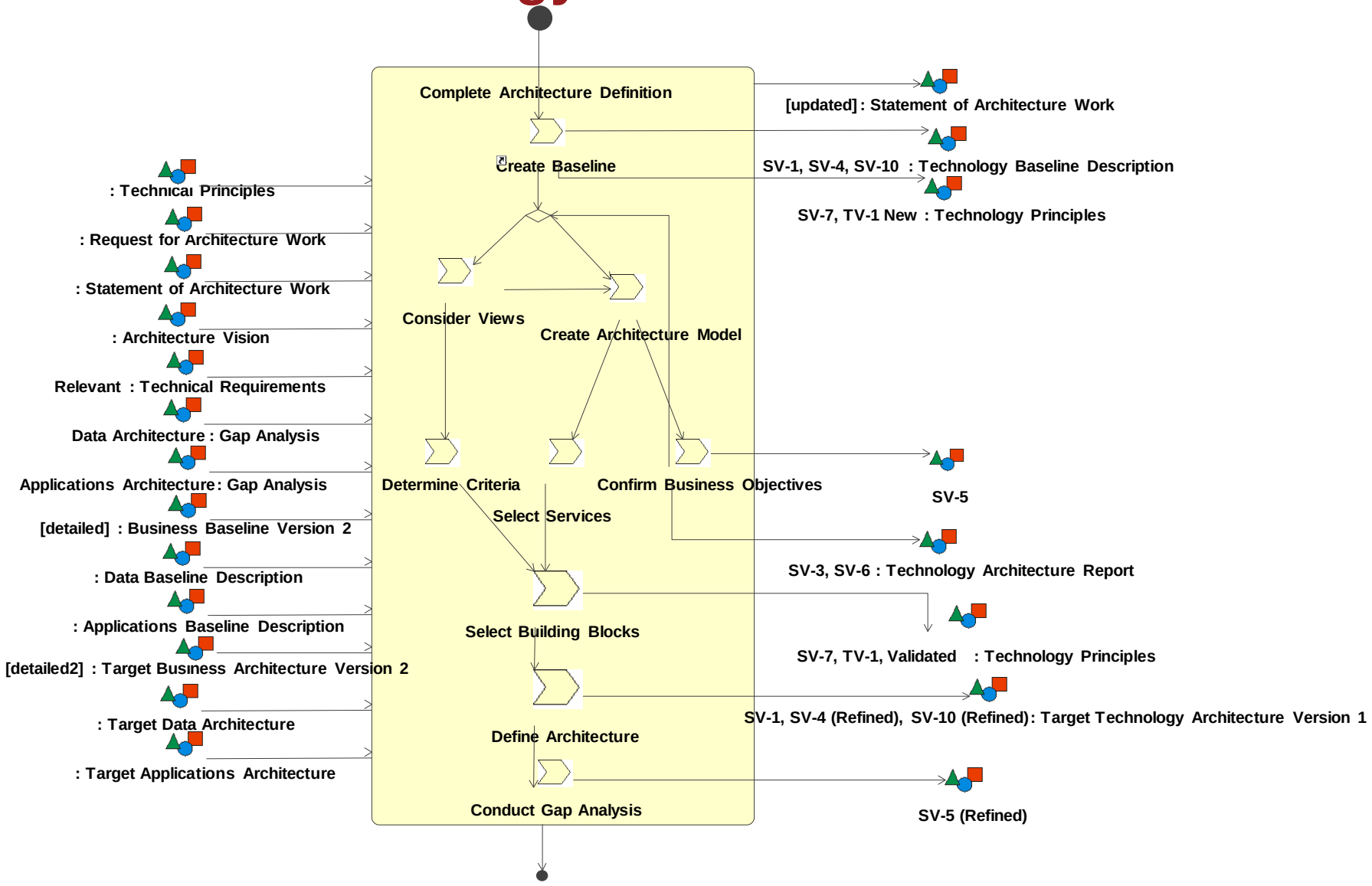
Phase B: Business Architecture



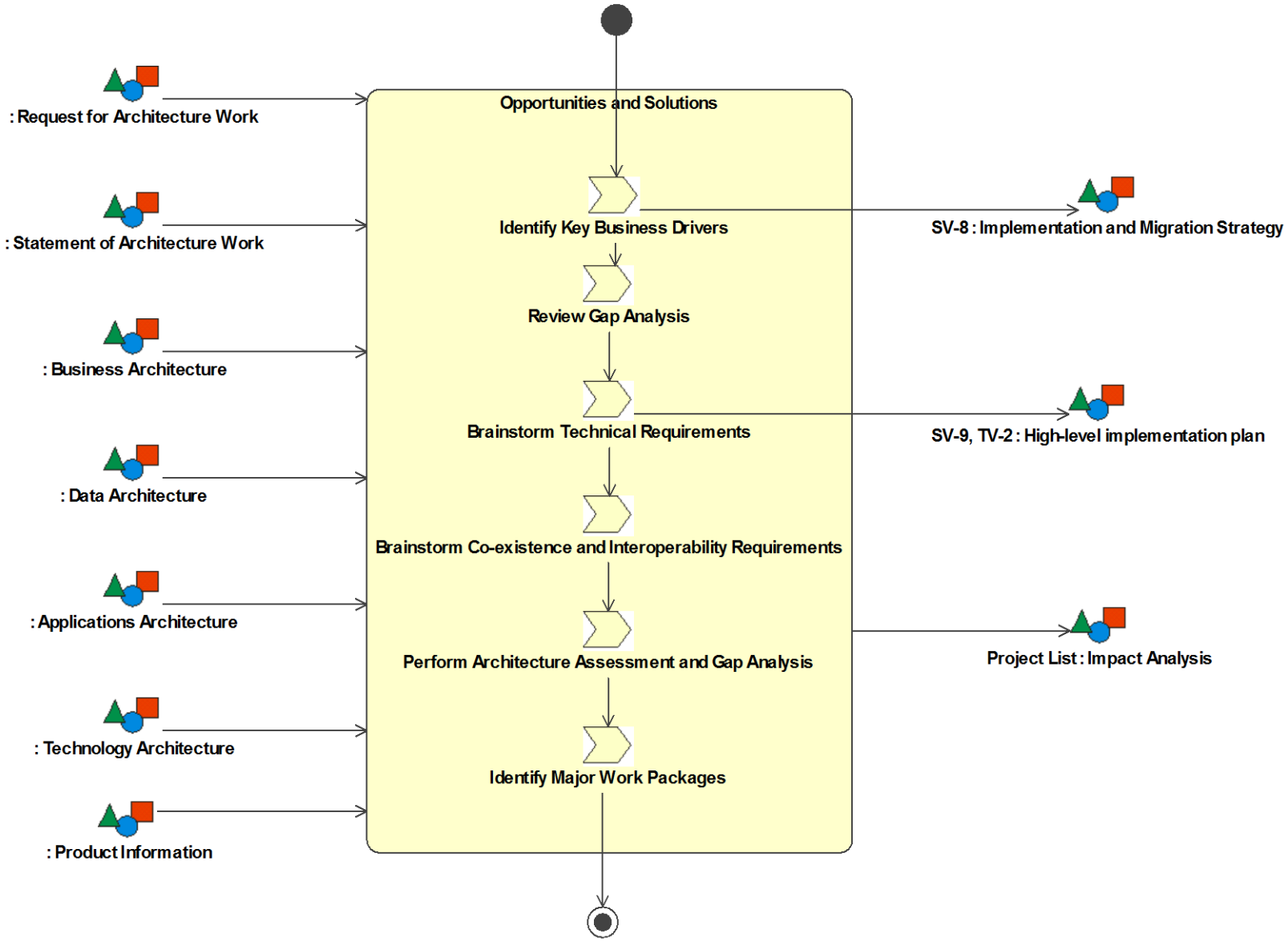
Phase C: Information System Architecture



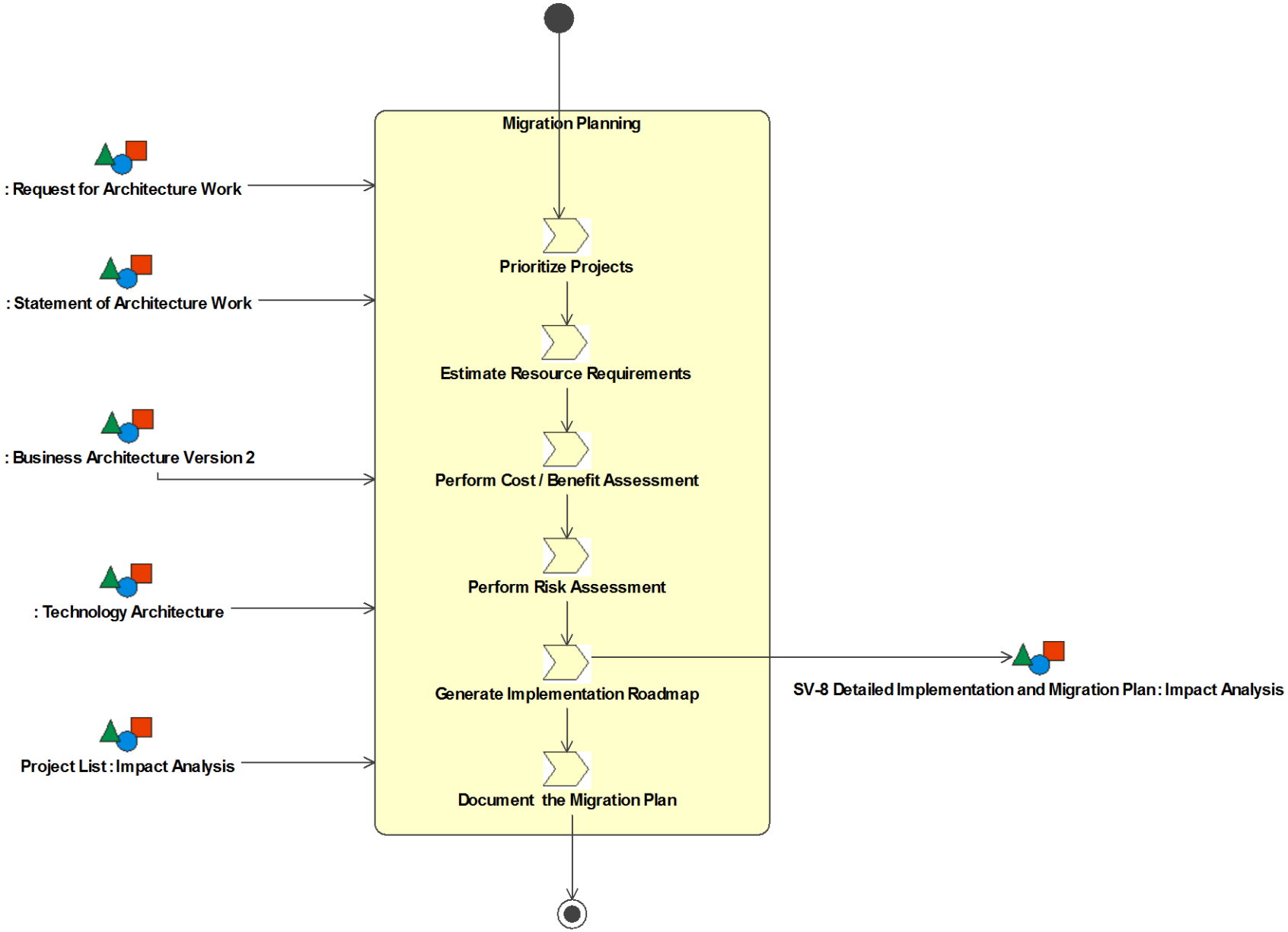
Phase D: Technology Architecture



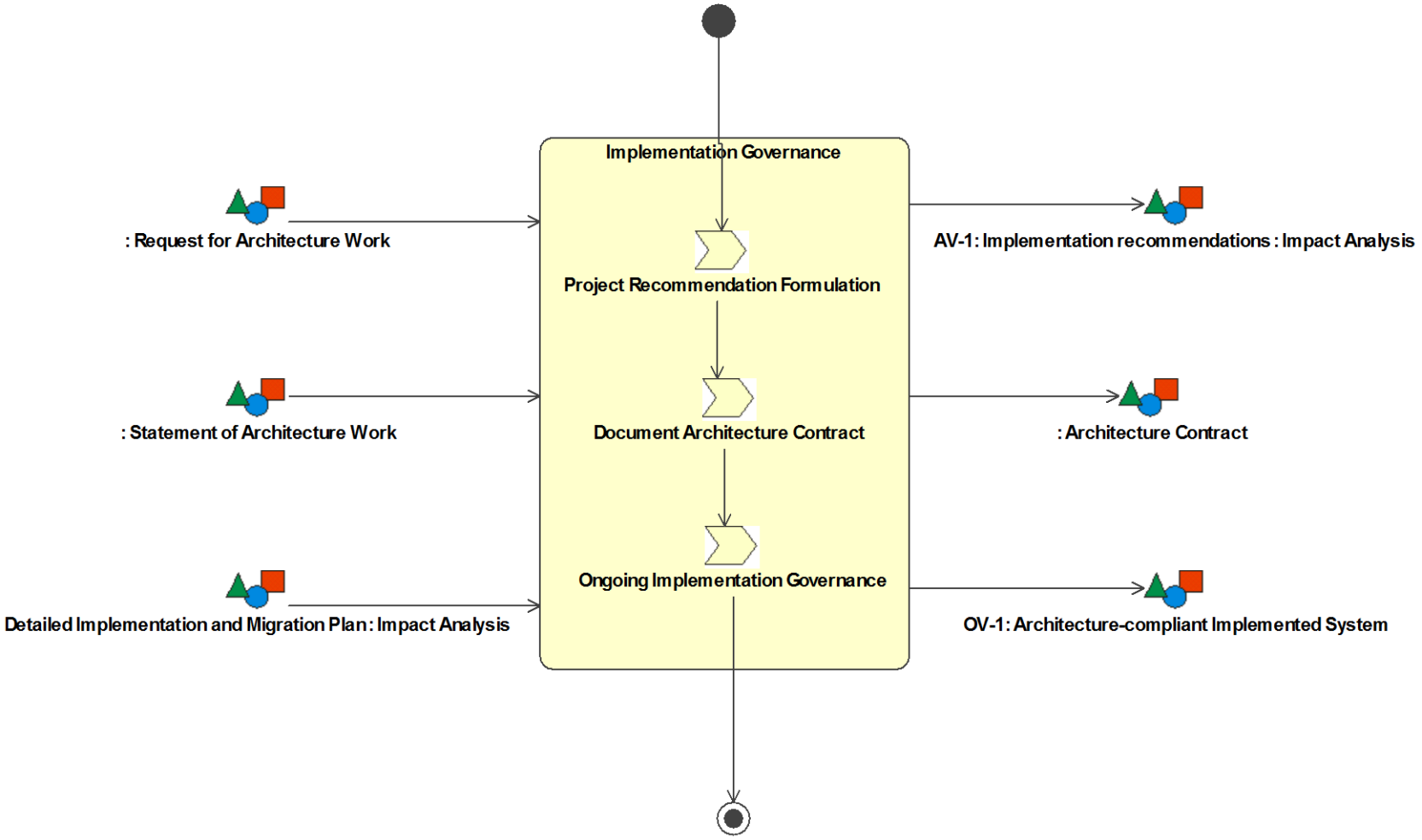
Phase E: Opportunities and Solutions



Phase F: Migration Planning



Phase G: Implementation Governance

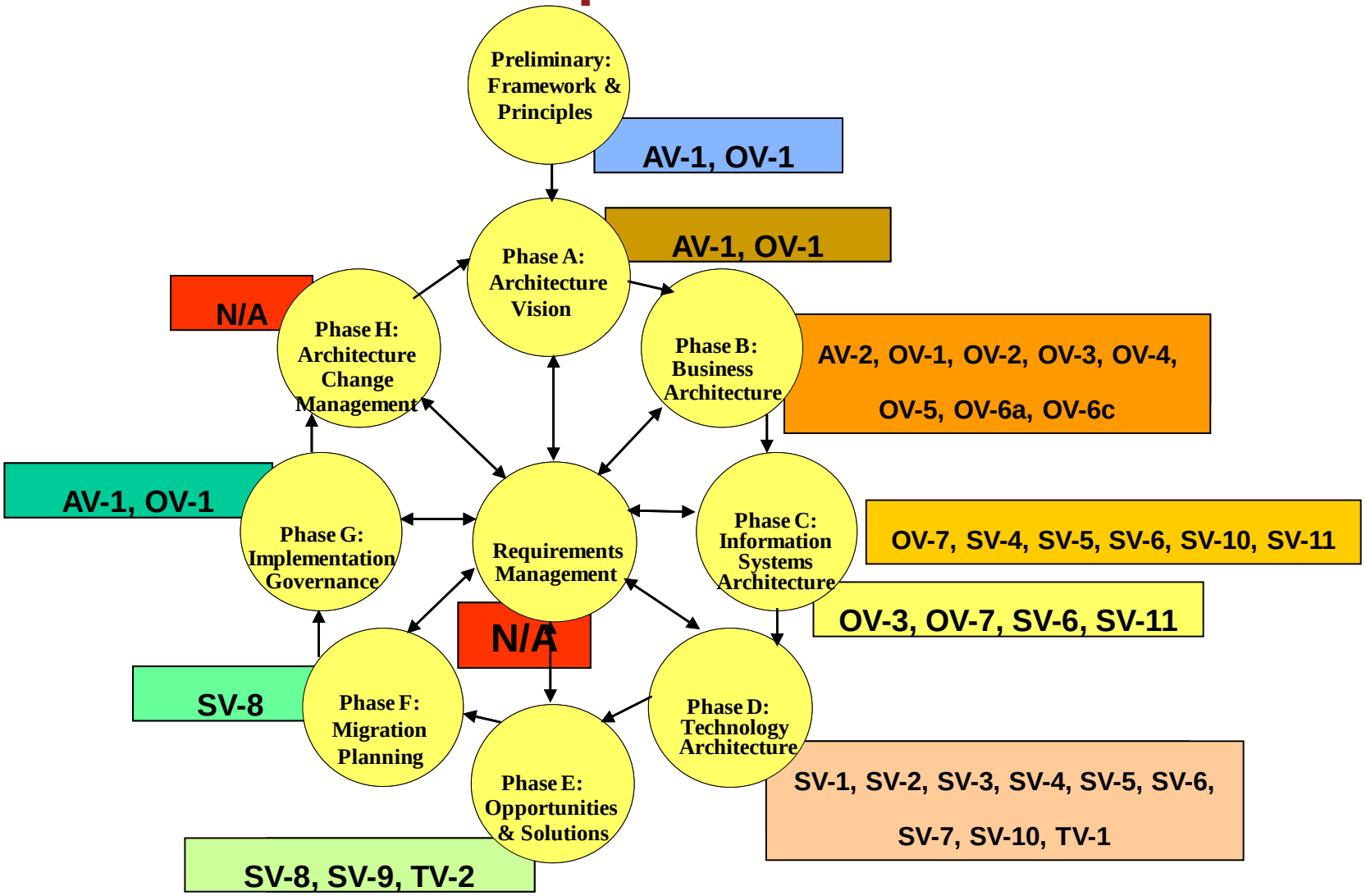


Phase H: Architecture Change Management

- **The objective of this phase is to establish an Architecture Change Management process for the new Enterprise Architecture baseline that is achieved with completion of the Implementation Governance phase**

- **Architecture Change Management is outside the scope of defining architecture modeling elements and relationships and DoDAF**

ADM with DoDAF Models as the Architecture Description



For More Information . . .

- **Architecture Portal:**
  - http://www.opengroup.org/architecture/index.htm
- **TOGAF Documentation:**
  - http://www.opengroup.org/public/arch
- **TOGAF Information:**
  - http://www.opengroup.org/public/togaf
- **Standards Information Base (SIB):**
  - http://www.db.opengroup.org/sib.htm

Outline

- DoDAF (1.0)
- TOGAF
- **Zachman**
- FEAF

ENTERPRISE ARCHITECTURE: A FRAMEWORK™

ZIFA
Zachman Institute for Framework Advancement
PHONE (810) 231-0531
FAX: (810) 231-6631
www.zifa.com
10895 Lakepointe Drive
Pinckney, MI 48169

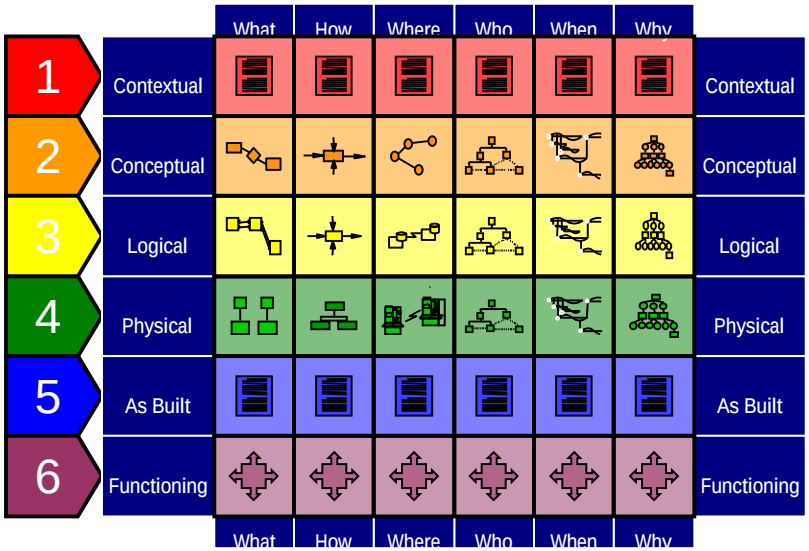
	WHAT	HOW	WHERE	WHO	WHEN	WHY	
	DATA	**FUNCTION**	**NETWORK**	**PEOPLE**	**TIME**	**MOTIVATION**	
SCOPE {contextual} Planner	List of Things Important to the Business Entity = Class of Business Thing	List of Processes the Business Performs Process = Class of Business Process	List of Locations in Which the Business Operates Node = Major Business Location	List of Organizations Important to the Business People = Major Organizational Unit	List of Events/Cycles Significant to the Business Time = Major Business Event/Cycle	Lists of Business Goals/Strategies Ends/Means = Major Business Goal/Strategy	**SCOPE** {contextual} Planner
BUSINESS MODEL {conceptual} Owner	e.g., Semantic Model Entity = Business Entity Relationship = Business Relationship	e.g., Business Process Model Process = Business Process I/O = Business Resources	e.g., Business Logistics System Node = Business Location Link = Business Linkage	e.g., Work Flow Model People = Organization Unit Work = Work Product	e.g., Master Schedule Time = Business Event Cycle = Business Cycle	e.g., Business Plan End = Business Objective Means = Business Strategy	**BUSINESS MODEL** {conceptual} Owner
SYSTEM MODEL {logical} Designer	e.g., Logical Data Model Entity = Data Entity Relationship = Data Relationship	e.g., Application Architecture Process = Application Function I/O = User Views	e.g., Distributed System Architecture Node = I/S Function (Processor, Storage, etc.) Link = Line Characteristics	e.g., Human Interface Architecture People = Role Work = Deliverable	e.g., Processing Structure Time = System Event Cycle = Processing Cycle	e.g., Business Rule Model End = Structural Assertion Means = Action Assertion	**SYSTEM MODEL** {logical} Designer
TECHNOLOGY MODEL {physical} Builder	e.g., Physical Data Model Entity = Segment/Table/etc. Relationship = Pointer/Key/etc.	e.g., System Design Process = Computer Function I/O = Data Elements/Sets	e.g., Technology Architecture Node = Hdw/System Software Link = Line Specifications	e.g., Presentation Architecture People = User Work = Screen Formats	e.g., Control Structure Time = Execute Cycle = Component Cycle	e.g., Rule Design End = Condition Means = Action	**TECHNOLOGY MODEL** {physical} Builder
DETAILED REPRESENTATIONS {out-of-context} Subcontractor	e.g., Data Definition Entity = Field Relationship = Address	e.g., Program Process = Language Statement I/O = Control Block	e.g., Network Architecture Node = Address Link = Protocol	e.g., Security Architecture People = Identity Work = Job	e.g., Timing Definition Time = Interrupt Cycle = Machine Cycle	e.g., Rule Specification End = Sub-condition Means = Step	**DETAILED REPRESENTATIONS** {out-of-context} Subcontractor
FUNCTIONING ENTERPRISE	e.g.: DATA	e.g.: FUNCTION	e.g.: NETWORK	e.g.: ORGANIZATION	e.g.: SCHEDULE	e.g.: STRATEGY	**FUNCTIONING ENTERPRISE**

© John A. Zachman

THE ZACHMAN FRAMEWORK FOR ENTERPRISE ARCHITECTURE

Zachman Framework

- Row 1 – Scope
  - External Requirements and Drivers
  - Business Function Modeling
- Row 2 – Enterprise Model
  Business Process Models
- Row 3 – System Model
  Logical Models
  Requirements Definition
- Row 4 – Technology Model
  Physical Models
  Solution Definition and Development
- Row 5 – As Built
  As Built
  Deployment
- Row 6 – Functioning Enterprise
  Functioning Enterprise
  Evaluation



Framework Rules

- ***Rule 1:***

  Columns have no order

- ***Rule 2:***

  Each column has a simple, basic model

- ***Rule 3:***

  Basic model of each column is unique

- ***Rule 4:***

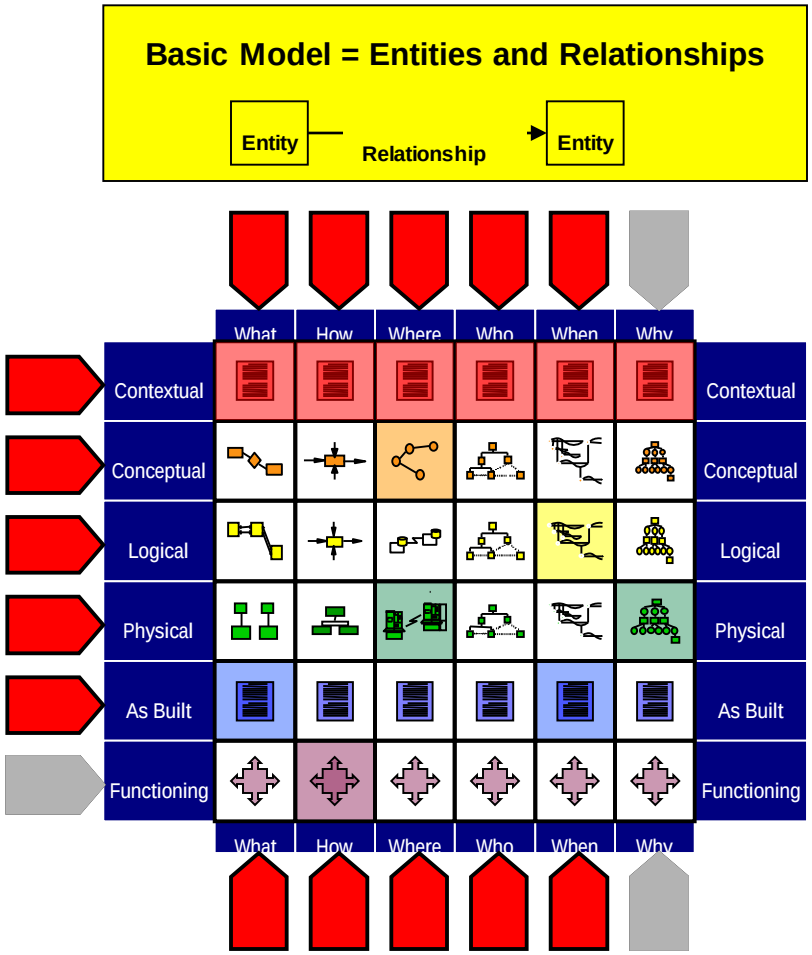
  Each row represents a distinct view

- ***Rule 5:***

  Each cell is unique

- ***Rule 6:***

  Combining the cells in one row forms a complete description from that view



Zachman Framework – Row 1 Scope/Planner's View

- Motivation/Why
  Business goals, objectives and performance measures related to each function

- Function/How
  High-level business functions

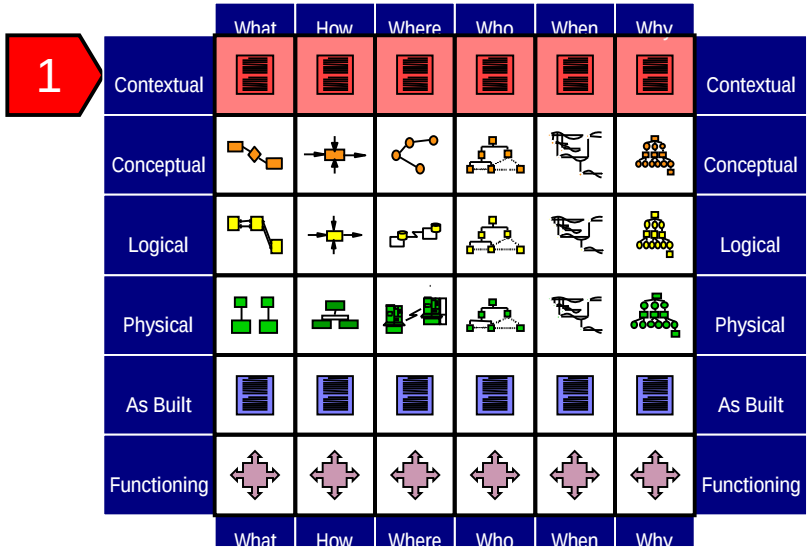
- Data/What
  High-level data classes related to each function

- People/Who
  Stakeholders related to each function

- Network/Where
  Locations related to each function

- Time/When
  Cycles and events related to each function

- External Requirements and Drivers
- Business Function Modeling

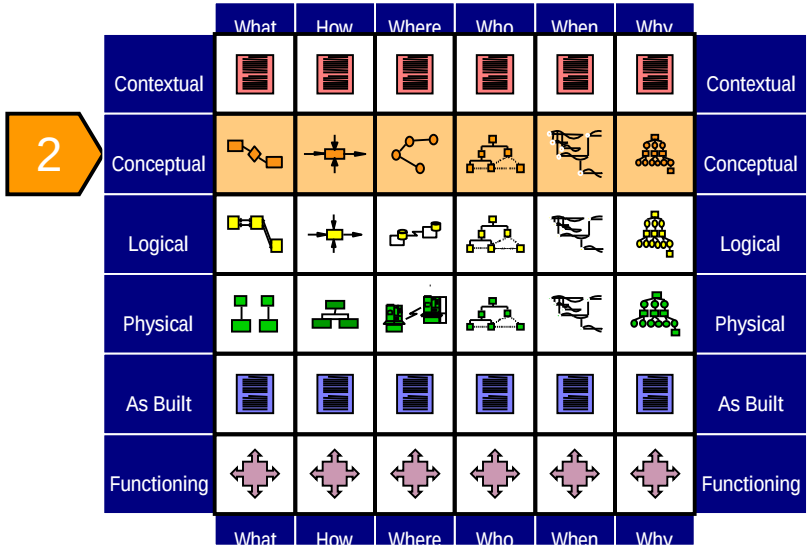


Zachman Framework – Row 2
Enterprise Model/Designer's View

- Motivation/Why
  Policies, procedures and standards for each process

- Function/How
  Business processes
- Data/What
  Business data
- People/Who
  Roles and responsibilities in each process
- Network/Where
  Locations related to each process
- Time/When
  Events for each process and sequencing of integration and process improvements

- Business Process Models
- Business Function Allocation
- Elimination of Function Overlap and Ambiguity



Zachman Framework – Row 3 System Model/Designer's View

- Motivation/Why
  Policies, standards and procedures associated with a business rule model

- Function/How
  Logical representation of information systems and their relationships

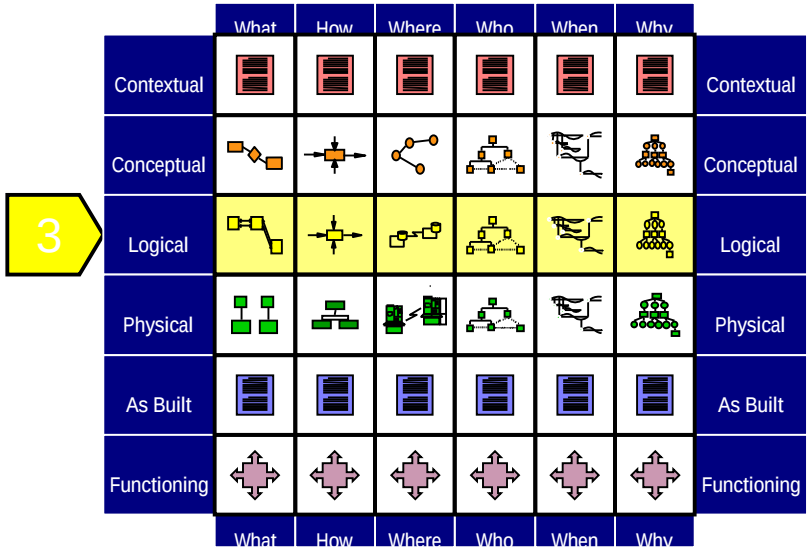
- Data/What
  Logical data models of data and data relationships underlying the information

- People/Who
  Logical representation of access privileges constrained by roles and responsibilities

- Network/Where
  Logical representation of the distributed system architecture for locations

- Time/When
  Logical events and their triggered responses constrained by business events and their responses

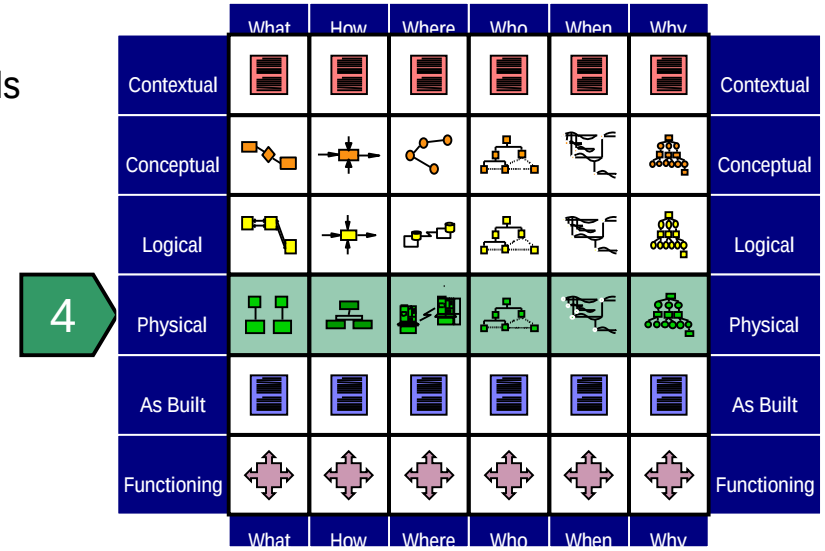
- Logical Models
- Project Management
- Requirements Definition



Zachman Framework – Row 4 Technology Model/Builder's View

- Motivation/Why
  Business rules constrained by information systems standards
- Function/How
  Specifications of applications that operate on particular technology platforms
- Data/What
  Database management system (DBMS) type requirements constrained by logical data models
- People/Who
  Specification of access privileges to specific platforms and technologies
- Network/Where
  Specification of network devices and their relationships within physical boundaries
- Time/When
  Specification of triggers to respond to system events on specific platforms and technologies

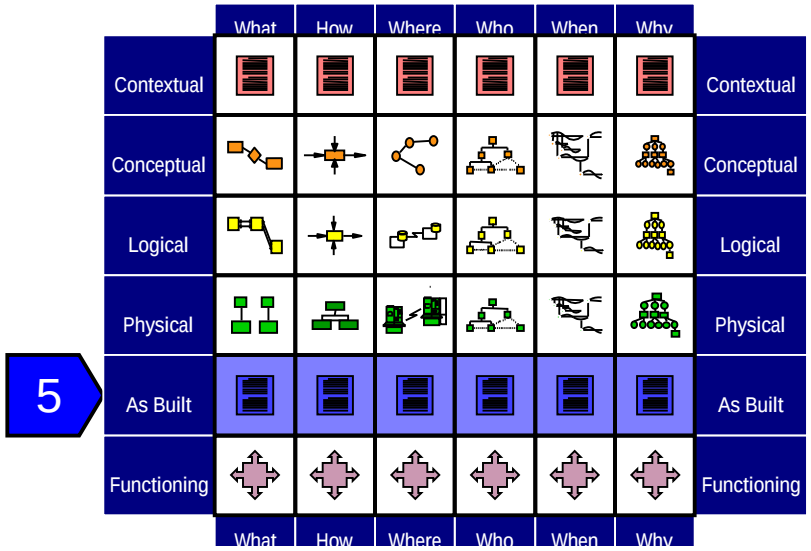
- Physical Models
- Technology Management
- Solution Definition and Development



Zachman Framework – Row 5
As Built/Integrator's View

- Motivation/Why
  Business rules constrained by specific technology standards
- Function/How
  Programs coded to operate on specific technology platforms
- Data/What
  Data definitions constrained by physical data models
- People/Who
  Access privileges coded to control access to specific platforms and technologies
- Network/Where
  Network devices configured to conform to node specifications
- Time/When
  Timing definitions coded to sequence activities on specific platforms and technologies

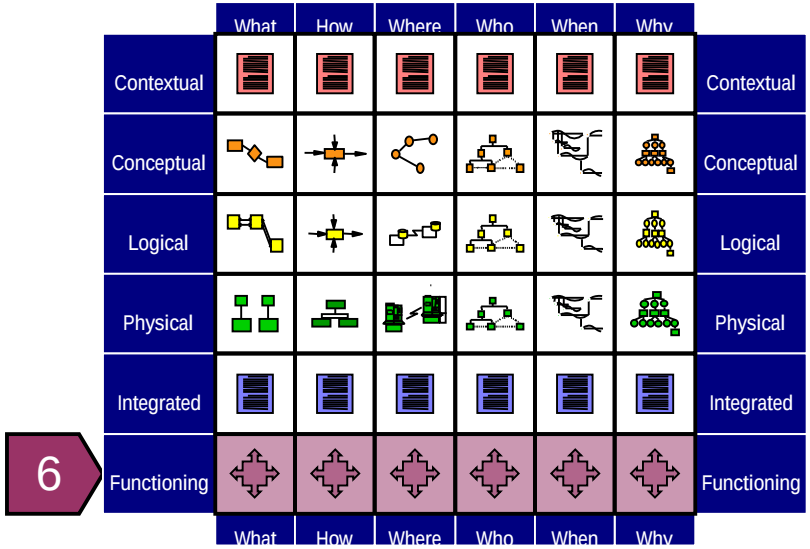
- As Built
- Configuration Management
- Deployment

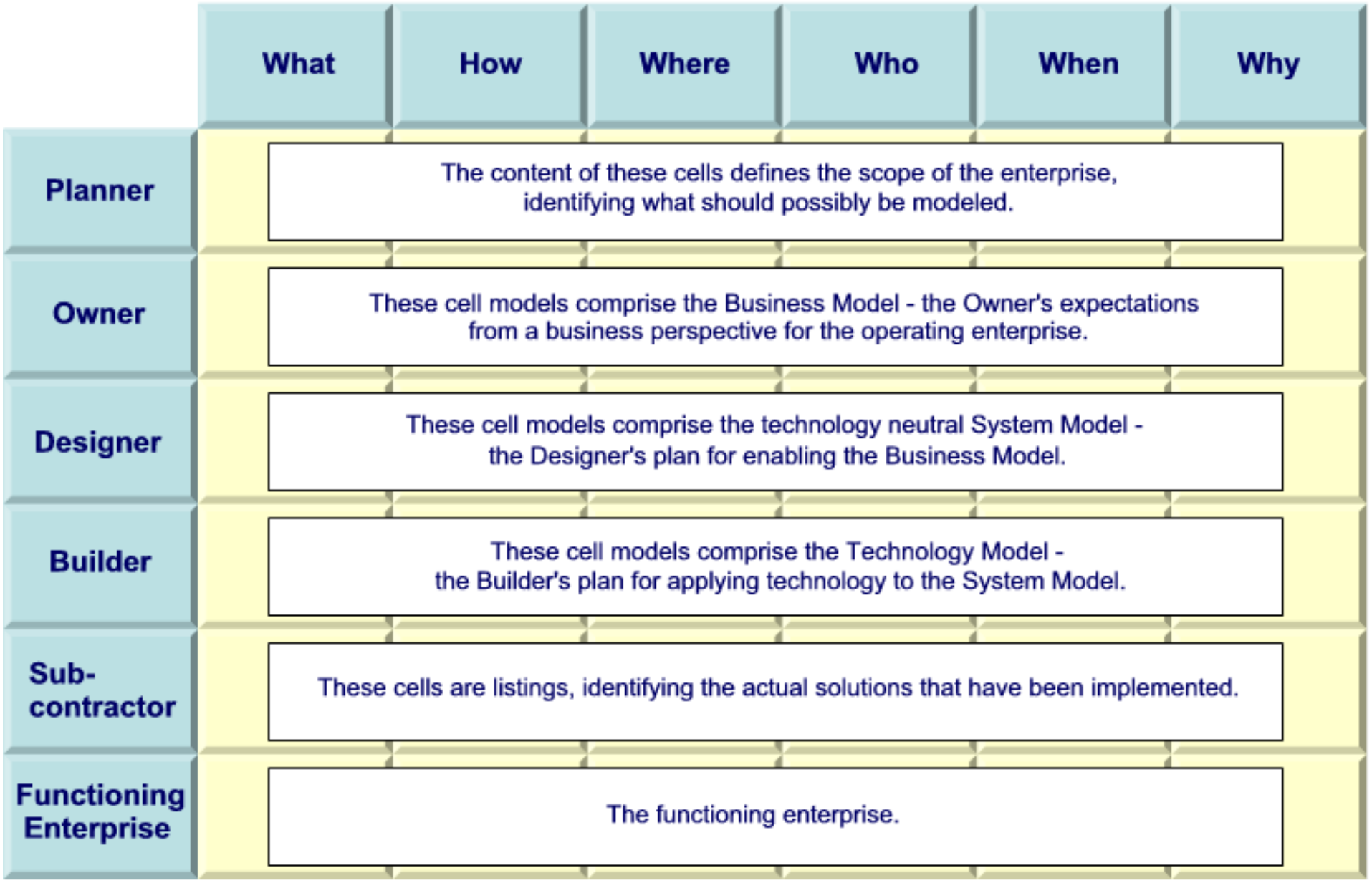


Zachman Framework – Row 6
Functioning Enterprise/User's View

- Motivation/Why
  Operating characteristics of specific technologies constrained by standards
- Function/How
  Functioning computer
- Data/What
  Data values stored in actual databases
- People/Who
  Personnel and key stakeholders working within their roles and responsibilities
- Network/Where
  Sending and receiving messages
- Time/When
  Timing definitions operating to sequence activities

- Functioning Enterprise
- Evaluation




What
How
Where
Who
When
Why
Planner
The content of these cells defines the scope of the enterprise, identifying what should possibly be modeled.
Owner
These cell models comprise the Business Model - the Owner's expectations from a business perspective for the operating enterprise.
Designer
These cell models comprise the technology neutral System Model - the Designer's plan for enabling the Business Model.
Builder
These cell models comprise the Technology Model - the Builder's plan for applying technology to the System Model.
Sub-contractor
These cells are listings, identifying the actual solutions that have been implemented.
Functioning Enterprise
The functioning enterprise.

Outline

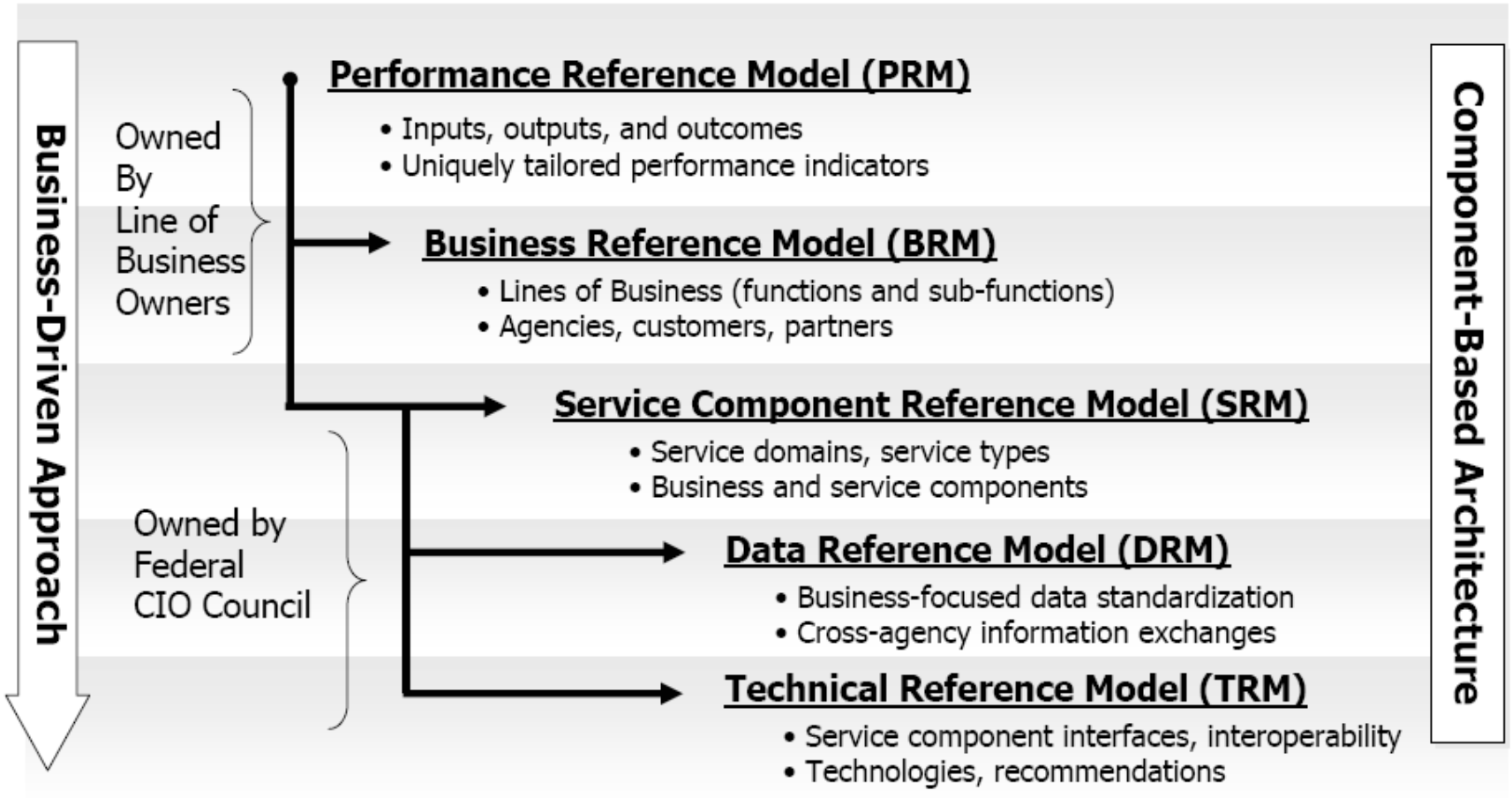
- DoDAF (1.0)
- TOGAF
- Zachman
- **FEAF**

Federal Enterprise Architecture (FEA) Framework

- **Established 1999 by the CIO in response to the Clinger-Cohen Act to facilitate shared development of common processes and information among Federal Agencies and other government agencies**

- **Objectives:**
  - Identifying opportunities to leverage technology and alleviate redundancy, or highlighting where agency IT overlap reduces the value of investments
  - Facilitating horizontal (cross-Federal) and vertical (Federal, state and local) integration of IT resources
  - Applying architecture practices to help drive business management improvements across the Federal government
  - Supporting a citizen-centered, results-focused government that maximizes IT to better achieve mission outcomes and fulfill legislative mandates

Federal Enterprise Architecture (FEA) Framework

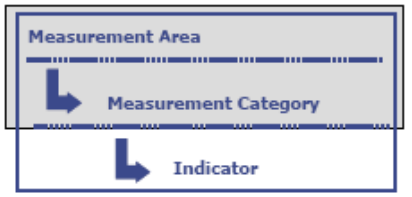


Reference Models

- Performance Reference Model (PRM)
  - Standardized framework to measure the performance of major IT investments and their contribution to program performance

FEA-- Performance Reference Model





Mission and business-critical results aligned with the Business Reference Model. Results measured from a customer perspective

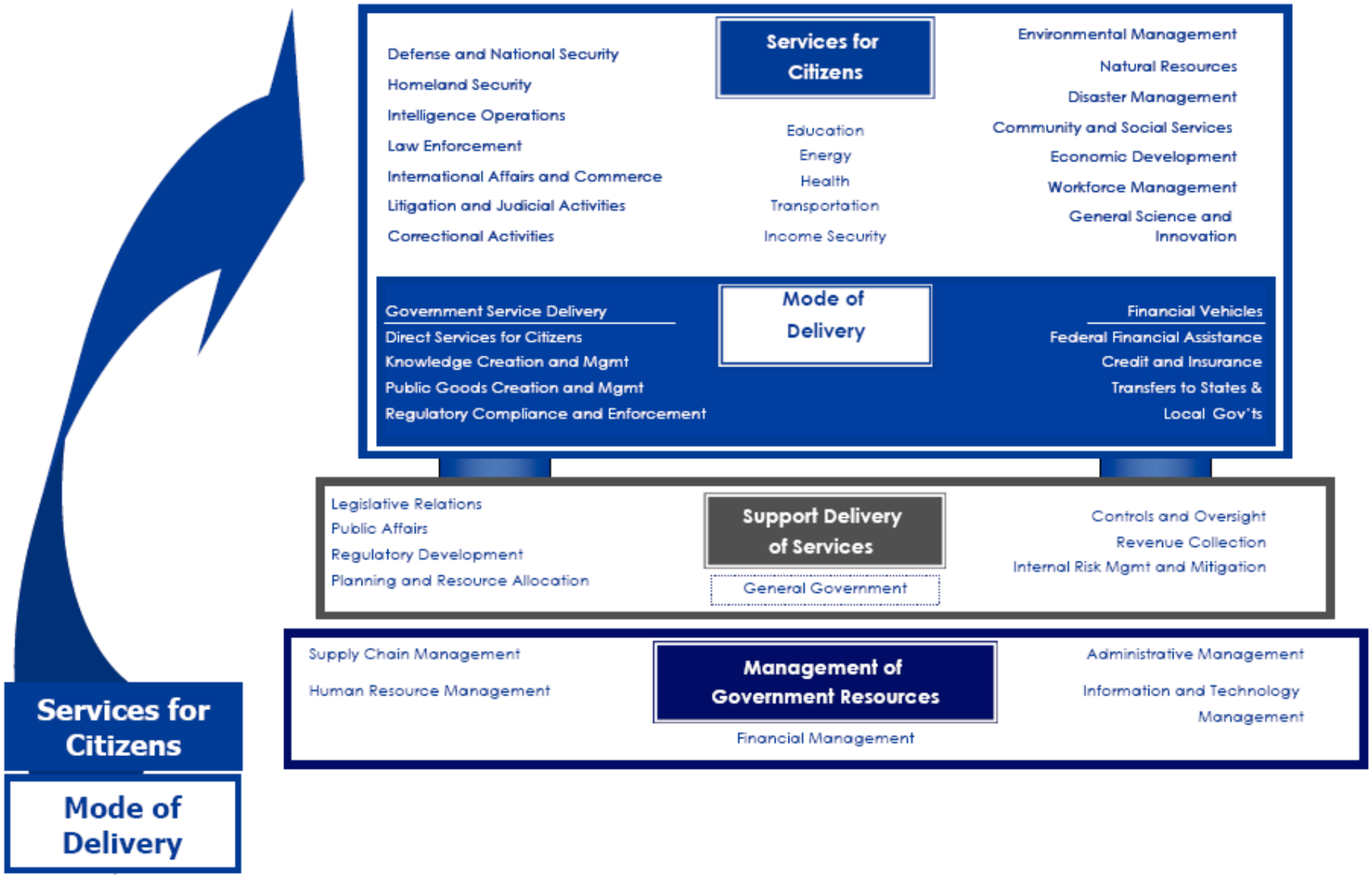
The direct effects of day-to-day activities and broader processes measured as driven by desired outcomes. Used to further define and measure the Mode of Delivery in the Business Reference Model

Key enablers measured through their contribution to outputs – and by extension outcomes

Reference Models

- **Business Reference Model (BRM)**
  - Function-driven framework for describing business operations of the Federal government independent of the agencies that perform them

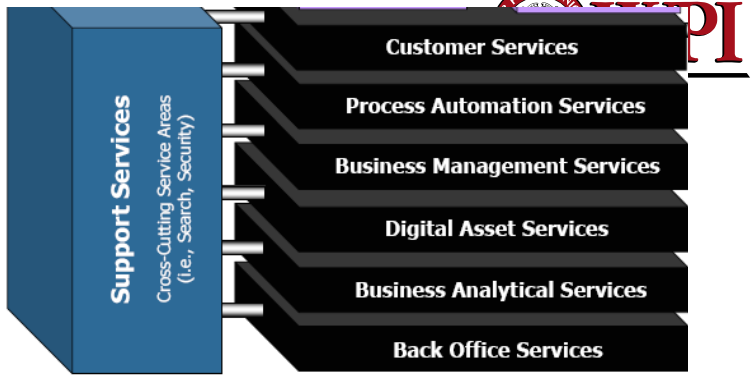
FEA-- Business Reference Model



Reference Models

- **Service Component Reference Model (SRM)**
  - Business and performance-driven functional framework that classifies service components with respect to how they support business and/or performance objectives

Service Component RM



Customer Services	Process Automation Services	Business Management Services	Digital Asset Services	Business Analytical Services	Back Office Services	Support Services
• Customer relationship management • Customer preferences • Customer initiated assistance	• Tracking & workflow • Routing & scheduling	• Management of process • Organizational management • Investment management • Supply chain management	• Content management • Document management • Knowledge management • Records management	• Analysis & statistics • Visualization • Business Intelligence • Reporting	• Data management • Human resources • Financial management • Assets / materials management • Integration • Human capital & workforce management	• Security management • Collaboration • Search • Communication • Systems management

The SRM is structured across horizontal service areas that can provide a leverageable foundation for reuse of applications, application capabilities, components, and business services.

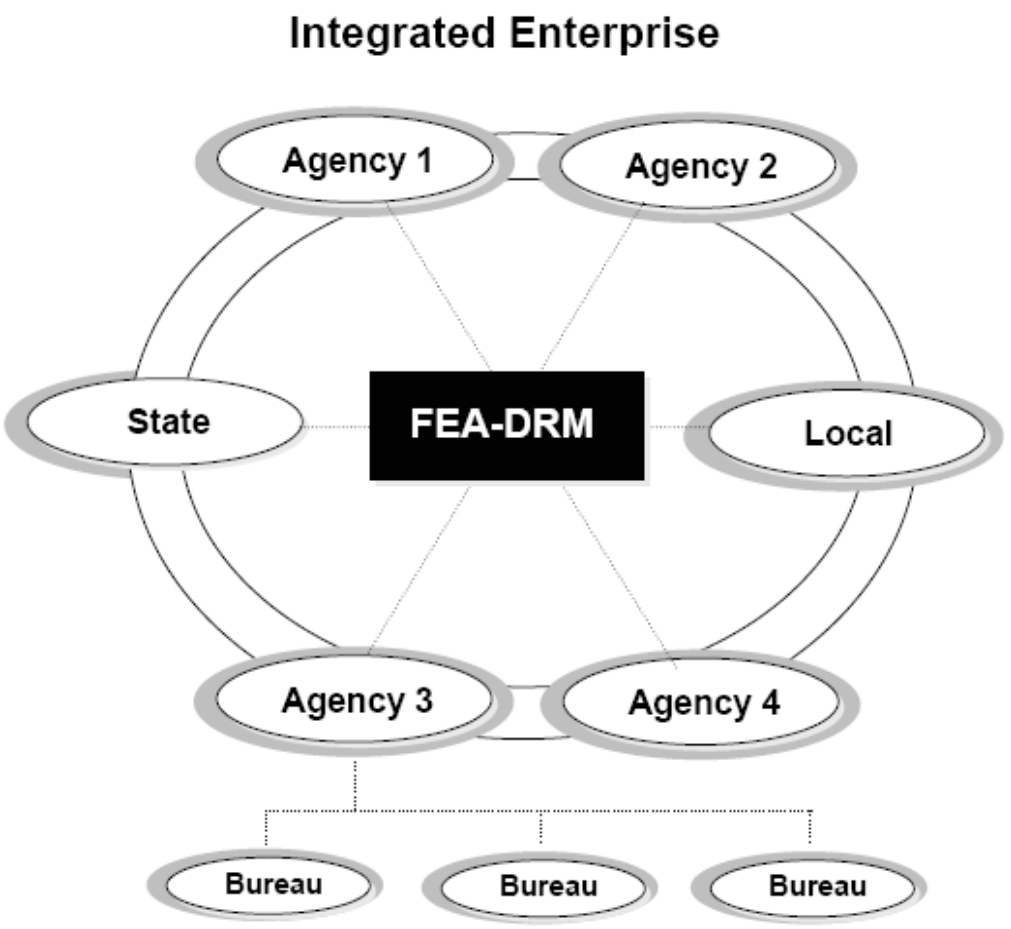
Reference Models

- Data Reference Model (DRM)
  - Model describing, at an aggregate level, the data and information that support program and business line operations

FEA-- Data Reference Model

Goals and Objectives:

- Promote horizontal and vertical information sharing between business lines
- Business-focused data standardization that can be categorized for re-use
- Re-use and integration of data as opposed to duplication
- Enabler to support cross-agency collaboration
- Facilitate cross-agency information exchanges
- Consistent means to categorize and classify data

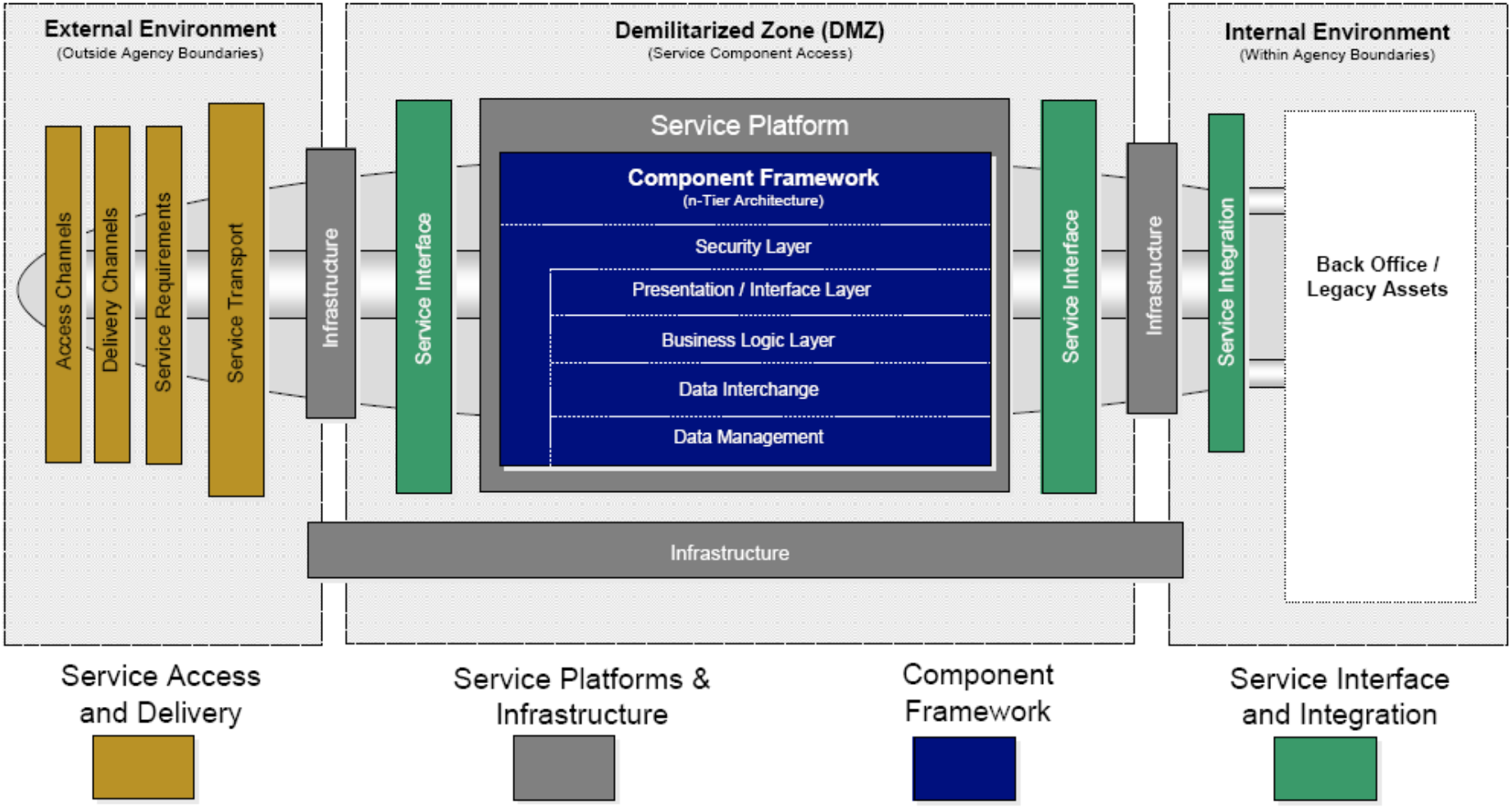


DRM provides framework for agencies to leverage existing data components across the Government

Reference Models

- **Technical Reference Model (TRM)**
  - Component-driven, technical framework used to identify the standards, specifications, and technologies that support and enable the delivery of service components and capabilities

FEA-- Technical Reference Model



Some Enterprise Architecture Resources

- Global Enterprise Architecture Organization
  - http://www.geao.org//
- Enterprise Architecture Community
  - http://www.eacommunity.com//
- Institute for Enterprise Architecture Developments
  - http://www.enterprise-architecture.info//
- Federal Enterprise Architecture Certification Institute
  - http://www.feacinstitute.org//
- International Enterprise Architecture Center
  - http://www.ieac.org
- Object Management Group (MDA, CORBA, UML, CWM (Common Warehouse Metamodel)
  - http://www.omg.org//
- The Open Group Architecture Framework, Version 8
  - http://www.opengroup.org/architecture//
- Zachman Institute for Framework Advancement
  - http://www.zifa.com//
- The Enterprise Architecture Toolbox
  - http://www.ovitztaylorgates.com/TheEnterpriseArchitectureToolBox.html

Systems Engineering Technical Short Course for Waters Corporation

This course planned to be conducted at the Waters facilities as follows:

Session 1:	**January 27 – 29, 2015:**	**Overview and Concepts of Systems Engineering**
Session 2:	**February 16 – 17, 2015:**	**Stakeholder Needs**
Session 3:	**March 16 – 17, 2015**	**System Requirements**
Session 4:	**April 13 – 14, 2015**	**System Architecture and Design**
Session 5:	**May 13 – 14, 2015**	**System Integration and Test**
	June 10, 2015	**Capstone preparations (Not part of WPI material)**
Session 6:	**June 11, 2015**	**Capstone presentations**

Robert B. Wheeler, ESEP, PMP
Thomas F. Gannon, Ph.D.

This course is developed by Worcester Polytechnic Institute (WPI) for Waters Corporation. Course materials remain the property of WPI. WPI and the course developer (Robert Wheeler of Robert B Wheeler & Associates for sessions 1, 2, and 3) retain the right to use, reuse, and modify this content in future course deliveries.



Systems Engineering Technical Short Course

Session 5: Systems Integration and Test

May 13 - 14, 2015

Robert B. Wheeler, ESEP, PMP
Thomas F. Gannon, Ph.D.



Systems Engineering Technical Short Course

(Session 5: Systems Integration and Test)

Lesson 1:
Program Review Update

Remember from Session 1 – During this course you will: (1/2)

All Sessions: Apply the Systems Engineering principles and practices to your assigned projects

Sessions 1 through 3 (Bob Wheeler, instructor):

- **Develop and use a common vocabulary**
- **Apply systems thinking and holism to your projects**
- **Articulate the Problem to be Solved by your project**
- **Develop a Technical Risk Management Plan**
- **Elicit and express stakeholders expectations and requirements**
- **Develop User Concept of Operations (CONOPS), solution selection criteria, and acceptance criteria**
- **Conduct system functional and performance analyses**
- **Define and express system requirements**
- **Manage requirements through the system lifecycle**

Session 1 | Session 2 | Session 3

Remember from Session 1 – During this course you will: (2/2)

Sessions 4 & 5 (Tom Gannon, instructor):

- Develop creative alternative system architectures
- Conduct analyses of your architectural alternatives to meet stakeholder and system requirements
- Define a candidate architecture and design for your solution
- Develop strategies and plans for system integration, validation and verification testing
- Manage requirements through the system lifecycle (continued)
- Document the results of your work in appropriate artifacts

Session 6:

- Present a status report on your project to Waters senior management

This Session 4 | Session 5

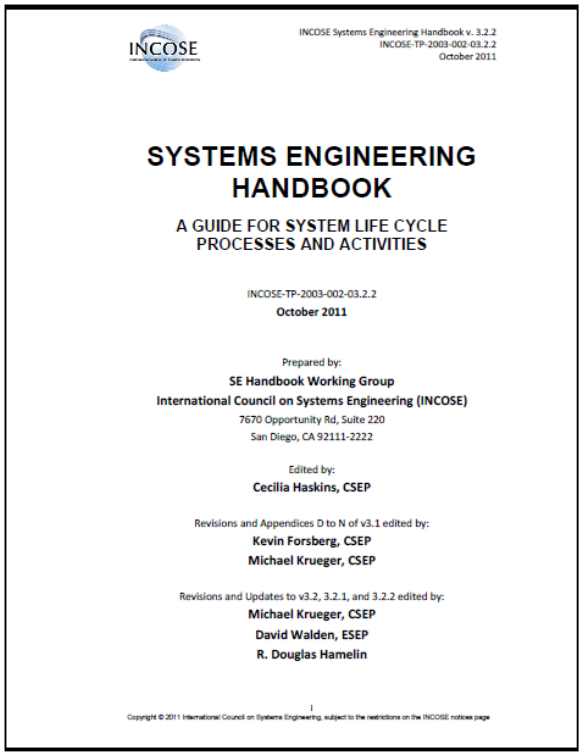
Schedule

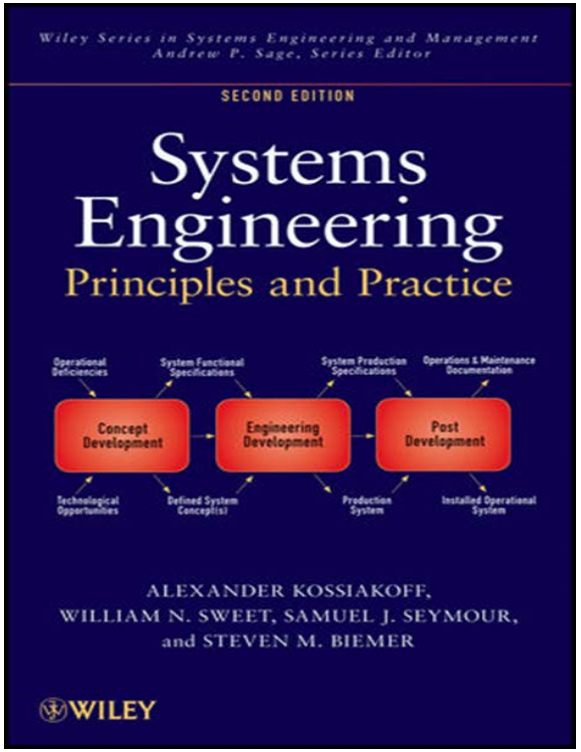
Session	Dates	Topic
1.	January 27 – 29	Introduction – Overview and Concepts of SE (US)
2.	February 16 – 17	Stakeholder Needs*
3.	March 16 – 17	Systems Requirements*
4.	April 13 – 14	Systems Architecture and Design (UK)*
5.	May 13 – 14	Systems Integration and Test*
6.	June 11	Capstone Presentations (US)

Notes:

* Denotes sessions held for some participants via video-teleconference.

Primary Requirements Course Texts





INCOSE Systems Engineering Handbook, (v 3.2.2), October 2011.

Alexander Kossiakoff, et al, Systems Engineering Principles and Practices, 2nd Edition, New Jersey: Wiley, 2011.

Please see Session 1 Appendix D for the other three primary course texts, and many other applicable references

What are your Questions before We Look at our Homework Assignments?



Sessions 2 through 5 Agenda (Revised)

- **Day 1**
  - **Review project team assignments from previous Session**
  - **Presentation and discussion of Session topic**
  - ~~**Project team working sessions on the application of the Session topics to their projects**~~
  - **Day 1 wrap-up and Q&A**

- **Day 2**
  - ~~**Review Day 1 topic and project working sessions**~~
  - **Presentation and discussion of Session topic (concluded)**
  - **Project team working sessions**
  - **Check-in with class on the application of the Session topic to their projects**
  - **Session wrap-up: Q&A and homework due next Session**

Timing will be similar to session 1, and will be announced at beginning of each session

Session 4 - System Architecture and Design:
Homework Assignment due at beginning of Session 4*

Develop the following deliverables:

- **ConOps and Operational View:** (Develop a high-level ConOps and Operational Architectural View for your system. Use any of the approaches from the course material)
- **Logical (Functional) View:** (Develop a Logical View for your system using your functional architecture from Session 3 decomposed to at least one additional level of detail. Use any of the approaches from the course material)
- **Alternative Design Concepts:** (Brainstorm and identify 2 or 3 alternative high-level design concepts which meet your functional requirements and system requirements. Use any of the approaches from the course material)
- **Evaluation Criteria:** (Develop the criteria that you would use to evaluate your alternative design concepts, and the method(s) that you would use to select the best overall concept for further development as your design solution. Use any of the approaches from the course material)

Update the previously developed deliverables

- Each project team must decide which of, and the extent to which, each of these deliverables will be done
- If you cannot complete these deliverables for your system, identify what is not completed, and how you would complete the work
- The course texts and appendices have artifact outlines

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

PROJECT GROUPING

SYSTEMS CONTROLLER

Sponsor: Allen Caswell

Cindy Miller (US)
Richard Moulds (UK)
Richard Chapman (UK)
Steve Binney (US)
Justin Deary (US)
Phillip Young (UK)
David Gordon (UK)

NEXUS

Sponsors: Joe Luongo/John Chan

Abhijit Tarafder (US)
Wade Leveille (US)
David Lamborghini (US)
Ian Trivett (UK)
Daniel Kenney (UK)
Jim Usowicz (US)
John Heden (US)

SMART CONSUMABLES

Sponsor: Tom Walter

Jacob Fairchild (US)
Jason Hill (US)
Mathew DeLano (US)
Jan Driesen (GERMANY)
Alex Hooper (UK)
Mark Pickering (UK)
Senthil Bala (US)

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

PROJECT GROUPING

SYSTEMS CONTROLLER
Sponsor: Allen Caswell

Cindy Miller (US)
Richard Moulds (UK)
Richard Chapman (UK)
Steve Binney (US)
Justin Deary (US)
Phillip Young (UK)
David Gordon (UK)

NEXUS
Sponsors: Joe Luongo/John Chan

Abhijit Tarafder (US)
Wade Leveille (US)
David Lamborghini (US)
Ian Trivett (UK)
Daniel Kenney (UK)
Jim Usowicz (US)
John Heden (US)

SMART CONSUMABLES
Sponsor: Tom Walter

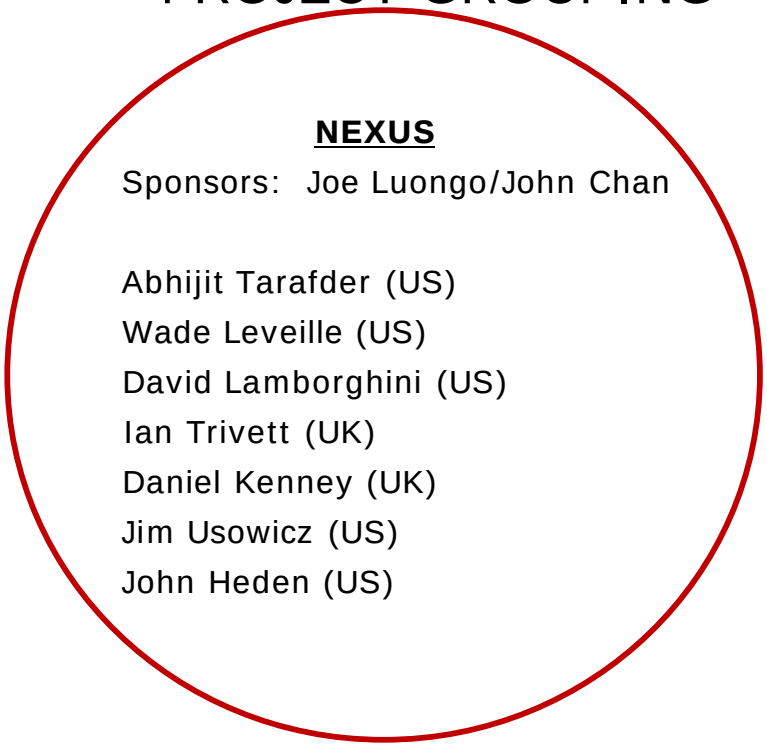
Jacob Fairchild (US)
Jason Hill (US)
Mathew DeLano (US)
Jan Driesen (GERMANY)
Alex Hooper (UK)
Mark Pickering (UK)
Senthil Bala (US)

©2015 Waters Corporation

WPI SYSTEMS ENGINEERING

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

PROJECT GROUPING



SYSTEMS CONTROLLER

Sponsor: Allen Caswell

Cindy Miller (US)
Richard Moulds (UK)
Richard Chapman (UK)
Steve Binney (US)
Justin Deary (US)
Phillip Young (UK)
David Gordon (UK)

NEXUS

Sponsors: Joe Luongo/John Chan

Abhijit Tarafder (US)
Wade Leveille (US)
David Lamborghini (US)
Ian Trivett (UK)
Daniel Kenney (UK)
Jim Usowicz (US)
John Heden (US)

SMART CONSUMABLES

Sponsor: Tom Walter

Jacob Fairchild (US)
Jason Hill (US)
Mathew DeLano (US)
Jan Driesen (GERMANY)
Alex Hooper (UK)
Mark Pickering (UK)
Senthil Bala (US)

©2015 Waters Corporation

Waters
THE SCIENCE OF WHAT'S POSSIBLE.®

PROJECT GROUPING

SYSTEMS CONTROLLER
Sponsor: Allen Caswell

Cindy Miller (US)
Richard Moulds (UK)
Richard Chapman (UK)
Steve Binney (US)
Justin Deary (US)
Phillip Young (UK)
David Gordon (UK)

NEXUS
Sponsors: Joe Luongo/John Chan

Abhijit Tarafder (US)
Wade Leveille (US)
David Lamborghini (US)
Ian Trivett (UK)
Daniel Kenney (UK)
Jim Usowicz (US)
John Heden (US)

SMART CONSUMABLES
Sponsor: Tom Walter

Jacob Fairchild (US)
Jason Hill (US)
Mathew DeLano (US)
Jan Driesen (GERMANY)
Alex Hooper (UK)
Mark Pickering (UK)
Senthil Bala (US)

©2015 Waters Corporation

Session 5 - System Integration and Test: Learning Objectives

- **Develop strategies and plans for system integration, validation and verification testing**
- **Manage system test strategies and plans through the system lifecycle**
- **Express system test strategies and plans in appropriate artifacts**

Sessions 2 through 5 Agenda (Revised)

- **Day 1**
  - **Review project team assignments from previous Session**
  - **Presentation and discussion of Session topic**
  - ~~**Project team working sessions on the application of the Session topics to their projects**~~
  - **Day 1 wrap-up and Q&A**

- **Day 2**
  - ~~**Review Day 1 topic and project working sessions**~~
  - **Presentation and discussion of Session topic (concluded)**
  - **Project team working sessions**
  - **Check-in with class on the application of the Session topic to their projects**
  - **Session wrap-up: Q&A and homework due next Session**

Timing will be similar to session 1, and will be announced at beginning of each session

Session 5 – System Integration and Test

Lessons

1. **Program Overview Update**
2. **Process Overviews**
3. **Integration, Planning, Management and Readiness**
4. **System Test Methods**
5. **Test Measurement Techniques and Metrics (if time permits)**

Appendices*

K. **Critical Systems Validation**

L. **Security Testing**

- Relationships with SE Lifecycle
- V&V Processes
- System Interfaces
- Technical Management and Test Processes

- Integration and Test Planning
- Requirements Management
- Interface Management
- Technical Readiness

- White Box Testing
- Black Box Testing
- Integration Testing

- TPM Metrics
- Quality Metrics
- Planning and Control Metrics

*Appendices A through J were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered.

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		

All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?



Systems Engineering Technical Short Course

(Session 5: System Integration and Test,

AKA: Verification, Validation and Test - VVT)

Lesson 2:
Technical Process Overview

Session 5 – System Integration and Test

- Relationships with SE Lifecycle
- V&V Processes
- System Interfaces
- Technical Management and Test Processes

Lessons

1. **Program Overview Update**
2. **Process Overviews**
3. **Integration, Planning, Management and Readiness**
4. **System Test Methods**
5. **Test Measurement Techniques and Metrics (if time permits)**

Appendices*

K. **Critical Systems Validation**

L. **Security Testing**

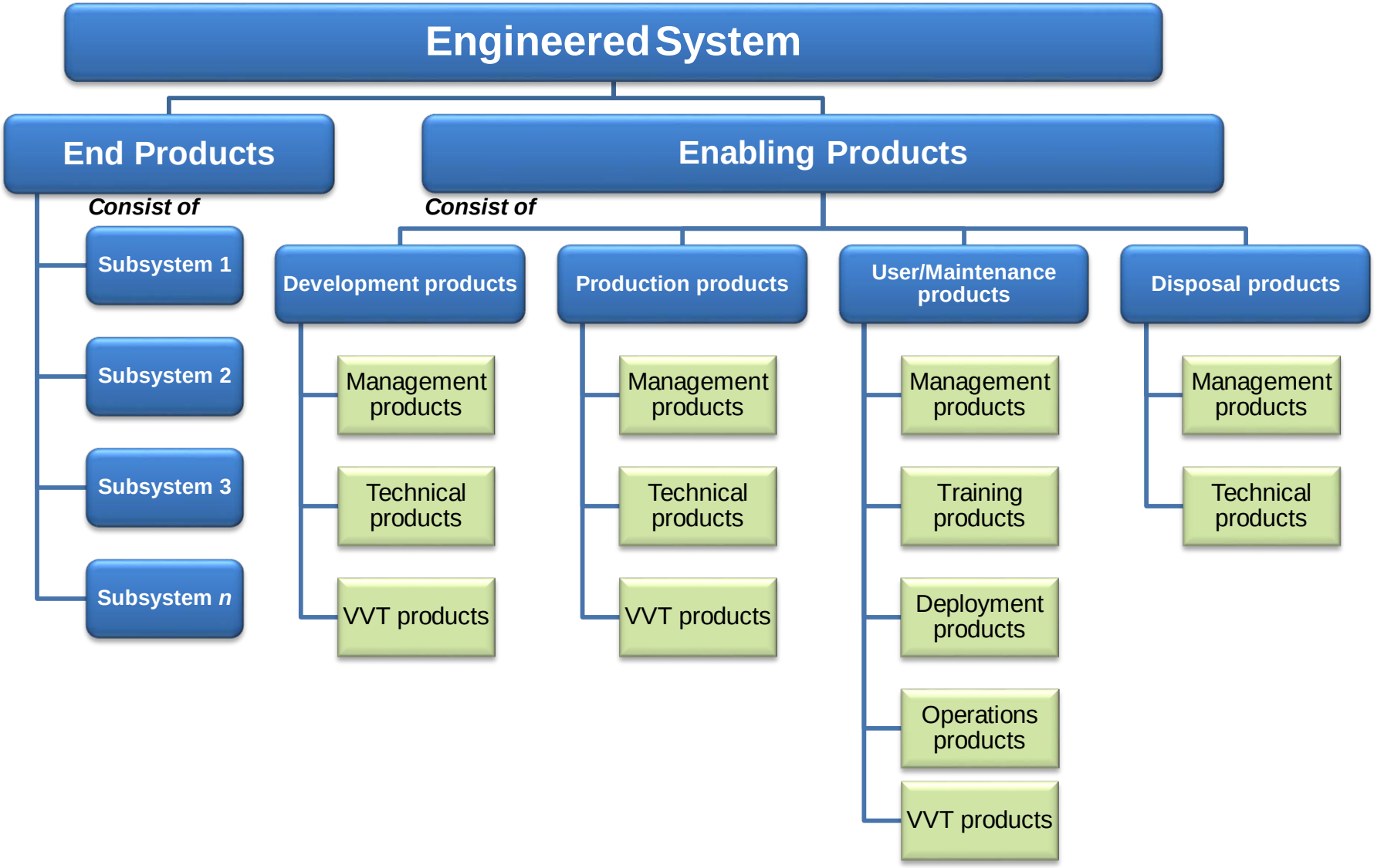
*Appendices A through J were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered.

Engineered Systems

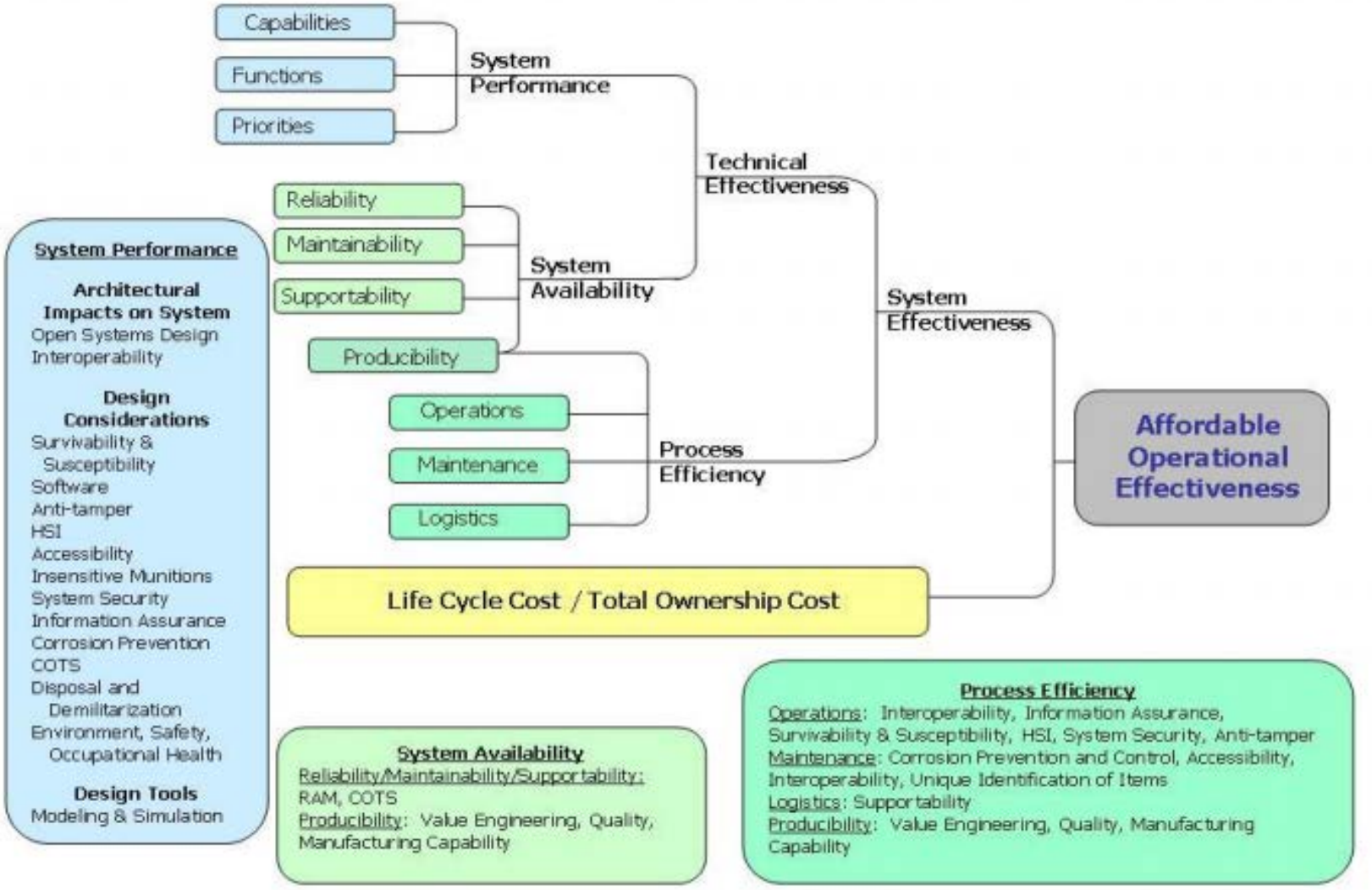
- **The goal of engineering processes is to develop and produce efficient and reliable system (products, services or solutions) that meet a specific need under a defined set of constraints**

- **Engineered systems often are composed of 'enabling products' required to provide lifecycle support in additional to the 'end products', which performs the required operational functions**

Engineered System Structure

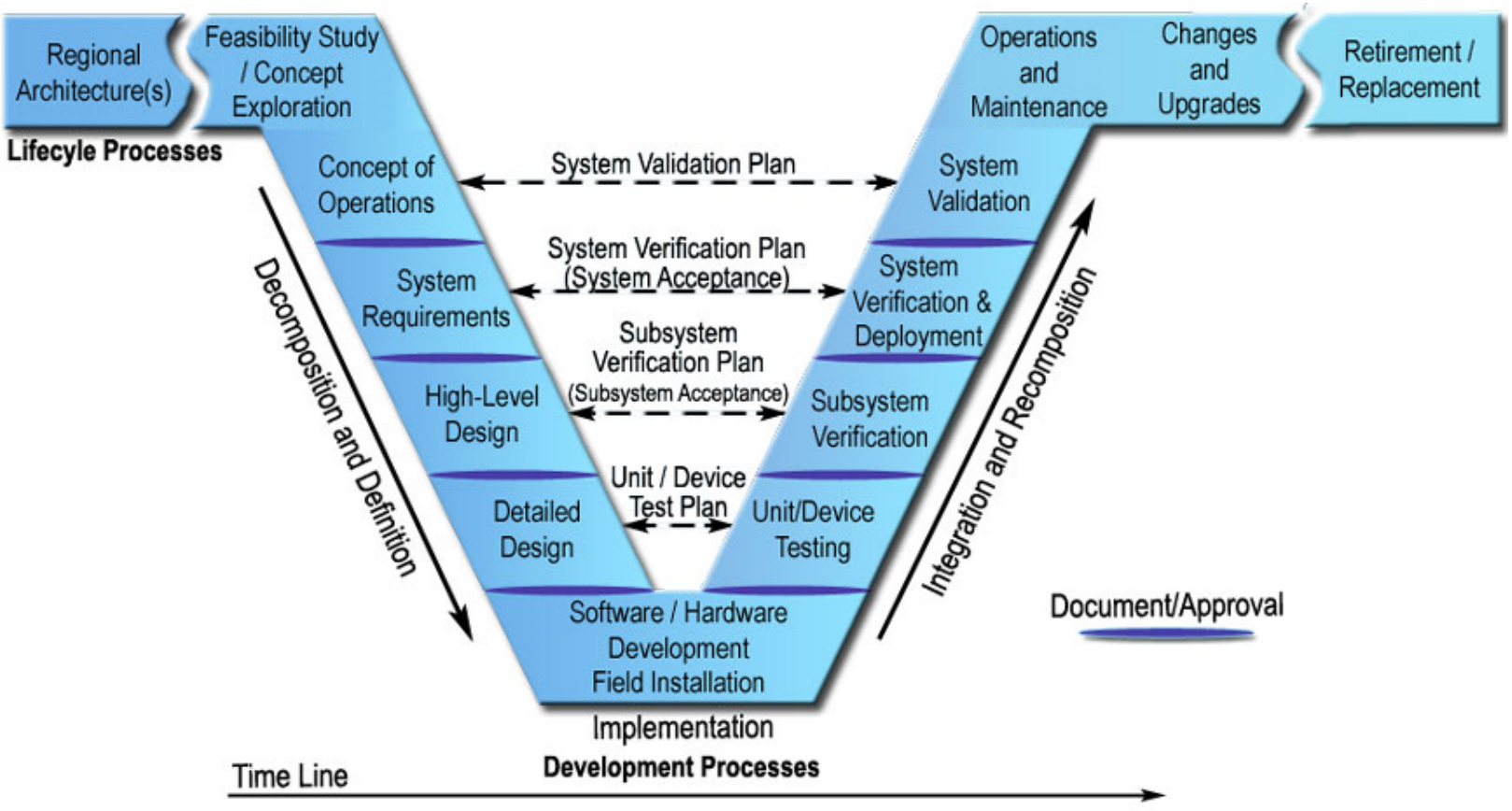


Design Considerations



SOURCE: Skalamera, Robert J. November 2004. "Implementing OSD Systems Engineering Policy" briefing to the USD(AT&L).

V & V Throughout the Lifecycle



True or False?

- Testing is a subset of Verification & Validation?
  - True
- Testing is something that happens late in a products development?
  - False – various levels of testing occur throughout a systems lifecycle
- The level of effort needed to plan for testing is minimal early in a program?
  - False – early planning forms the foundation of a successful test process

System Life Cycle Processes (ISO/IEC 15288)

Agreement Processes

Acquisition

Supply

Enterprise Processes

Enterprise Environment Management

Investment Management

Systems Life Cycle Process Management

Resource Management

Quality Management

Project Processes

Planning

Assessment and Control

Decision Management

Risk Management

Configuration Management

Information Management

Measurement

Technical Processes

Requirements Definition

Requirements Analysis

Architectural Design

Implementation

Integration

Verification

Transition

Validation

Operation

Maintenance

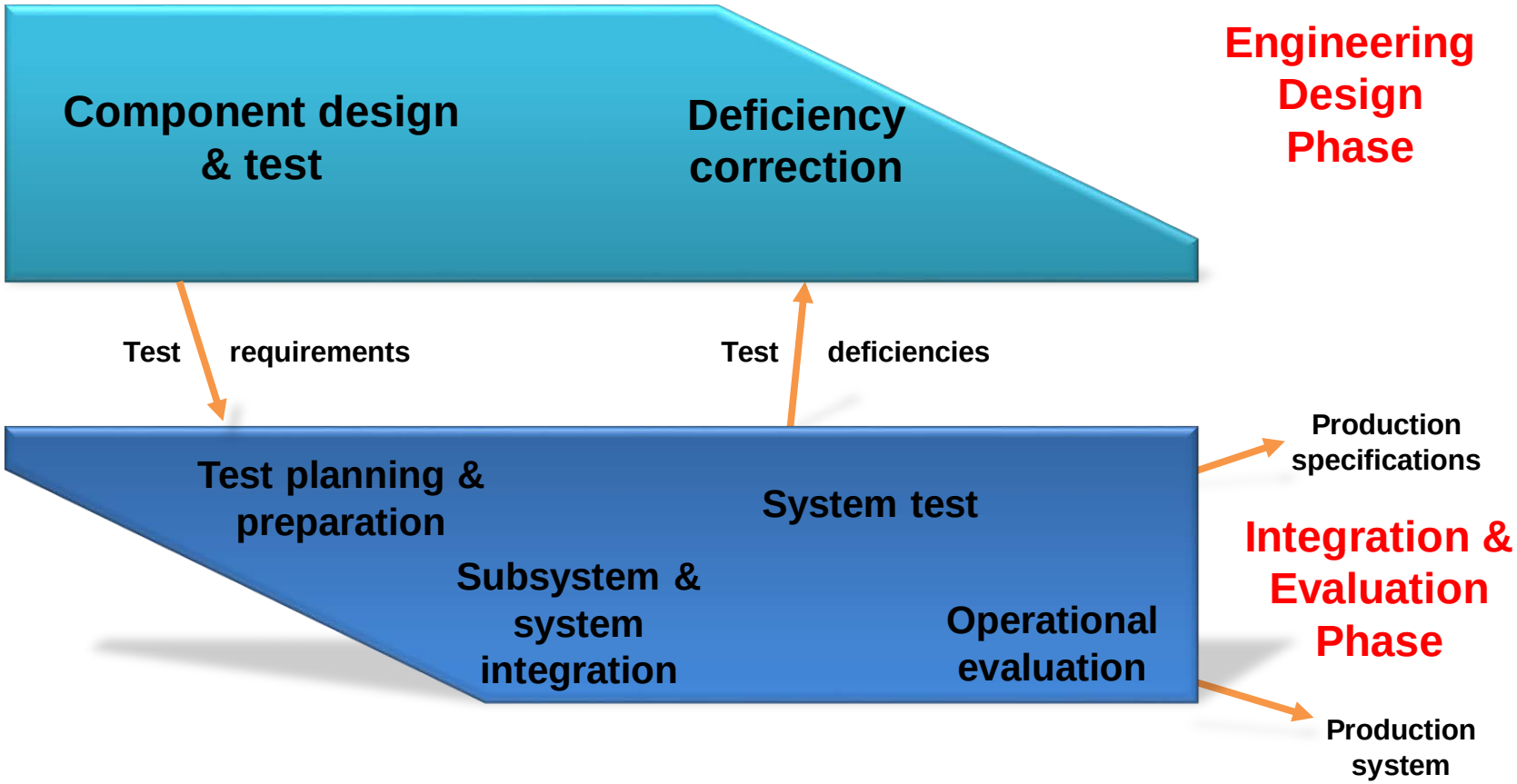
Disposal

Definitions

- **Verification - A quality process that is used to evaluate whether or not a product, service, or system complies with a regulation, specification, or conditions imposed at the start of a development phase. Verification can be in development, scale-up, or production.**
  - **Are we building the product right?**
- **Validation - The process of establishing documented evidence that provides a high degree of assurance that a product, service, or system accomplishes its intended requirements. This often involves acceptance and suitability with external customers.**
  - **Are we building the right product?**
- **Test - the process of gathering data, both quantitative and qualitative, relevant to developing new capabilities**
- **Evaluation - the process whereby data are logically assembled, analyzed, and compared to expected performance to aid in making systematic decisions**
- **Integration - The bringing together of the component subsystems into one system and ensuring that the subsystems function together as a system.**

See Notes Page

Integration Phase in Relation to Engineering Design



Reference: Kossiakoff, A., & Sweet, W. N. (2003). *Systems Engineering: Principles & Practice,* p. 275.

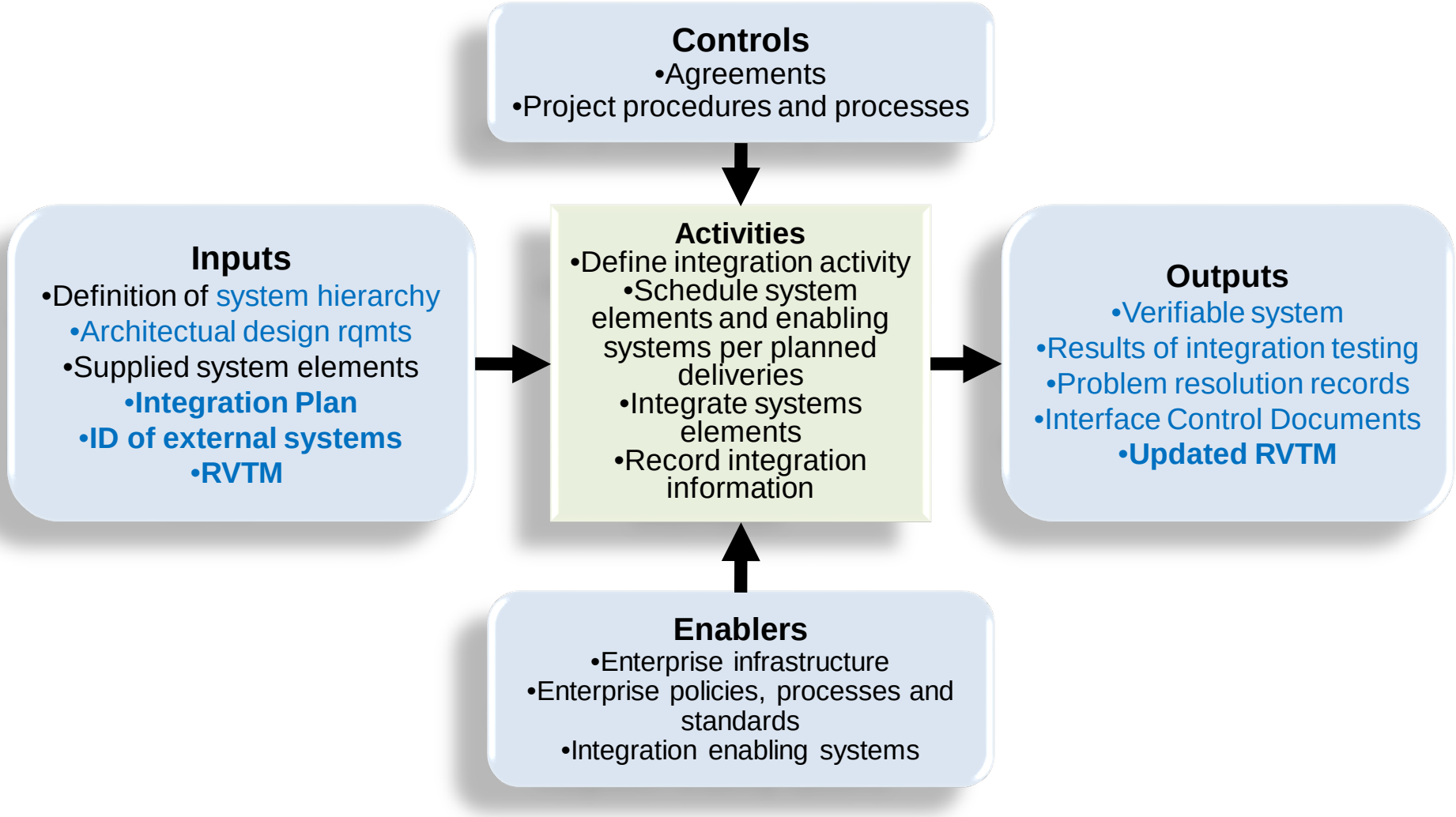
Integration

- The purpose of the Integration Process is to realize the system by progressively combining system elements in accordance with the architectural design requirements and the integration strategy
- This process is successively repeated in combination with the Verification and Validation Processes as appropriate
- Integration begins with early functional analysis.
- Early in the life cycle simulations, prototypes and mockups are used and built upon until the final product is achieved
- Ultimately we want to ensure the individual system components will function as an integrated system or system of systems

Objectives for Systems Integration

- Encompass the entire life cycle through planning, design, construction, test, deployment and maintenance
- Support both top down and bottoms up design philosophy
- Support definition and documentation of all aspects of the program
- Support compliance with fundamental SE processes such as requirements management, quality, risk management and configuration management
- Support capturing of problems, issues and performance shortfalls and communication between all parties at all levels
- Support a comprehensive audit trail

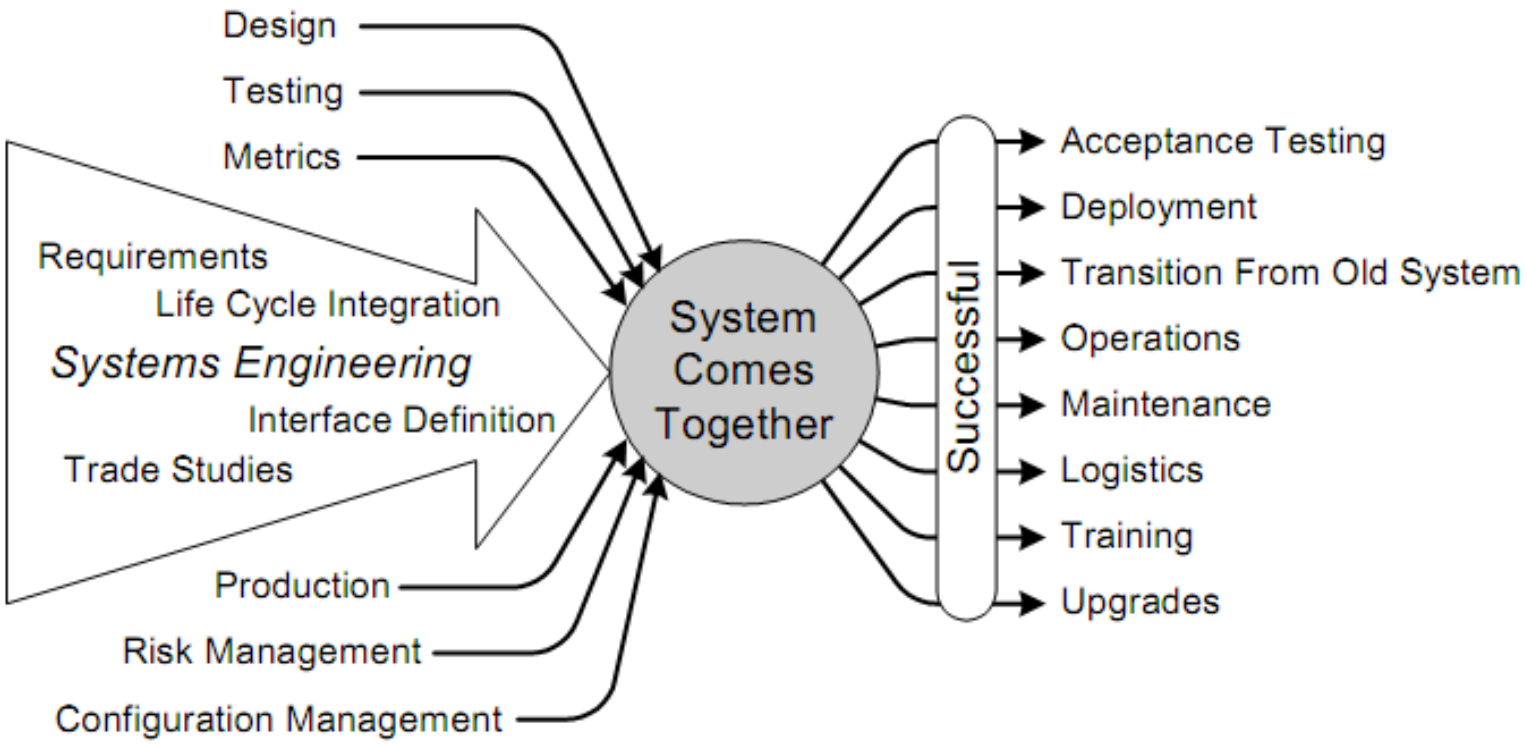
Integration Process



Integration Process

- Inputs
  - System Architectural artifacts
    - DOD Architectural Framework views
    - Use cases
    - Other commercial architectural products
  - Interface specifications
  - Integration plan – may be part of a SEP
- Outputs
  - Integration testing results
  - Validated interfaces
  - Updated system drawings, ICD's and requirements verification and traceability matrix (Updated Technical Data Package)
  - Documentation of problems/issues/non-compliance encountered

Systems Integration Inputs and Results



Integration Challenges

- Maintaining requirements traceability to specific tests
- Determining the lowest level for meaningful integration (and test)
- How to minimize the integrate-test-correct.....integrate-test-correct syndrome
- Need for concrete plan for isolating problems and rapid re-testing.
- For hardware particularly, plan for strategically located and functionally oriented test points
- Careful selection of interface points during architecture and design
- Maintaining configuration control - software, hardware, documentation
- Identifying, mitigating and managing risk

Integration Pitfalls

- No plan for strategically placed test points
- Not designing for the final system environment
- Neglecting the user
  - Age, size, physical condition, attitudes, ...
  - Disabilities (not always obvious - e.g., color blindness)
  - Education
- Not maintaining documented interfaces
- Not maintaining requirements traceability
- Not specifying measurable acceptance criteria with the customer or with subcontractors
- Lack of functional modularity to ease fault isolation
- Insufficient and/or improper integration planning
- Inadequate lower level testing
- Insufficient or nonexistent configuration management

Integration Lessons Learned

- More is better when design of test points is involved (both hardware and software)
- Lower level component tests along the way is better
  - Minimize need for system-level tests
- Build-a-little, test-a-little to aid test verification
- Systems Engineer guides process through System Design Team leadership and the System Engineering Management Plan (SEMP) and a System Integration Plan
- Integration is a team effort which involves the user and all developers (including subcontractors)
- WBS is the framework for the SEMP, TEMP, and Integration Plan
- Measurable and mutually understood acceptance criteria with customer and subcontractors minimizes future disconnects

Session 5 – System Integration and Test

- Relationships with SE Lifecycle
- V&V Processes
- System Interfaces
- Technical Management and Test Processes

Lessons

1. **Program Overview Update**
2. **Process Overviews**
3. **Integration, Planning, Management and Readiness**
4. **System Test Methods**
5. **Test Measurement Techniques and Metrics (if time permits)**

Appendices*

K. **Critical Systems Validation**

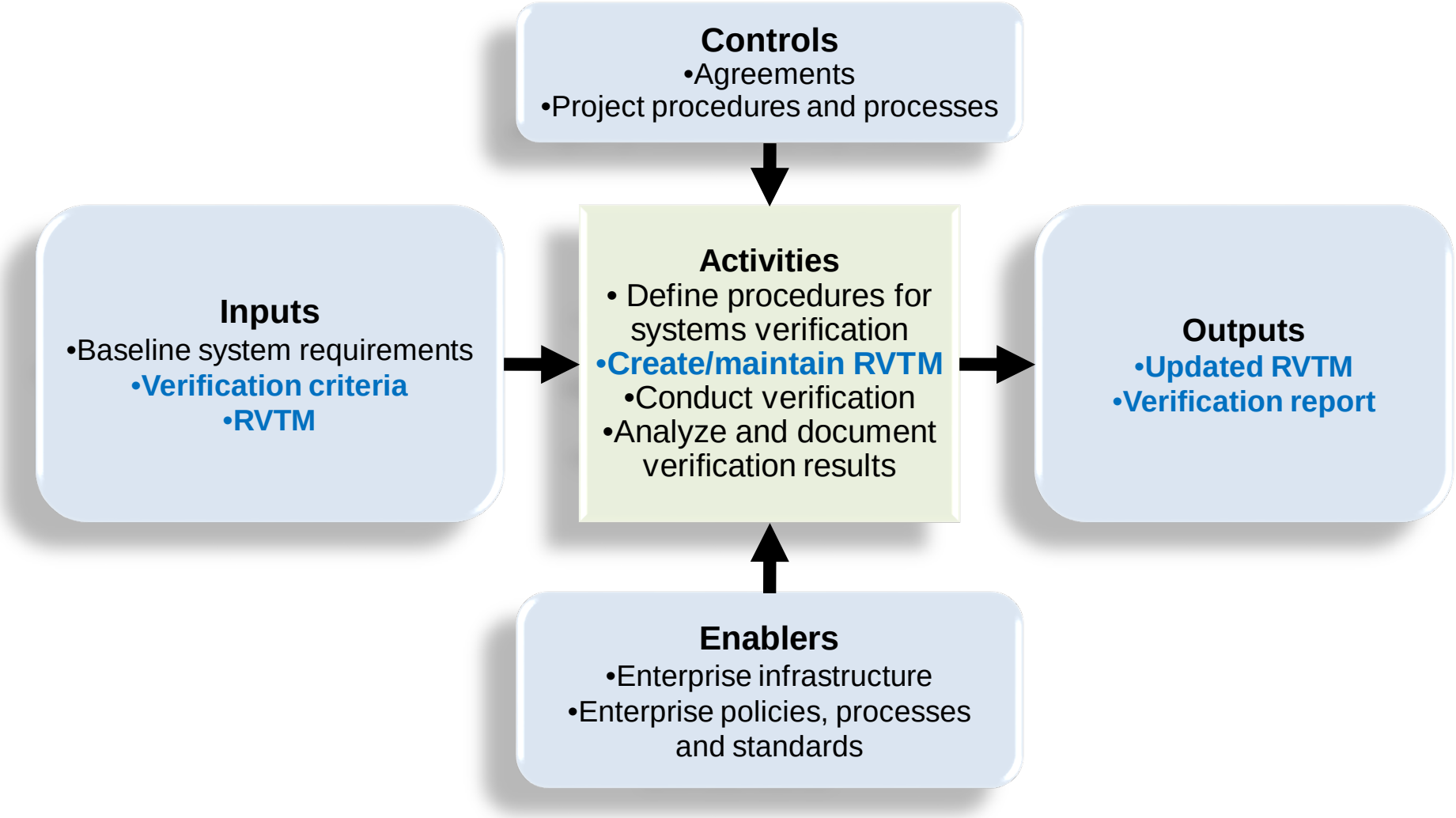
L. **Security Testing**

*Appendices A through J were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered.

Verification

- Has the system been built right?
- Confirm that the documented requirements have been achieved and, if not, document the shortfalls
- Verification methods include:
  - Test
  - Inspection
  - Analysis
  - Demonstration
- What gets verified and how it gets done is a function of risks, safety, and the importance of a particular element/subsystem

Verification Process



Verification Process

- **Inputs**
  - **Baselined requirements**
  - **RVTM**
  - **Policy, directives, statute**

- **Outputs**
  - **Verification report**
  - **Documentation of requirements shortfalls and corrective actions**
  - **Updated RVTM**

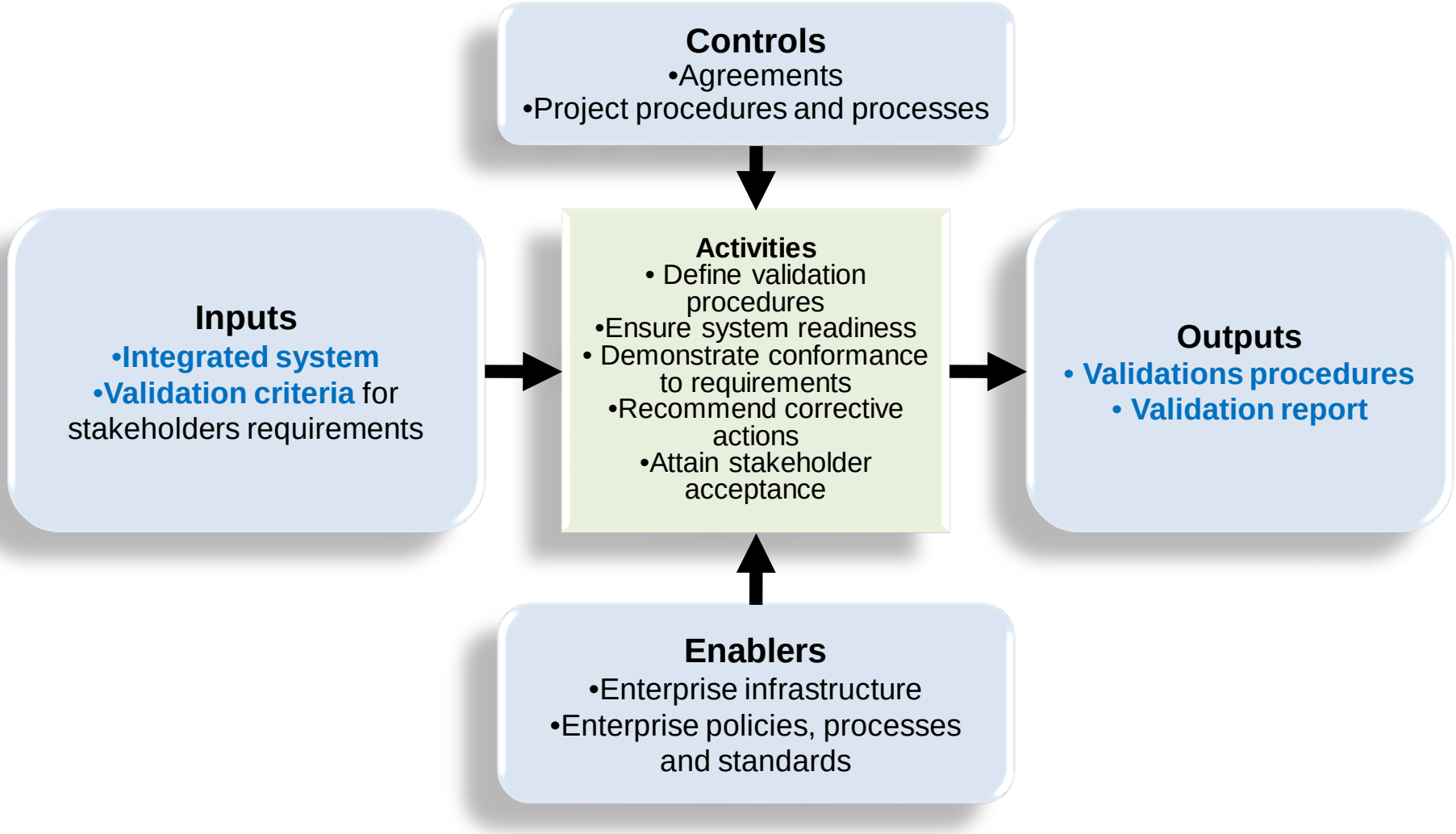
Validation Process Objectives

- **The objectives of the Validation Process are to confirm:**
  - **That the right product was realized -- the one wanted by the customer**
  - **That the realized product can be used by intended operators/users**
  - **That the system's KPPs, MOEs and MOPs are satisfied**
  - **That the realized product fulfills its intended use when operated in its intended environment**

- **To generate evidence to confirm that products at each layer of the system meet the capability and other operational expectations of the customer/user/operator and other interested parties**

- **To ensure that any problems discovered are appropriately resolved:**
  - **Prior to delivery of the realized product (if validation is done by the supplier of the product), or**
  - **Prior to integration with other products into a higher-level assembled product**

Validation Process



Validation Process

- **Inputs**
  - **Stakeholder requirements**
    - **In DoD this may be a Capability Description Document (CDD)**
- **Outputs**
  - **Validation report**
  - **Design feedback and corrective actions**
  - **Approved baseline**

Validation Techniques

- **Formal methods** - Formal methods means the use of mathematical and logical techniques to express, investigate, and analyze the specification, design, documentation, and behavior of both hardware and software

- **Fault injection** - Intentional activation of faults by either hardware or software means to observe the system operation under fault conditions

- **Dependability analysis** - Involves identifying hazards and then proposing methods that reduces the risk of the hazard occurring

- **Hazard analysis** - Involves using guidelines to identify hazards, their root causes, and possible countermeasures

- **Risk analysis** - Identifying the possible consequences of each hazard and their probability of occurring

Who Does Validation?

- **Validation is conducted by the user/operator or by the developer**
- **System-level validation (e.g., Operational Test & Evaluation and some other types of validation) is almost always performed by user/operator**
- **For those portions of validation done by the developer, appropriate agreements must be negotiated to ensure that validation proof-of-documentation is delivered with the product**

Session 5 – System Integration and Test

- Relationships with SE Lifecycle
- V&V Processes
- System Interfaces
- Technical Management and Test Processes

Lessons

1. **Program Overview Update**
2. **Process Overviews**
3. **Integration, Planning, Management and Readiness**
4. **System Test Methods**
5. **Test Measurement Techniques and Metrics (if time permits)**

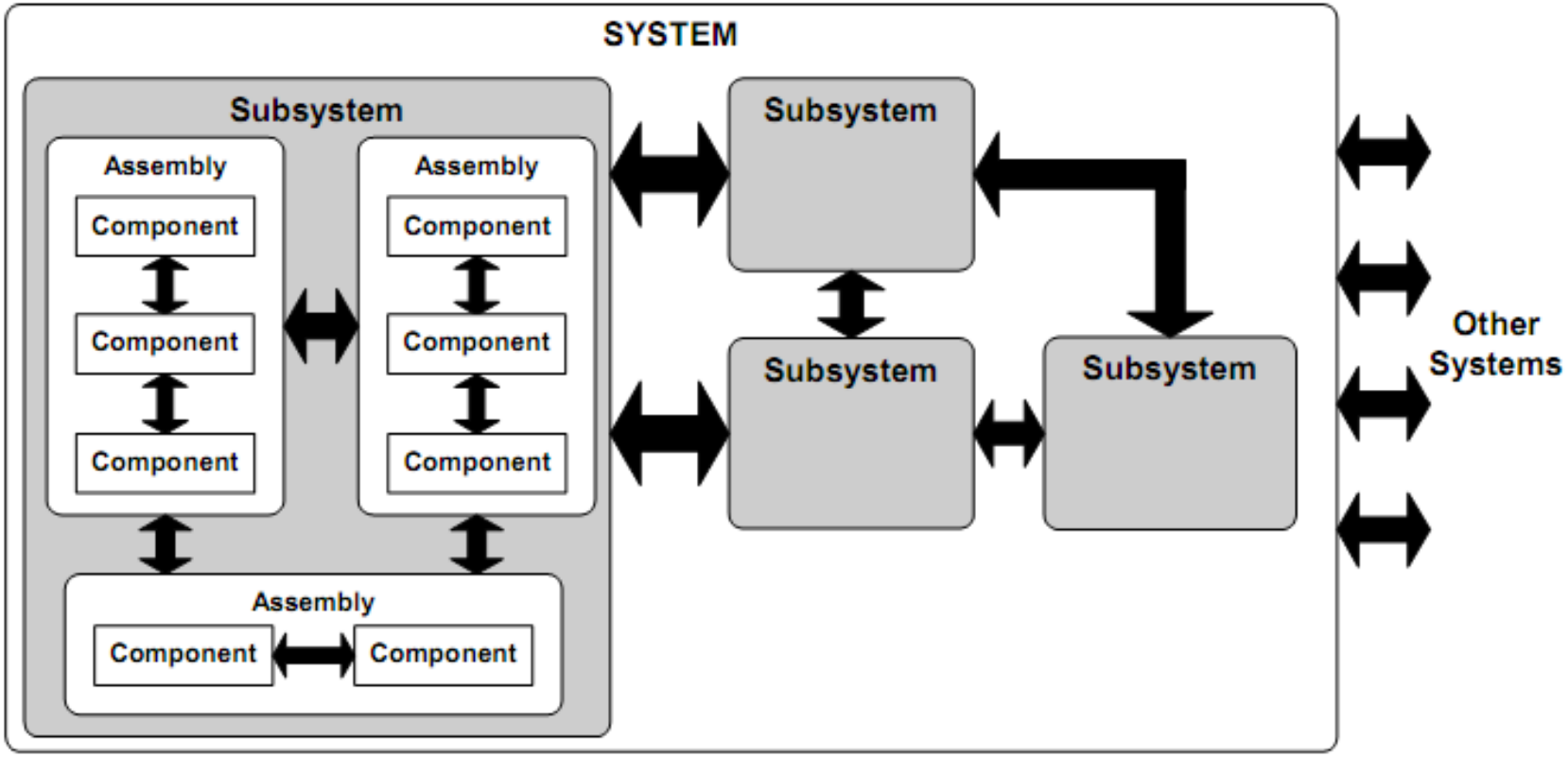
Appendices*

K. **Critical Systems Validation**

L. **Security Testing**

*Appendices A through J were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered.

System Interface Identification

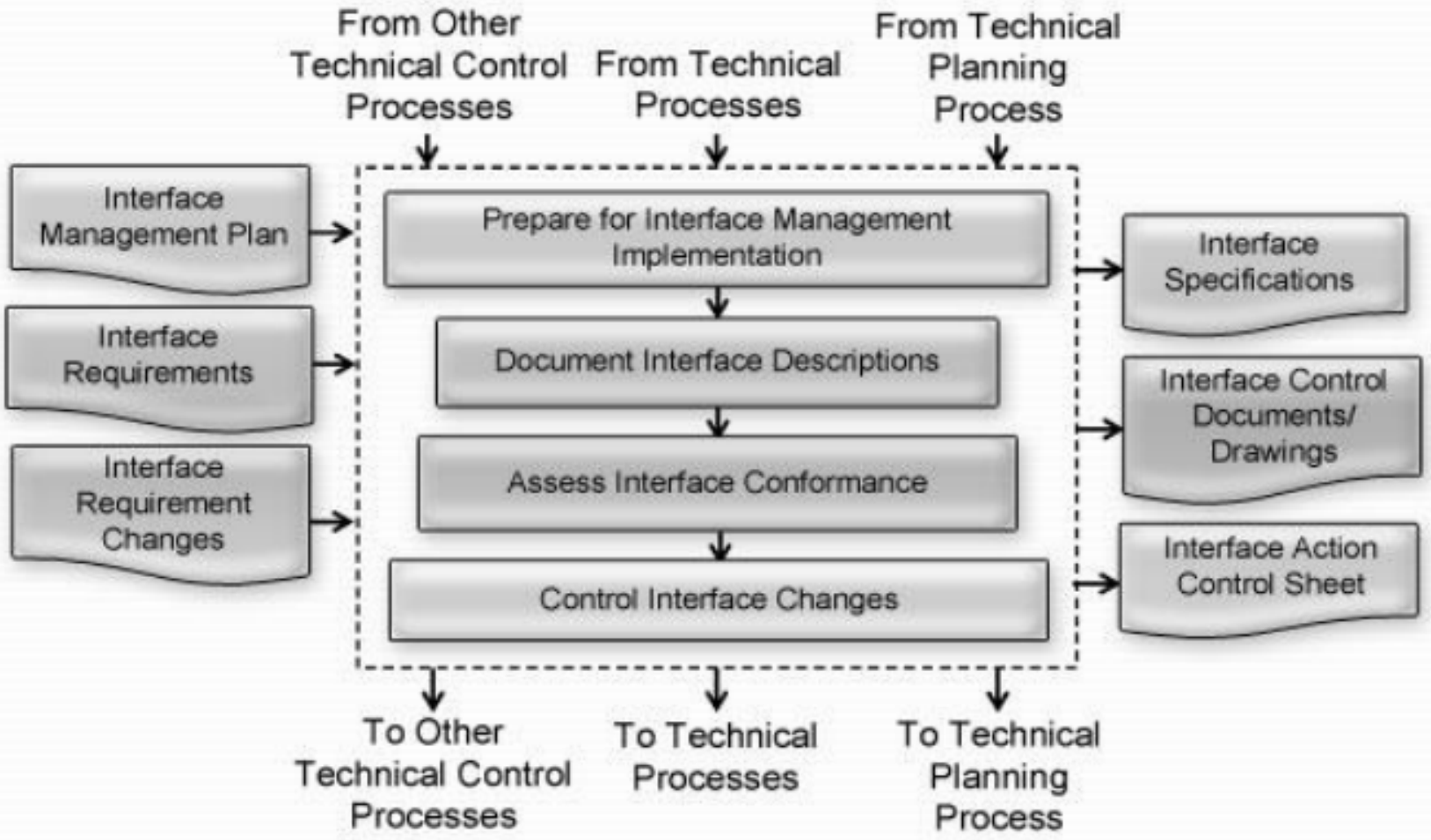


- An absolute essential to any integration effort is complete knowledge of all interfaces.
- Includes interfaces between components, assemblies, subsystems, and between the system and other systems it will need to work with

Interface Management

- **Interface failures are one of the key reasons systems fail V&V**
- **Identification and management of internal and external interfaces is critical to program success**
- **Interface management encompasses**
  - **Interface identification**
  - **Interface assessment and control (Configuration Management)**
  - **Communication and agreements as needed**
- **Interfaces should be properly documented**
  - **Functional, physical, electromagnetic, data, human and interoperability requirements/characteristics**

Interface Management Inputs/Outputs



Documenting Interface Descriptions

- **Integrate:** Integrate interface management procedures with the requirements management procedures
- **ID:** Identify, capture and document interfaces, both external to the system model and internal to the system model products
- **Analyze:** Analyze the Concept of Operations (CONOPS) and similar documents to identify interfaces not included in the original set of Stakeholder Requirements
-
- **Capture:** Identify and capture the requirements for the identified interfaces including origin, destination, stimulus and special characteristics based on the type of interface
- **Establish:** Document the design solution interfaces for the system model under development

Documenting Interface Descriptions *continued*

- **Include:** Ensure that the design solution for the end product includes the interface requirements defined during Requirements Development and Logical Analysis and includes the requirement origin, destination, stimulus and special characteristics

- **Trace:** Maintain horizontal traceability of interface requirements across interfaces and document status in the established program requirements compliance matrix

- **Validate ICD:** Ensure that each Interface Control Document (ICD) or drawing that is established has been validated with parties on both sides of the interface

- **Distribute:** Provide authorized users with the needed interface information for integration into technical efforts and for external interface control

Controlling Interface Changes

- **Manage:** Monitor proposed changes (e.g., ECPs) to system model requirements, looking for those affecting established interfaces
- **ID & Resolve:** Analyze interface conflicts, non-compliance and change issues; propose changes to resolve discrepancies
- **Capture:** Identify and record proposed and directed changes to stakeholder or design solution interface requirements/specifications and Interface Control Documents/Drawings
- **Analyze Impacts:** Analyze the cost, schedule, performance and risks associated with making a proposed or directed interface change within planned time limits and resource availability

Controlling Changes *continued*

- **Both Sides:** Ensure that the interface issues are analyzed/ resolved when a change affects products on both sides of the interface
- **Traceability:** Maintain/control traceability of changes including source of the change, processing methods/approvals in accordance with the Interface Management Plan
- **Maintain:** Ensure the consistency of the interfaces throughout the life of the product
- **Repository:** Establish and maintain a repository for interface data.
- **Disseminate:** Distribute approved interface change information/data for implementation at every level of the program and for integration into technical efforts.

Interface Examples

- **Interface examples include:**
  - **Mechanical Interfaces**
  - **Noise Interfaces**
  - **Climatic Interfaces**
  - **Thermal Interfaces**
  - **Computer Interfaces**
  - **Fluid Interfaces**
  - **Electrical Interfaces**
  - **Electromagnetic Interfaces**
  - **Human-Machine Interfaces**

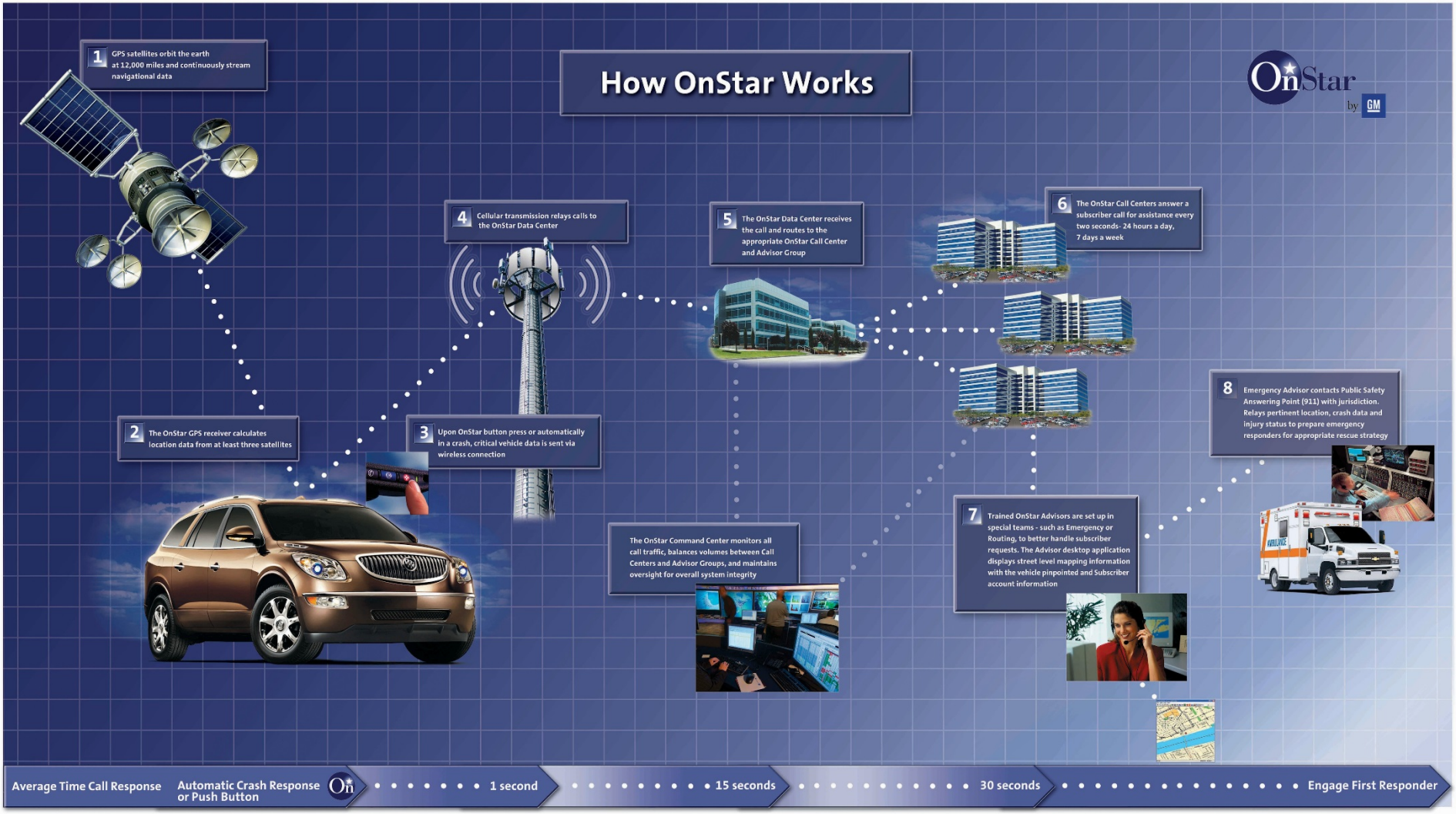


- **Any of these could be internal or external**

Interface Summary

- System Interfaces are the Weakest Link in the System
- System Interfaces need to be developed in the beginning of life cycle
- Interface development is difficult because it involves multiple (independent) Sub-systems
- System Interfaces need to be managed, controlled and verified

OnStar System Exercise:
System Interfaces



OnStar System Exercise:
System Interfaces

1. GPS satellites continuously orbit the earth at 12,000 miles and continuously stream navigational data.

2. The OnStar GPS receiver calculates location data from at least three satellites.

3. Upon OnStar button press or automatically in a crash, critical vehicle data is sent via wireless connection.

4. Cellular transmission relays calls to the OnStar Data Center.

5. The OnStar Data Center receives the call and routes the call to the appropriate OnStar Call Center and Advisor Group.

   The OnStar Command Center monitors all call traffic, balances volumes between Call Centers and Advisor Groups, and maintains oversight for overall system integrity.

6. The OnStar Call Centers answer a subscriber call for assistance every two seconds – 24 hours a day, 7 days a week.

7. Trainer OnStar Advisors are set up in special teams – such as Emergency or Routing, to better handle subscriber requests. The Advisor desktop application provides street level mapping information with the vehicle pinpointed and subscriber account information.

8. Emergency Advisor contacts Public Safety Answering Point (911) within jurisdiction. Relays pertinent local, crash data and injury status to prepare emergency responders for appropriate rescue strategy.

OnStar System Exercise:
System Interfaces

- Identify the major interfaces for the OnStar system, and indicate whether each interface is internal or external.
- Identify the origin, destination, and stimuli for each interface.
- Identify the special characteristics of each interface, such as
  - Functional, physical, electromagnetic, data, human and interoperability characteristics/requirements
- Be prepared to share your results with the class, including:
  - Descriptions of one or two of your major interfaces
  - Your observations for the exercise



Session 5 – System Integration and Test

- Relationships with SE Lifecycle
- V&V Processes
- System Interfaces
- Technical Management and Test Processes

Lessons

1. **Program Overview Update**
2. **Process Overviews**
3. **Integration, Planning, Management and Readiness**
4. **System Test Methods**
5. **Test Measurement Techniques and Metrics (if time permits)**

Appendices*

K. **Critical Systems Validation**

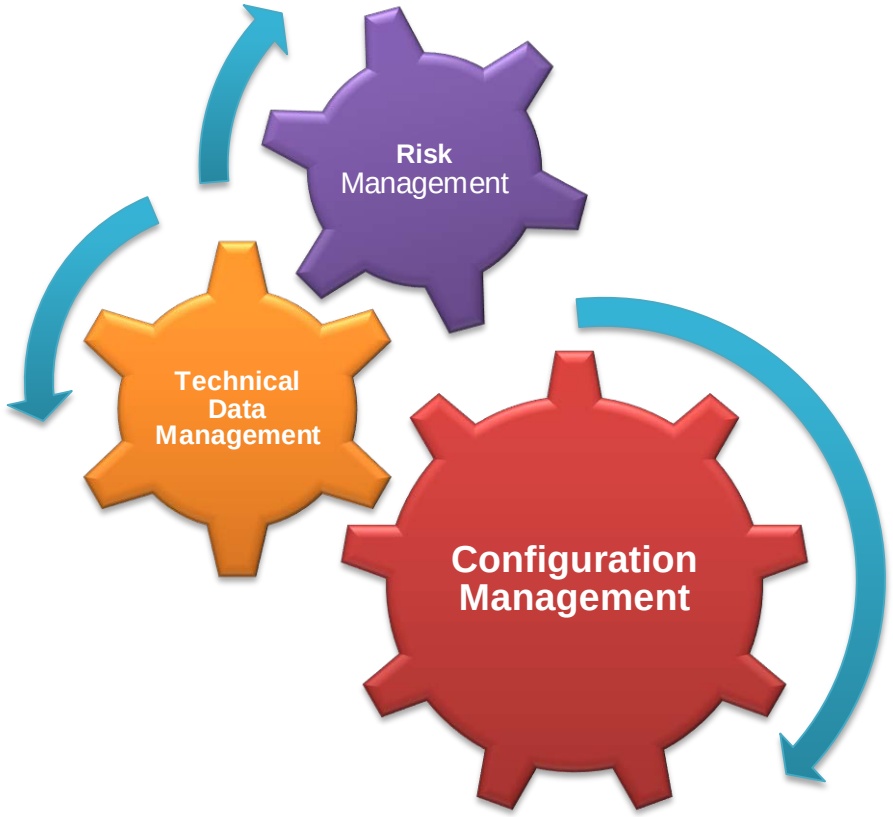
L. **Security Testing**

*Appendices A through J were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered.

Relationship of Technical Management Processes

- **Interface Management is but one of several interrelated Technical Management Processes that are used to manage the overall technical development of a system**

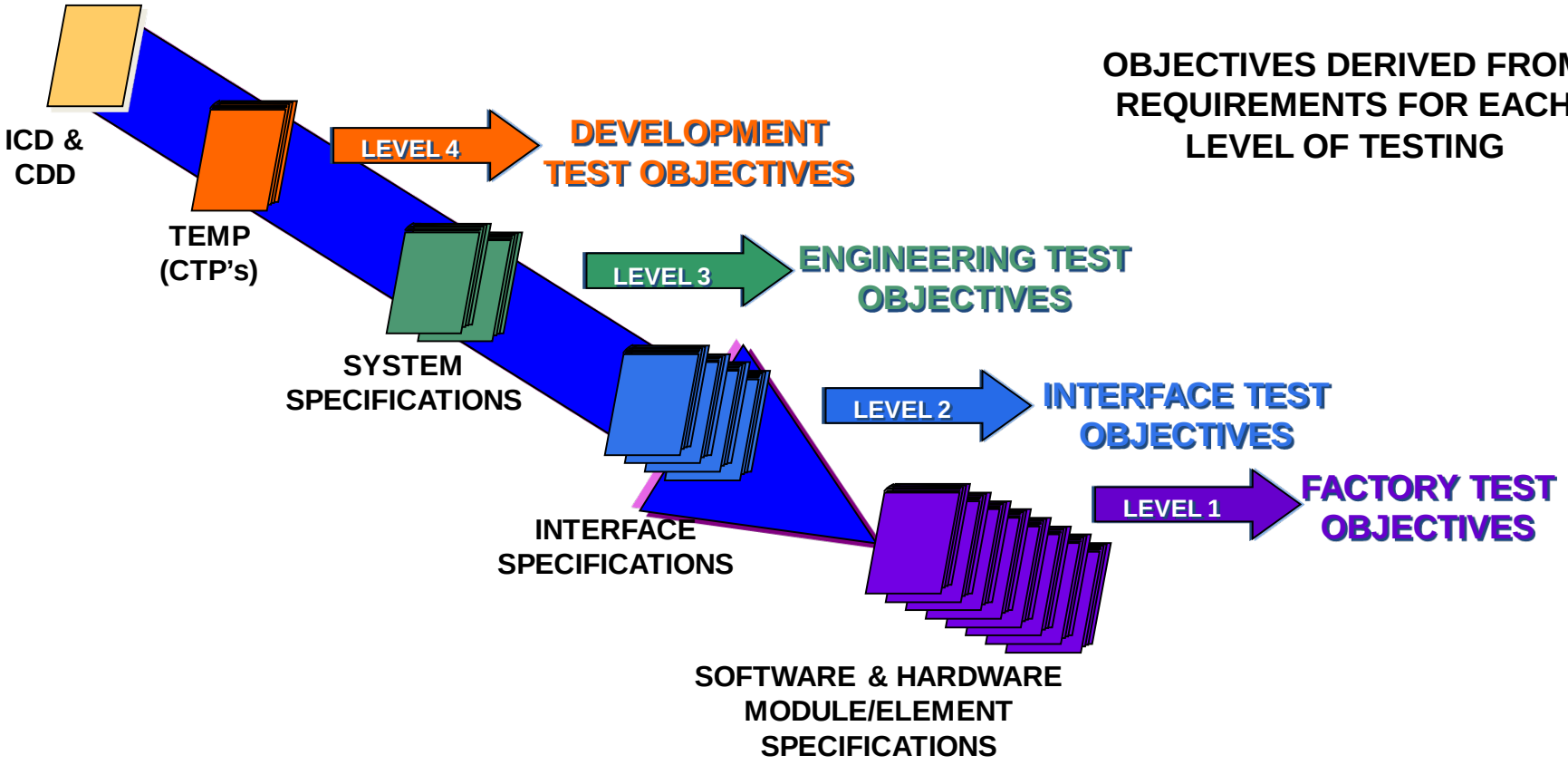
- **Risk Management, Configuration Management and Technical Data Management are key supporting players in the Interface Management Process**



Requirements Management

- Testing cannot, will not, be successful without a comprehensive requirements management process
- Test personnel are essential early in the requirements definition process to ensure requirements are testable and to start planning the test program
- A fundamental component of Systems Engineering which forms the foundation for the test program

Requirement/Test Objective Traceability



REQUIREMENTS TRACEABLE FROM TEMP THROUGH SYSTEM AND INTERFACE SPECIFICATIONS TO SOFTWARE & HARDWARE SPECIFICATIONS

Requirements Verification Traceability Matrix (RVTM)

Spec. Paragraph	Title	Verification Method				Verification Paragraph	Verification Task Number
		Test	Analys.	Demo	Exam		
1.0	Tag Behavior			●		V_1.0	12.0
1.1	RF Communications	●				V_1.1	4.0
1.1.1	Warehouse Environ.			●		V_1.1.1	11.0
1.1.2	Program w/MtagPro			●		V_1.1.2	11.1
1.1.3	Antenna Design		●			V_1.1.3	1.1
1.1.4	Antenna Size/Shape				●	V_1.1.4	1.0
1.1.5	RF Trans. Bandwidth	●				V_1.1.5	3.2
1.1.6	RF Trans. Carrier Freq.	●				V_1.1.6	3.0
1.1.7	RF Trans. SNR	●				V_1.1.7	3.1
1.1.8	RF Trans. FSK-4	●				V_1.1.8	3.3
1.2	Tag Data			●		V_1.1.2	11.1
1.2.1	Microcontroller Mem.				●	V_1.2.1	2.1
1.2.2	Microcontroller Mem.				●	V_1.2.2	2.0
1.3	Clock Frequency	●				V_1.3	4.1
1.4	Software				●	V_1.4	2.2
1.4.1					●		
1.4.2					●		
1.5	Tag Power Consump.	●				V_1.5	7.1
1.5.1	Battery Power	●	●			V_1.5.1	5.1A
						V_1.5.1	5.2T
1.5.2	Battery Size				●	V_1.5.2	5.0
1.5.3	Voltage Regulator	●			●	V_1.5.3	6.0E
						V_1.5.3	6.1T
1.5.4	Low-voltage Alarm	●				V_1.5.4	6.2
1.5.4.1		●				V_1.5.4.1	7.0
1.5.4.2		●					
1.6	Waterproof Enclosure	●				V_1.6	9.0
1.7	Tag Weight				●	V_1.7	10.0
1.8	Enclosure Dimensions				●	V_1.8	8.0

- **RVTM is not a process but a tool**
- **Each requirement must have a verification activity associated with it**
- **Essential for successful planning and determination if higher level requirements have been met – especially on complex systems**
- **Choose an activity that is the most cost effective mix of simulation and physical testing**

Configuration Management

- Through configuration management we establish and maintain our technical baseline
- Changes will occur during design and testing
- Engineers must know the impact of changes on the overall design and modify the test program accordingly
- Changes to the system need to be evaluated for regression testing
- Beware of rapid field changes that don't get documented or assessed for impacts

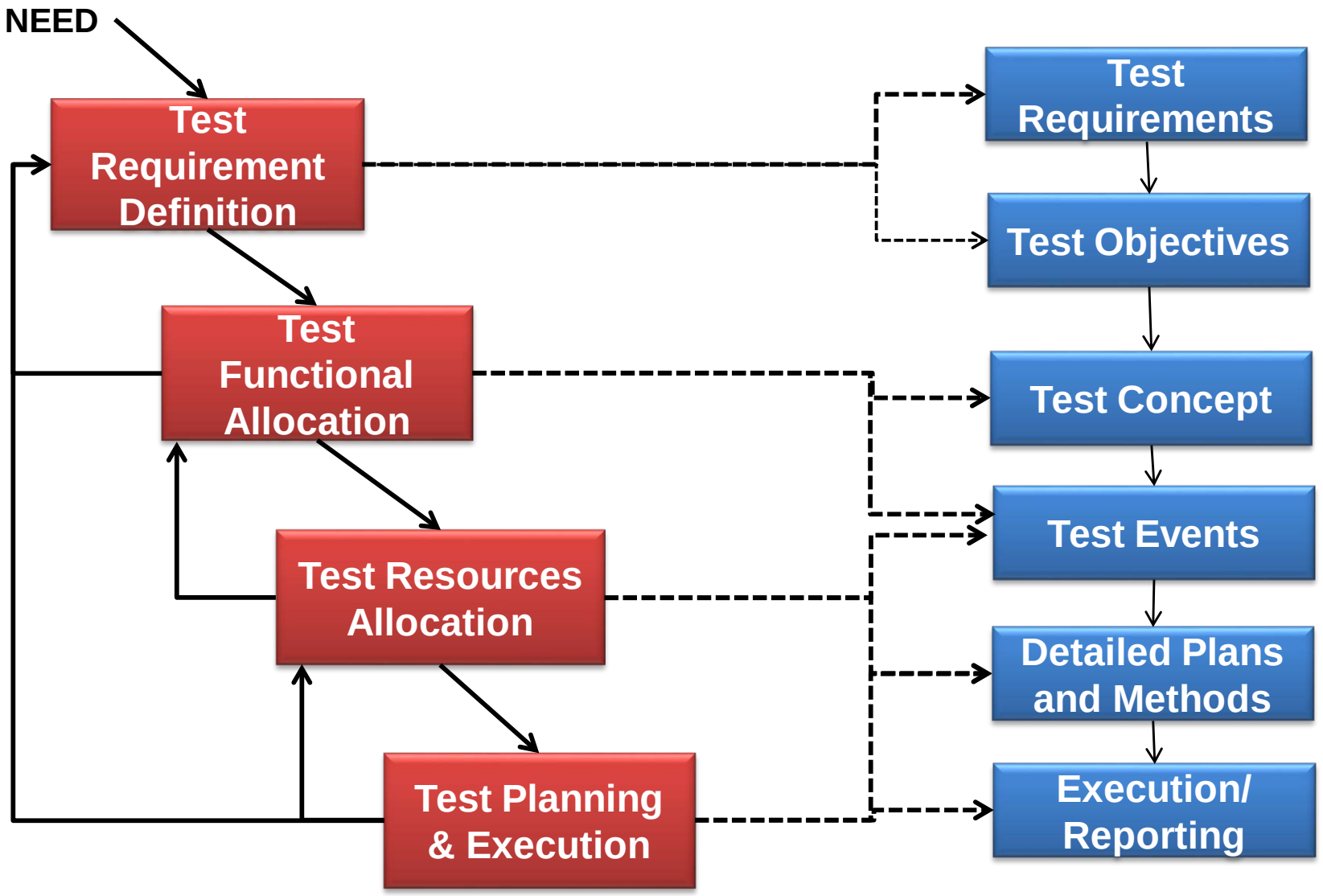
Risk Management

- **Early test planning will leverage the risk process to identify critical components, sub-systems and functions**



- **Parachute subsystem failure - Did they consider this a high risk area?**

Test Process Flow



Test & Evaluation (T&E)

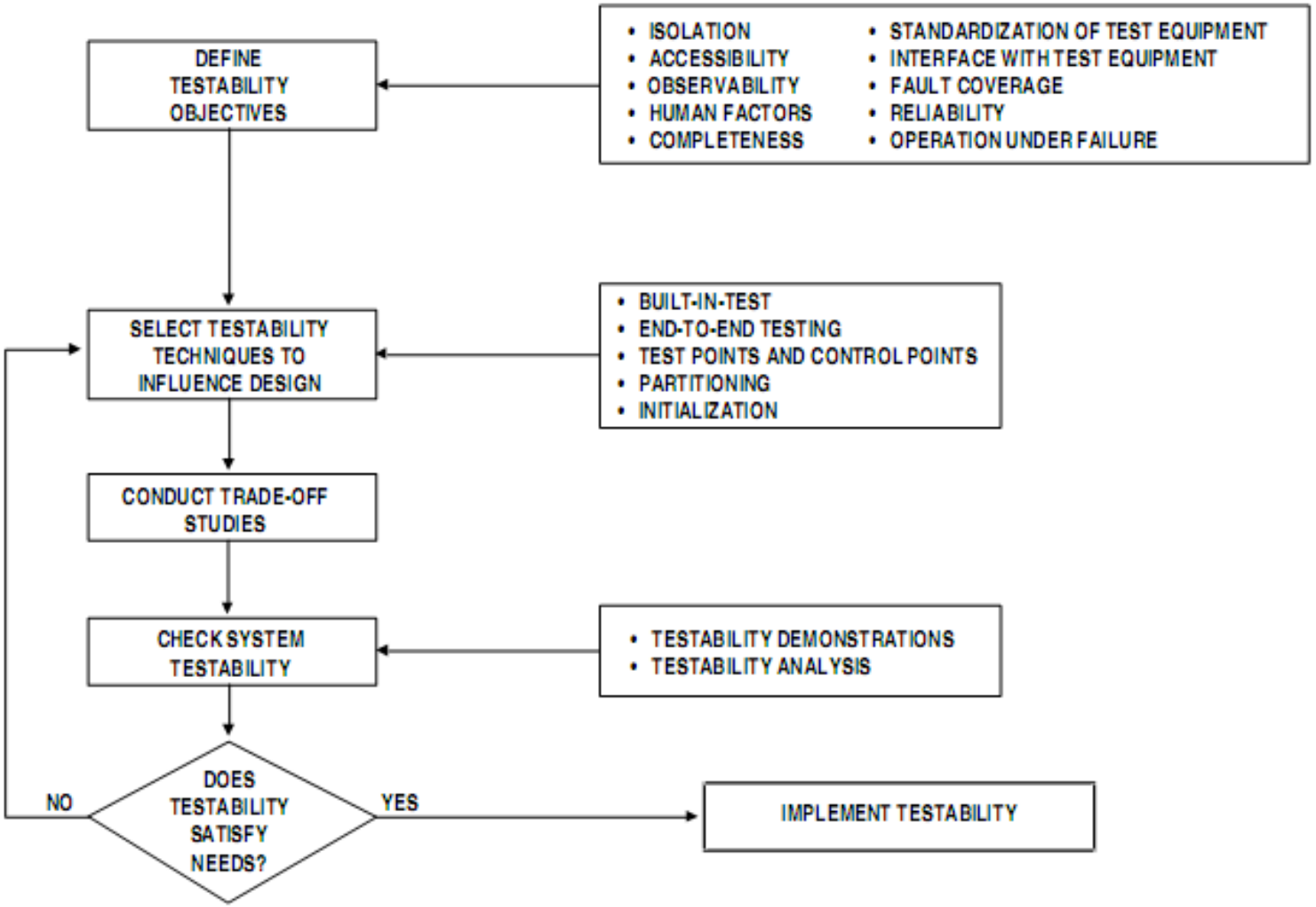
- **Test is the use of an end product to obtain detailed data to validate performance or to provide sufficient information to validate performance through further analysis**

- **Testing is the detailed quantifying method of both Verification and Validation, but it is required in order to validate final End Products to be produced and deployed**

Role of T&E in Systems Engineering

- **Integral part of the SE process**
- **Significant in supporting the decision making process, providing data to support trade-off analysis, risk reduction, and requirements refinement**
- **Will tell how well a system is performing during development and assess attainment of technical performance parameters, determine whether systems are operationally effective, suitable, survivable, and safe for intended use**
- **Will assess technology maturity and interoperability and confirm performance against documented requirements and specifications**

Design for Test



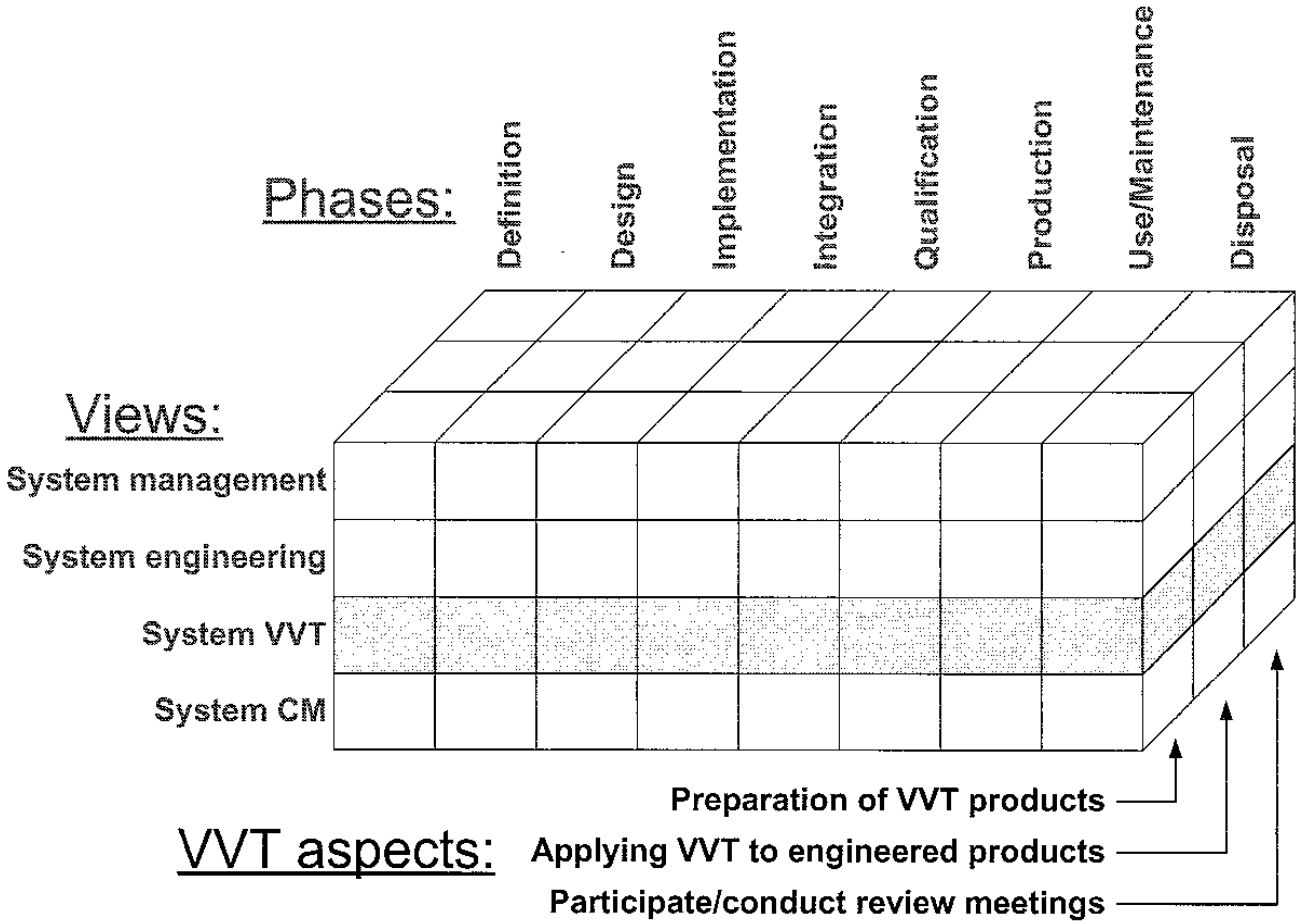
Test Categories

- **Development Test:** Conducted on new items to demonstrate proof of concept or feasibility
- **Qualification Test:** Tests are conducted to prove the design on the first article produced, has a predetermined margin above expected operating conditions, for instance by using elevated environmental conditions for hardware
- **Acceptance Test:** Conducted prior to transition such that the customer can decide that the system is ready to change ownership status from supplier to acquirer
- **Operational Test:** Conducted to verify that the item meets its specification requirements when subjected to the actual operational environment

Documentation

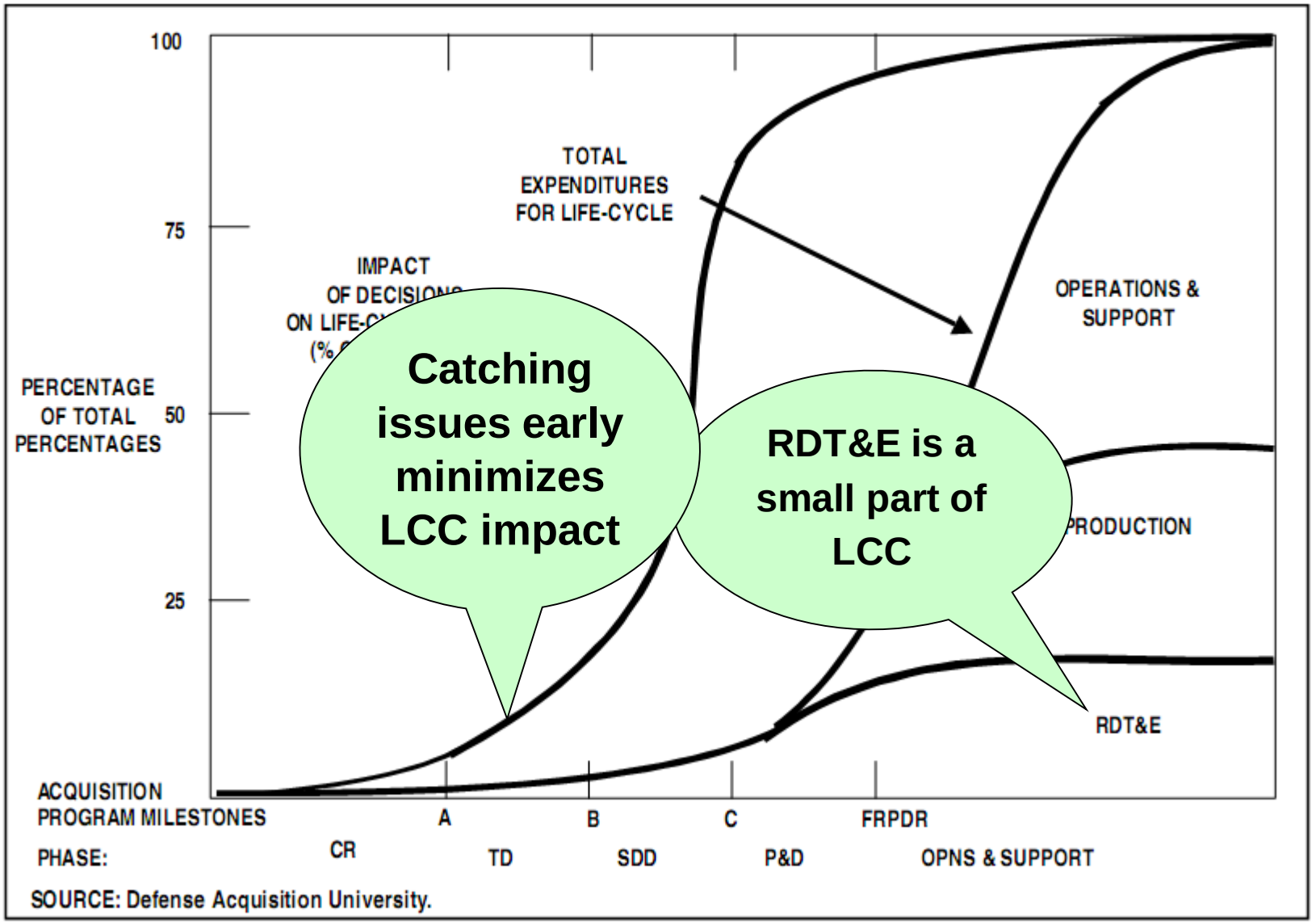
- **Systems Engineering Plan (SEP) - The verification and validation process and its relationship to the other SE process needs to be spelled out in the SEP**
- **Test and Evaluation Management Plan (TEMP) - Describes the overall test approach for integrating developmental, operational test and evaluation and addresses test resource planning**
- **Test Plans**
- **Test Procedures**
- **Test Reports**

Canonical System VVT Paradigm

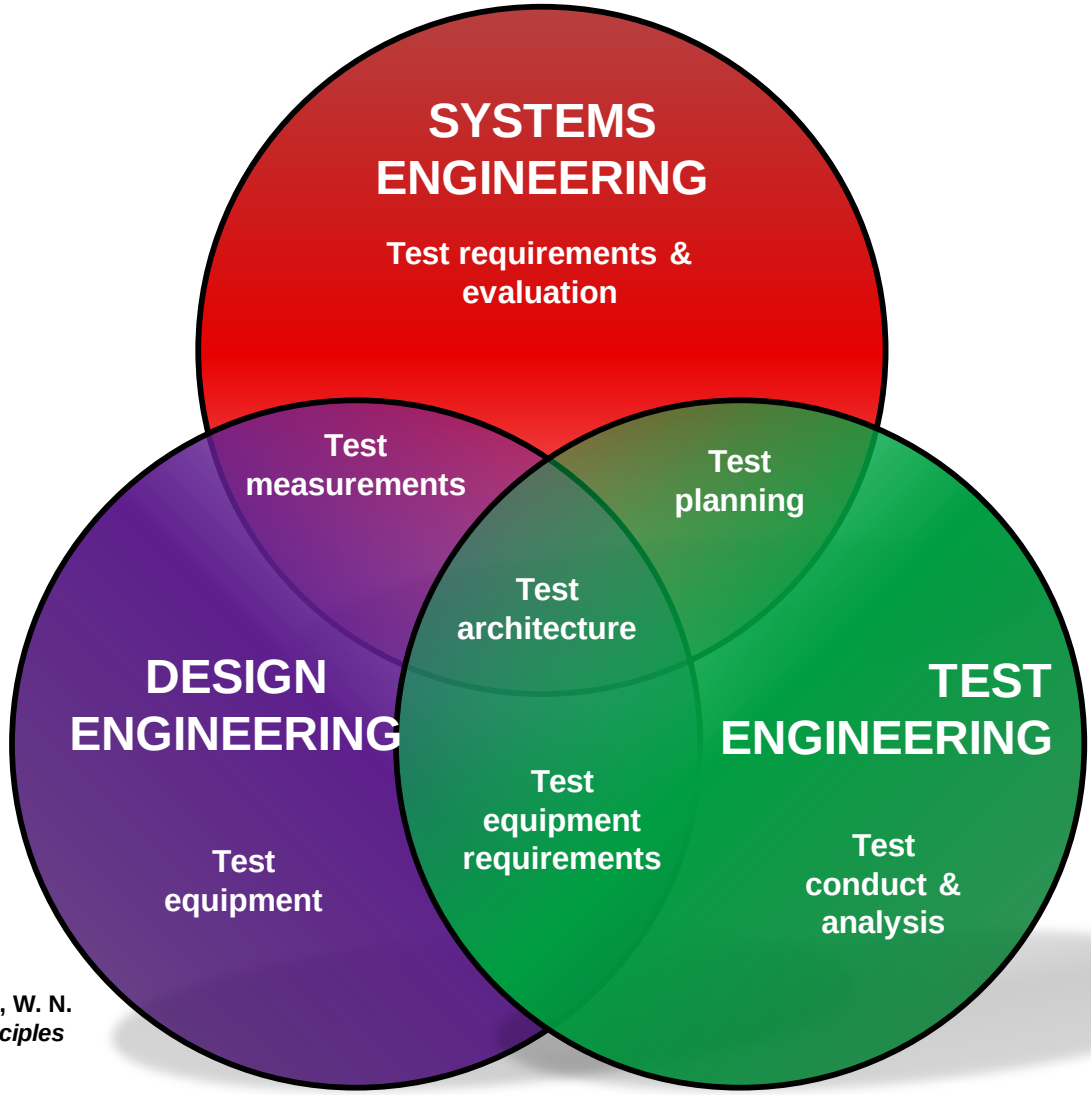


Reference: Engel (2010). *Verification, Validation and Testing of Engineered Systems*, p. 33.

Life Cycle Cost Impacts



Relationship Between Systems, Design, & Test Engineering In Relation To T&E



Reference: Kossiakoff, A., & Sweet, W. N. (2003). *Systems Engineering: Principles & Practice,* p. 276.

Summary

- Integration, Verification and Validation are fundamental tenants of bottoms up product realization
- T&E is part of every phase of a systems life cycle
- Integral to fundamental Systems Engineering and core SE processes

Your Questions & Takeaways

	Takeaways you can use back on the job	Reference
1		
2		
3		
4		
5		

All lessons in this course will have a slide like this for you to record "Takeaways" that you can use on your projects or in the future



What are your Questions before We Continue?



Systems Engineering Technical Short Course

(Session 5: System Integration and Test,

AKA: Verification, Validation and Test - VVT)

Lesson 3:
Integration Planning, Management and Readiness

Session 5 – System Integration and Test

Lessons

1. **Program Overview Update**
2. **Process Overviews**
3. **Integration, Planning, Management and Readiness**
4. **System Test Methods**
5. **Test Measurement Techniques and Metrics (if time permits)**

- Relationships with SE Lifecycle
- V&V Processes
- System Interfaces
- Technical Management and Test Processes

- Integration and Test Planning
- Requirements Management
- Interface Management
- Technical Readiness

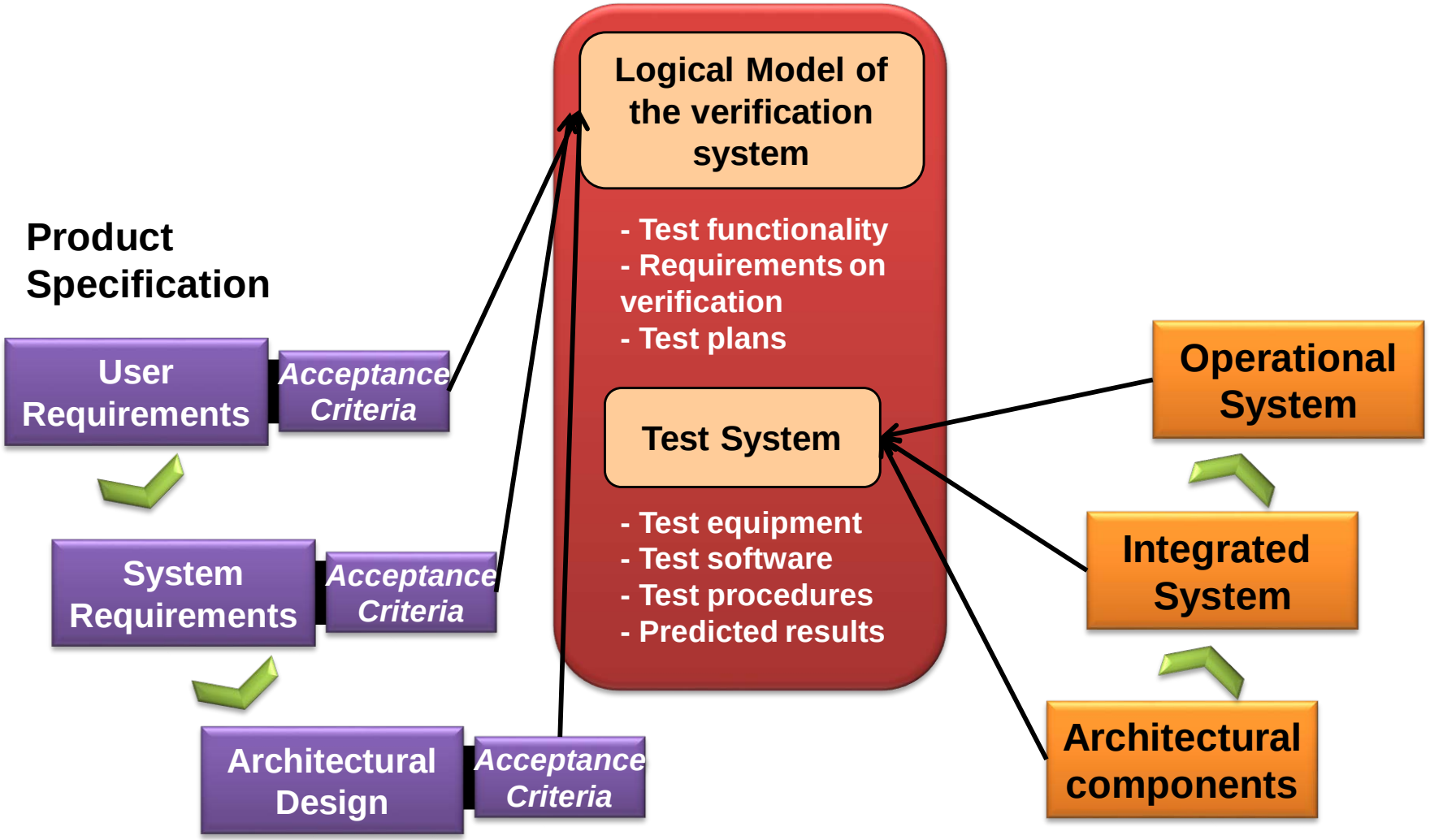
Appendices*

K. **Critical Systems Validation**

L. **Security Testing**

*Appendices A through J were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered.

Building the Verification System



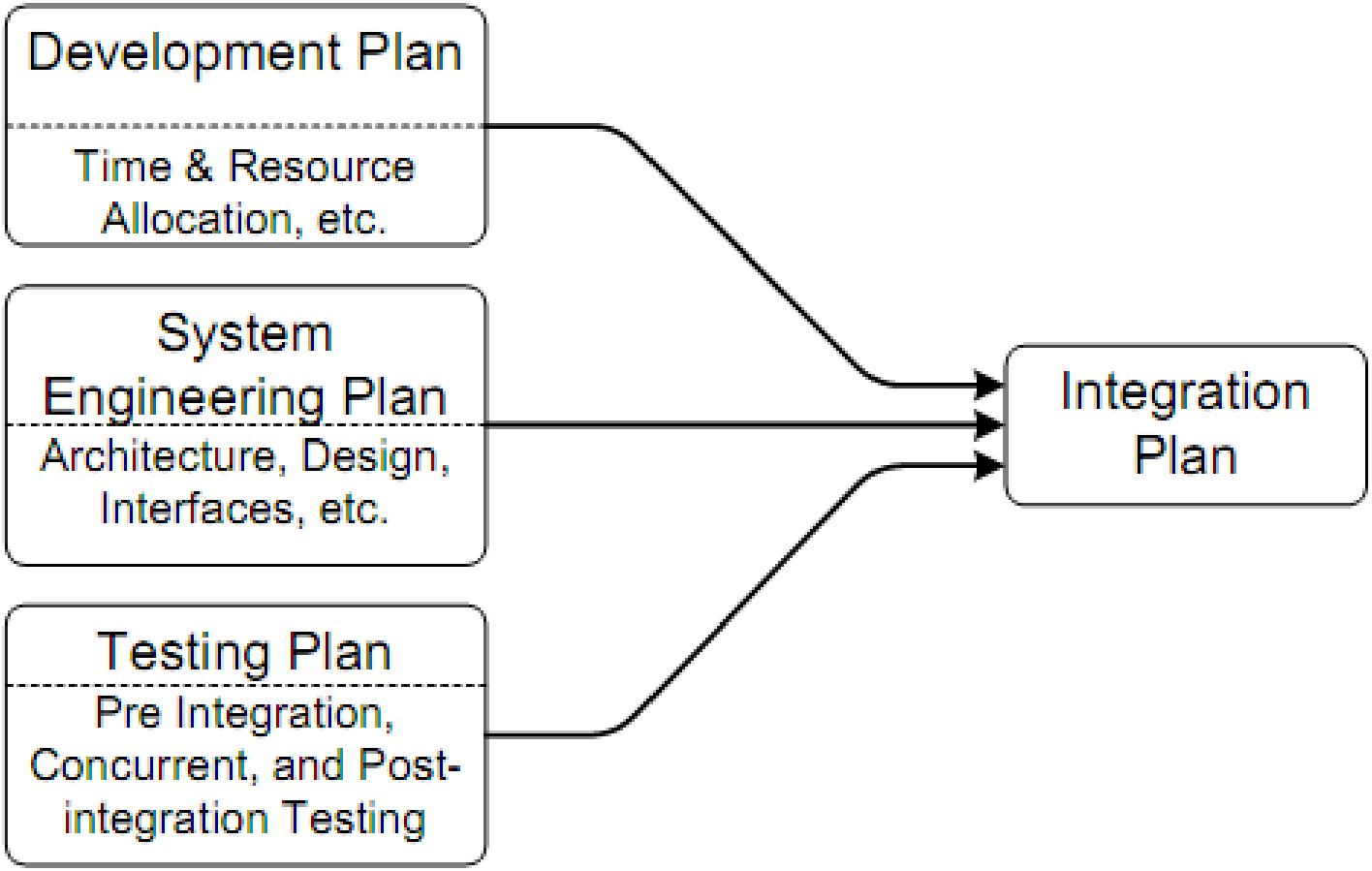
Integration Planning Considerations

- Any constraints imposed by planned integration procedures, physical integration requirements and operational integration requirements
- Time-sequencing of the flow of parts and products to be assembled and integrated into the desired product, including storage of parts and subassemblies
- Availability of integration enabling products such as integration facilities, equipment, tools, fixtures, connectors; integration processes and procedures; equipment and facilities for storage, handling, and integration and for waste product handling and disposal
- Interface Management needed for:
  1. Form, fit and functional internal interfaces of end products to be assembled and integrated together;
  2. External interfaces (physical, communication, electromagnetic, radiation, human, environment, etc.) during development and once deployed
- Production/assembly and integration facilities; layout of assembly lines to include machines and equipment, tools, fixtures, software compilers and other tools; workforce skills needed and workforce safety and health factors

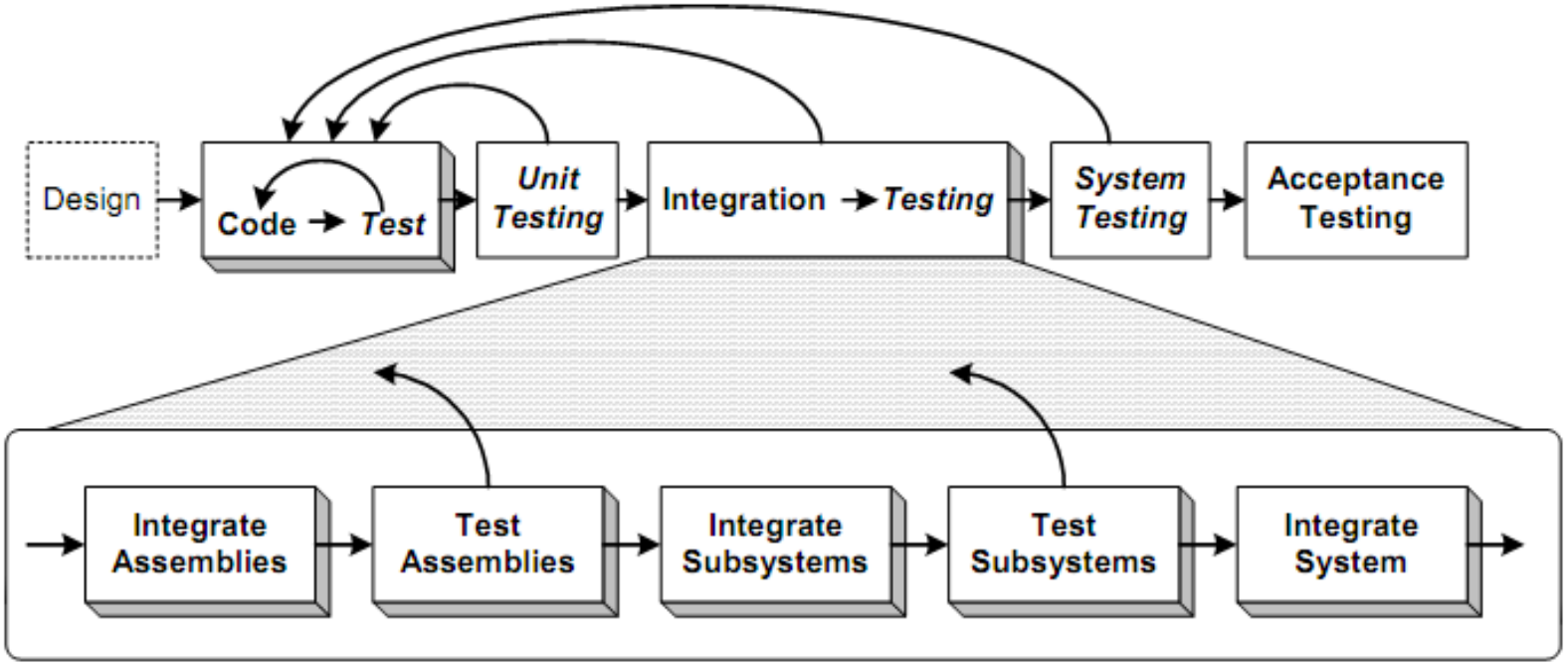
Integration and SE

- **Successful system integration results in proper implementation of activities on the left side of the SE "V"**
- **This IS Systems Engineering**
- **A primary outcome of SE is successful systems integration!**
- **Testing goes hand in hand with integration**
  - **Evidence of successful integration**
- **Planning is a logical consequence of SE and other activities**

Systems Integration Planning

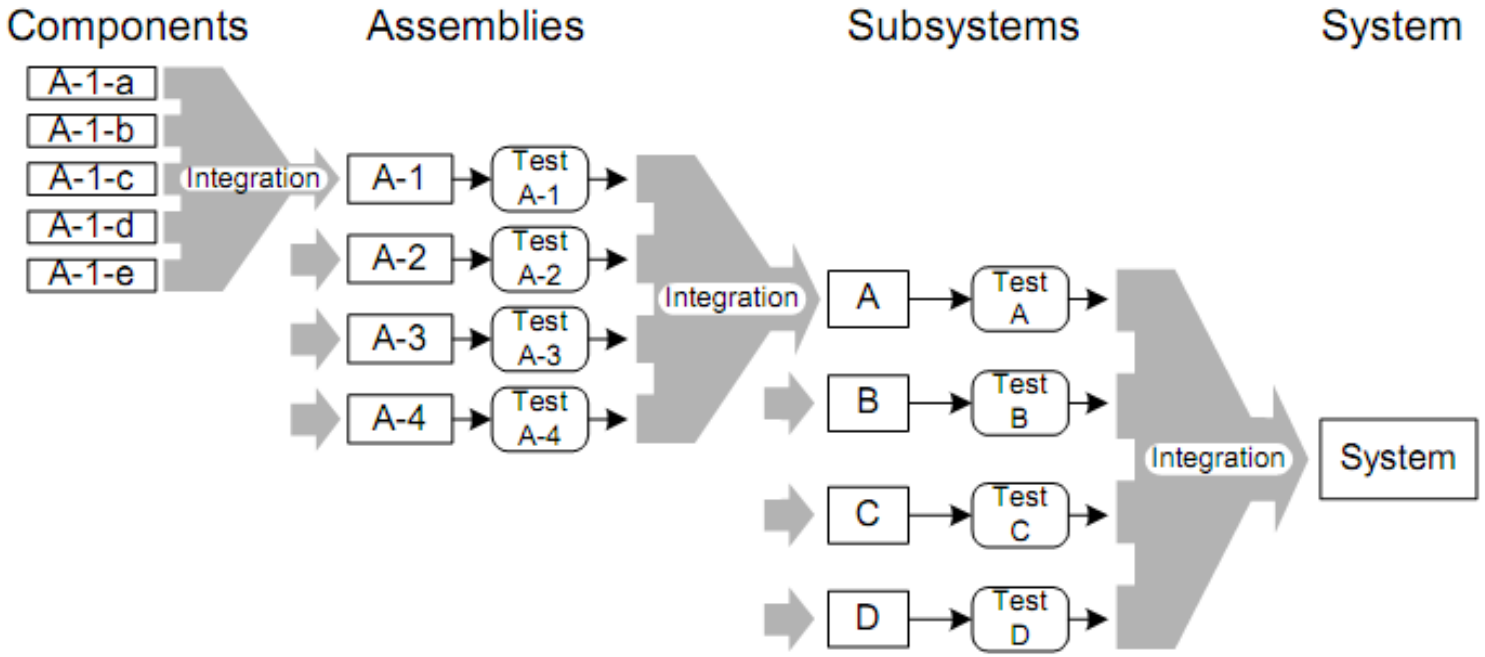


Integration Process



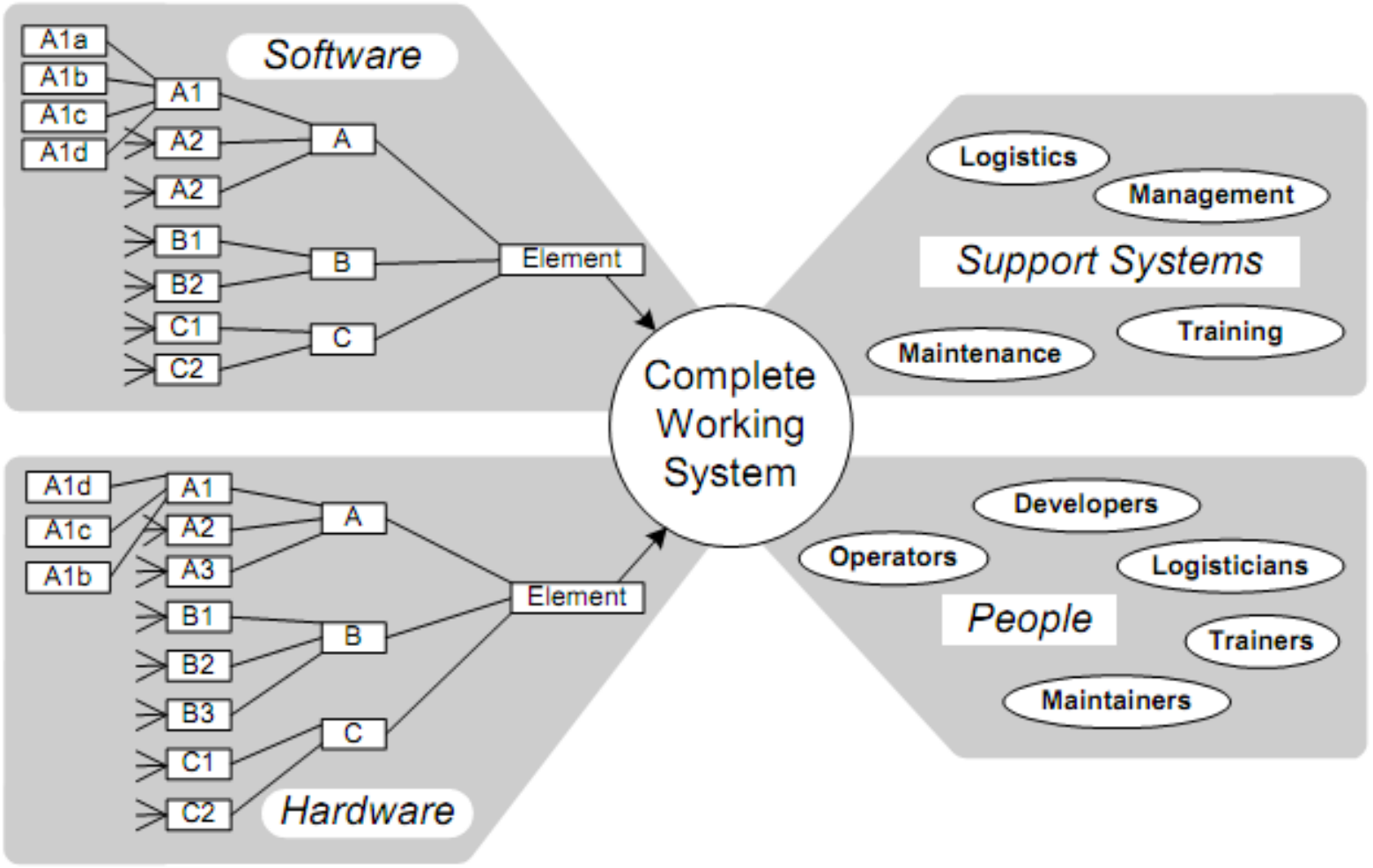
- The integration process is iterative
- Failures do occur which results in analysis and fixes
- Proper test planning is essential to success

Iterative, Progressive Nature of Integration



- **Each level builds on top of the previous level**
- **Complexity of the system drives the level to be accomplished**

System Elements



System Integration Checklist

Before Starting:

- Have you implemented systems engineering as an integrated life cycle effort ?
- Do your test plans include and support integration efforts?
- Does your development plan allocate adequate time and resources for system integration efforts, including rework time?
- Are the interfaces between components, assemblies, subsystems, and systems defined in adequate detail?
- Will hardware be available for testing software during integration?
- Is there a contingency plan if the schedule slips if and the integration schedule is compressed?
- Are all elements of the system included in the integration plan?
- Is all documentation current and available for reference?

During Integration:

- Is there an efficient rework cycle in place to fix problems found during integration testing?
- Are "fixed" modules or components integrated and retested at all levels of integration up to the level where the problem was found?
- Is the people element (operators, maintainers, logisticians, trainers, etc.) being prepared to work with the system when it is deployed?
- Is the support systems element (logistics, maintenance, training, etc.) being prepared to support the new system when it is deployed?
- Are you following an iterative, progressive integration process?
- Are experienced integrators involved with the integration?
- Are area/subject matter experts involved with the integration?
- Is adequate time being allowed for integration, testing, rework, reintegration, and retesting?
- Are all necessary resources being made available for integration?
- Is adequate testing being performed on integrated units (assemblies, subsystems, elements, system) to ensure that there are no surprises during acceptance testing?
- Are you updating documentation during rework?
- Are integration and system test errors being traced back to requirements and design? And if so, are the requirements and design being updated?

Test Planning

- **When do you start?**
- **Plays a key role in:**
  - **System Integrated Master Schedule development**
  - **Budget development**
  - **WBS development**
  - **Systems Engineering & Logistics documentation**
  - **Staffing plans**
  - **Maintenance planning**
- **Integration and Test Team touches all planning efforts … early and often!**

Test Plan

- The Test Plan describes the overall test objectives and test events for the Test Program
- Provides the basis from which detailed Event Plans and Test Procedures will be written
- Serves to identify the resources required to execute the Test Plan and the schedule for conducting the Test Program
- In short, the Test Plan provides the structure for the test and evaluation activities

Sample Test Plan Contents

- Preliminary pages
  - Title page
  - Approval page
  - Conditions for release of technical data
  - Preface (abstract)
  - Executive summary
- Table of contents
- Appendices
  - Test condition matrix
  - Requirements traceability
  - Test information sheets
  - Parameters list
  - Data analysis plan
  - Instrumentation plan
  - Logistics support plan
- List of abbreviations
- Main body
  - Introduction
  - Background
  - Test item description
  - Overall test objectives
  - Constraints and limitations
  - Test resources
  - Safety requirements
  - Security requirements
  - Test project management
  - Test procedures
  - Test reporting
  - Logistics
  - Environmental protection

The Test Concept

- The *Test Objective* describes the purpose for the test, exactly what the test is attempting to PROVE

- The *Test Concept* describes the idea or approach for conducting test and evaluation activities
  - What specific results are required by the customer?
  - What types of Test Phases are planned (Bench, DT, OT, none)?
  - What should be the test articles? How many? Configurations?
  - At what level of indenture should one begin?
  - What should be assumed as a given (or previously demonstrated)?
  - What environment (mix) will tests be conducted under?
  - What should be measured? How will it be measured?
  - What stimuli should be used?

Test Events

- **Description of each test event's**
  - **Objectives**
  - **Test configuration**
  - **Functions performed by the test elements**
  - **Stimuli for/expected responses of test cases**
  - **Data to be extracted and general pass/fail criteria**

- **This is the heart of your plan; it makes clear how you intend to utilize your test site, test tools and test equipment**

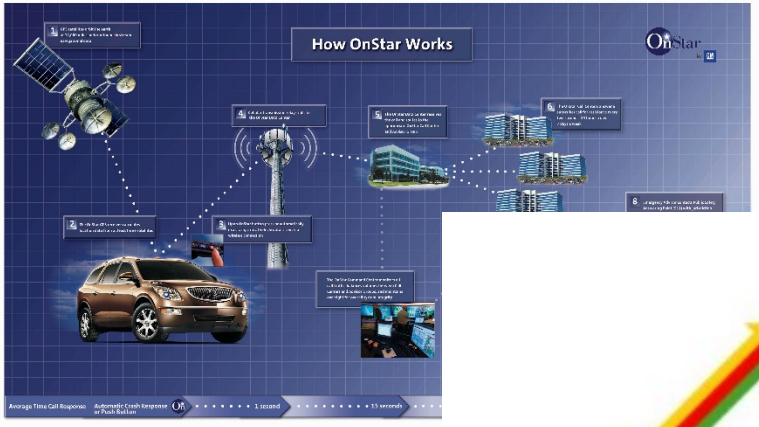
Role of A Concept of Operations (CONOPS)

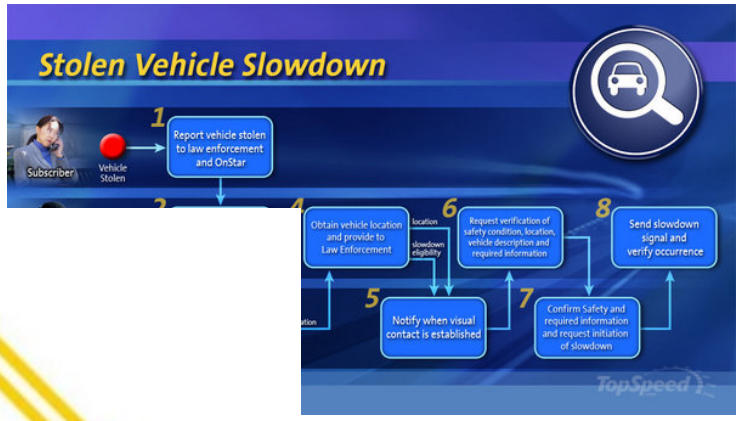
- Describes how the user will employ the system
  - Discusses who will operated the system
  - How it will be employed
  - Where it will be employed
  - Can be a graphic or text document

- Why is this important to test planning?
  - Customer expectations – want a reliable, robust system
  - Defines the operating environment
  - Describes system and environment interactions
  - Identifies desired and undesired stimuli

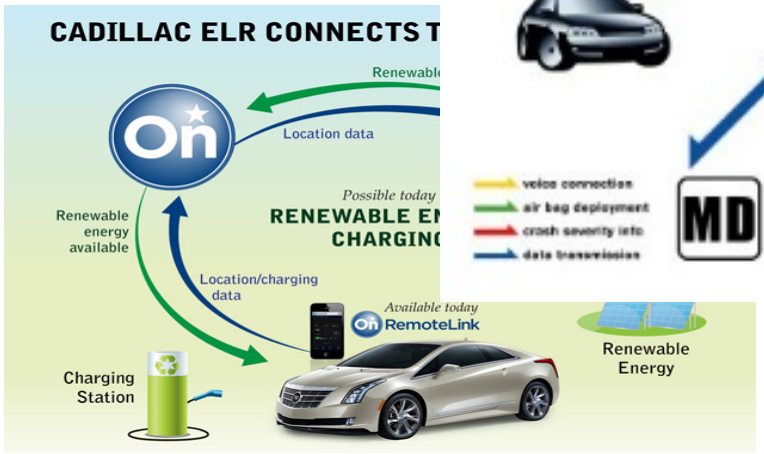
- Bottom Line – Provides the "Big Picture" how the system will be used

Examples of Test Event Scenarios (CONOPS)











Test Schedule

- The date when each test event will commence and the calendar time required to complete the test event
- What factors would you consider in deciding the length of time required to conduct a test event?
  - Level of maturity of the test procedures
  - Reasonable time to conduct mature procedures on the system
  - Transportation of test assets and data, if needed
  - Time to analyze test results
  - Time to correct system implementation errors
  - Time to retest to demonstrate correct system operation
- Test events often driven by test asset availability
- Paced by the development schedule so requires flexibility

Test Resources

- **Description of all the equipment and resources required to**
  - **Stimulate the system under test**
  - **Record the system's responses**
  - **Analyze the results**

- **Should include**
  - **Models and simulations needed along with their intended use**
  - **Test articles**
  - **Test beds**
  - **Support equipment**
  - **Instrumentation/sites**

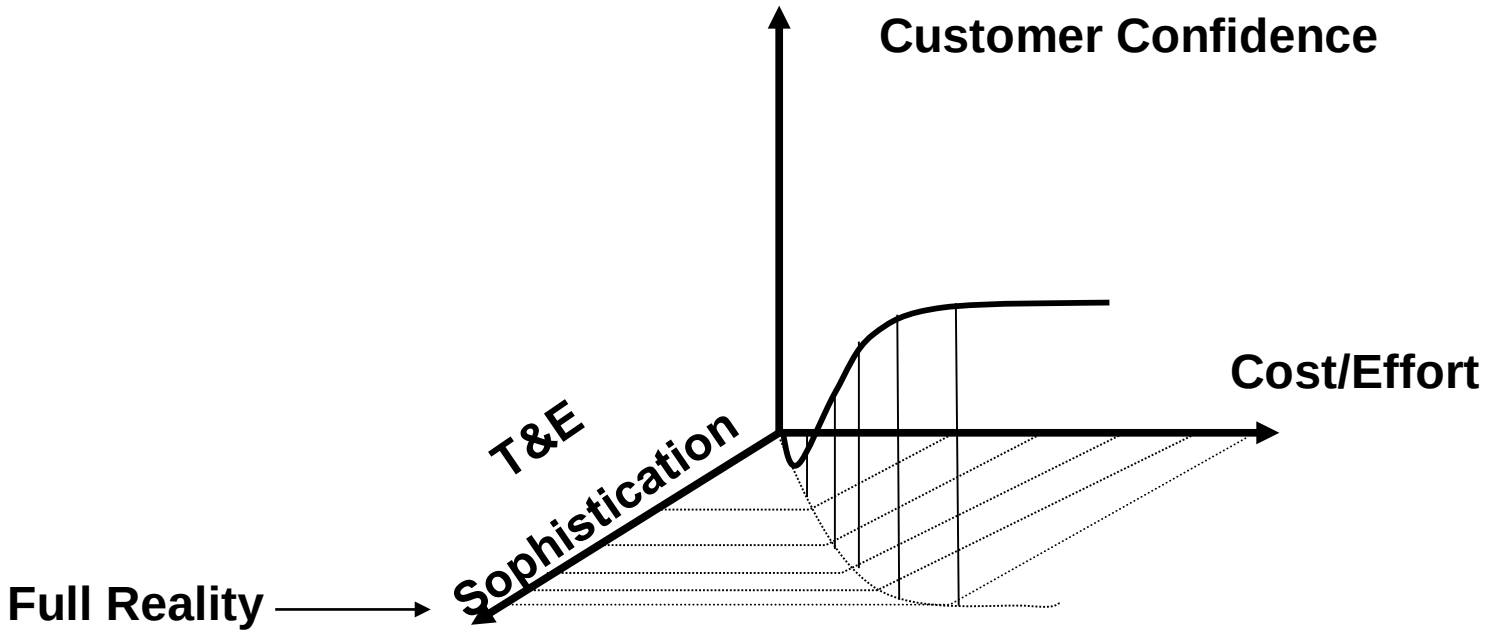
Test Personnel

- All the people required to complete the test program
- Obvious personnel
  - Lead Test Conductor
  - Equipment Operators
  - System/Equipment Engineers
  - Data Analysts
- Not-so-obvious test personnel
  - Configuration Manager
  - Data Technician
  - Design Engineers
  - Customer witness (formal, informal)
  - Quality Assurance
  - Security
  - Safety

Building A Test Team

- **Choosing the right personnel is essential for a test program's success**
- **Personnel must**
  - **Have the right skill sets**
  - **Be trained on the system**
  - **Have the necessary access**
- **How will the users/sponsors be integrated?**
- **Do you have certification requirements?**
  - **FAA, FCC, FDA**
- **May need to form IPT's to address specific issues or testing**

Customer Expectations



- Complete confidence requires the entire system to demonstrate repeatable, reliable performance in all operational environments
- T&E engineer must determine how much confidence is affordable, and trade off cost and effort for
  - Simulated vs. real system elements and/or operational environments
  - Significant vs. all possible test conditions

Where to Begin

- **Develop a plan to PROVE the system functions as intended and performs in its operational environment**
  - **Interfaces are properly implemented by the associated subsystems**
  - **Data and control signals flow through and between subsystems to produce the desired system functionality**
  - **The system's operations result in the customer's desired capability**
  - **Consult with past T&E programs and design engineers**

- **Start at lowest indenture possible**
  - **Units should be shown to comply with requirements before being assembled into elements**
  - **Elements should be shown to comply with requirements before being assembled into subsystems**
  - **Subsystems should be shown to be fully functional and complaint with specifications**

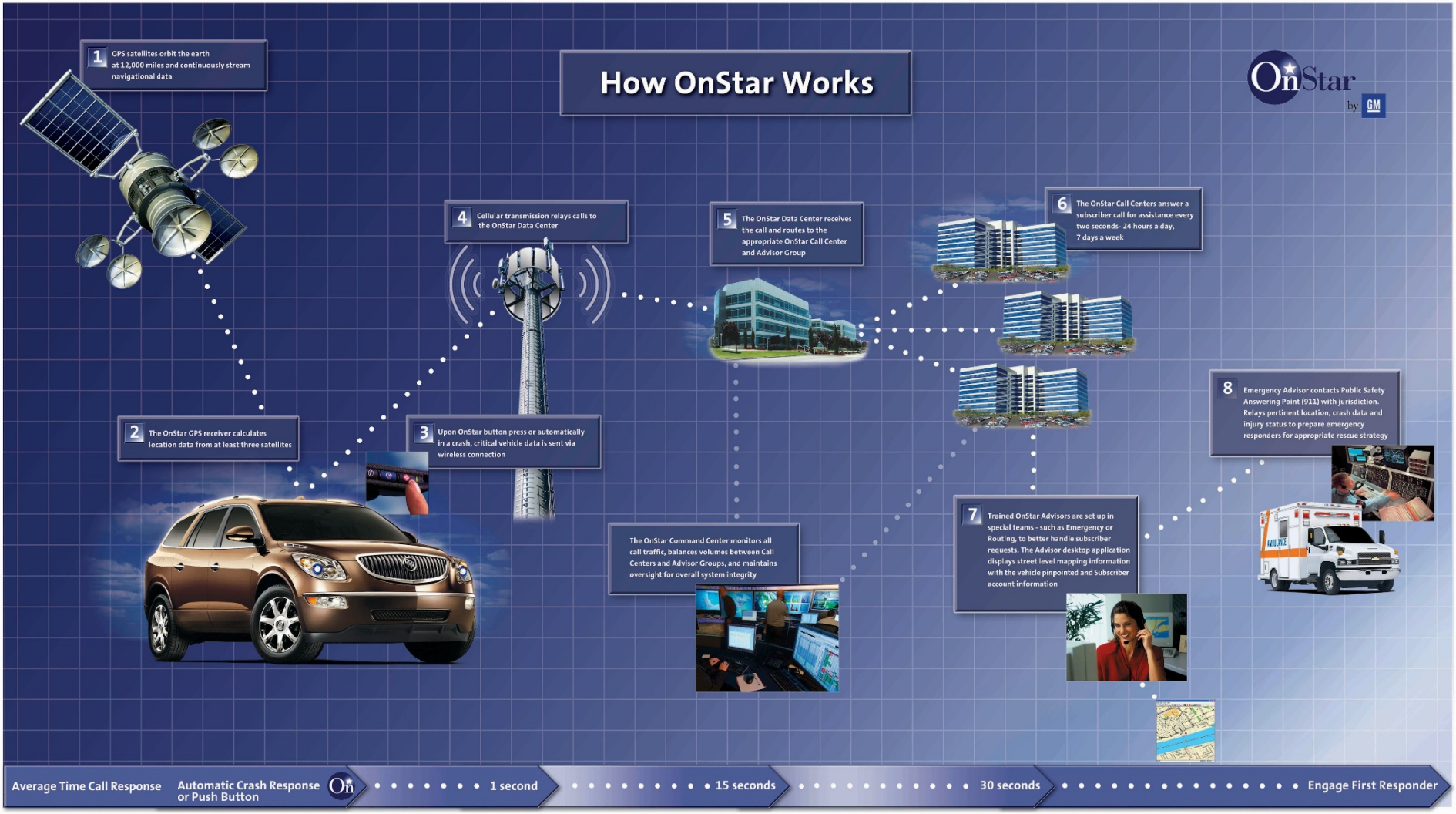
Where to Begin

- **Review the designer's intended test regime to determine which units, elements, and subsystems will be tested**
  - **If you are satisfied**
    - **assume that the remaining issues involve external interactions and test accordingly**
  - **If you are not satisfied**
    - **have your concerns addressed (best approach), or**
    - **build the necessary testing into your plan (costliest approach),**
    - **or ignore your concern (riskiest approach)**
- **Never assume budget is not an issue, forecast and mitigate high-cost tests accordingly**
  - **Instrumentation, remote sites, etc.**

Summary

- Test planning must happen early and often in the acquisition process even though it may be based on rough estimates
- Must have traceable, observable, and quantifiable progress milestones
- Minimize simultaneous introduction of new items at same task stage
- Test the system as the customer will use it
- Plan for Configuration Control and use of certified tools
- Pull in the customer early and often
- The Test Objective describes the purpose for the test, exactly what the test is attempting to PROVE
- The Test Concept describes the idea or approach for conducting test and evaluation activities

OnStar System Exercise:
Test Planning



OnStar System Exercise:

Test Planning

1. GPS satellites continuously orbit the earth at 12,000 miles and continuously stream navigational data.

2. The OnStar GPS receiver calculates location data from at least three satellites.

3. Upon OnStar button press or automatically in a crash, critical vehicle data is sent via wireless connection.

4. Cellular transmission relays calls to the OnStar Data Center.

5. The OnStar Data Center receives the call and routes the call to the appropriate OnStar Call Center and Advisor Group.

   The OnStar Command Center monitors all call traffic, balances volumes between Call Centers and Advisor Groups, and maintains oversight for overall system integrity.

6. The OnStar Call Centers answer a subscriber call for assistance every two seconds – 24 hours a day, 7 days a week.

7. Trainer OnStar Advisors are set up in special teams – such as Emergency or Routing, to better handle subscriber requests. The Advisor desktop application provides street level mapping information with the vehicle pinpointed and subscriber account information.

8. Emergency Advisor contacts Public Safety Answering Point (911) within jurisdiction. Relays pertinent local, crash data and injury status to prepare emergency responders for appropriate rescue strategy.

OnStar System Exercise:
Test Planning

- **Develop a Test Objective for testing the emergency response scenario of the OnStar system**
- **Develop a Test Concept for testing the emergency response scenario of the OnStar system**
- **Develop descriptions of 1 – 2 Test Events for conducting tests and evaluations of the OnStar system for the emergency response scenario.**
- **Be prepared to share your results with the class, including:**
  - **Descriptions your Test Objective, Concept and Events**
  - **Your observations for the exercise**



Session 5 – System Integration and Test

Lessons

1. **Program Overview Update**
2. **Process Overviews**
3. **Integration, Planning, Management and Readiness**
4. **System Test Methods**
5. **Test Measurement Techniques and Metrics (if time permits)**

- Relationships with SE Lifecycle
- V&V Processes
- System Interfaces
- Technical Management and Test Processes

- Integration and Test Planning
- Requirements Management
- Interface Management
- Technical Readiness

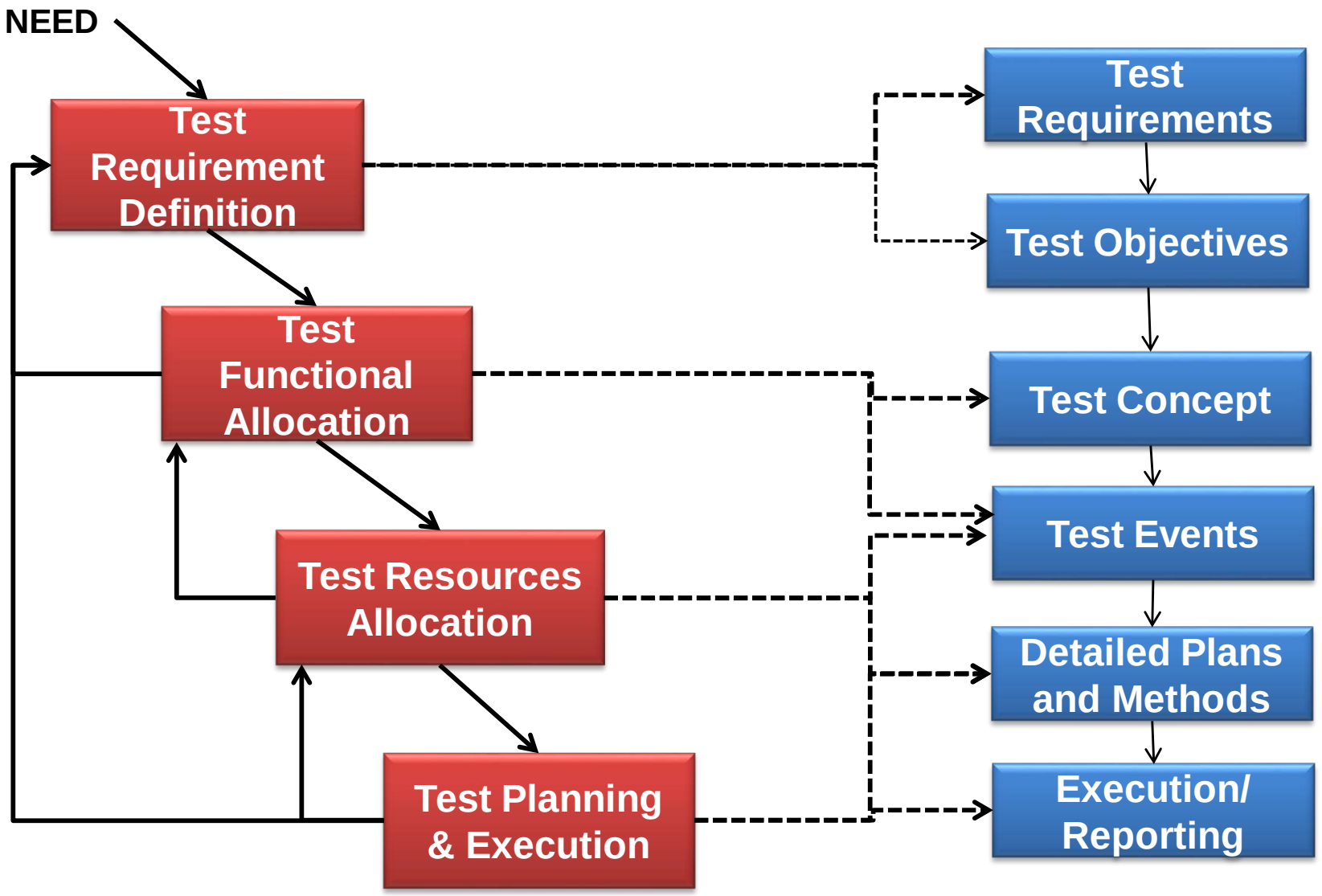
Appendices*

K. **Critical Systems Validation**

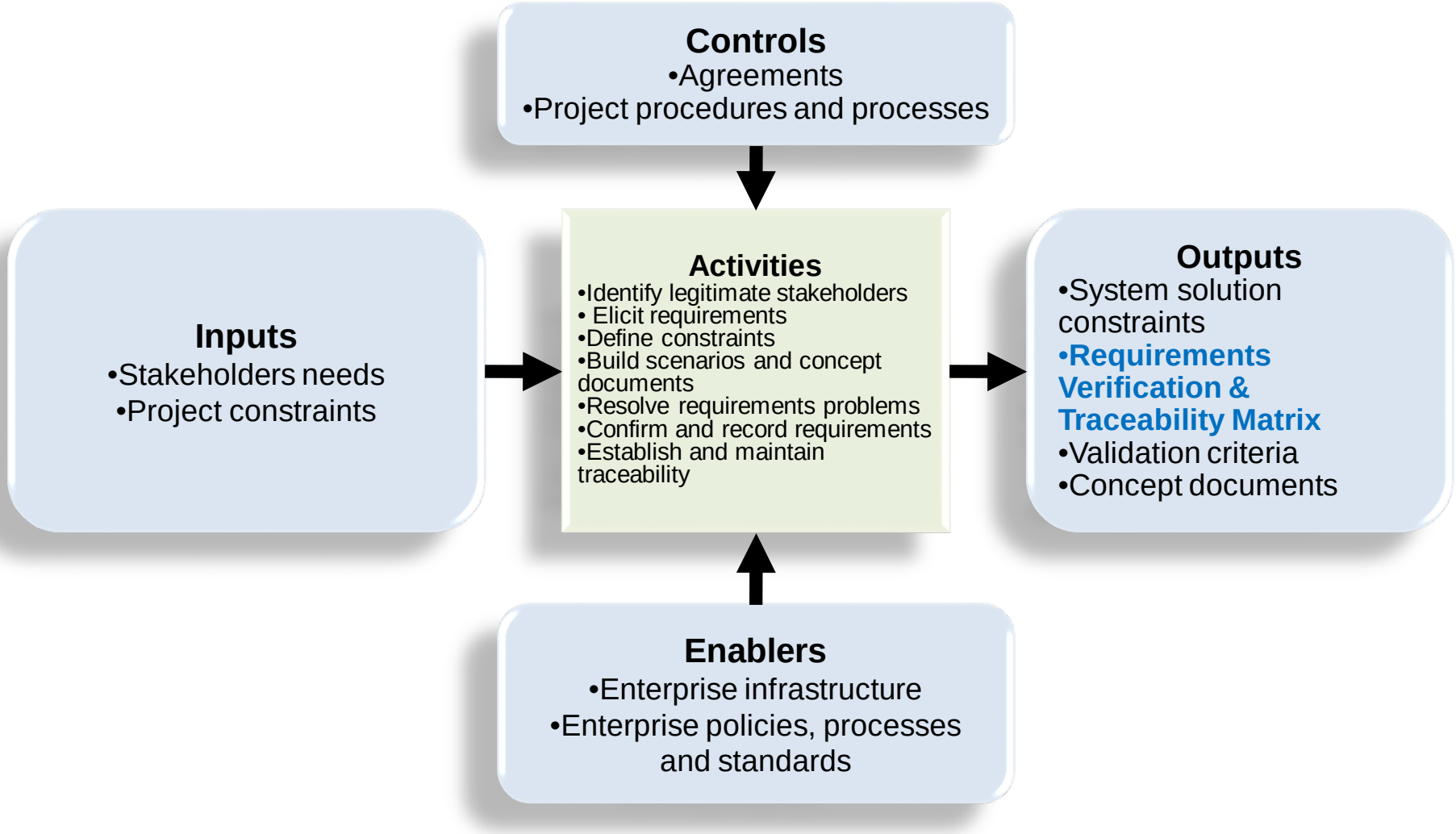
L. **Security Testing**

*Appendices A through J were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered.

Test Process Flow



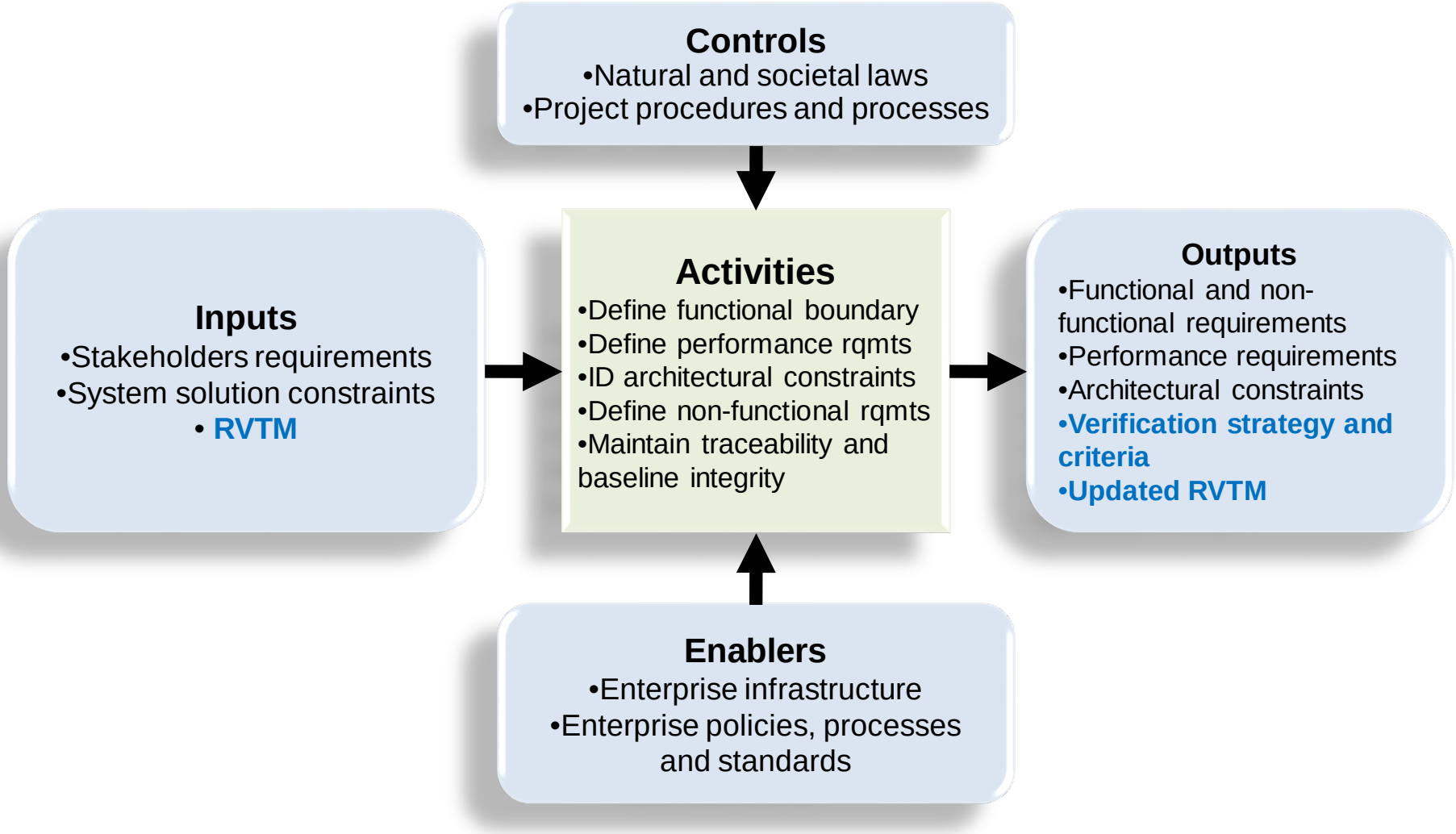
Stakeholder Requirements Definition Process



Stakeholder Requirements Definition Process

- **Inputs**
  - **Include the description of users' needs or services that the system will provide, cost, schedule, and solution constraints, terms and conditions of the agreement, and industry specifications and standards**
- **Outputs**
  - **Formally documented and approved stakeholder requirements that will govern the project, including: required system capabilities, functions and/or services; quality standards; cost and schedule constraints; concept of operations; and concept of support**
  - **Measures of effectiveness and suitability that will be used for assessing the realized system and enabling systems**
  - **Initial Requirements Verification and Traceability Matrix (RVTM)**
  - **Validation criteria may specify who will perform validation activities, and the environments of the system-of-interest and enabling systems**

Requirements Analysis Process



Requirements Analysis Process

- Inputs
  - Stakeholder requirements
  - Initial RVTM
  - Constraints such as
    - Applicable statutes, regulations, and policies
    - Concept of operations
    - Design or enterprise constraints
    - Manufacturing
    - Life cycle support considerations
    - Decisions or data resulting from previous phases of development
- Outputs
  - Technical description of characteristics the future system must have in order to meet Stakeholder Requirements
  - Additional derived requirements
  - Interfaces and information exchange requirements with systems external to the functional boundaries.
  - Updated RVTM

Test and Evaluation (T&E) Methods

Test	• Checks functions, interfaces, compatibility and performance requirements • Can be performed in conditions similar to the operational environment
Analysis	• Mathematical models, computer analysis software, simulation
Inspection	• Visual inspection of requirement (equipment ID marking, safety, material types, etc)
Demonstration	• Demonstrates functionality or capability, usually in an operationally relevant environment.

Verification by SE Assessment

- **Conducted to support the development of products, services, and processes necessary to verify that system end-items satisfy their requirements.**

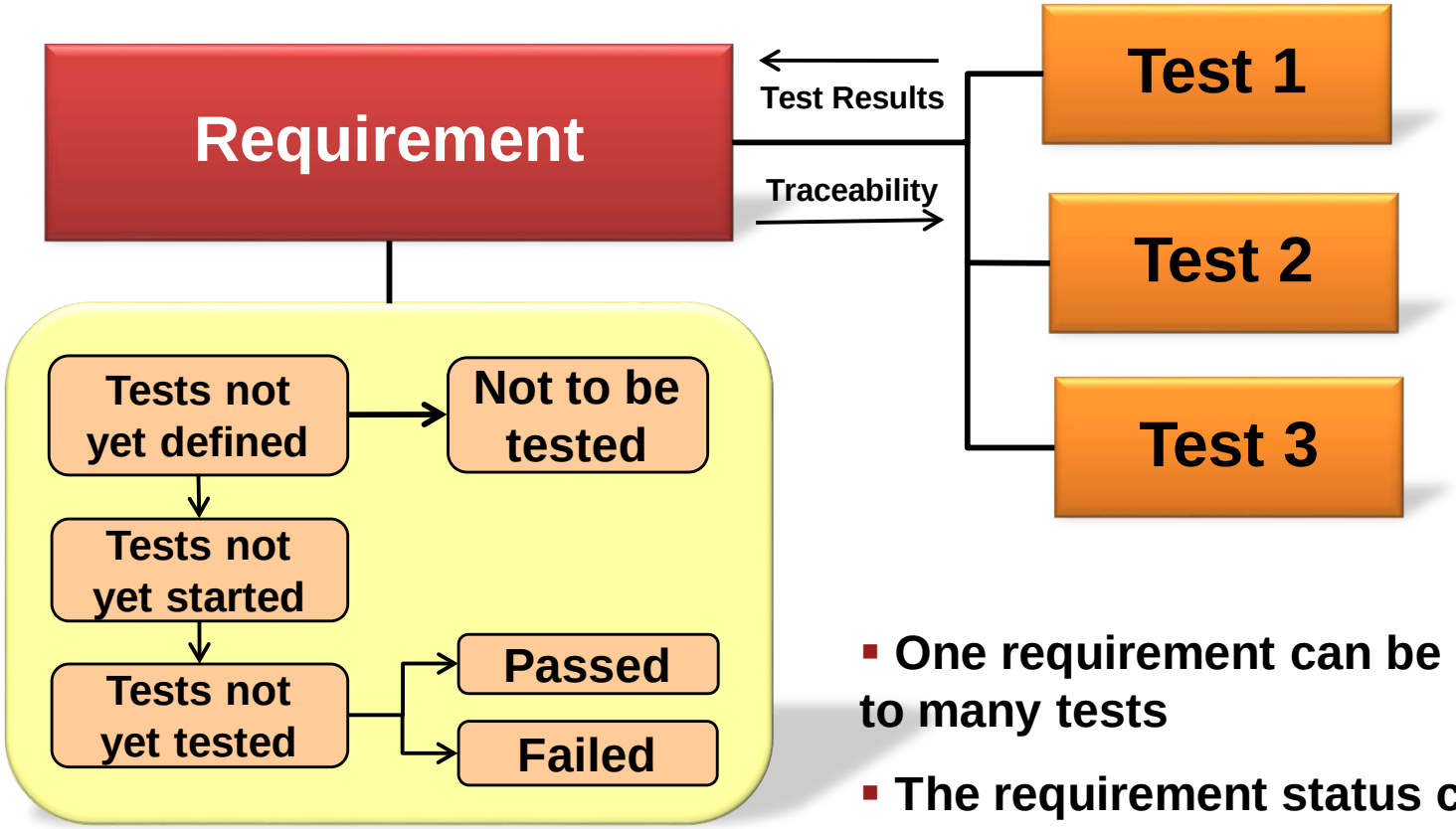
- **Verification assessment addresses Verification requirements and criteria for solution alternatives; definition of Verifications to demonstrate proof of concept; and development, qualification, acceptance, pertinent operational, and other testing.**

- **May also consider the requirements and procedures needed to verify critical Verification methods and processes (e.g., Verification of key methods and assumptions and the data used in Verification by analysis).**

Assessment Methods

- ***Verification by Engineering Analysis*** **includes the techniques of SE analysis, Specialty Engineering, statistic and qualitative analysis, simulations, and modeling.**
- ***Verification by Similarity*** **assesses compliance with requirements by reviewing a similar system's test data, configuration, and applications.**
- ***Validation of Records*** **reviews manufacturing records at end-item acceptance to verify features and requirements of the system.**
- ***Simulation*** **verifies design features, system behavior, and performance using simulated models of the system.**
- ***Review of Design Documentation*** **uses the disciplined review of design documentation, such as reports and drawings from Acquisition Reviews, Design Reviews (preliminary and critical), and other evaluations.**
- ***Physical Examination*** **assesses compliance with requirements by visually inspecting a physical item or configuration according to pre-established criteria.**

Requirement Status and Tests



Requirements status against test

- One requirement can be related to many tests
- The requirement status changes as the tests are defined and the test results are analyzed

Requirements Drive Test Objectives

- **OPERATIONAL Requirements**
  - **Defines the customer's expectations of the systems in terms of the mission need and constraints, includes Measures of Effectiveness**
- **SYSTEM Requirements**
  - **Defines the requirements required to design and develop the system, traceable to Operational Requirements.**
- **FUNCTIONAL Requirements**
  - **Defines the activities or functions to be performed by the system.**
- **PERFORMANCE Requirements**
  - **Defines the expectation for performance of the mission, function, or system, includes Measures of Performance (quantity, quality, coverage, timeliness or readiness)**
- **TEST Requirements**
  - **System is tested against System Requirements, which includes functional and performance requirements.**

Requirements Allocation & Traceability

- **Traceability is not an end goal but a tool**
  - **Can improve integrity and accuracy of all requirements, from the system level down to the component**
  - **Allows tracking of the requirements development, allocation and generating overall metrics**
  - **Support easier maintenance and change implementation of the system**
- **Every requirement at every level should have clear definition of its source and why it is needed**
- **As testers, why do we care?**

Requirements Allocation & Traceability

- Should be maintained throughout all levels of the system documentation
- Bi-directional traceability is top-down and to verification and validations plans and procedures
- Should be traceable to the test program to provide closed loop verification
- There are commercial tools that can perform this function
  - Whatever tool is used it should provide
    - Requirements statement with unique identifiers
    - Requirements traceability matrices – list requirements and their traces
    - Verification cross reference matrices – list requirements and their verification attributes
    - Specifications
    - Requirement metrics (helps determine requirements stability)
- What does this do for your test program?

Requirements Traceability and V&V

PUID	Requirement	Source	Rationale	"Other"
1				
2				
3				

Basic traceability:

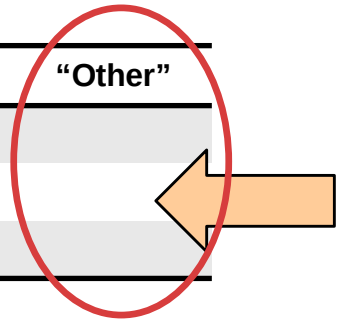
- **Is often represented in matrix* form**
- **Assigns a Project Unique identifier (PUID) to every expectation, need, or requirement when first identified**
- **Identifies the source (stakeholder; artifact; parent expectation, need or requirement; or other source)**
- **Points to the associated rationale . . . .**
- **. . . . and, can provide other important information, particularly with respect to V&V**

What are the "other" columns in the traceability matrix that might be useful to us?

*** Traceability best practices call for the use of a relational database, but the representation is often via a "spreadsheet-like matrix**

What "Other Columns" are Useful?

PUID	Req	Source	Rationale	"Other"
1				
2				
3				



Other Information

Validation Information

Verification Information

- **Plan**
  - **Method (T,D,I, A)**
  - **Brief description**
  - **Criteria**
  - **In-process or at-completion**
- **Responsibility**
- **Test plan paragraph ID**
- **Test procedure step**
- **Result:**
  - **Pass/fail**
  - **Quantified data**
  - **Anomaly number(s)**
- **Other……**

Class Discussion:

Beware the "T", "D", "I", and "A"

- **Typically we dutifully put one or more of these letters in our verification column, claim victory, and go on to our next task**
- **But what value do these 4 letters provide?**
- **What should we provide, and why?**

RTVM Example

REQUIREMENTS TRACEABILITY MATRIX

Project Name:	<optional>
National Center:	<required>
Project Manager Na	<required>
Project Description	<required>

ID	Assoc ID	Technical Assumption(s) and/or Customer Need(s)	Functiona Requireme
001	1.1.1		
002	2.2.2		
003	3.3.3		
004	4.4.4		
005	5.5.5		
006			
007			
008			
009			
010			
011			
012			
013			
014			
015			
016			
017			
018			
019			
020			
021			
022			
023			
024			
025			
026			
027			
028			
029			
030			
031			
032			
033			
034			

Instructions For Completing This Document

1.)	Complete the Project Name, C/I/O, Project Manager Name, and Project Description fields
2.)	For each issue identified, complete the following:
	ID: A unique ID number used to identify the traceability item in the requirements traceability matrix.
	Associated ID(s): This column should contain the ID of any associated utilities used for requirements tracking such as a repository, pipeline document, etc.
	Technical Assumption(s) or Customer Need(s): This column should be populated with a description of the technical assumption or customer need linked to the functional requirement.
	Functional Requirement: This column should be populated with a description of the functional requirement.
	Status: This column should be populated with the current status of the functional requirement.
	Architectural/Design Document: This column should be populated with a description of the architectural/design document linked to the functional requirement.
	Technical Specification: This column should be populated with a description of the technical specification linked to the functional requirement.
	System Component(s): This column should be populated with a description of the system component(s) linked to the functional requirement.
	Software Module(s): This column should be populated with a description of the software module(s) linked to the functional requirement.
	Test Case Number: This column should be populated with the test case number linked to the functional requirement.
	Tested In: This column should be populated with the module that the functional requirement has been tested in.
	Implemented In: This column should be populated with the module that the functional requirement has been implemented in.
	Verification: This column should be populated with a description of the verification document linked to the functional requirement.
	Additional Comments: This column should be populated with any additional comments

EMENTS TRACEABILITY MATRIX

	<optional>
	<required>
Name:	<required>
on:	<required>

Test Case Number	Tested In	Implemented In	Verification

Session 5 – System Integration and Test

Lessons

1. **Program Overview Update**
2. **Process Overviews**
3. **Integration, Planning, Management and Readiness**
4. **System Test Methods**
5. **Test Measurement Techniques and Metrics (if time permits)**

- Relationships with SE Lifecycle
- V&V Processes
- System Interfaces
- Technical Management and Test Processes

- Integration and Test Planning
- Requirements Management
- Interface Management
- Technical Readiness

Appendices*

K. **Critical Systems Validation**

L. **Security Testing**

*Appendices A through J were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered.

Interface Management

- **Most system failures occur at the interface level**
  - **If we know this why do we continue to have problems in this area?**
- **The Interface Management Process encompasses identifying, documenting and controlling all product attributes relevant to the interfacing of two or more products provided by one or more organizations**
- **Control measures ensure that all internal and external interface requirement changes are:**
  - **Properly assessed, controlled and documented, in accordance with the program's Configuration Management Plan**
  - **Communicated to others, in accordance with the program's Interface Management Plan**

IRD vs. ICD

- **Interface Requirements Document (IRD) is used to control interface requirements**
- **Interface Control Document (ICD) controls interface design**
- **These documents:**
  - **Define and illustrate performance, physical, and functional characteristics in sufficient detail to ensure that all details on the interface can be determined solely from the information in the IRD/ICD**
  - **Identify required interface data and monitor submission of this data**
  - **Control the interface requirements and design to prevent any changes to characteristics that might affect compatibility with other systems and equipment**
  - **Communicate coordinated interface requirements and design decisions as well as interface requirements/design changes to program participants**

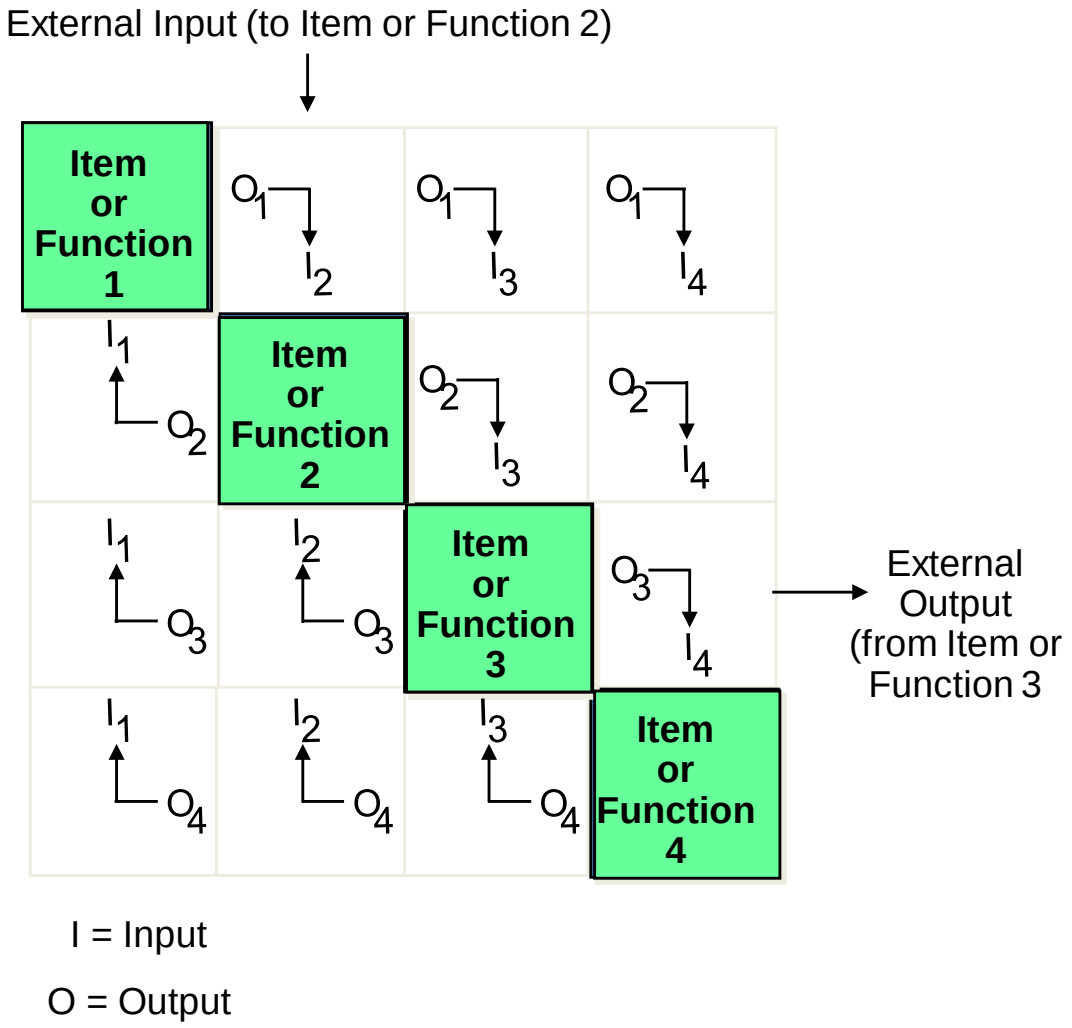
N^2 Diagram

- The N^2 diagram is a visual matrix representing functional or physical interfaces between system elements.
- It is used as a systematic approach to identify, define, tabulate, design, and analyze functional and physical interfaces
- It applies to system interfaces and hardware and/or software interfaces
- N^2 diagrams are a valuable tool for not only identifying functional or physical interfaces, but also for pinpointing areas in which conflicts may arise with interfaces so that system integration proceeds smoothly and efficiently

Generic N-squared Diagram as an Interface Artifact

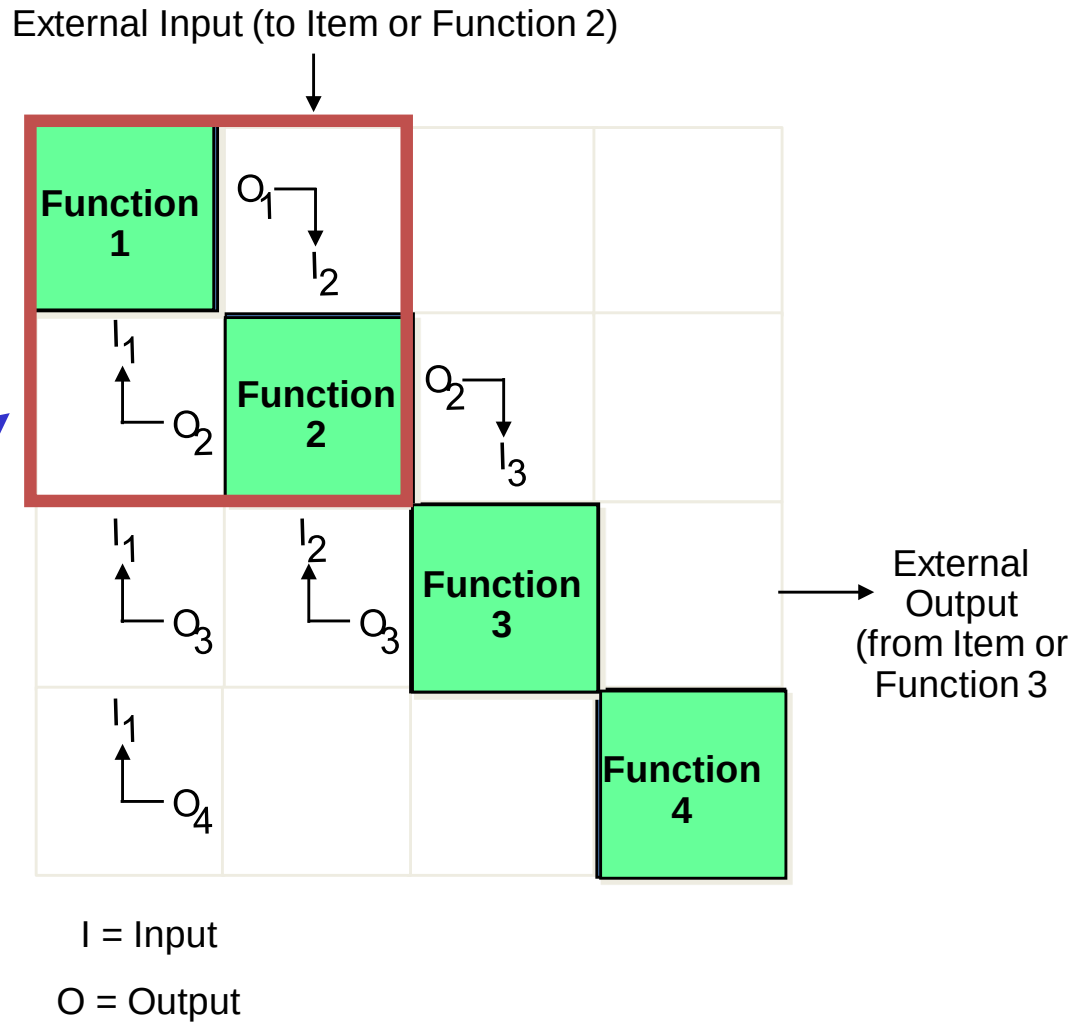
N^2 diagram rules:

- The system components or functions are placed on the diagonal; the remainder of the squares in the N x N matrix represent the interface inputs and outputs.
- Where a blank appears, there is no interface between the respective components or functions.
- Items or functions have input and outputs
- Item or function outputs are contained in rows; inputs are contained in columns



Identify Subsystem Feedback Loops and Candidate Subsystem Boundaries

- **If there is bi-directional information flow between functions - this is a feedback loop.**
- **In this example there are feedback loops between functions 1 & 2 and 2 & 3.**
- **Consider combining functions where there is a lot of 'coupling' (including feedback loops) within a subsystem. This may simplify the subsystem designs and their interfaces.**



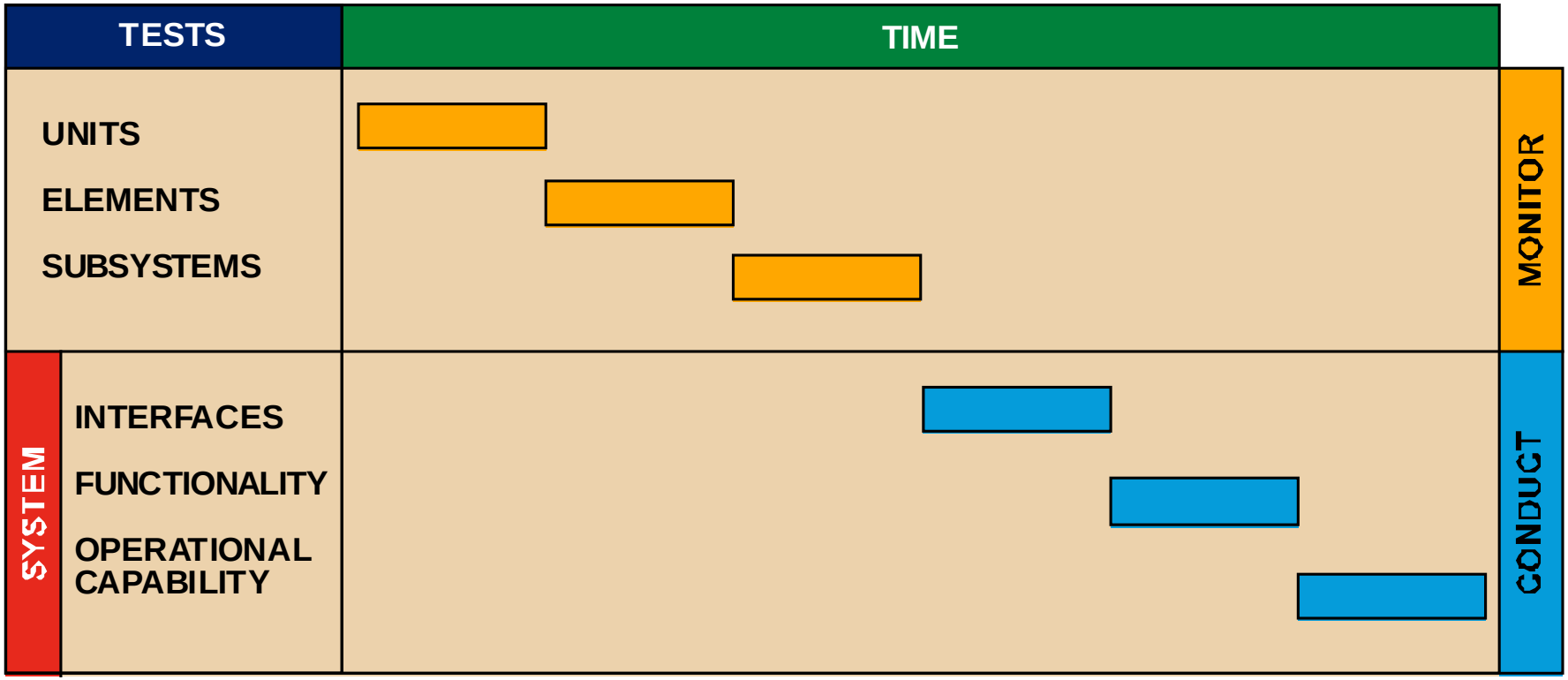
External vs. Internal Interfaces

- **External Interfaces are the boundaries:**
  - **Between a system End Product and another external system End Product; or**
  - **Between a human and the environment in which the system End Products will be used or operated**
- **Internal Interfaces are those boundaries between products within a system model that are controlled by the developer**
- **What are test challenges with respect to external and internal interfaces?**

Interface Test Challenges

- Legacy Systems
- Tough operational environments
- Limited access to test points or data
- New technologies
- Poor documentation
- Most of these problems can be mitigated with early planning

The Ideal Sequence of Test Events



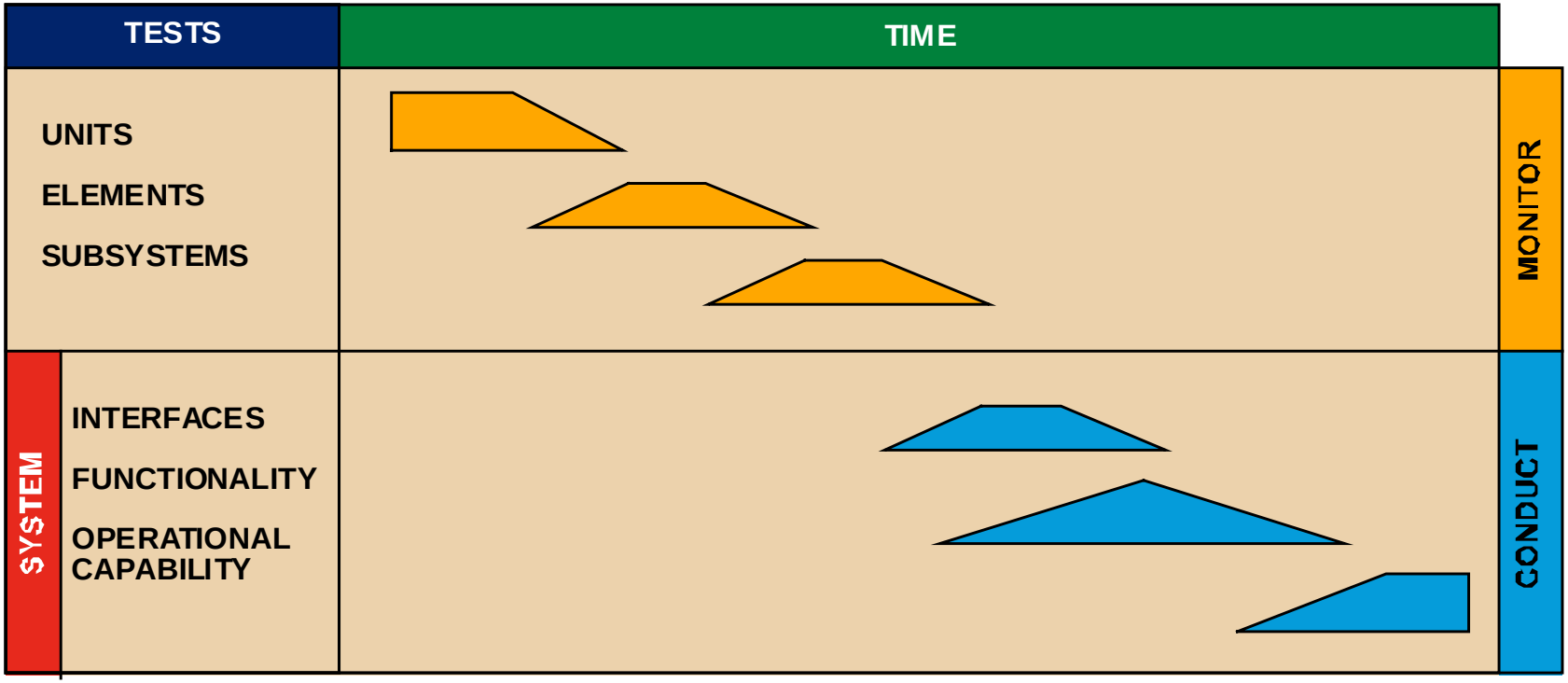
- **Each step completes before the next one begins**
- **Designers and subsystem engineers conduct unit, element, and subsystem tests**
- **The T&E engineer is responsible for demonstrating interfaces, functionality, and operational capability**

Factors Preventing the Ideal Sequencing

- **Schedules**
  - **Requirements indicate approximately when a subsystem will be ready**
  - **Problems in development will dictate when it really will be ready**
  - **System T&E must be flexible enough to respond to schedule uncertainty**

- **Cost**
  - **Ideal sequence is inefficient and therefore costly**
  - **Budget considerations will limit the available amount of funding**

The Real World's Test Sequence



- Unit, element, and subsystem testing must complete before interfaces and functionality can be proven
- System integration (interfaces and functionality) becomes the focus in the project schedule
- To prevent integration from becoming a bottleneck, its contents must be cleverly constructed

How to Construct Tests

- Understand the system operation
- Identify the first data produced
- Establish that the interface(s) which transport(s) the data is(are) correct
- Test the functional thread(s) which utilize the data
- Add tests until all interfaces are verified and all functions are demonstrated … Build a little, Test a little, Learn a lot …
- Verify system performance with all functions operating under full load, if possible

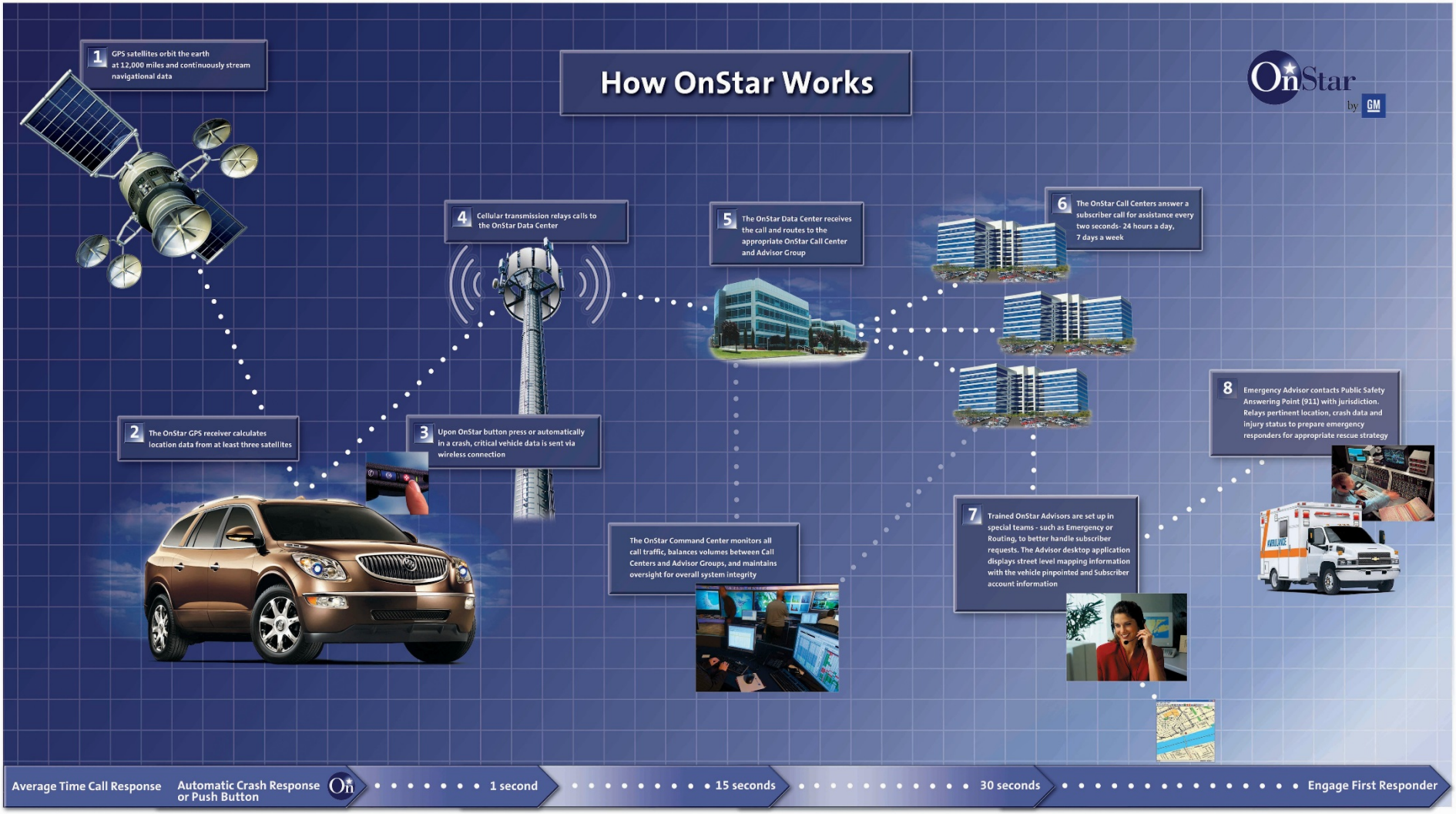
Systems Engineering Plan (SEP) and T&E

- **SEP T&E content should consider:**
  - **Integration of test and evaluation (T&E) planning within the SE approach**
  - **How the measures of effectiveness and performance, KPPs, and critical technical measures are stated as quantifiable parameters**
  - **How verification and validation (V&V) plans will be integrated in the SE approach, particularly how test objectives in the TEMP will be traceable to performance requirements in the system specifications**
  - **The methods the program will use for verification (inspection, analysis (M&S), demonstration, and test) to ensure requirements will be addressed in T&E at the system and subsystem level**

Summary

- **Stakeholder requirements form the program foundation**
- **Integration and T&E activities need to be planned early … and reviewed often**
- **Risk analysis is critical to ensure the right things are tested to an extent that we have adequately mitigated the risks**
- **Interfaces are a common area for test and integration failure and thus require attention to detail and careful planning**
  - **May need agreements with critical external interface owners**
- **SEP, TEMP and WBS are key program management documents AND … play an important part in laying out the integration and test part of the program**

OnStar System Exercise:
Test Planning



OnStar System Exercise:

Test Planning

1. GPS satellites continuously orbit the earth at 12,000 miles and continuously stream navigational data.

2. The OnStar GPS receiver calculates location data from at least three satellites.

3. Upon OnStar button press or automatically in a crash, critical vehicle data is sent via wireless connection.

4. Cellular transmission relays calls to the OnStar Data Center.

5. The OnStar Data Center receives the call and routes the call to the appropriate OnStar Call Center and Advisor Group.

   The OnStar Command Center monitors all call traffic, balances volumes between Call Centers and Advisor Groups, and maintains oversight for overall system integrity.

6. The OnStar Call Centers answer a subscriber call for assistance every two seconds – 24 hours a day, 7 days a week.

7. Trainer OnStar Advisors are set up in special teams – such as Emergency or Routing, to better handle subscriber requests. The Advisor desktop application provides street level mapping information with the vehicle pinpointed and subscriber account information.

8. Emergency Advisor contacts Public Safety Answering Point (911) within jurisdiction. Relays pertinent local, crash data and injury status to prepare emergency responders for appropriate rescue strategy.

OnStar System Exercise:
Test Planning

- For one of the OnStar system Test Events you identified earlier, which interfaces must be tested as part of that event?
- For each of those interfaces, describe the test and evaluation method(s) you would plan to use to verify the correct operation of that interface (T, D, I, A)
- Are there any agreements that need to be negotiated with critical interface owners?
- Be prepared to share your results with the class, including:
  - Descriptions the interfaces, test and evaluation method(s), and any agreements needed with external interface owners
  - Your observations for the exercise



Session 5 – System Integration and Test

Lessons

1. **Program Overview Update**
2. **Process Overviews**
3. **Integration, Planning, Management and Readiness**
4. **System Test Methods**
5. **Test Measurement Techniques and Metrics (if time permits)**

- Relationships with SE Lifecycle
- V&V Processes
- System Interfaces
- Technical Management and Test Processes

- Integration and Test Planning
- Requirements Management
- Interface Management
- Technical Readiness

Appendices*

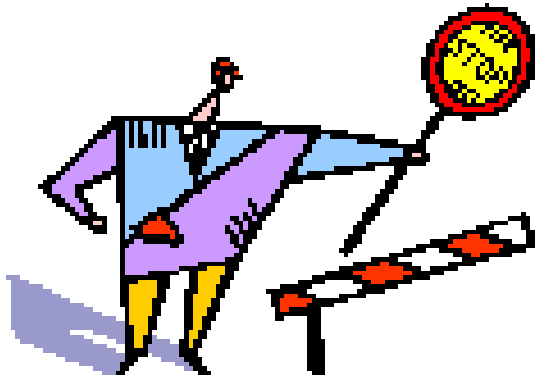
K. **Critical Systems Validation**

L. **Security Testing**

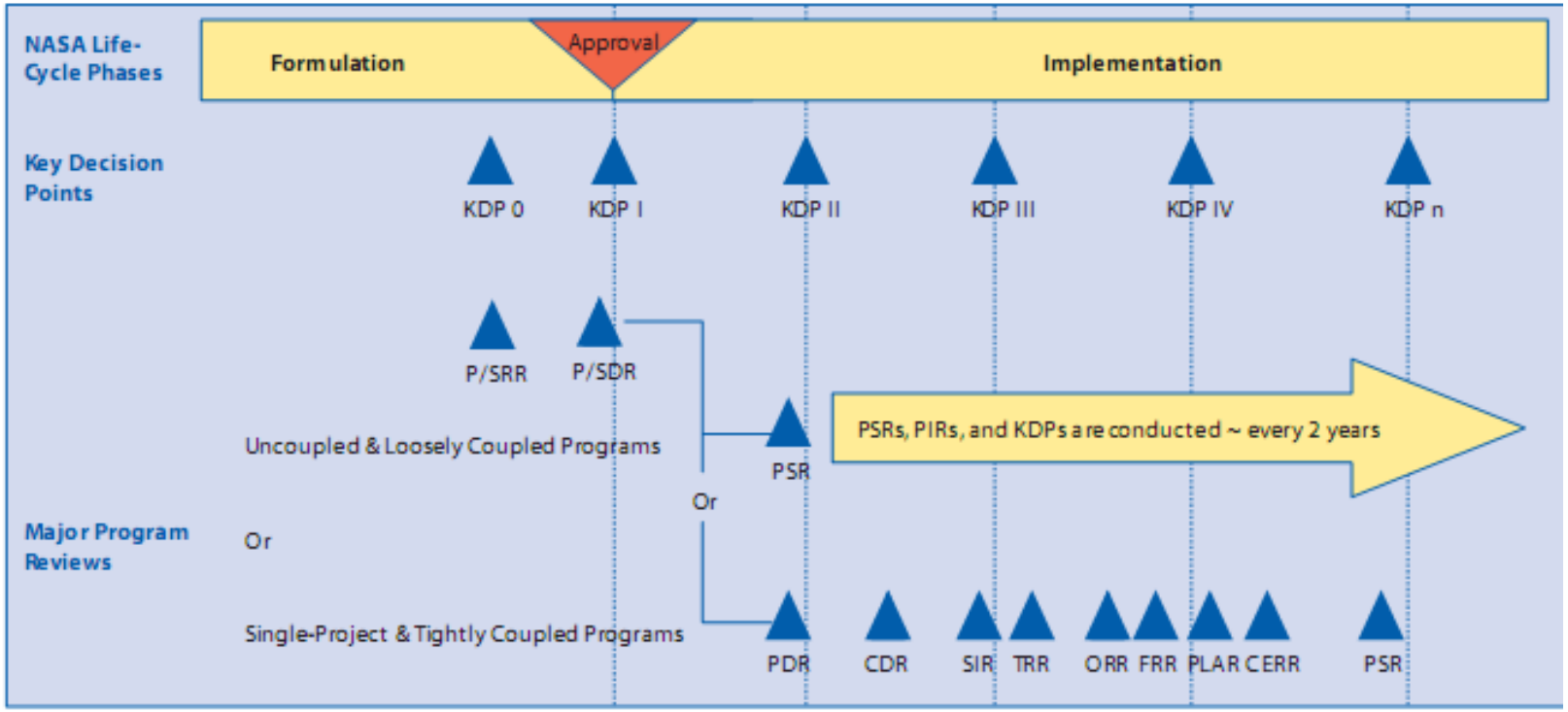
*Appendices A through J were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered.

When Are You Ready For Integration & Test?

- Many Program Managers want to rush into test
- Many grow concerned when too many deficiencies are found
- Gates are established to ensure a program is ready to proceed into its next phase
- The gates take the form of Design and Technical Readiness Reviews
- Tester's play a major role in Design Reviews



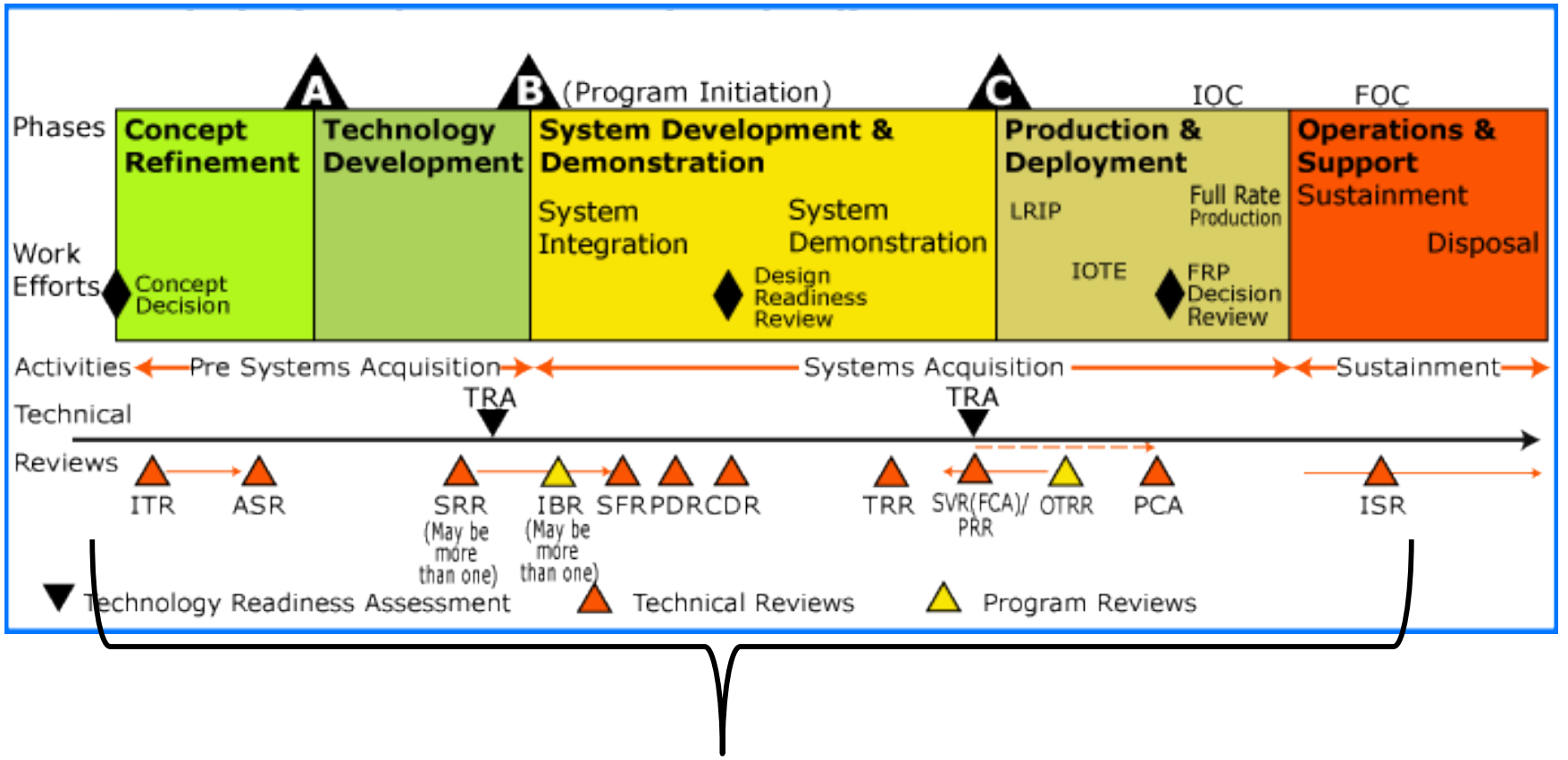
NASA Life Cycle



- CDR Critical Design Review
- CERR Critical Events Readiness Review
- DR Decommissioning Review
- FRR Flight Readiness Review
- KDP Key Decision Point
- MCR Mission Concept Review
- MDR Mission Definition Review
- ORR Operational Readiness Review
- PIR Program Implementation Review
- PLAR Post-Launch Assessment Review
- PRR Production Readiness Review
- P/SDR Program/System Definition Review
- P/SRR Program/System Requirements Review
- PSR Program Status Review
- SAR System Acceptance Review
- SDR System Definition Review
- SIR System Integration Review

There are a lot of gates!

DoD Life Cycle



As testers what do we need to consider at these points?

Technical Design Reviews

- Technical reviews assess the maturity of the product or service under consideration
- Design reviews to be held and process are documented in SEP
  - Reviews are event driven not schedule driven
- Entrance and Exit criteria need to be established
  - Negotiate criteria up front with system developer
  - Systems Engineer needs to ensure program held to criteria
- The focus of reviews depends on where the program is in its lifecycle and what areas need focus/decisions



Integration & Test (I&T) Influence?

- **Integration and Test elements need to be added to entrance and exit criteria**
- **Often data requires detail review before the "formal" review to ensure quality and content is adequate and meets entrance criteria**
- **Tiered testing is often done as the system development progresses**
  - **Progress of achieving performance measures and integration objectives needs to be evaluated**

Metrics and Measures

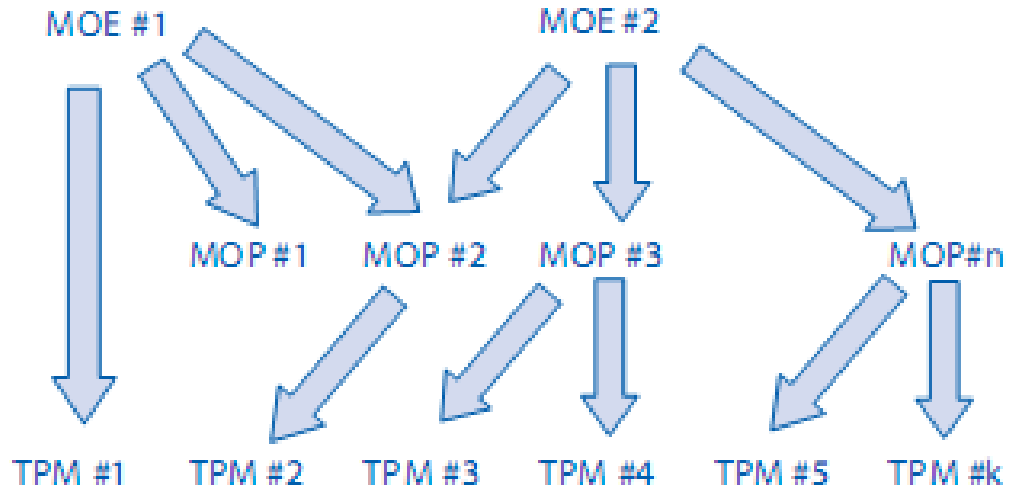
- As testers we need to establish metrics and measures in order to plan test events and establish and monitor program risks

- Examples include:
  - Measure of Performance (MOP)
  - Measure of Effectiveness (MOE)
  - Technical Performance Measure (TPM)
  - Deficiencies found/corrected



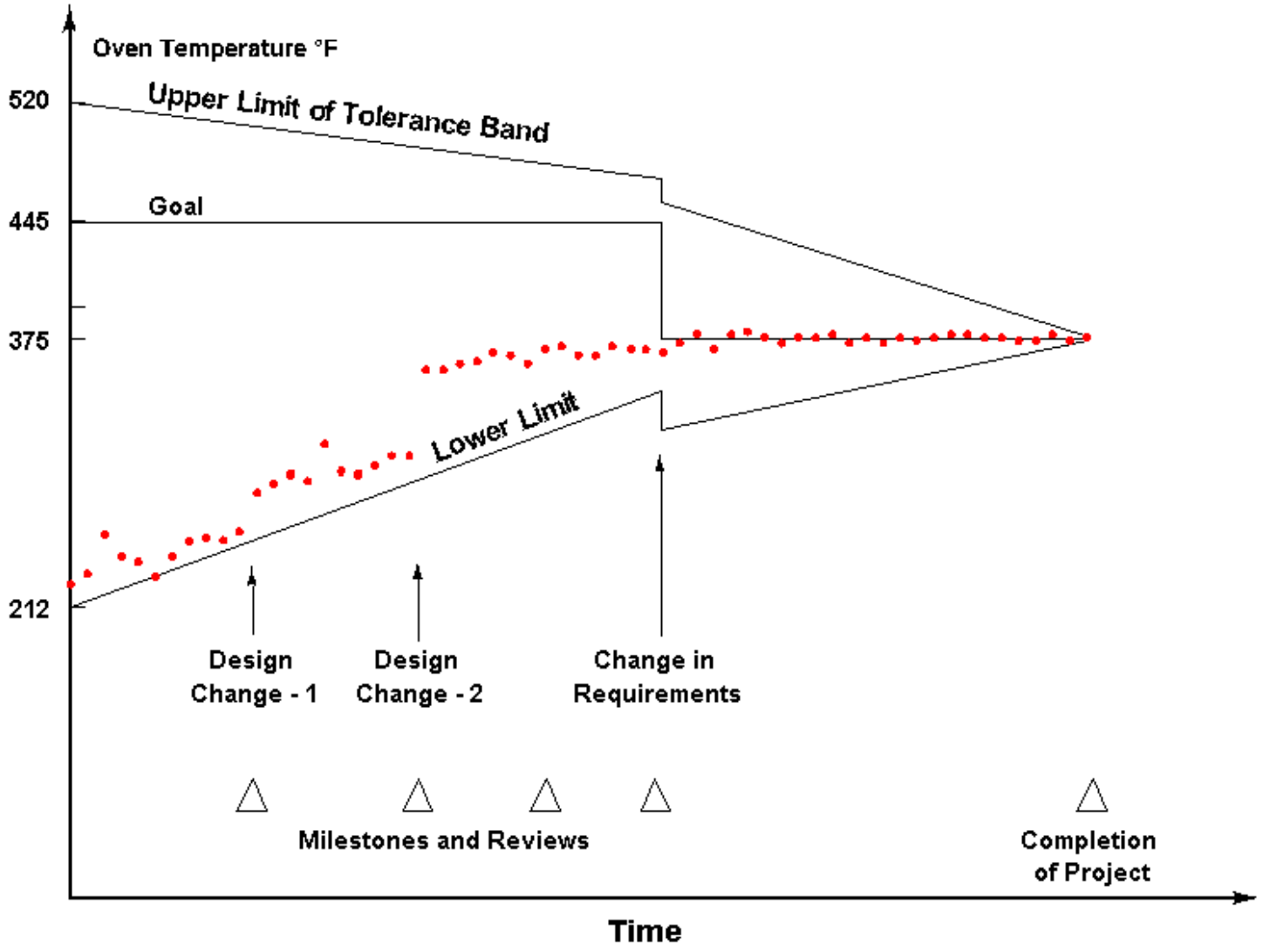
- We will go into more detail in a later lesson on measures and metrics, if time permits

MOE, MOP & TPM Relationships



MOEs	Derived from stakeholder expectation statements; deemed critical to mission or operational success of the system
MOPs	Broad physical and performance parameters; means of ensuring meeting the associated MOEs
TPMs	Critical mission success or performance attributes; measurable; progress profile established, controlled, and monitored

TPM Example


Oven Temperature °F
520
Upper Limit of Tolerance Band
Goal
445
375
Lower Limit
212
Design Change - 1
Design Change - 2
Change in Requirements
Milestones and Reviews
Completion of Project
Time

Systems Requirements Review (SRR)

- Requirements have been identified and are complete or, if there are shortfalls, where they are and the risk associated with each shortfall
- Mutual agreement between all parties
- Results of integrated test planning are reviewed
- Issues of testability of requirements should be addressed
- Successful completion of the SRR freezes program/project requirements

SRR Criteria Example

System Requirements Review	
Entrance Criteria	**Success Criteria**
1. Successful completion of the MCR and responses made to all MCR RFAs and RIDs. 2. A preliminary SRR agenda, success criteria, and charge to the board have been agreed to by the technical team, project manager, and review chair prior to the SRR. 3. The following technical products for hardware and software system elements are available to the cognizant participants prior to the review: a. system requirements document; b. system software functionality description; c. updated ConOps; d. updated mission requirements, if applicable; e. baselined SEMP; f. risk management plan; g. preliminary system requirements allocation to the next lower level system; h. updated cost estimate; i. technology development maturity assessment plan; j. updated risk assessment and mitigations (including PRA, as applicable); k. logistics documentation (e.g., preliminary maintenance plan); l. preliminary human rating plan, if applicable; m. software development plan; n. system SMA plan; o. CM plan; p. initial document tree; q. verification and validation approach; r. preliminary system safety analysis; and s. other specialty disciplines, as required.	1. The project utilizes a sound process for the allocation and control of requirements throughout all levels, and a plan has been defined to complete the definition activity within schedule constraints. 2. Requirements definition is complete with respect to top-level mission and science requirements, and interfaces with external entities and between major internal elements have been defined. 3. Requirements allocation and flowdown of key driving requirements have been defined down to subsystems. 4. Preliminary approaches have been determined for how requirements will be verified and validated down to the subsystem level. 5. Major risks have been identified and technically assessed, and viable mitigation strategies have been defined.

Preliminary Design Review (PDR)

- Confirms the preliminary design logically follows the SRR findings and meets the requirements
- Reviews draft Configuration Item (CI) specifications
- Reviews development test and operational test activities to measure the performance of each configuration item
- Looks at the existence and compatibility of the physical and functional interfaces among the CI's and other equipment
- Often seen by many as the last viable point for effective technology insertion before the start of detail design
- As a result of successful completion of the PDR
  - The design to baseline is approved
  - Authorizes the project to proceed into implementation and toward final detailed design

PDR Criteria Example

Preliminary Design Review	
Entrance Criteria	**Success Criteria**
1. Successful completion of the SDR or MDR and responses made to all SDR or MDR RFAs and RIDs, or a timely closure plan exists for those remaining open. 2. A preliminary PDR agenda, success criteria, and charge to the board have been agreed to by the technical team, project manager, and review chair prior to the PDR. 3. PDR technical products listed below for both hardware and software system elements have been made available to the cognizant participants prior to the review: a. Updated baselined documentation, as required. b. Preliminary subsystem design specifications for each configuration item (hardware and software), with supporting tradeoff analyses and data, as required. The preliminary software design specification should include a completed definition of the software architecture and a preliminary database design description as applicable. c. Updated technology development maturity assessment plan. d. Updated risk assessment and mitigation. e. Updated cost and schedule data. f. Updated logistics documentation, as required. g. Applicable technical plans (e.g., technical performance measurement plan, contamination control plan, parts management plan, environments control plan, EMI/EMC control plan, payload-to-carrier integration plan, producibility/manufacturability program plan, reliability program plan, quality assurance plan). h. Applicable standards. i. Safety analyses and plans. j. Engineering drawing tree. k. Interface control documents. l. Verification and validation plan. m. Plans to respond to regulatory (e.g., National Environmental Policy Act) requirements, as required. n. Disposal plan. o. Technical resource utilization estimates and margins. p. System-level safety analysis. q. Preliminary LLIL.	1. The top-level requirements—including mission success criteria, TPMs, and any sponsor-imposed constraints—are agreed upon, finalized, stated clearly, and consistent with the preliminary design. 2. The flowdown of verifiable requirements is complete and proper or, if not, an adequate plan exists for timely resolution of open items. Requirements are traceable to mission goals and objectives. 3. The preliminary design is expected to meet the requirements at an acceptable level of risk. 4. Definition of the technical interfaces is consistent with the overall technical maturity and provides an acceptable level of risk. 5. Adequate technical interfaces are consistent with the overall technical maturity and provide an acceptable level of risk. 6. Adequate technical margins exist with respect to TPMs. 7. Any required new technology has been developed to an adequate state of readiness, or backup options exist and are supported to make them a viable alternative. 8. The project risks are understood and have been credibly assessed, and plans, a process, and resources exist to effectively manage them. 9. SMA (e.g., safety, reliability, maintainability, quality, and EEE parts) has been adequately addressed in preliminary designs and any applicable SMA products (e.g., PRA, system safety analysis, and failure modes and effects analysis) have been approved. 10. The operational concept is technically sound, includes (where appropriate) human factors, and includes the flowdown of requirements for its execution.

Critical Design Review (CDR)

- Can be conducted at the CI level or the whole system
- All CI's are reviewed with respect to technical, cost and schedule
- Qualification test results contribute to the decision process
- Completion of CDR
  - allows a program to proceed to Low Rate Production
  - the build-to baseline, production, and verification plans are approved
  - authorizes coding of deliverable software (according to the build-to baseline and coding standards presented in the review), and system qualification testing and integration
- All open issues should be resolved with closure actions and schedule

CDR Criteria Example

Critical Design Review	
Entrance Criteria	**Success Criteria**
1. Successful completion of the PDR and responses made to all PDR RFAs and RIDs, or a timely closure plan exists for those remaining open. 2. A preliminary CDR agenda, success criteria, and charge to the board have been agreed to by the technical team, project manager, and review chair prior to the CDR. 3. CDR technical work products listed below for both hardware and software system elements have been made available to the cognizant participants prior to the review: a. updated baselined documents, as required; b. product build-to specifications for each hardware and software configuration item, along with supporting tradeoff analyses and data; c. fabrication, assembly, integration, and test plans and procedures; d. technical data package (e.g., integrated schematics, spares provisioning list, interface control documents, engineering analyses, and specifications); e. operational limits and constraints; f. technical resource utilization estimates and margins; g. acceptance criteria; h. command and telemetry list; i. verification plan (including requirements and specifications); j. validation plan; k. launch site operations plan; l. checkout and activation plan; m. disposal plan (including decommissioning or termination); n. updated technology development maturity assessment plan; o. updated risk assessment and mitigation; p. update reliability analyses and assessments; q. updated cost and schedule data; r. updated logistics documentation; s. software design document(s) (including interface design documents); t. updated LLIL; u. subsystem-level and preliminary operations safety analyses; v. system and subsystem certification plans and requirements (as needed); and w. system safety analysis with associated verifications.	1. The detailed design is expected to meet the requirements with adequate margins at an acceptable level of risk. 2. Interface control documents are appropriately matured to proceed with fabrication, assembly, integration, and test, and plans are in place to manage any open items. 3. High confidence exists in the product baseline, and adequate documentation exists or will exist in a timely manner to allow proceeding with fabrication, assembly, integration, and test. 4. The product verification and product validation requirements and plans are complete. 5. The testing approach is comprehensive, and the planning for system assembly, integration, test, and launch site and mission operations is sufficient to progress into the next phase. 6. Adequate technical and programmatic margins and resources exist to complete the development within budget, schedule, and risk constraints. 7. Risks to mission success are understood and credibly assessed, and plans and resources exist to effectively manage them. 8. SMA (e.g., safety, reliability, maintainability, quality, and EEE parts) have been adequately addressed in system and operational designs, and any applicable SMA plan products (e.g., PRA, system safety analysis, and failure modes and effects analysis) have been approved.

Test Readiness Review(s) (TRR)

- Ensures that the test article (hardware/software), test facility, support personnel, and test procedures are ready for testing and data acquisition, analysis, and control
- Held prior to commencement of verification or validation testing
  - Good engineering practice to have prior to any testing, even if informal
- A successful TRR signifies that test engineers have certified that preparations are complete, and that the project manager has authorized formal initiation of verification and validation testing

TRR

Test Readiness Review	
Entrance Criteria	**Success Criteria**
1. The objectives of the testing have been clearly defined and documented and all of the test plans, procedures, environment, and the configuration of the test item(s) support those objectives. 2. Configuration of the system under test has been defined and agreed to. All interfaces have been placed under configuration management or have been defined in accordance with an agreed-to plan, and a version description document has been made available to TRR participants prior to the review. 3. All applicable functional, unit-level, subsystem, system, and qualification testing has been conducted successfully. 4. All TRR-specific materials such as test plans, test cases, and procedures have been made available to all participants prior to conducting the review. 5. All known system discrepancies have been identified and disposed in accordance with an agreed-upon plan. 6. All previous design review success criteria and key issues have been satisfied in accordance with an agreed-upon plan. 7. All required test resources people (including a designated test director), facilities, test articles, test instrumentation, and other enabling products have been identified and are available to support required tests. 8. Roles and responsibilities of all test participants are defined and agreed to. 9. Test contingency planning has been accomplished, and all personnel have been trained.	1. Adequate test plans are completed and approved for the system under test. 2. Adequate identification and coordination of required test resources are completed. 3. Previous component, subsystem, and system test results form a satisfactory basis for proceeding into planned tests. 4. Risk level is identified and accepted by program/competency leadership as required. 5. Plans to capture any lessons learned from the test program are documented. 6. The objectives of the testing have been clearly defined and documented, and the review of all the test plans, as well as the procedures, environment, and the configuration of the test item, provide a reasonable expectation that the objectives will be met. 7. The test cases have been reviewed and analyzed for expected results, and the results are consistent with the test plans and objectives. 8. Test personnel have received appropriate training in test operation and safety procedures.

Summary

- We reviewed only a few key reviews but … Integration and Test is part of every review
- Reviews are your gauge to how the system is proceeding to Integration and Test
- Reviews are gates; you should not proceed to the next step until the agreed upon entrance and exit criteria are met
- Formality is not as essential as the process, informal reviews can be very valuable



Systems Engineering Technical Short Course

(Session 5: System Integration and Test,

AKA: Verification, Validation and Test - VVT)

Lesson 4:
System Test Methods

Session 5 – System Integration and Test

Lessons

1. Program Overview Update
2. Process Overviews
3. Integration, Planning, Management and Readiness
4. System Test Methods
5. Test Measurement Techniques and Metrics (if time permits)

- Relationships with SE Lifecycle
- V&V Processes
- System Interfaces
- Technical Management and Test Processes

- Integration and Test Planning
- Requirements Management
- Interface Management
- Technical Readiness

- White Box Testing
- Black Box Testing
- Integration Testing

Appendices*

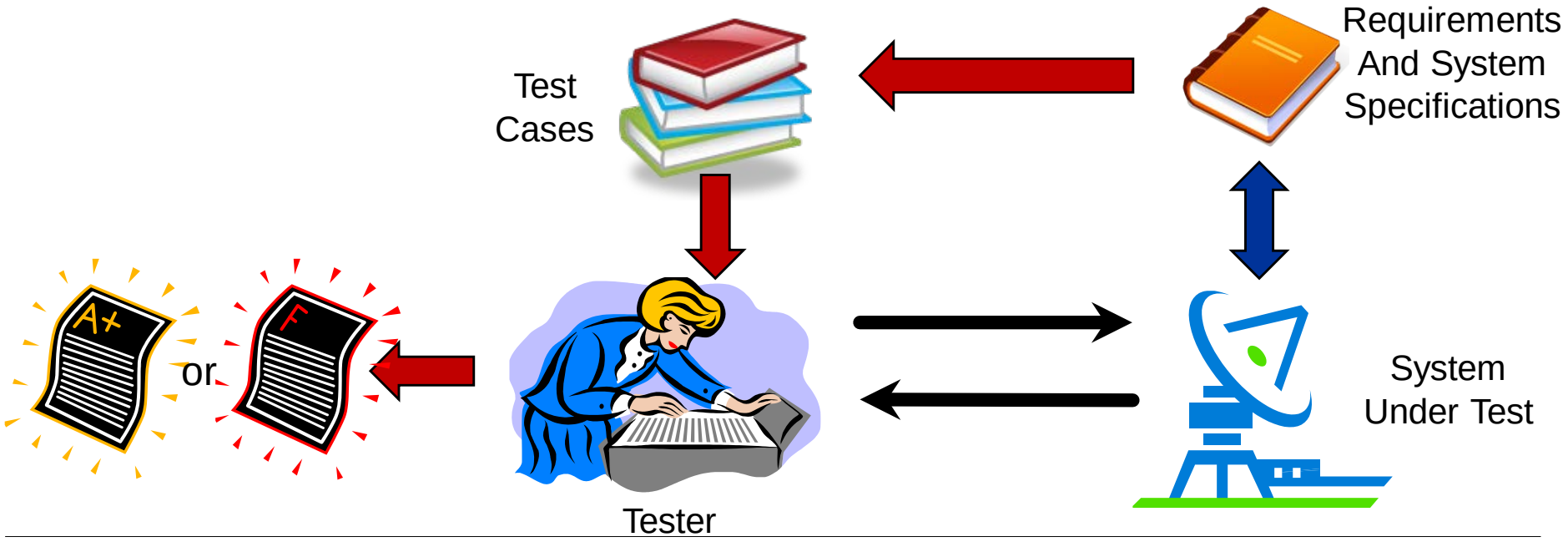
K. Critical Systems Validation

L. Security Testing

*Appendices A through J were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered.

Discussion

- Remember, Testing is a subset of verification & validation
- We test to system specification and requirements
  - Requirements and specifications form the foundation from which test cases are generated
- Testing must confirm that the system does what it supposed to do and that the system does NOT do what it should NOT be doing especially with respect to safety of life



AT&T Outage

- Several hours of outage for the entire eastern US phone service (1990)
- Caused by minor software error in AT&T switching system
- Single misplaced statement inserted as part of a three line software "fix"
  - Less than 0.0005% of the software was changed
- Cost documented at $1+ billion

Shuttle Rendezvous

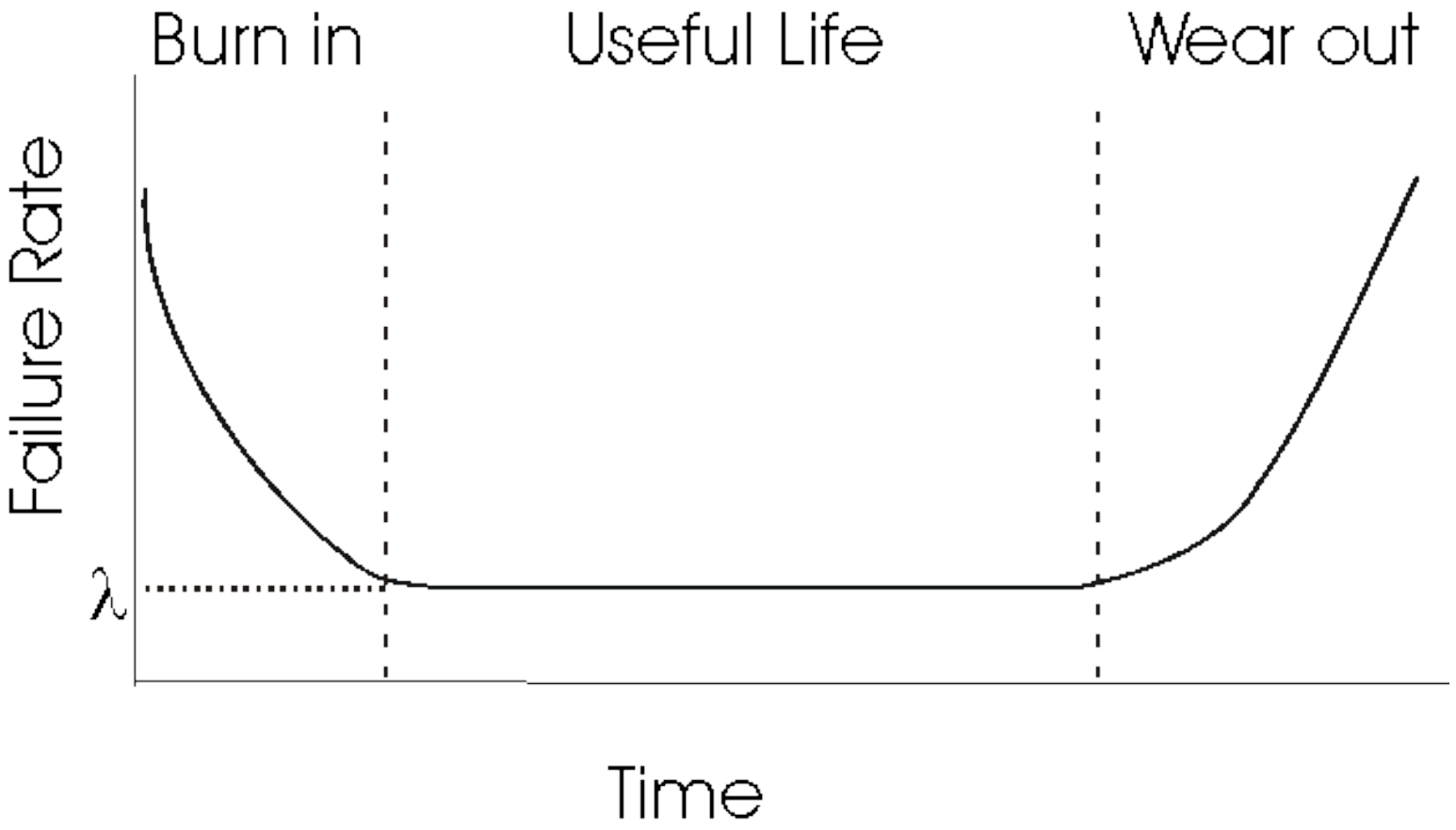
- Shuttle Endeavor rendezvous with Hughes F/6 Intelsat satellite (1992)
  - Position related 64 bit floating point variable stored in 32 bit floating point number
  - Caused significant error in space location

- IEEE 64 bit floating point number
  - S eeeeeeeeeee ffffffffffffffffffffffffffffffffffffffffffffffffffff
  - There is a leading sign bit, S, 11 bit exponent, 52 bit mantissa

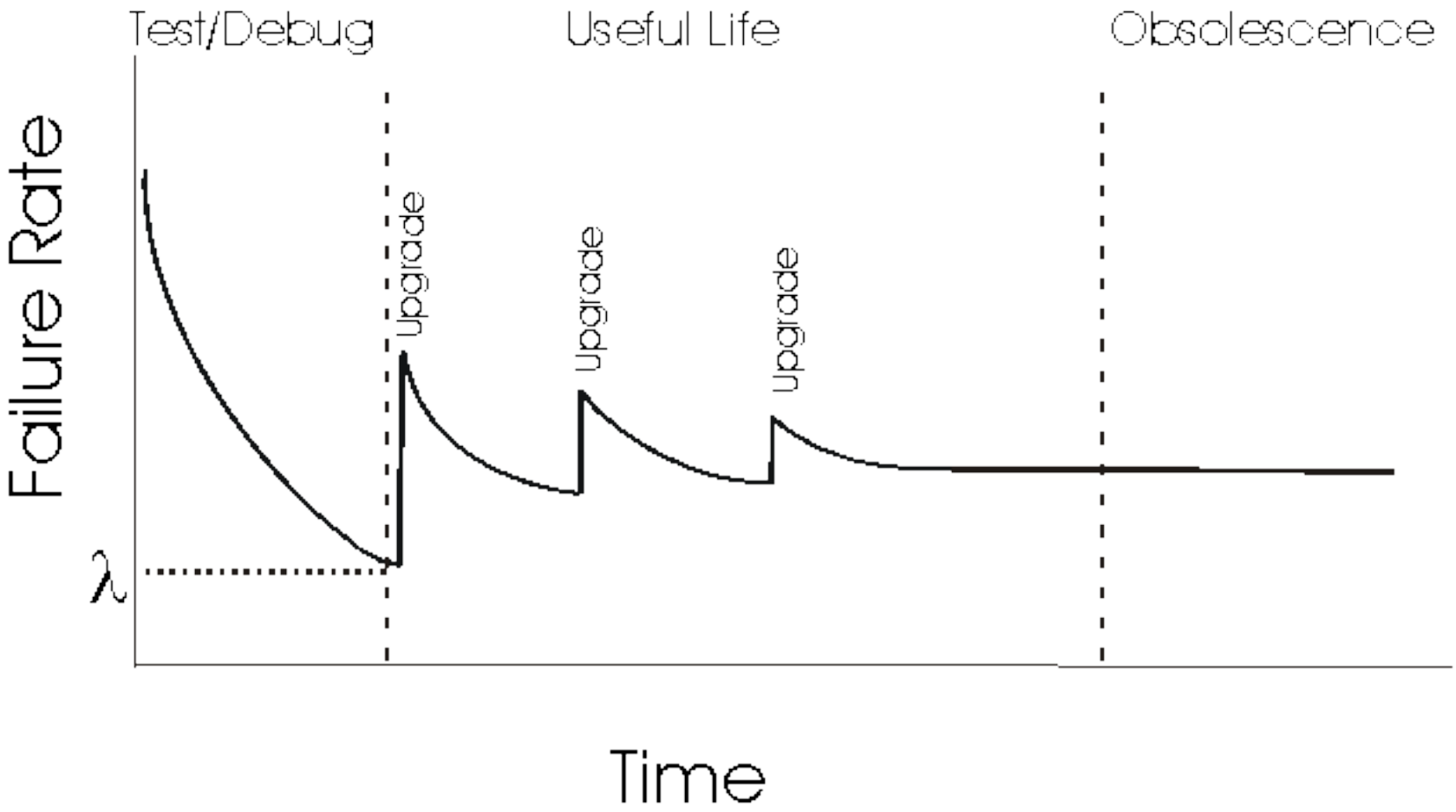
- IEEE 32 bit floating point number
  - S eeeeeeee fffffffffffffffffffff
  - There is a leading sign bit, S, 8 bit exponent, 23 bit mantissa

What Are Some of the Lessons to Be Learned From These Examples?

Hardware Failure Curve

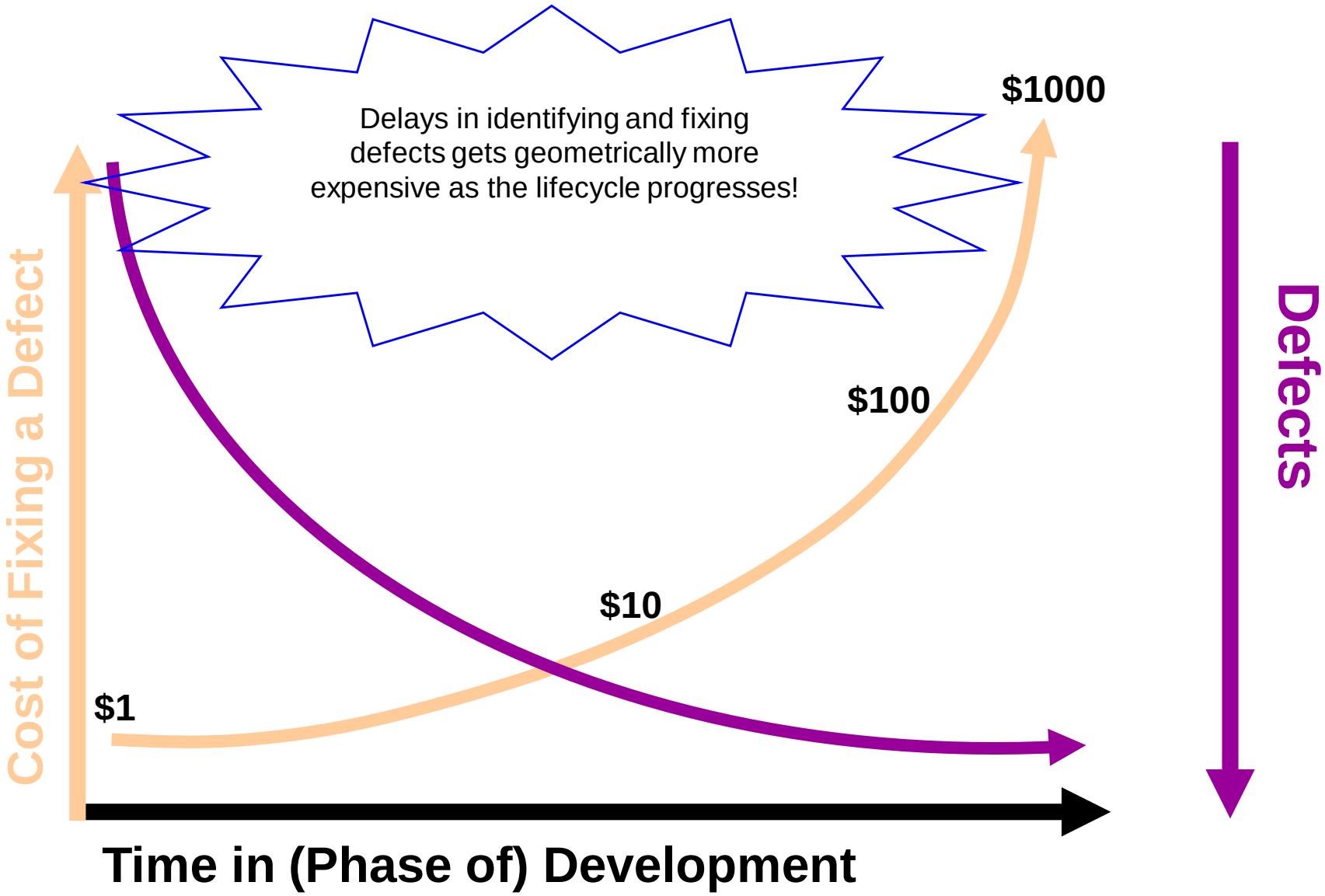


Software Failure Curve

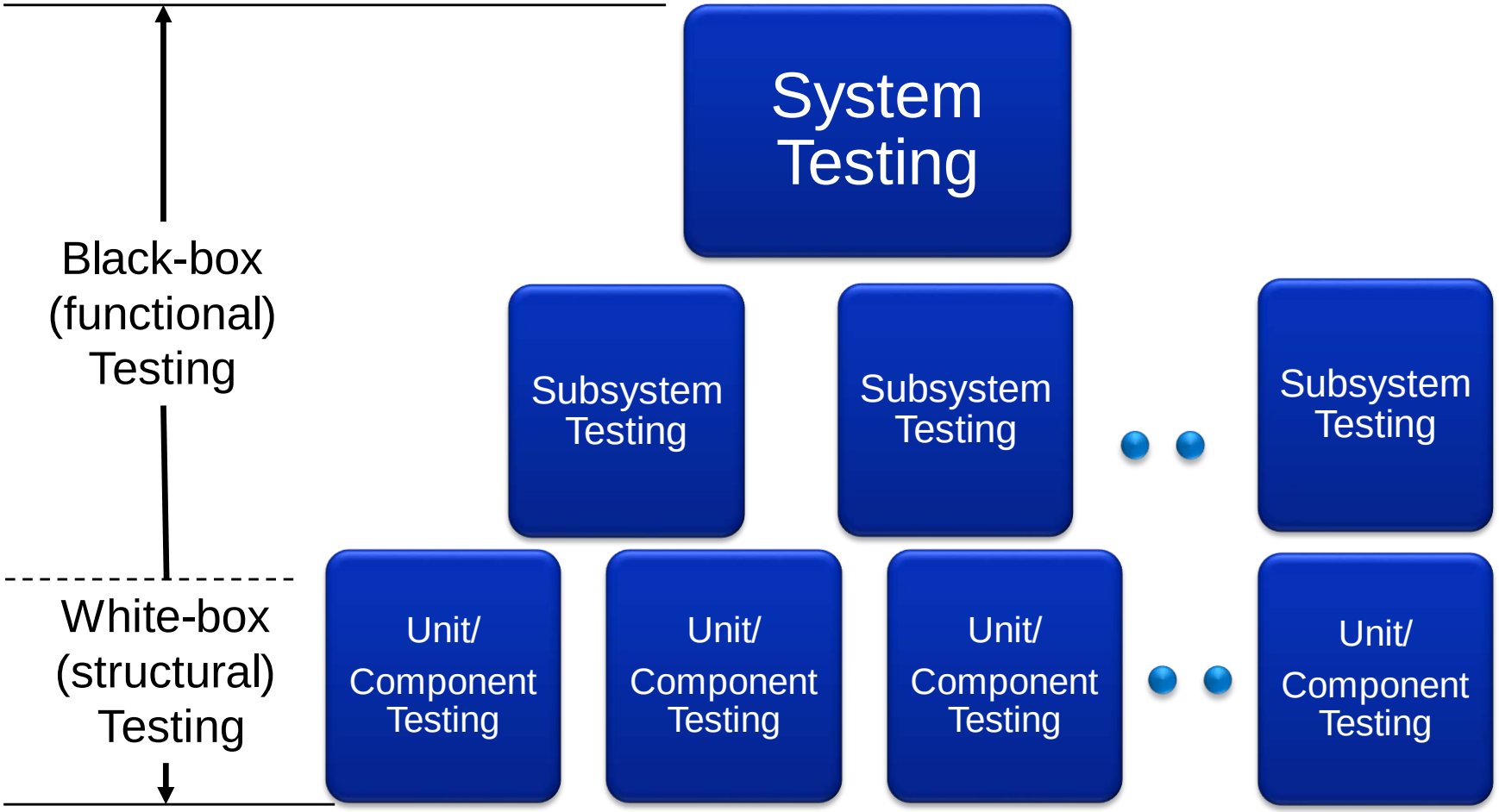


What Are The Implications on Test Due to the Software Failure Curve?

Costs of Finding and Fixing Late



White Box vs. Black Box Testing



Session 5 – System Integration and Test

Lessons

1. Program Overview Update
2. Process Overviews
3. Integration, Planning, Management and Readiness
4. System Test Methods
5. Test Measurement Techniques and Metrics (if time permits)

- Relationships with SE Lifecycle
- V&V Processes
- System Interfaces
- Technical Management and Test Processes

- Integration and Test Planning
- Requirements Management
- Interface Management
- Technical Readiness

- White Box Testing
- Black Box Testing
- Integration Testing

Appendices*

K. Critical Systems Validation

L. Security Testing

*Appendices A through J were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered.

White-Box (Structural) Testing

- **Definition:** testing that uses knowledge of internal design and structure of a system

- Allows the engineer to:
  - Tests independent paths
  - Exercise all logical decisions in both true and false conditions
  - Execute all loops at boundaries and within operational bounds
  - Exercise all internal data structures

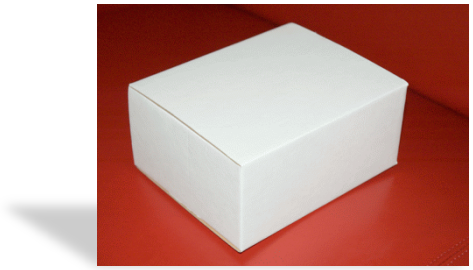
- Reasons for white box testing
  - Defects are typically random, but uniformly distributed in code
  - Typical operation of software spends 90% of the execution time in only 25% of the code
  - We tend often to miscalculate our test coverage when planning tests from a functional perspective

- Typically performed in earlier test phases

- Typically tests design implementation

White-Box Testing

- **Advantages**
  - Test engineer's knowledge of system design enables targeting of potentially weak design points
  - Forces test developer to reason carefully about implementation
  - Reveals errors in "hidden" software code
  - Beneficial side-effects
  - Optimizations



- **Disadvantages**
  - Expensive
  - Complexity of modern systems makes total testing difficult or impossible
  - Must have skills in the domain and internal structure
  - Often will not detect missing or incorrect functionalities

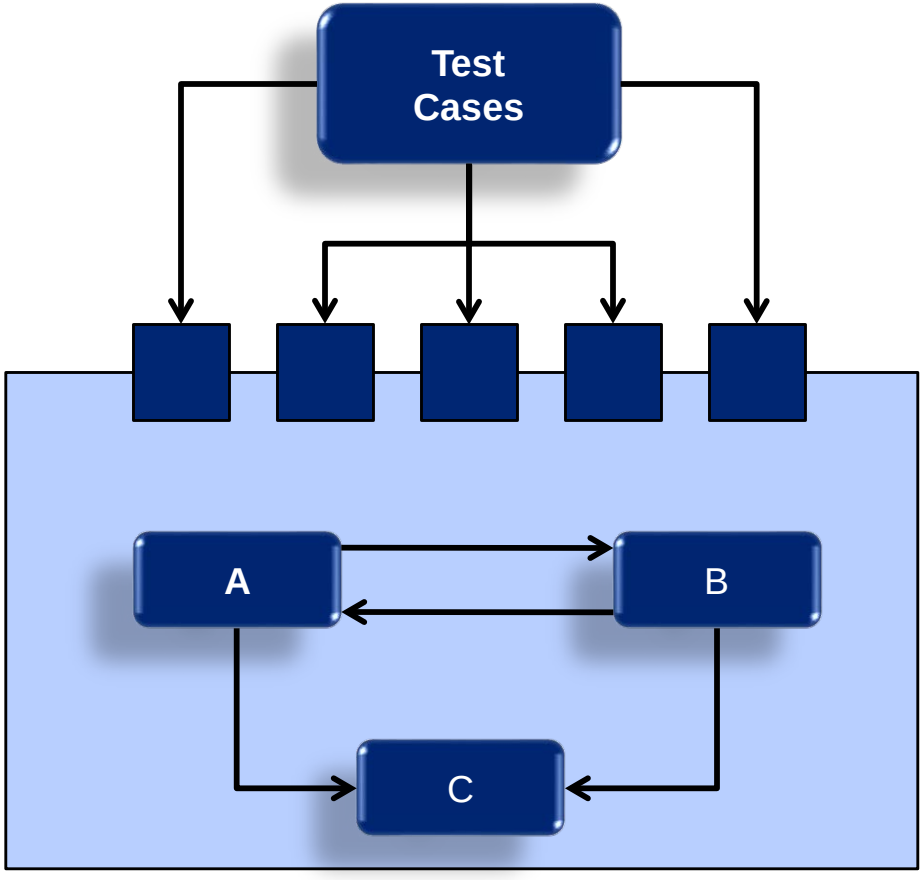
Interface Testing

- Objectives are to detect faults due to interface errors or invalid assumptions about interfaces

- **Categories**
  - Material – need for materials exchange
  - Spatial – need for adjacency, force transfer or orientation
  - Energy – energy transfer
  - Information – information or signal

- **Information Interface classes**
  - Hardware/Hardware
  - Hardware/Software
  - Software/Software
  - Human/Systems

Interface Testing



Interface Errors

- **Interface Misuse** – one component or system does not follow the prescribed interface rules
- **Interface Erroneous Assumptions** – one component or system makes wrong assumptions about the dynamic behavior of another system
- **Interface Timing Errors** – components operating at difference speeds or non-synchronization of data



Software Interface Testing

- **Parameter interfaces**
  - Data passed from one procedure to another
- **Shared memory interfaces**
  - Block of memory is shared between procedures or functions
- **Procedural interfaces**
  - Sub-system encapsulates a set of procedures to be called by other sub-systems
- **Message passing interfaces**
  - Sub-systems request services from other sub-systems

Interface Testing Guidelines

- Design tests so that parameters to a called procedure are at the extreme ends of their ranges
- Always test pointer parameters with null pointers
- Design tests which cause the component to fail
- Use stress testing in message passing systems
- In shared memory systems, vary the order in which components are activated

Session 5 – System Integration and Test

Lessons

1. Program Overview Update
2. Process Overviews
3. Integration, Planning, Management and Readiness
4. System Test Methods
5. Test Measurement Techniques and Metrics (if time permits)

- Relationships with SE Lifecycle
- V&V Processes
- System Interfaces
- Technical Management and Test Processes

- Integration and Test Planning
- Requirements Management
- Interface Management
- Technical Readiness

- White Box Testing
- Black Box Testing
- Integration Testing



Appendices*

K. Critical Systems Validation

L. Security Testing

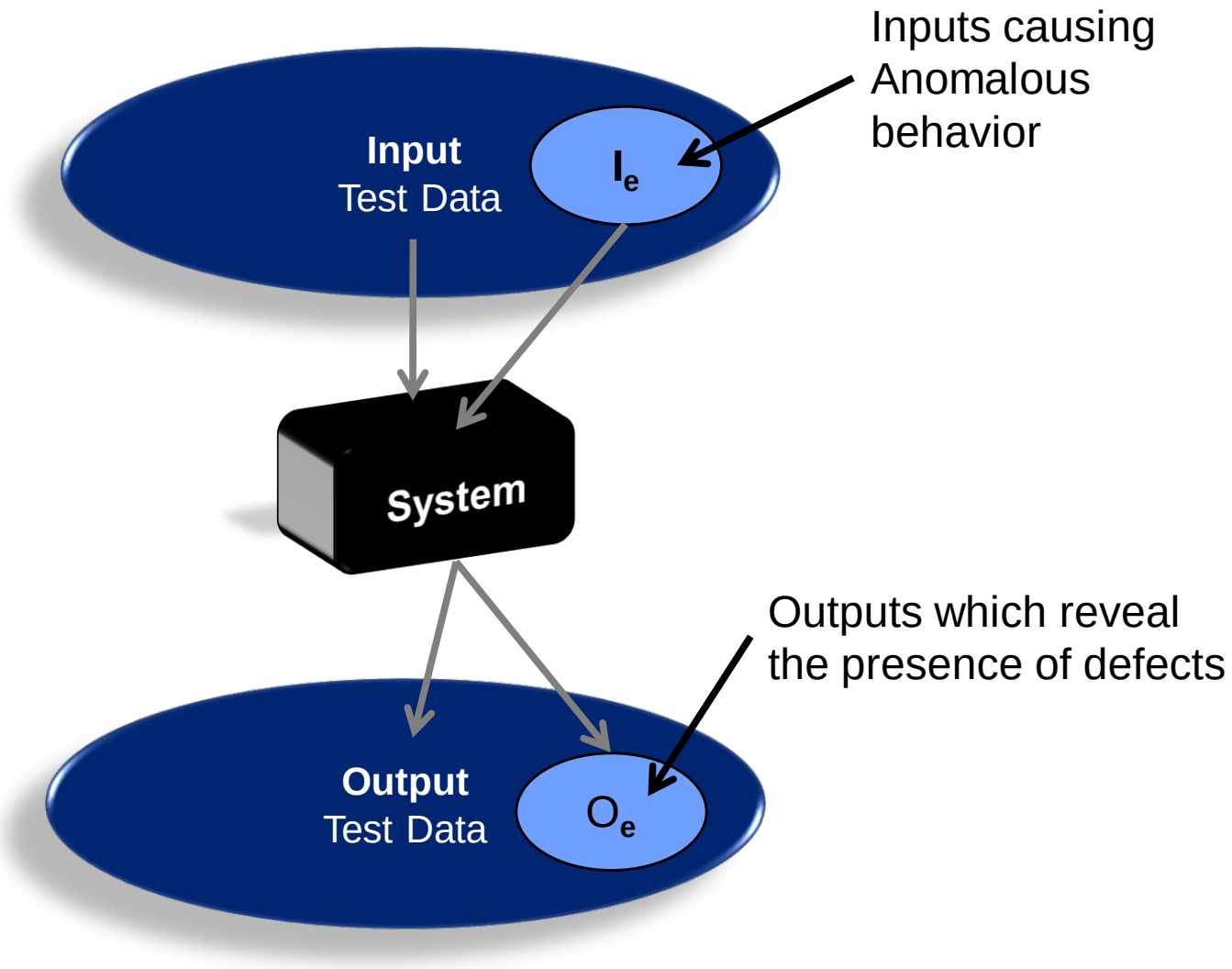
*Appendices A through J were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered.

Black Box Testing

- **Definition:** black box testing / functional testing
  - Testing which ignores the internal mechanism of a system and focuses on response/output of the system to specific controlled inputs
  - Evaluates the system for compliance with functional requirements
- Test cases fully exercise all functional requirements for a system
- Views the system as a black box which performs the requirements
- Typically performed in later test phases
- Typically tests requirements



Black Box Testing



Black Box Testing

- **Advantages**
  - More effective on larger units of code than white box testing
  - Tester needs no knowledge of implementation, including specific programming languages
  - Tester and developer are independent of each other
  - Tests are done from a user's point of view
  - Will help to expose any ambiguities or inconsistencies in the specifications
  - Test cases can be designed as soon as the specifications are complete
  - Appropriate for all levels of testing

- **Disadvantages**
  - Only a small number of possible inputs can actually be tested
  - Without clear and concise specifications, test cases are hard to design
  - There may be unnecessary repetition of test inputs if the tester is not informed of test cases the developer has already tried
  - May leave many software program paths untested
  - Cannot be directed toward specific segments of code which may be very complex (and therefore more error prone)
  - Most testing related research has been directed toward white box testing

Black Box Basic Testing

- **Boundary Values**
  - Verify the behavior of the system operating boundary areas
  - Select data that operates the system at the extremes
- **Decision Tables**
  - Validates system under specified conditions and constraints
  - Conditions correspond to variables that could affect behavior
  - Actions are operations preformed under a stated conditions
- **Finite State Machines**
  - Evaluates system control functions through system state behavior changes

Human-System Interface (HSI) Testing

- Verifies that humans can:
  - Control the system - ergonomics
  - Receive appropriate and timely information

- Must consider:
  - Unpredictability of users
  - System mission(s)
  - System technology
  - Operational environment



Human Input Testing

- Must validate the system responds to proper commands or sequences
- Verify the system recognizes, tolerates and properly handles operator errors
  - Violate data type or size
  - Violate user input restrictions
  - Skip mandatory fields
  - Inundate system
  - Generate unexpected sequences
- **Exceptions** – an internal event that signals an error condition
  - Real-time validation – system checks if input meets expectation
  - Committed-value validation – checks that the entire input field meets expected values
  - Pass-through validation – checks all fields at once
  - Input errors will generate an exception that a tester can use

Human Output Testing

- Verifies the proper output is achieved per the specification *AND* that humans react properly to the output information



Performance Testing

- Demonstrate system meets its performance characteristics
- Tests run-time performance of software within context of the integrated system
- Needs to occur at all levels of testing based upon performance allocations specified by systems engineers
- Usually includes load and volume testing:
  - Task Response Time
  - System External Capacity
  - System Resources
  - System Reliability/Stability



Recovery Testing

- Verifies a system's ability to recover from a variety of faults and resume operation without a lost of data

- Testing is designed to force the system to fail and recover per the system requirements
  - Can inject a fault (software or physical)
  - Software tools can be used to modify code to inject a failure



Stress Testing

- Similar to performance testing but … the objective is to operate the system beyond normal operating conditions and observe the results

- Can be done by
  - Operating the system beyond nominal environment specifications
  - Overwhelm system resources

- System should degrade gracefully rather than catastrophically



EMC/EMI Testing

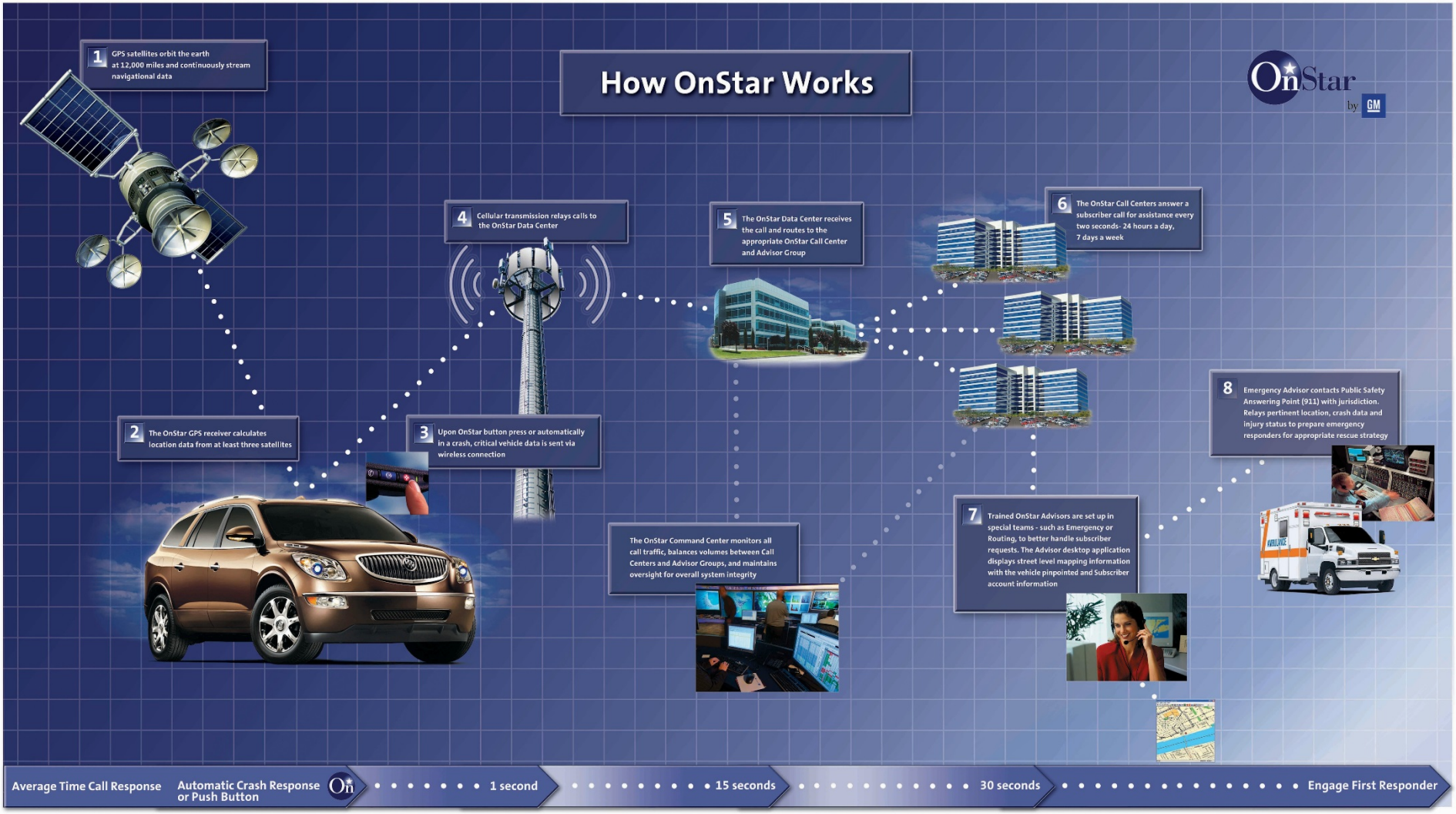
- Verify correct operation of a system in an electromagnetic environment where equipment may emit or be susceptible to electromagnetic effects
- Does the system operate in its approved spectrum?
- Commercial and military standards define testing to be done
  - Conducted emissions
  - Radiated emissions
  - Conducted susceptibility
  - Radiated susceptibility



Regression Testing

- Selective retesting of a system that has been modified to ensure modifications have been properly implemented with no adverse affects on other system elements
  - Can comprise 20% or more of the total test costs of a complex system development
  - Test Strategies
    - Test All – test the entire system (expensive and time consuming)
    - Prioritized and Rotating – prioritize test cases and run on a rotating basis
    - Heuristic Minimization – run tests based on knowledge of modification, system design and impact to functionality
    - Specification-Based – subset of test cases to verify new functionality
    - Control Flow and Data Flow – selected subset of test cases to verify control and data flows.
    - Risk-Based – subset of test cases selected based on risk assessment of modification impacts

OnStar System Exercise:
Test Methods



OnStar System Exercise:
Test Methods

1. GPS satellites continuously orbit the earth at 12,000 miles and continuously stream navigational data.

2. The OnStar GPS receiver calculates location data from at least three satellites.

3. Upon OnStar button press or automatically in a crash, critical vehicle data is sent via wireless connection.

4. Cellular transmission relays calls to the OnStar Data Center.

5. The OnStar Data Center receives the call and routes the call to the appropriate OnStar Call Center and Advisor Group.

   The OnStar Command Center monitors all call traffic, balances volumes between Call Centers and Advisor Groups, and maintains oversight for overall system integrity.

6. The OnStar Call Centers answer a subscriber call for assistance every two seconds – 24 hours a day, 7 days a week.

7. Trainer OnStar Advisors are set up in special teams – such as Emergency or Routing, to better handle subscriber requests. The Advisor desktop application provides street level mapping information with the vehicle pinpointed and subscriber account information.

8. Emergency Advisor contacts Public Safety Answering Point (911) within jurisdiction. Relays pertinent local, crash data and injury status to prepare emergency responders for appropriate rescue strategy.

OnStar System Exercise:
Test Methods

- For each interface your identified earlier for your OnStar system Test Event, which White Box and/or Black Box test method(s) would be appropriate for each interface?
- Briefly describe how you would apply each test method to each of your interfaces.
- Be prepared to share your results with the class, including:
  - Descriptions the test methods your would use for each interface
  - Descriptions of how you would apply those test methods to 1 – 2 interfaces
  - Your observations for the exercise



Session 5 – System Integration and Test

Lessons

1. Program Overview Update
2. Process Overviews
3. Integration, Planning, Management and Readiness
4. System Test Methods
5. Test Measurement Techniques and Metrics (if time permits)

- Relationships with SE Lifecycle
- V&V Processes
- System Interfaces
- Technical Management and Test Processes

- Integration and Test Planning
- Requirements Management
- Interface Management
- Technical Readiness

- White Box Testing
- Black Box Testing
- Integration Testing



Appendices*

K. Critical Systems Validation

L. Security Testing

*Appendices A through J were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered.

Integration Testing

- Testing of system elements to ensure they interact properly
- Intent is to expose faults in the interfaces and element interactions
- Simple systems may not need dedicated integration testing, but it is essential for complex, safety critical or high reliability systems
- Integration Plan is used as the basis for test planning
  - Captures the integration strategy and resource requirements
  - Build a little … Test a little … Learn a lot



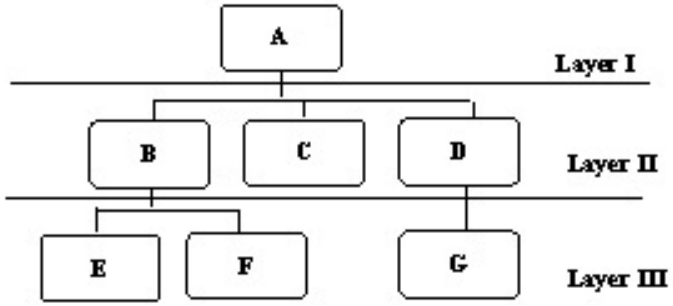
Top-Down Testing

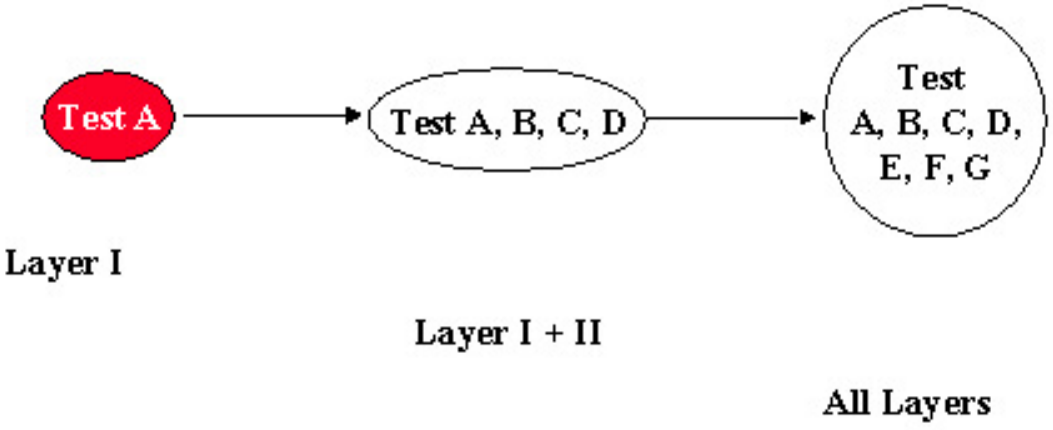
- Incremental approach starting with the top component or subsystem in the hierarchy

- **Advantages include:**
  - Incremental approach simplifies fault localization
  - Higher level components provide system drivers and control
  - Can get an early prototype depicting or simulating general system behavior
  - Critical modules are tested on priority, major design flaws could be found and fixed first

- **Disadvantages include:**
  - Numerous temporary test stubs or mock objects (mimic subsystems)
  - Test stubs may not faithfully represent the actual component
  - Low level components are tested late posing a potential development risk

Top-Down Testing

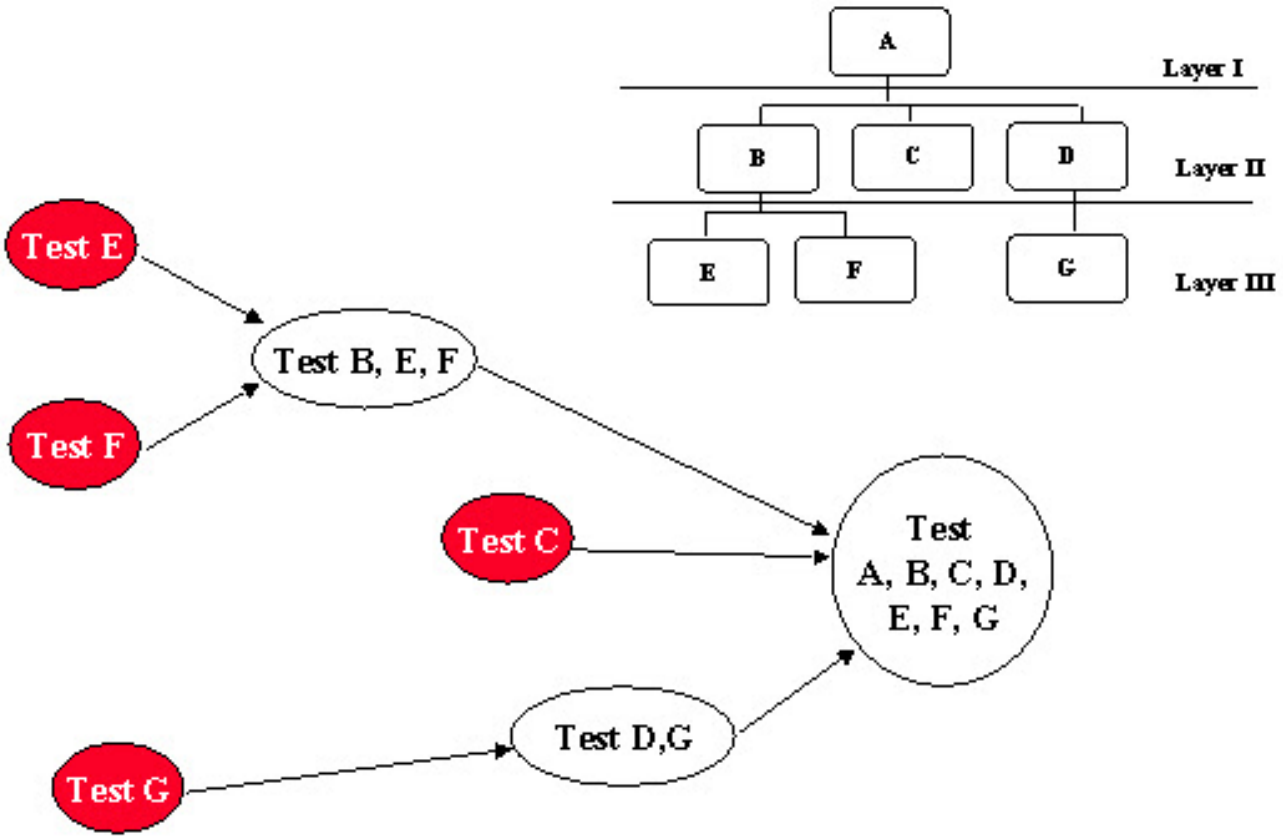




Bottom-Up Testing

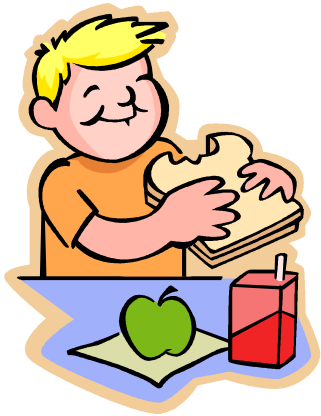
- Integrated moving up through the system hierarchy starting with the lowest components
- Test stubs must be developed to emulate higher level functions
- **Advantages include:**
  - Incremental approach simplifies fault isolation
  - Can be started early
  - Testing upper level subsystems is easier because no lower level stubs have to be developed
- **Disadvantages include:**
  - Numerous temporary test drivers
  - Early prototypes not possible
  - Lack of a skeletal system until late in the development
  - Critical modules which control the flow of application are tested last

Bottom-Up Testing

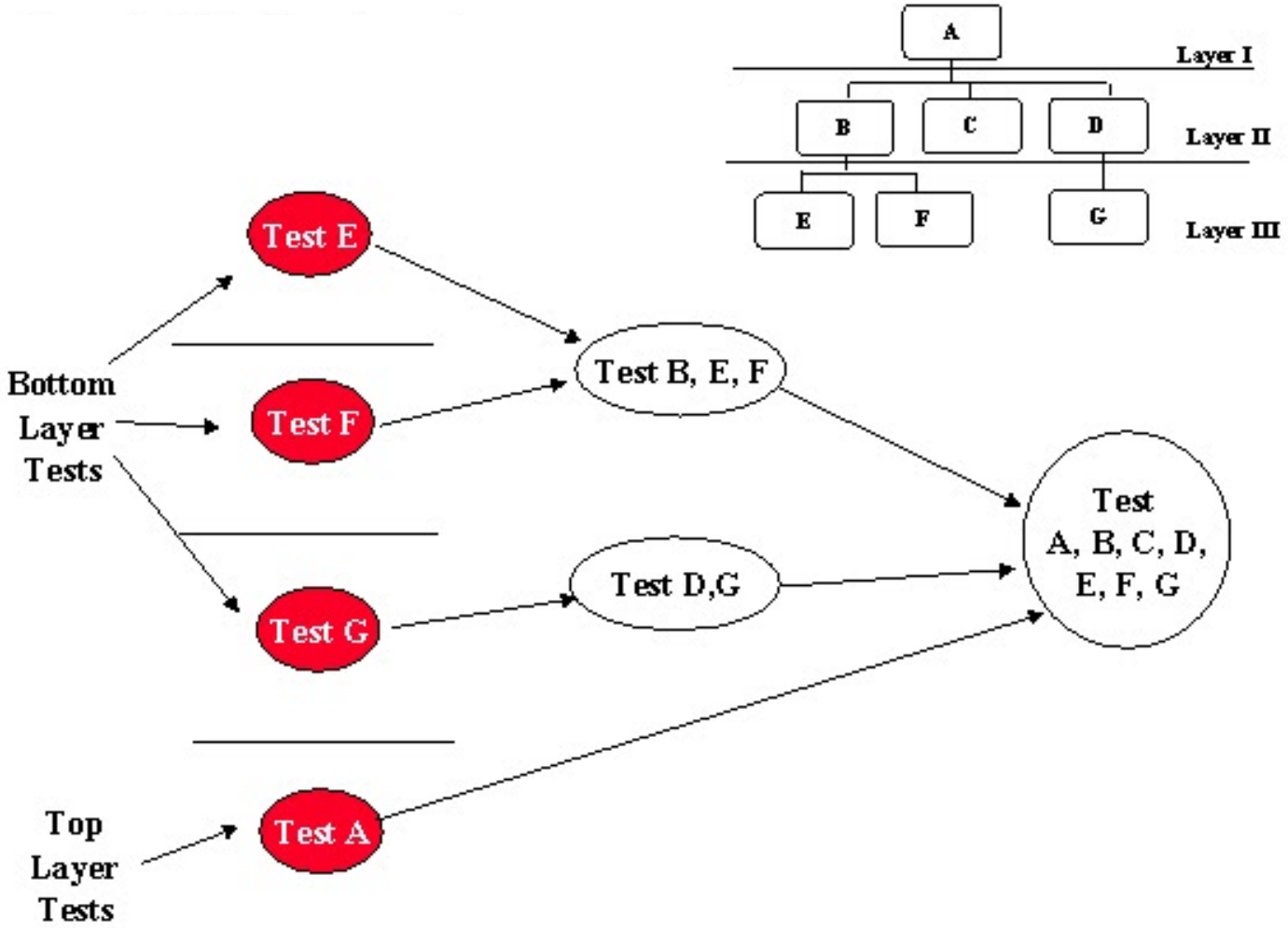


Sandwich Testing

- Combination of Top-Down and Bottom-Up testing
- Usually split between two teams
- **Advantages include:**
  - Incremental approach simplifies fault isolation
  - Parallel approach can shorten integration time and test period
- **Disadvantages include:**
  - Needs more manpower and resources
  - Requires more collaboration and communications



Sandwich Testing



Big-Bang Testing

- Testing all system parts at once --- not incremental

- **Advantages include:**
  - Convenient for the small systems

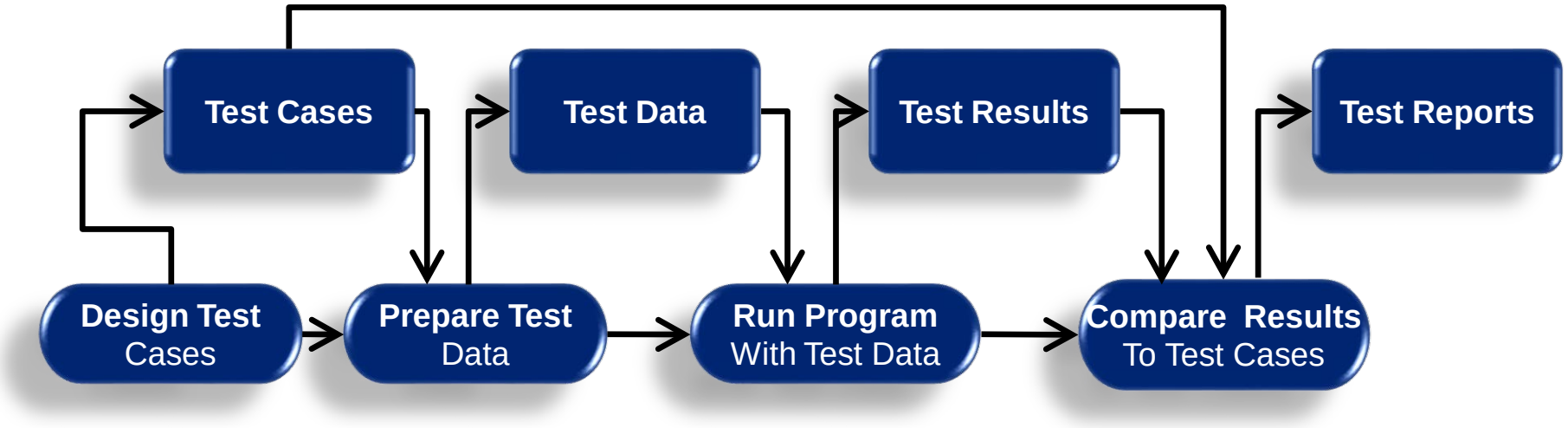
- **Disadvantages include:**
  - Fault localization is difficult
  - Some interface links to be tested could be missed
  - Since integration testing is done after all the modules are integrated, test team could have less time to test … assuming that everything works together the first time!
  - Since all modules tested at once, high risk modules are not isolated and tested based on priority
  - All system components and subsystems must be completed before testing can begin

The Testing Process

- **Component testing**
  - Testing of individual program components
  - Usually the responsibility of the component developer (except sometimes for critical systems)
  - Tests are derived from the developer's experience

- **System testing**
  - Testing of groups of components integrated to create a system or sub-system
  - The responsibility of an independent testing team
  - Tests are based on the requirements and system specifications

A Software Testing Process Model



Testing Policies

- Only exhaustive testing can show a program is 'free' from defects
- However, exhaustive testing is impossible
- Testing policies define the approach to be used in selecting system tests:
  - All functions accessed through menus should be tested
  - Combinations of functions accessed through the same menu should be tested
  - Where user input is required, all functions must be tested with correct and incorrect inputs

Testing Guidelines

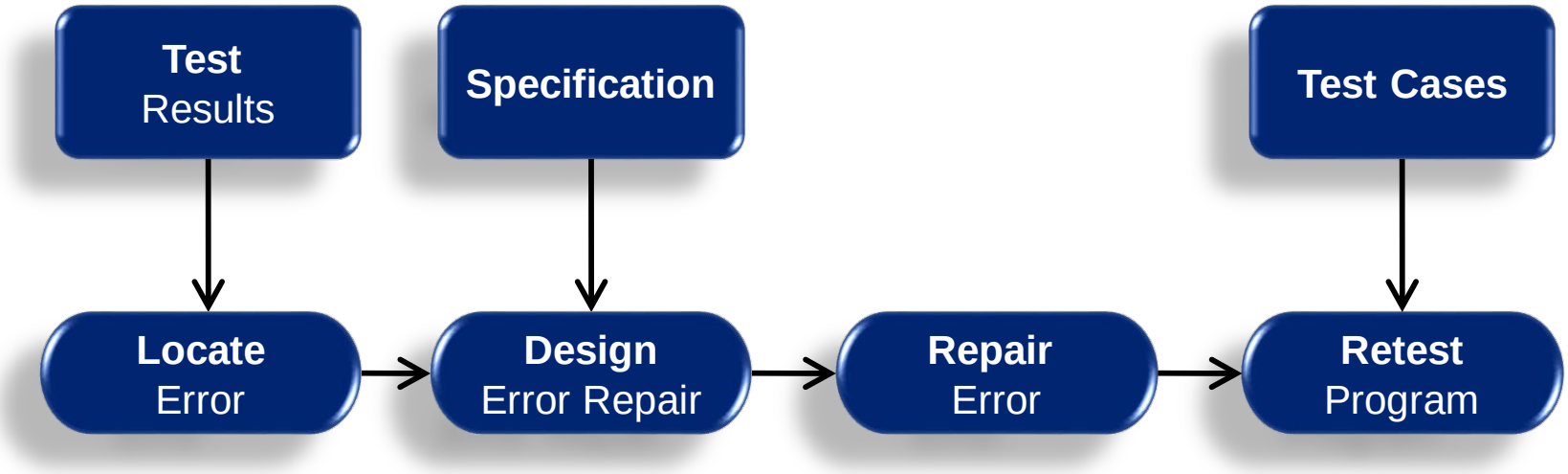
- Testing guidelines are hints for the testing team to help them choose tests that will reveal defects in the system:
  - Choose inputs that force the system to generate all error messages;
  - Design inputs that cause buffers to overflow;
  - Repeat the same input or input series several times;
  - Force invalid outputs to be generated;
  - Force computation results to be too large or too small.

Testing and Debugging

- Defect testing and debugging are distinct processes
- Verification and validation is concerned with establishing the existence of defects in a program
- Debugging is concerned with locating and repairing these errors
- Debugging involves formulating a hypothesis about program behaviour then testing these hypotheses to find the system error



The Debugging Process



Stages of Testing

- **Development testing,** where the system is tested during development to discover bugs and defects.
- **Release testing,** where a separate testing team tests a complete version of the system before it is released to users.
- **User testing,** where users or potential users of a system test the system in their own environment.

Release Testing

- The process of testing a release of a system that will be distributed to customers
- Primary goal is to increase the supplier's confidence that the system meets its requirements
- Release testing is usually black-box or functional testing
  - Based on the system specification only
  - Testers do **NOT** have knowledge of the system implementation

Release Testing and System Testing

- Release testing is a form of system testing.

- Important differences:
  - A separate team that has not been involved in the system development, should be responsible for release testing.
  - System testing by the development team should focus on discovering bugs in the system (defect testing).
  - The objective of release testing is to check that the system meets its requirements and is good enough for external use (validation testing).

User Testing

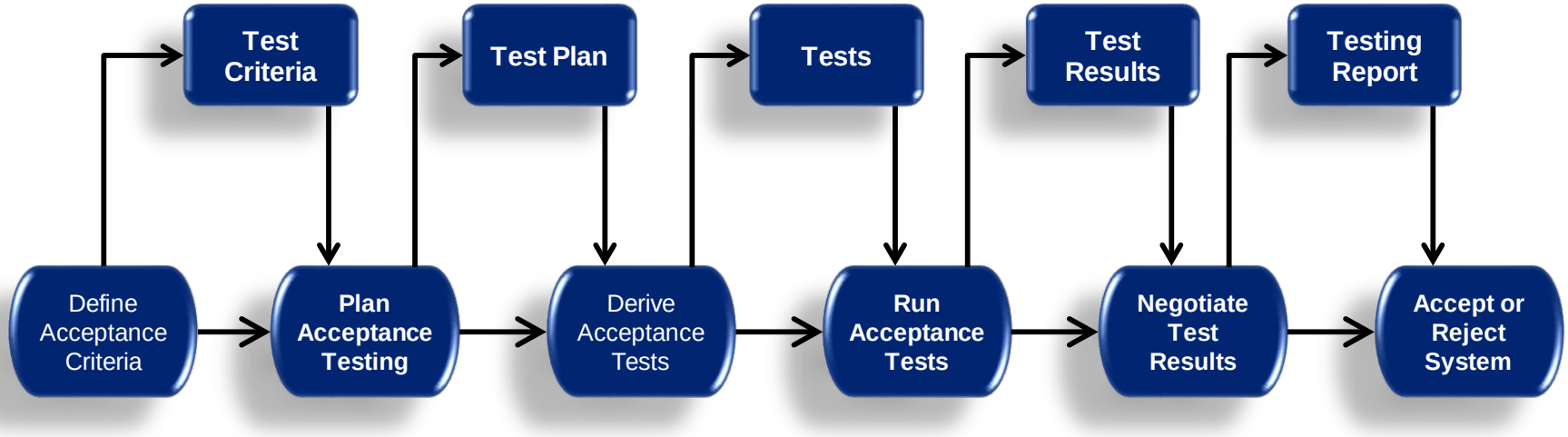
- User or customer testing is a stage in the testing process in which users or customers provide input and advice on system testing.

- User testing is essential, even when comprehensive system and release testing have been carried out.
  - The reason for this is that influences from the user's working environment have a major effect on the reliability, performance, usability and robustness of a system. These cannot be replicated in a testing environment.

Types of User Testing

- **Alpha testing**
  - Users of the software work with the development team to test the software at the developer's site.
- **Beta testing**
  - A release of the software is made available to users to allow them to experiment and to raise problems that they discover with the system developers.
- **Acceptance testing**
  - Customers test a system to decide whether or not it is ready to be accepted from the system developers and deployed in the customer environment. Primarily for custom systems.

The Acceptance Testing Process



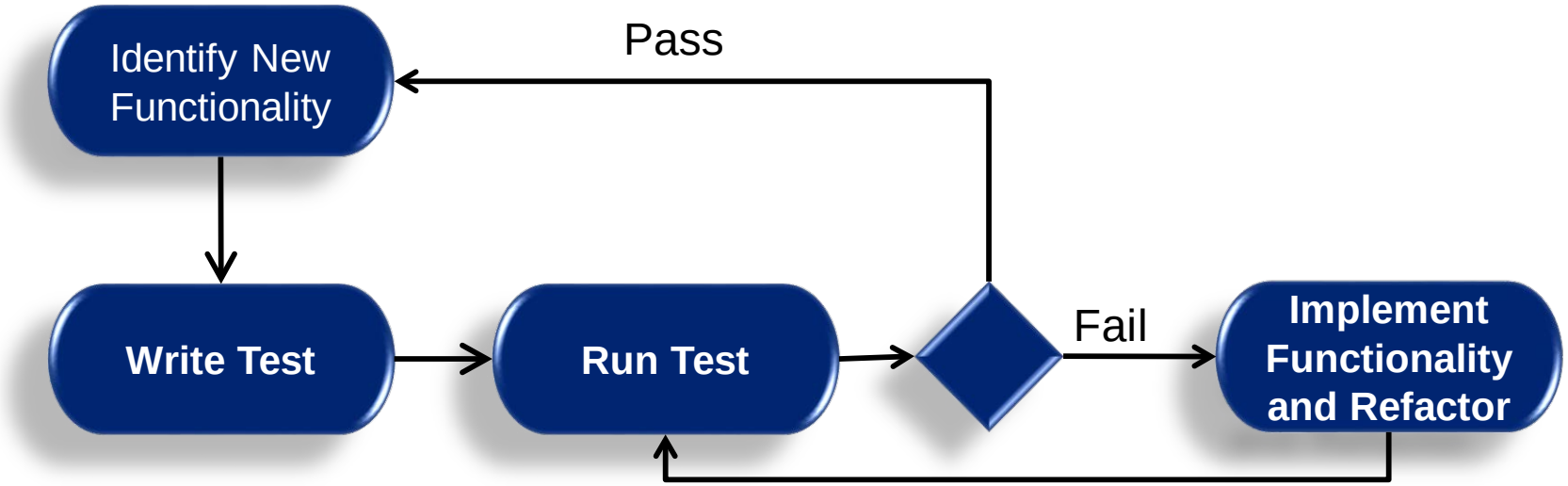
Agile Methods and Acceptance Testing

- In agile methods, the user/customer is part of the development team and is responsible for making decisions on the acceptability of the system
- Tests are defined by the user/customer and are integrated with other tests in that they are run automatically when changes are made – regression testing
- There is no separate acceptance testing process
- Main problem here is whether or not the embedded user is 'typical' and can represent the interests of all system stakeholders

Test-Driven Development

- Test-driven development (TDD) is an approach to program development in which you inter-leave testing and code development.
- Tests are written before code and ‘passing’ the tests is the critical driver of development.
- You develop code incrementally, along with a test for that increment. You don’t move on to the next increment until the code that you have developed passes its test.
- TDD was introduced as part of agile methods such as Extreme Programming. However, it can also be used in plan-driven development processes.

Test-Driven Development



Benefits of Test-Driven Development

- **Code coverage**
  - Every code segment that you write has at least one associated test so all code written has at least one test.
- **Regression testing**
  - A regression test suite is developed incrementally as a program is developed.
- **Simplified debugging**
  - When a test fails, it should be obvious where the problem lies. The newly written code needs to be checked and modified.
- **System documentation**
  - The tests themselves are a form of documentation that describe what the code should be doing.

Other Special Testing

- **Acceptance Testing** – Does the system meet the needs of the user?
  - Defined by the customer
  - Often defined in the contract
- **Certification & Accreditation Testing**
  - Attesting that a system meets assurance and qualification requirements
  - FAA, FCC, NIST, FDA
  - May have to use an accredited laboratory for testing (e.g. Underwriters)
- **First Article Inspection** – Do the early products produced meet the specifications?
  - Process consists of complete, independent and documented physical and functional inspection and testing
- **Production Testing** – Initial and ongoing verification that production products meet specifications
  - Tests must be well defined with clear pass/fail criteria
  - Often done by non-technical persons

Other Special Testing (continued)

- **Installation Testing** – testing the system once installed in its intended environment
  - Could involve construction of new facilities, modification of existing facilities or installation in existing facilities
  - May require integration during installation
  - Testing verifies system is integrated and operates in its new environment
- **Maintenance Testing** – confirms that repairs did not degrade the baseline performance of the system
  - Verify performance or functionality of upgrades as a result of modifications
  - Testing will depend on following category:
    - Preventative Maintenance
    - Corrective Maintenance – restoring a failed system to operational status
    - Adaptive Maintenance – adapt to changes due to changes in physical, regulatory, or operational environment
    - Perfective Maintenance – user defined enhancement

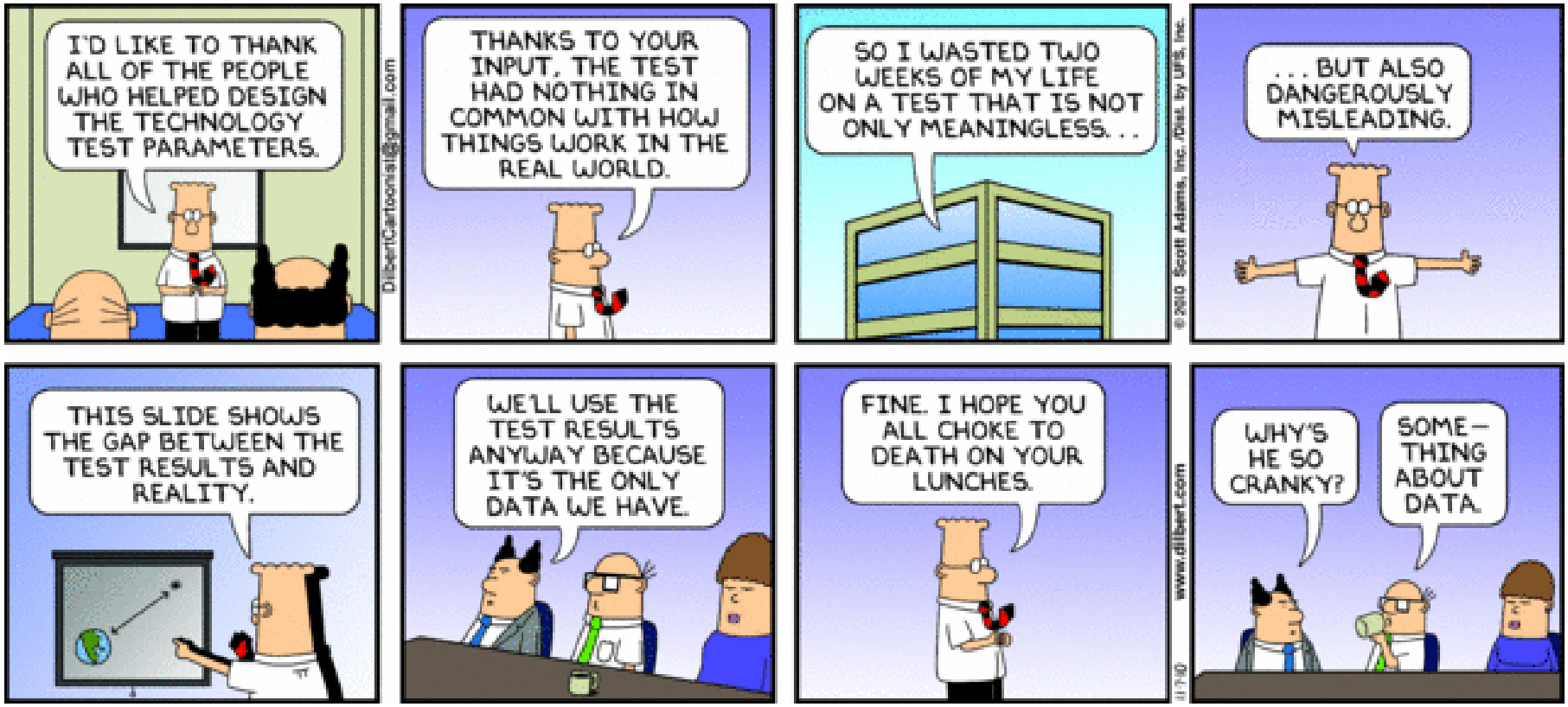
Summary

- White Box and Black Box testing form the foundation of test activities
- Planning is critical to a successful test program – test cases, budget, schedule need to be planned
- Humans need to be integrated into the system design and their requirements verified
- Integration strategy and verification methods depends on system complexity, cost and schedule
- **Bottom line** … there are a lot of tools in the testing tool box … which ones are best for your system depends on many factors which are determine very early in the system lifecycle

Summary

- Testing can show the presence of faults in a system; it cannot prove there are no remaining faults
- Component developers are responsible for component testing; system testing is the responsibility of a separate team
- Use experience and guidelines to design test cases in defect testing
- Interface testing is designed to discover defects in the interfaces of composite components

In closing…


I'D LIKE TO THANK ALL OF THE PEOPLE WHO HELPED DESIGN THE TECHNOLOGY TEST PARAMETERS.
THANKS TO YOUR INPUT, THE TEST HAD NOTHING IN COMMON WITH HOW THINGS WORK IN THE REAL WORLD.
SO I WASTED TWO WEEKS OF MY LIFE ON A TEST THAT IS NOT ONLY MEANINGLESS. . .
. . . BUT ALSO DANGEROUSLY MISLEADING.
THIS SLIDE SHOWS THE GAP BETWEEN THE TEST RESULTS AND REALITY.
WE'LL USE THE TEST RESULTS ANYWAY BECAUSE IT'S THE ONLY DATA WE HAVE.
FINE. I HOPE YOU ALL CHOKE TO DEATH ON YOUR LUNCHES.
WHY'S HE SO CRANKY?
SOME-THING ABOUT DATA.
DilbertCartoonist@gmail.com
© 2010 Scott Adams, Inc. /Dist. by UFS, Inc.
www.dilbert.com

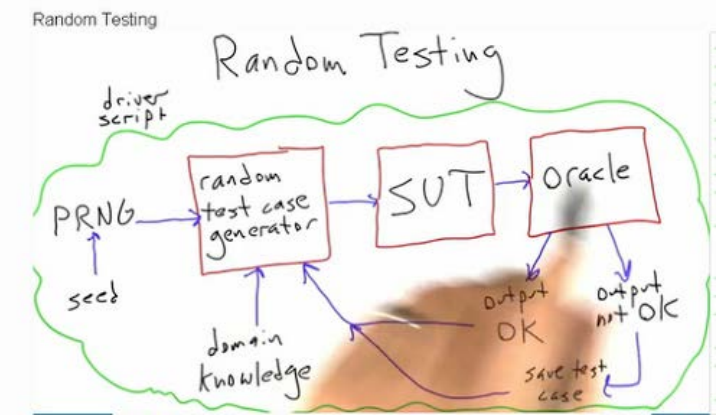
Backup Slides:
Black Box - Special Testing

Enhanced Random Testing

- Parameterized Random Test Generation – data is parameterized to control the range and characteristics of random values
- Directed Automated Random Testing – uses automatically generated software test sequences based on automatic analysis of source code
- Specification Based Random Testing – combines random testing with formal specification of properties-embedded systems
- State Model Random Testing – employs finite state machine methodology for constructing test cases
- Random Testing of Interrupt - Embedded Systems – automates testing of interrupt-driven embedded systems
- Regression Random Testing – enhances regression testing. Random sequential testing can reveal failures even if all tests were passed individually
- Random Testing of Integrated Circuits

Automatic Random Testing

- Injects large quantities of random inputs into a system to test behavior
- Objective is to evaluate performance under unexpected conditions over time
- Automated process
- Does not demonstrate system meets specification. Only detects failures based on crashes, error conditions or improper inter-system actions
- Does not do a good job on boundary condition failures
- Has other limitations – must evaluate to determine if this is the method to use on your system



Usability Testing

- Finds out practical information on how users actually use a system
- Ensures the design will meet the needs of the target user and meet the business needs of the company
- Involves observing users performing real tasks, recording them and analyzing the results
  - Evaluates functional suitability
  - How easy Is it to learn and operate the system
  - Determines if the users can maintain knowledge of the system over time
- Evaluate users
  - Emotional response - satisfaction
  - Time on Task - learnability
  - Accuracy
  - Recall - memorability

Search Based Testing

- Search-Based Software Testing is the use of a meta-heuristic optimizing search technique, such as a Genetic Algorithm, to automate or partially automate a testing task; for example the automatic generation of test data

- Key to the optimization process is a problem-specific fitness function
  - The role of the fitness function is to guide the search to good solutions from a potentially infinite search space, within a practical time limit

- Has been applied to many other software testing applications, including structural testing, model-based testing, mutation testing, temporal testing, exception testing, regression testing, integration testing, configuration, and interaction testing

- Can, in some circumstances, increase effectiveness and efficiency of the testing process

Mutation Testing

- Mutation testing is a method of testing in which hardware or software is deliberately manipulated, followed by a suite of testing against the mutated system
- The mutations introduced (seeds) are designed to imitate common engineering errors
- Allows us to determine the adequacy of test cases
- Mutation testing facilitates the following advantages:
  - Program code fault identification
  - Effective test case development
  - Detection of loopholes in test data
  - Improved software program quality
  - Elimination of code ambiguity
- Disadvantages of mutation testing include:
  - Difficult implementation of complex mutations
  - Expensive and time-consuming
  - Requires skilled testers with programming knowledge

Estimating Number of Program Errors by Error Seeding

- One member of the test team places a known number of errors in the program while other members try to find them.

- Assumption: (s/S) = (n/N)
  - s = # seeded errors found during testing
  - S = # of seeded errors placed in program
  - n = # non-seeded (actual) errors found during testing
  - N = total # of non-seeded (actual) errors in program

- This can be written as N = (S*n)/s

Error Seeding Example

- Using the error seeding assumptions
  - If 75 of 100 seeded errors are found
  - We believe the we have found 75% of the actual errors

- If we found 25 non-seeded errors, that means the actual # errors in the program is

N = (100 * 25)/75 = 33 1/3

Confidence in Software

- C = 1
  - If n >N
  - Meaning the actual # errors found during testing exceeds the # actual errors in the program

- C = S / (S - N -1)
  - If n <= N
  - Meaning the actual # errors fond during testing is less than the actual # program errors

$$C \cong \begin{cases} 1 & \text{if n} > N \\ \frac{S}{S-X+1} & \text{if n} \leq N \end{cases}$$

Confidence Example

- To achieve a 98% confidence level that our program is bug free (N = 0), how many seeded errors would need to be introduced and found by our test team?

$$S/(S - N + 1) = 98/100$$

$$S/(S - 0 + 1) = 98/100$$

$$S/(S + 1) = 98/100$$

$$100S = (98S + 98)$$

$$2S = 98$$

$$S = 49$$

Better Approach

- A more realistic approach to estimating confidence would be based on the number of seeded errors found, regardless of whether they have all been found or not.

- If $n \leq N$ then

  $C = C(S, s-1)/C(S + N + 1, N+s)$

- Where

  $C(S, s - 1) = S!/(s - 1)!*(S - s + 1)!$

  $C(S+N+1, N+s) = (S+N+1)!/(N+s)!*(S+1-s)!$

Destructive Testing

- Some testing destroys the test article
- A lot can be learned by stressing a system or component to failure
- Can reveal previously unknown faults
- Computer models help predict behavior
  - Models should be refined with the results of actual testing
- Can be expensive (depending on SUT costs) and time consuming
- Must budget and plan accordingly for costs, test articles and test sequencing



Reactive Testing

- Not as fully defined during planning – affected by the behavior of the SUT
- The test facility can react and evolve depending on the behavior of the SUT
- Used for systems that are complex, exhibit non-linear or erratic behavior
  - This drives a large number of input data combinations
- Advantages are:
  - Automation
  - Reusability
  - Effectiveness
  - Robustness
  - Efficiency
- Disadvantages include:
  - Coverage Gaps
  - Repeatability Limitation
  - Test Oracle Problems

Temporal Testing

- Targeted primarily to real-time systems due to the criticality of timing in most embedded systems

- Estimating temporal behavior is difficult due to induced errors (unknowns, computing execution times, data transfer rates, etc.)

- Methods of test may include:
  - Constrained random-based temporal testing – Automated test cases with randomized inputs
  - Stress-based temporal testing – maximum use of resources
  - Search-based temporal testing – evolutionary testing using genetic algorithms
  - Mutation-based temporal testing – uses real-time application models. Mutation operators change a system property to mimic a fault and temporal performance is evaluated



Systems Engineering Technical Short Course

(Session 5: System Integration and Test,

AKA: Verification, Validation and Test - VVT)

Lesson 5:
Test Measurement Techniques and Metrics

Session 5 – System Integration and Test

Lessons

1. **Program Overview Update**
2. **Process Overviews**
3. **Integration, Planning, Management and Readiness**
4. **System Test Methods**
5. **Test Measurement Techniques and Metrics (if time permits)**

Appendices*

K. **Critical Systems Validation**

L. **Security Testing**

- Relationships with SE Lifecycle
- V&V Processes
- System Interfaces
- Technical Management and Test Processes

- Integration and Test Planning
- Requirements Management
- Interface Management
- Technical Readiness

- White Box Testing
- Black Box Testing
- Integration Testing

- TPM Metrics
- Quality Metrics
- Planning and Control Metrics

*Appendices A through J were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered.

Metric vs. Measures

- **Metric: a composite of meaningful, quantifiable, product or process attributes (measures) taken over time that communicate important information about quality, processes, technology, products and/or resources.**

- **Measure: provides a quantitative indication of the extent, amount, dimensions, capacity, or size of some attribute of a product or process.**

- **Indicator: a metric or combination of metrics that provide insight into a process, a project, or the product itself.**

- **Must only collect measures you can use**

Metrics Taxonomy

- **Three types of metrics exist**
  - **Technical Performance Measures (TPMs)**
  - **Planning and Control Metrics**
  - **Engineering Process and Quality Metrics**

What is a Good Metric?

- **Basic characteristics of a good metric:**
  - **It is accepted as having value to the customer or as an attribute essential to customer satisfaction with the product.**
  - **It tells how well organizational goals and objectives are being met through processes and tasks.**
  - **It is simple, understandable, logical and repeatable.**
  - **It shows a trend, more than a snapshot or a one-time status point.**
  - **It is unambiguously defined.**
  - **Its data is economical to collect.**
  - **The collection, analysis, and reporting of the information is timely, permitting rapid response to problems.**
  - **The metric provides product and/or process insight and drives the appropriate action(s).**
- **For a metric to be effective it must (1) present data that is useful, (2) be able to show status over a period of time, (3) support corporate and product goals and objectives, and (4) be meaningful to the customer.**

Metrics are Not

- Metrics are not charts or any other form of display tool, although charts and graphics may display the results of a metric.
- Metrics are not a team or personnel control tool. Metrics are a process and product control tool.
- Metrics are not one-time snapshots or status measures.
- Metrics are not forever. Different phases of the product lifecycle require different metrics.
- Metrics are not schedules, though some schedules lead to good metrics.
- Metrics are not "counts of activity," although counts of activity or statistics may be significant

Metrics Benefits

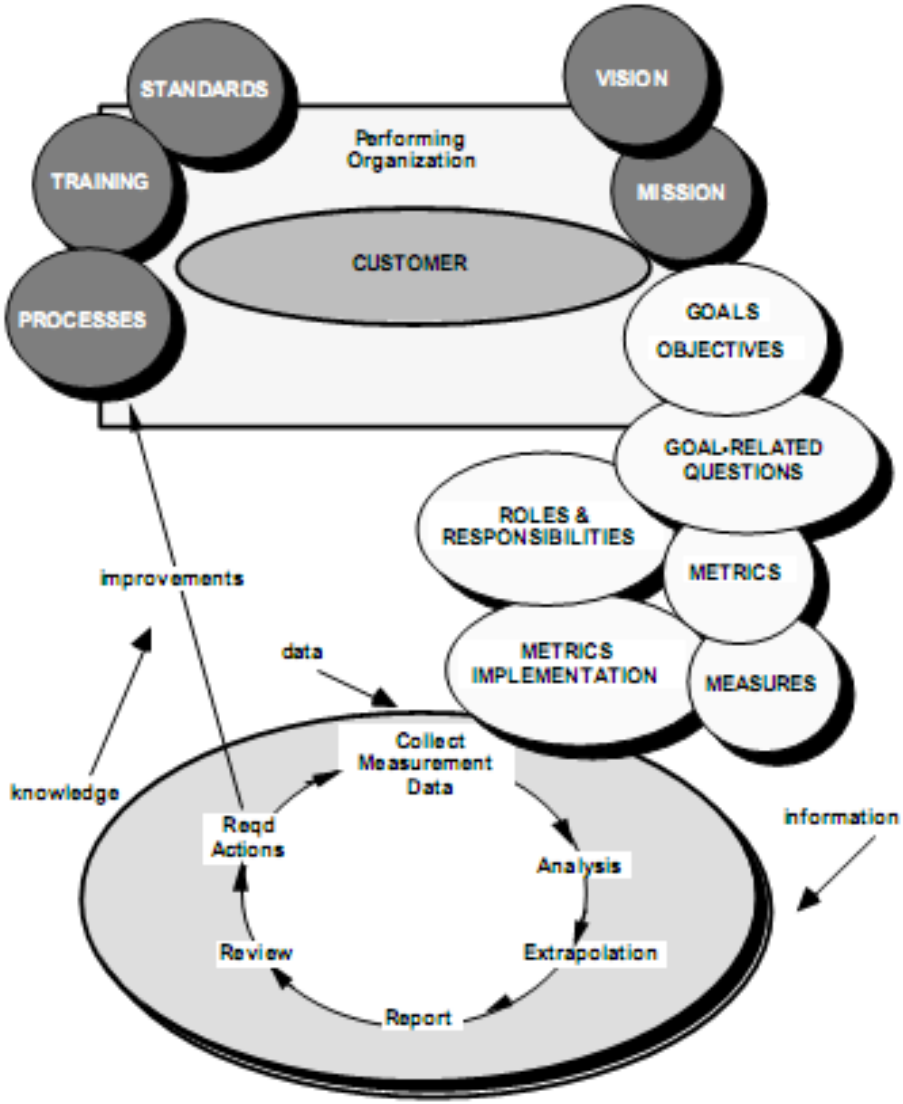
- Ensure that detailed measures of engineering processes, performance and product quality are defined, collected, and analyzed to provide quantitative understanding and control in support of improving performance, products and processes.
- With quantitative measurements, problems and results become more apparent to management, and required actions are clearer to engineers.
- Metrics help to identify risks early
- Provides customer insight and raises confidence in product development
- Facilitates communications, process discipline and quality assessment

Metrics Development & Application

- **Selecting existing metrics or developing new metrics should be directly related to the product's or organization's critical success factors, usually defined as technical, cost, or schedule targets or requirements.**

- **Metrics must be relevant to the vision and mission of the organization and yield insight into customer satisfaction.**

Metrics Development Relationships



Selection of Metrics

- **Goal/ Question/Metric (GQM) Approach**

- **The GQM paradigm:**
  - **State the information goal: Identify the information consumer groups (stakeholders) and determine what each wants to know and wants to do with the information.**
  - **Ask the question: What question(s) should be asked to determine whether the goal is being met?**
  - **Identify the specific parameters that must be measured: What metric is needed and what must be measured to obtain it?**
  - **Apply the metrics selected, evaluate their usefulness.**

- **Selecting metrics is typically an iterative process.**

- **Each cycle ends with an evaluation of the measurement system that has been put into place.**

System Design Questions & Metrics Example

1. What has been processed by this activity?	Metric Category: Number of system requirements statements evaluated (size)
2. What has been produced by this activity?	Metric Category: Number of software and hardware requirements statements developed (size)
3. How much effort has been expended on this activity?	Metric Category: Labor Hours (effort)
4. How much time has been spent on this activity?	Metric Category: Clock Hours (schedule)
5. How many times has this activity been repeated?	Metric Category: Number of Repetitions (schedule)
6. How much rework has been done?	Metric Category: Rework Labor Hours (effort)
7. How much more has to be done?	Metric Category: Number of system requirements statements to be evaluated (size), Labor Hours to be expended (effort)
8. When will the activity be completed?	Metric Category: Clock Hours (schedule)

Session 5 – System Integration and Test

Lessons

1. **Program Overview Update**
2. **Process Overviews**
3. **Integration, Planning, Management and Readiness**
4. **System Test Methods**
5. **Test Measurement Techniques and Metrics (if time permits)**

Appendices*

K. **Critical Systems Validation**

L. **Security Testing**

- Relationships with SE Lifecycle
- V&V Processes
- System Interfaces
- Technical Management and Test Processes

- Integration and Test Planning
- Requirements Management
- Interface Management
- Technical Readiness

- White Box Testing
- Black Box Testing
- Integration Testing

- TPM Metrics
- Quality Metrics
- Planning and Control Metrics

*Appendices A through J were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered.

TPM Metrics

- How well does the evolving product satisfy technical performance requirements?

- TPMs may measure
  - Physical size
  - Weight
  - Power consumption
  - Heat dissipation
  - Reliability
  - Throughput
  - Human factors
  - Response time
  - Availability
  - Speed

- Typically track actual and predicted quantity against maximum allowable with growth margins

TPM Characteristics

- Should be important and relevant
- Should be relatively easy to measure
- Performance should be expected to improve with time
- If the measure crosses its threshold, corrective action should be known
- The measured parameter should be controllable
- Management should be able to tradeoff cost, schedule and performance
- Should be documented
- Should be tailored for the project

Technical Performance Measures (TPMs)

- System or product specific, quantifiable measures that express critical system or product performance parameters relative to performance requirements.
- Used to track those parameters that are critical to performance and at risk.
- Used to understand allocation and implications of design on system performance budgets, to manage product and technical risk.
- Used in development and field support phases to demonstrate (verify) system or product performance.
- Depicted using control charts or stacked column or line charts with target and hard requirement values.
- Collection can be event driven, monthly, weekly, or even daily when following critical activities.

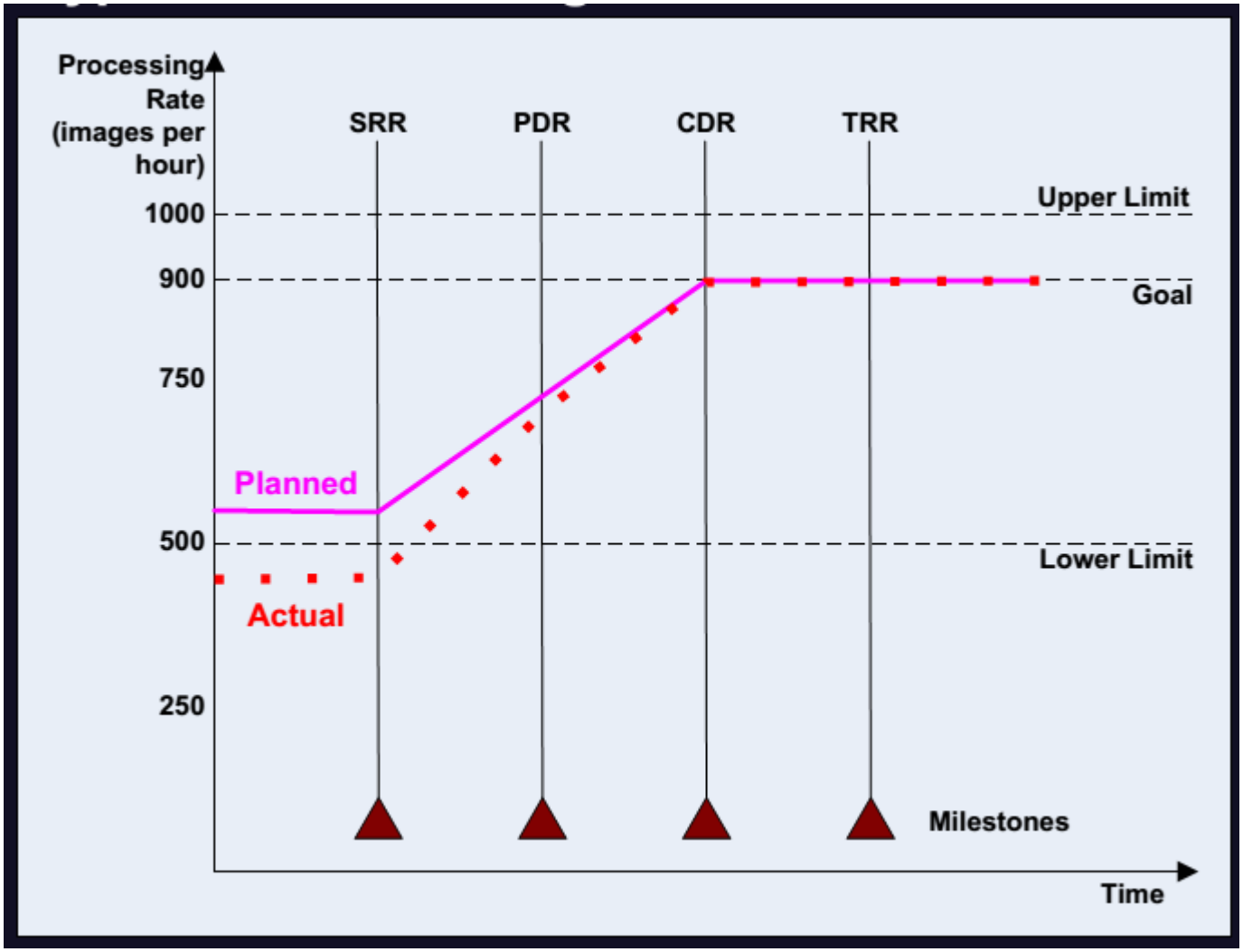
TPMs Continued

- **TPMs are used to**
  - **Forecast the values to be achieved through the planned technical program effort,**
  - **Measure differences between the achieved values and those allocated to the product element by the systems engineering process,**
  - **Determine the impact of these differences on system effectiveness.**

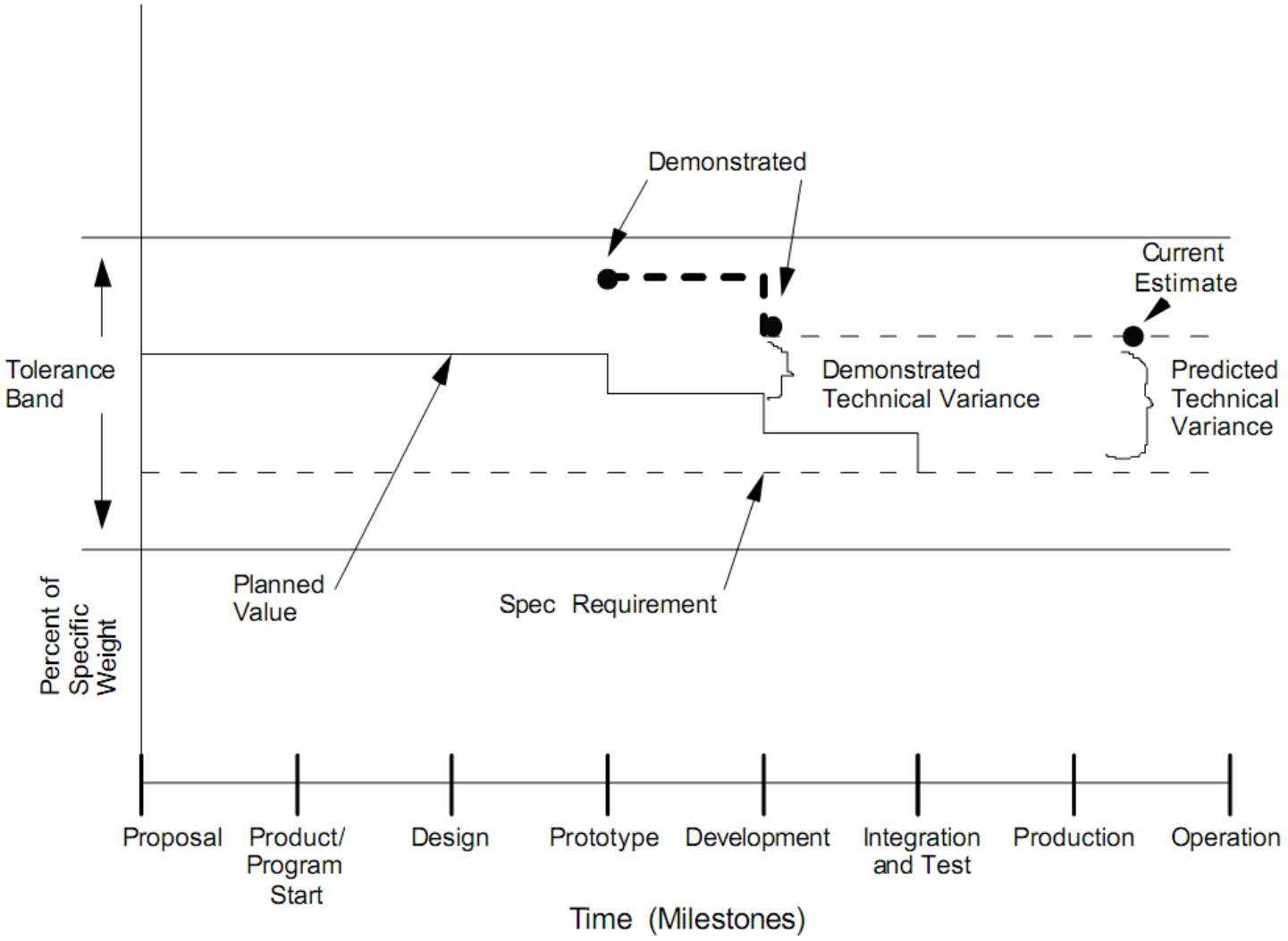
- **Designed to:**
  - **Provide visibility of actual versus planned performance.**
  - **Provide early detection or prediction of problems that require management attention.**
  - **Support assessment of the program impact of proposed change alternatives.**

- **Alert program management to potential performance deficiencies before irrevocable cost or schedule impact occurs.**

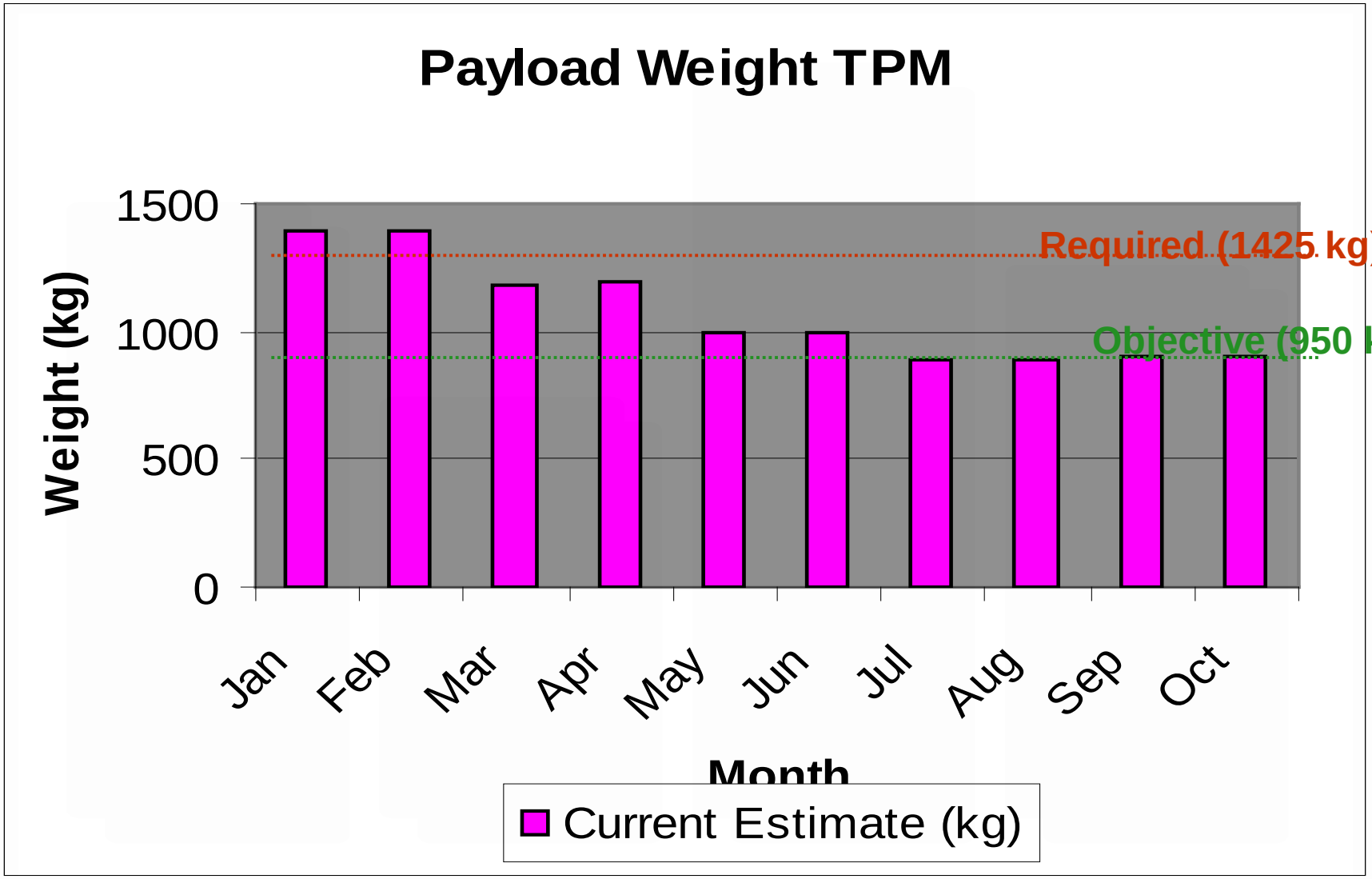
Typical TPM Tracking Chart



TPM Profile


Demonstrated
Current Estimate
Tolerance Band
Demonstrated Technical Variance
Predicted Technical Variance
Planned Value
Spec Requirement
Percent of Specific Weight
Proposal
Product/ Program Start
Design
Prototype
Development
Integration and Test
Production
Operation
Time (Milestones)

TPM – Payload Weight



Class Exercise

TPMs

- Consider the following requirement, where the user basically wants as high a value as possible for a TPM
  - Ignore the costs
- The requirement from the user was for ‘at least 120 per second under all circumstances’
- The user will not pay for more than 120 per second, but will be happy if the value is higher
- The design (with margin) was for 160 per second
- The test at the factory on the first pre-production model revealed 140 per second.
- One thousand of these systems will be produced over the next year

Class Exercise Continued

TPMs

- **What, if anything, needs to be done?**
- **Why was the design requirement higher than the user requirement?**
- **What key value is missing from this question?**
- **Be prepared to share your answers with the class!**

Session 5 – System Integration and Test

Lessons

1. **Program Overview Update**
2. **Process Overviews**
3. **Integration, Planning, Management and Readiness**
4. **System Test Methods**
5. **Test Measurement Techniques and Metrics (if time permits)**

Appendices*

K. **Critical Systems Validation**

L. **Security Testing**

- Relationships with SE Lifecycle
- V&V Processes
- System Interfaces
- Technical Management and Test Processes

- Integration and Test Planning
- Requirements Management
- Interface Management
- Technical Readiness

- White Box Testing
- Black Box Testing
- Integration Testing

- TPM Metrics
- Quality Metrics
- Planning and Control Metrics

*Appendices A through J were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered.

Metrics Taxonomy

- **Three types of metrics exist**
  - Technical Performance Measures (TPMs)
  - **Quality Metrics**
    - **Defect tracking and removal**
    - **Quality, such as the severity of defects**
    - **Requirements stability**
  - Planning and Control Metrics

Defect Removal

- Requires detailed metrics from previous projects and a means to normalize them and use them to predict the total defects
- Requires a process by which defects are identified, tracked and recorded
  - Tied to Risk and Configuration Management processes
- Compare the actual defects identified and corrected to a profile for defect discovery and removal based on a projected total
- Plot the data using a line graph to graphically depict completion status

Classification of Defects by Priority Level

- **Priority 1**
  - **Prevents the accomplishment of an operational or mission essential capability specified by baselined requirements.**
  - **Prevents the operator's accomplishment of an operational or mission essential capability.**
  - **Jeopardizes personnel safety.**

- **Priority 2**
  - **Adversely affects the accomplishment of an operational or mission essential capability specified by baselined requirements, so as to degrade performance and for which no alternative work-around solution is known**
  - **Adversely affects the operator's accomplishment of an operational or mission essential capability specified by baselined requirements, so as to degrade performance and for which no alternative work-around solution is known.**

Classification of Defects by Priority Level (cont'd)

- **Priority 3**
  - **Adversely affects the accomplishment of an operational or mission essential capability specified by baselined requirements, so as to degrade performance and for which an alternative work-around solution is known**
  - **Adversely affects the operator's accomplishment of an operational or mission essential capability specified by baselined requirements so as to degrade performance and for which an alternative work-around solution is known.**

- **Priority 4**
  - **An operator inconvenience or annoyance and which does not effect a required operational or mission essential capability.**

- **Priority 5**
  - **All other errors.**

How Successful Are You at Identifying and Fixing Defects?

- **Candidate metrics to be tracked**
  - **Cumulative defects removed compared to predicted total defects as per some profile (linear or statistical)**
  - **Comparing number of defects found versus number resolved**
  - **Monitoring how long identified defects remain unresolved**

Cumulative Defects Removed Compared to Predicted Total Defects

- Objective is to count and track defects identified and corrected and compare to some expected standard to determine whether you are approaching completion

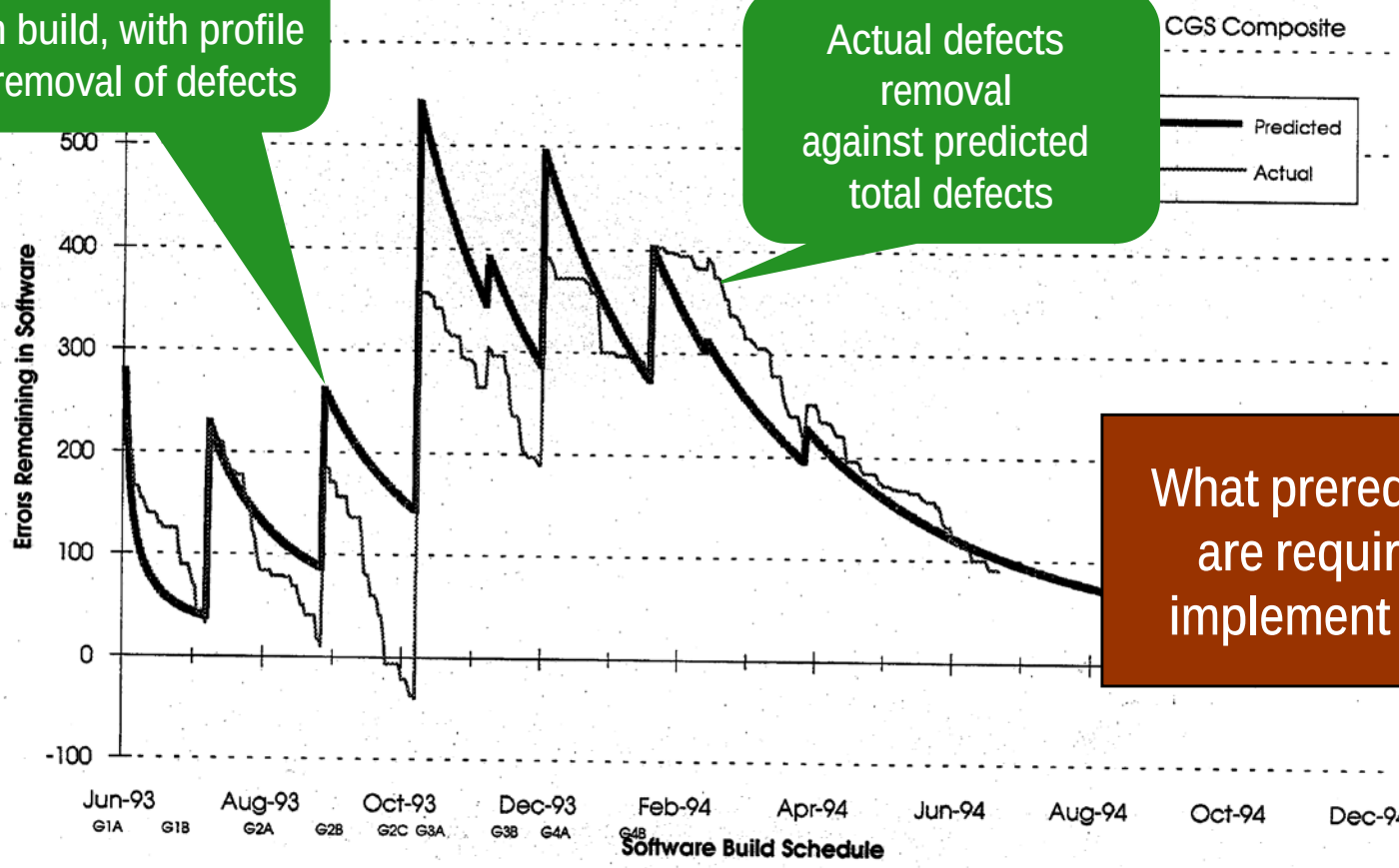
- Two methods
  - Model the defect removal process and track against the model
    - Requires normalize estimate of total defects per some unit of complexity of the project and allowable residual defects at time of delivery
  - Estimate total defects and use a linear profile for removal or simply count defects to ensure you are approaching expected total
    - Requires normalize estimate of total defects per some unit of complexity

- Both methods require extensive metrics and analysis from previous projects

Defect Removal Tracking Using A Statistical Model

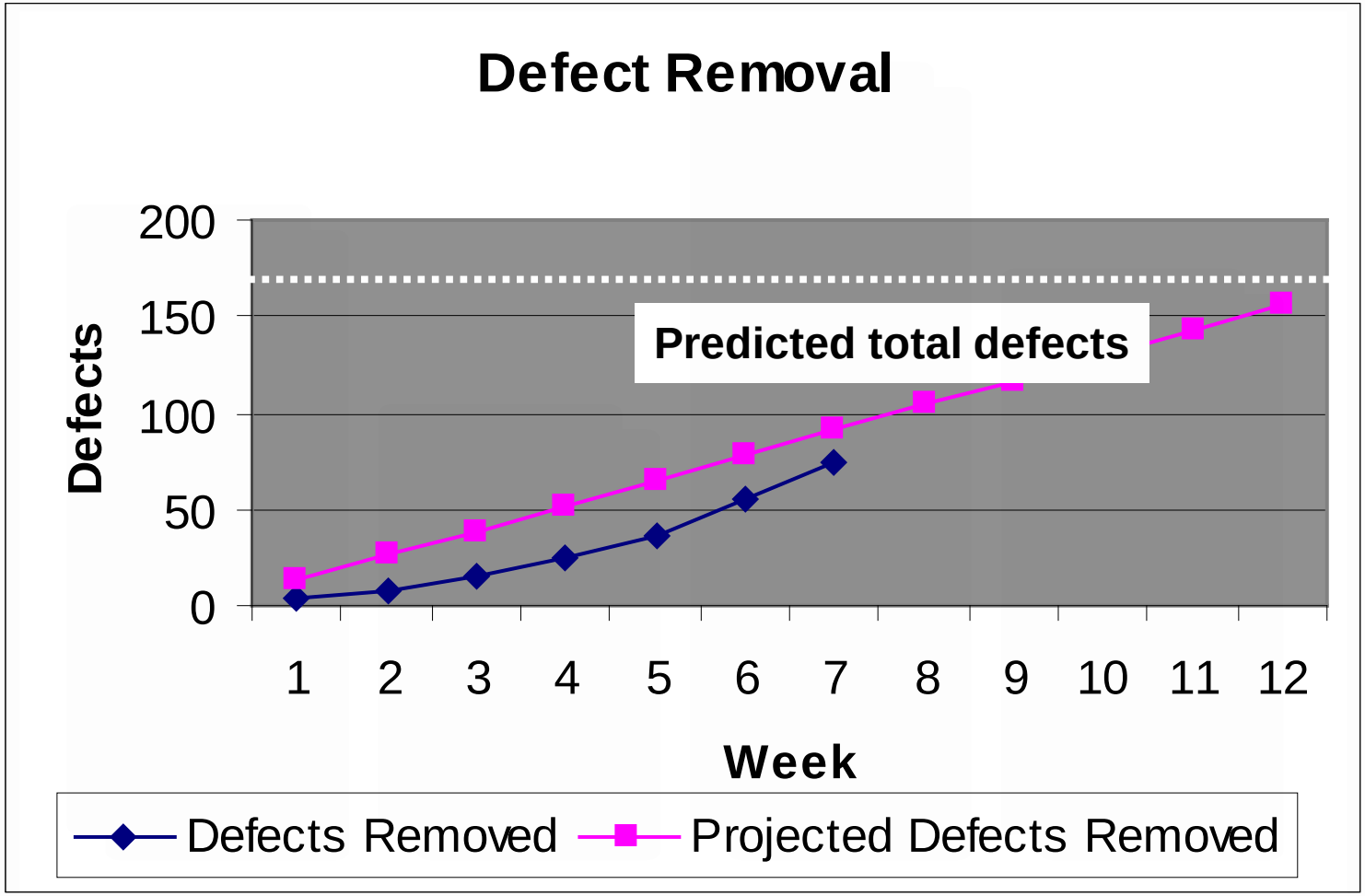
Predicted cumulative defects introduced at each build, with profile for removal of defects

Actual defects removal against predicted total defects



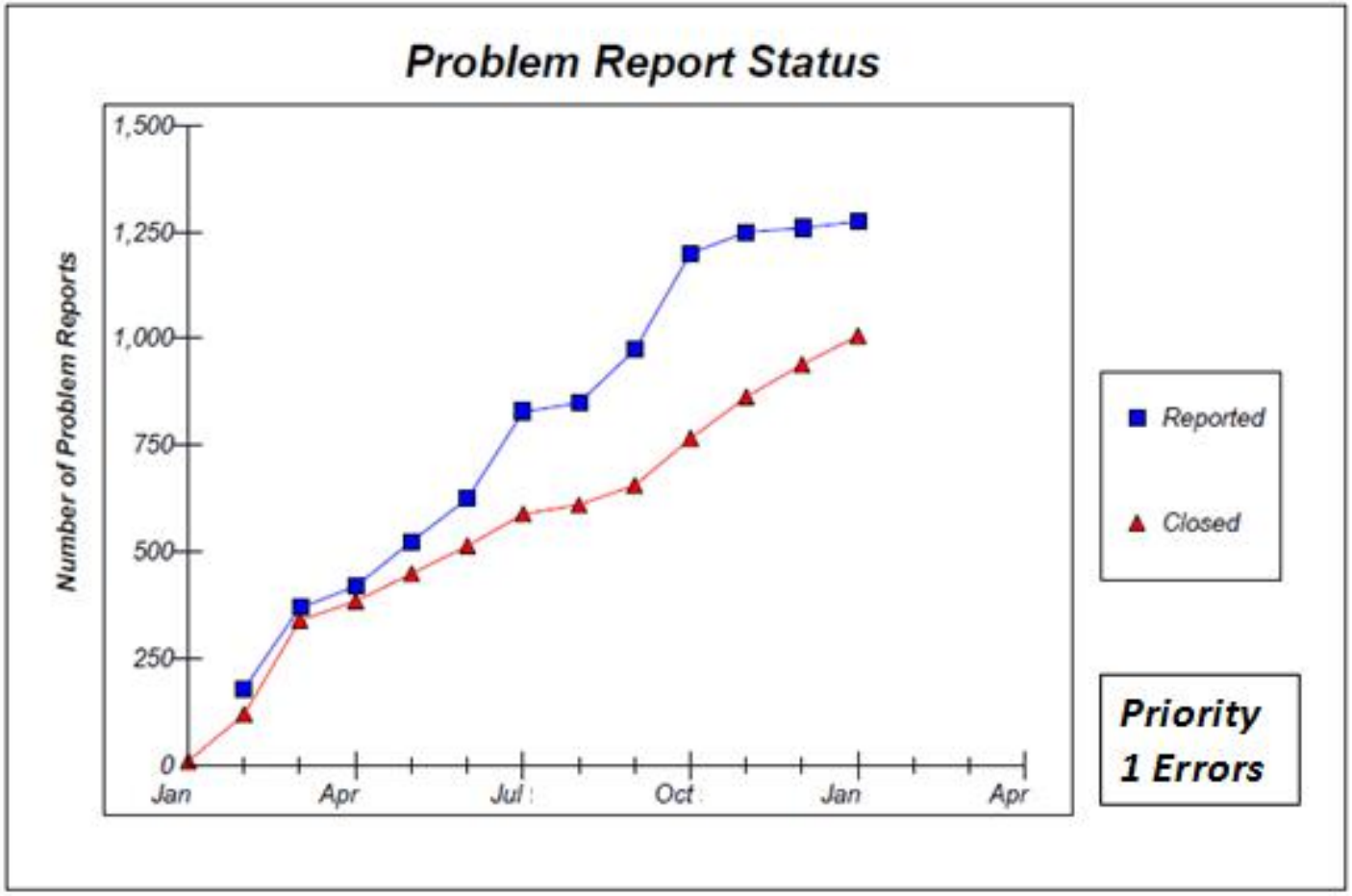
What prerequisites are required to implement this??

Defect Removal - Linear Profile

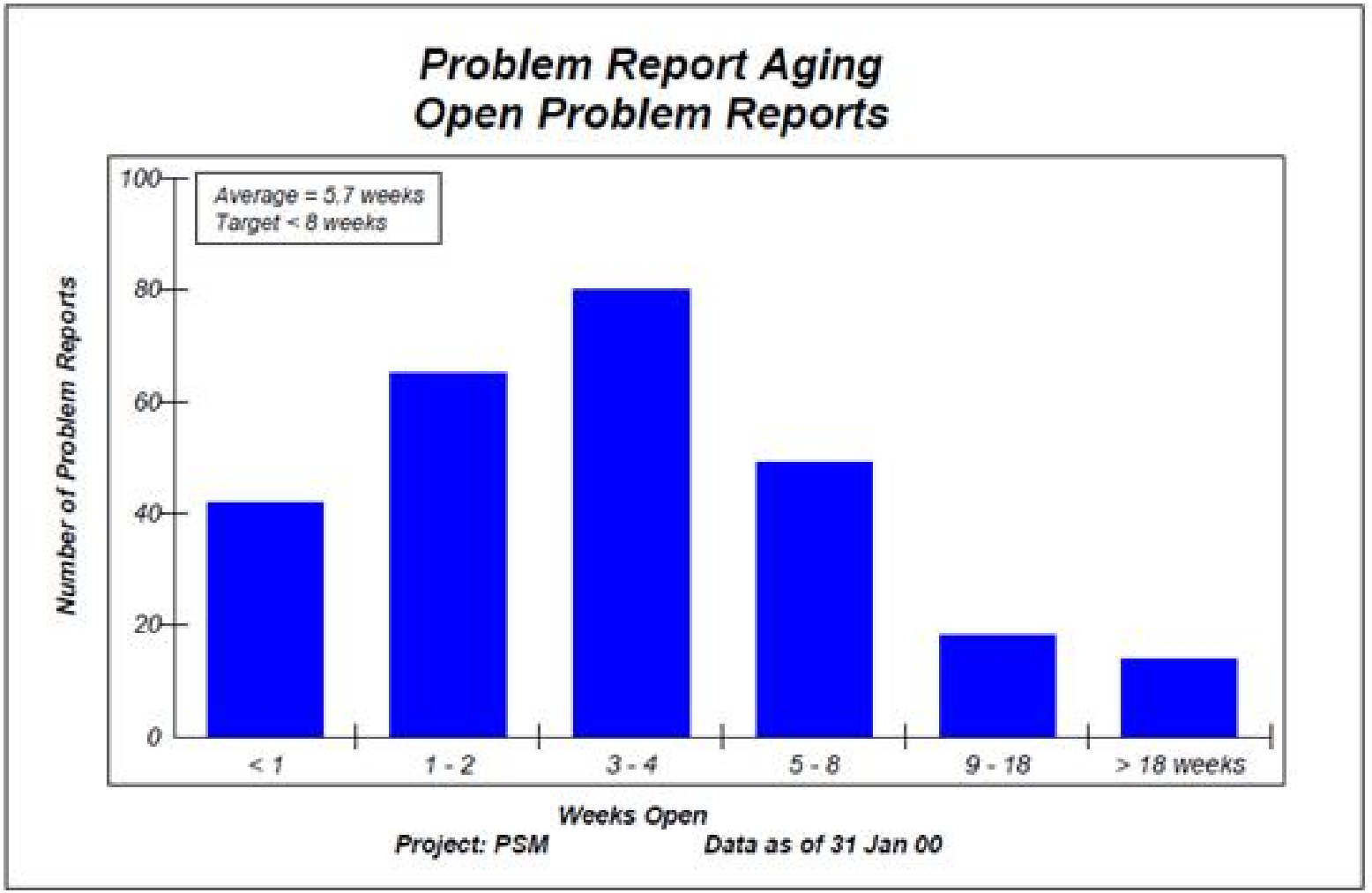


Comparing Number of Defects Found Versus Number Resolved

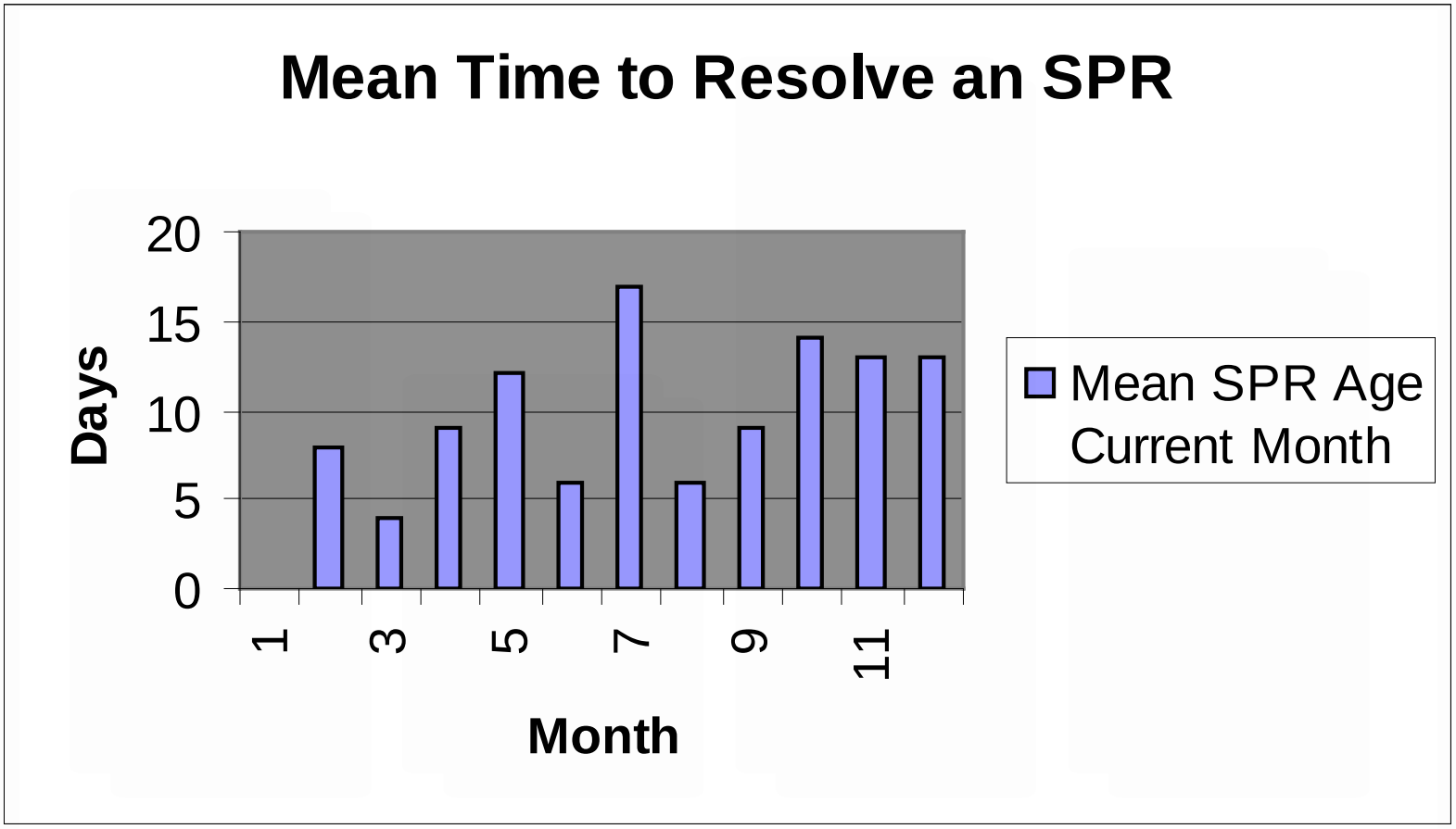
Problem Report Status - Severity Level n



Monitoring How Long Identified Defects Remain Unresolved



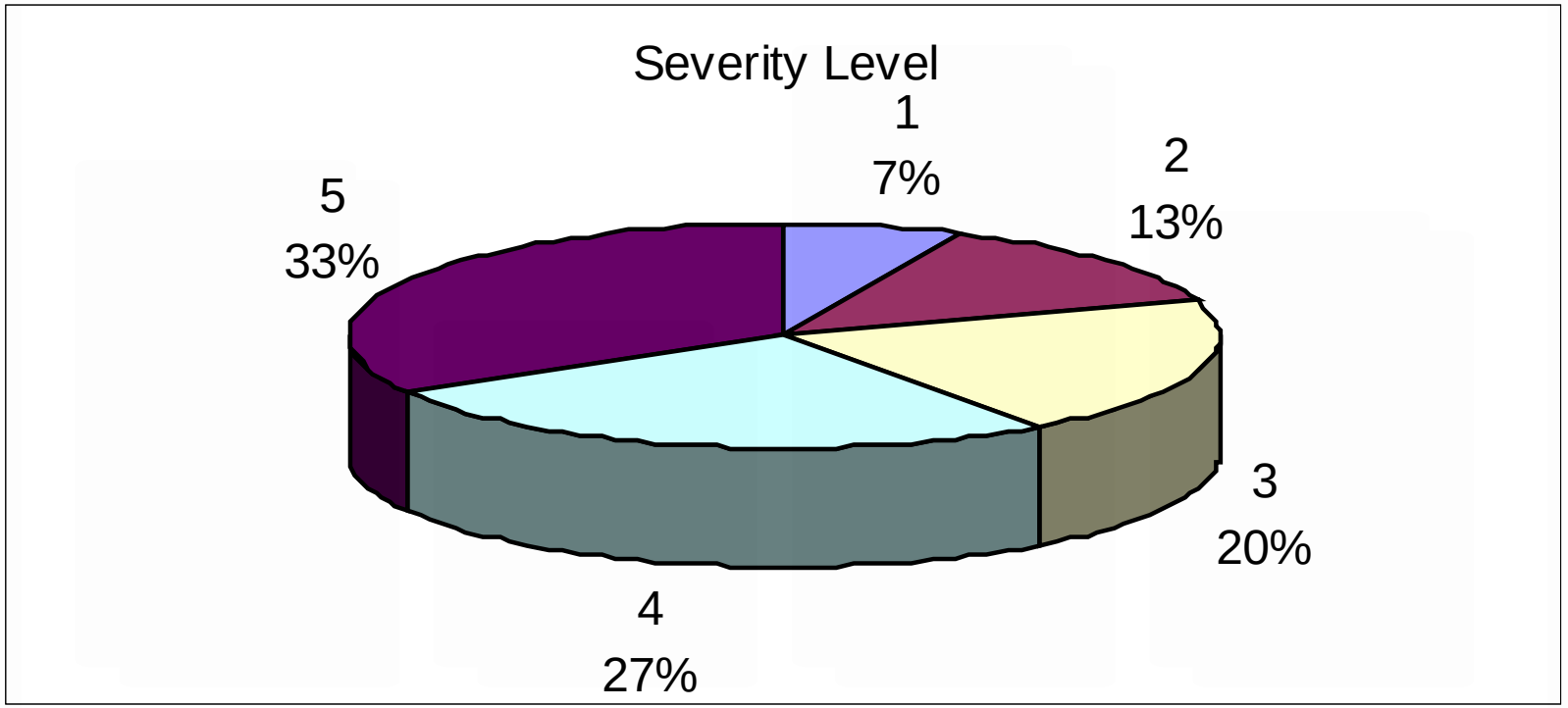
Monitoring How Long Identified Defects Remain Unresolved



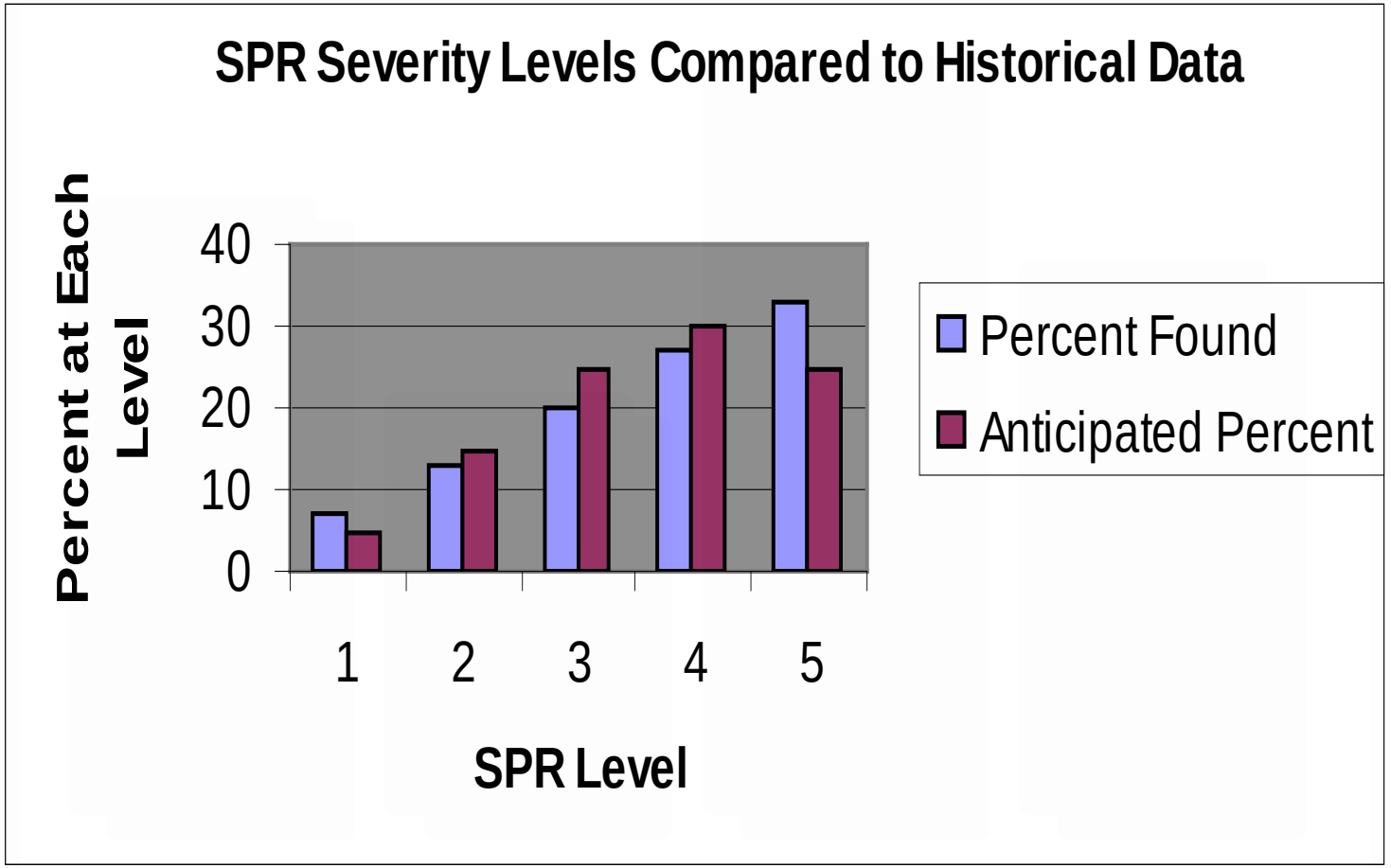
Product Quality

- How severe are the defects found?
  - Depict representation of defect severity
- How "good" is the product?
  - Project when product will be ready for delivery
- How successful are you at identifying and fixing defects?
  - Overlaps Schedule Completion Progress

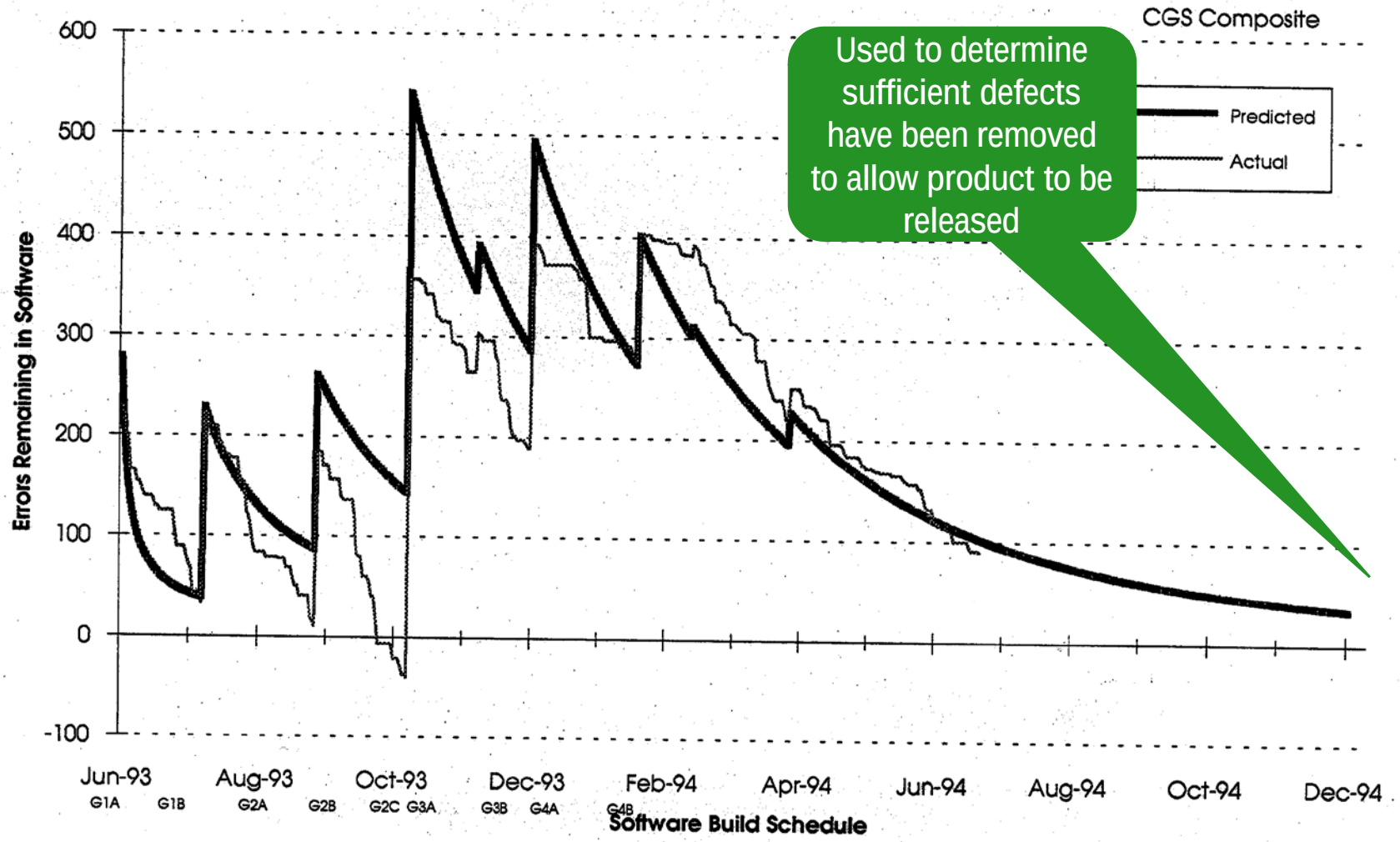
SPR Severity Level



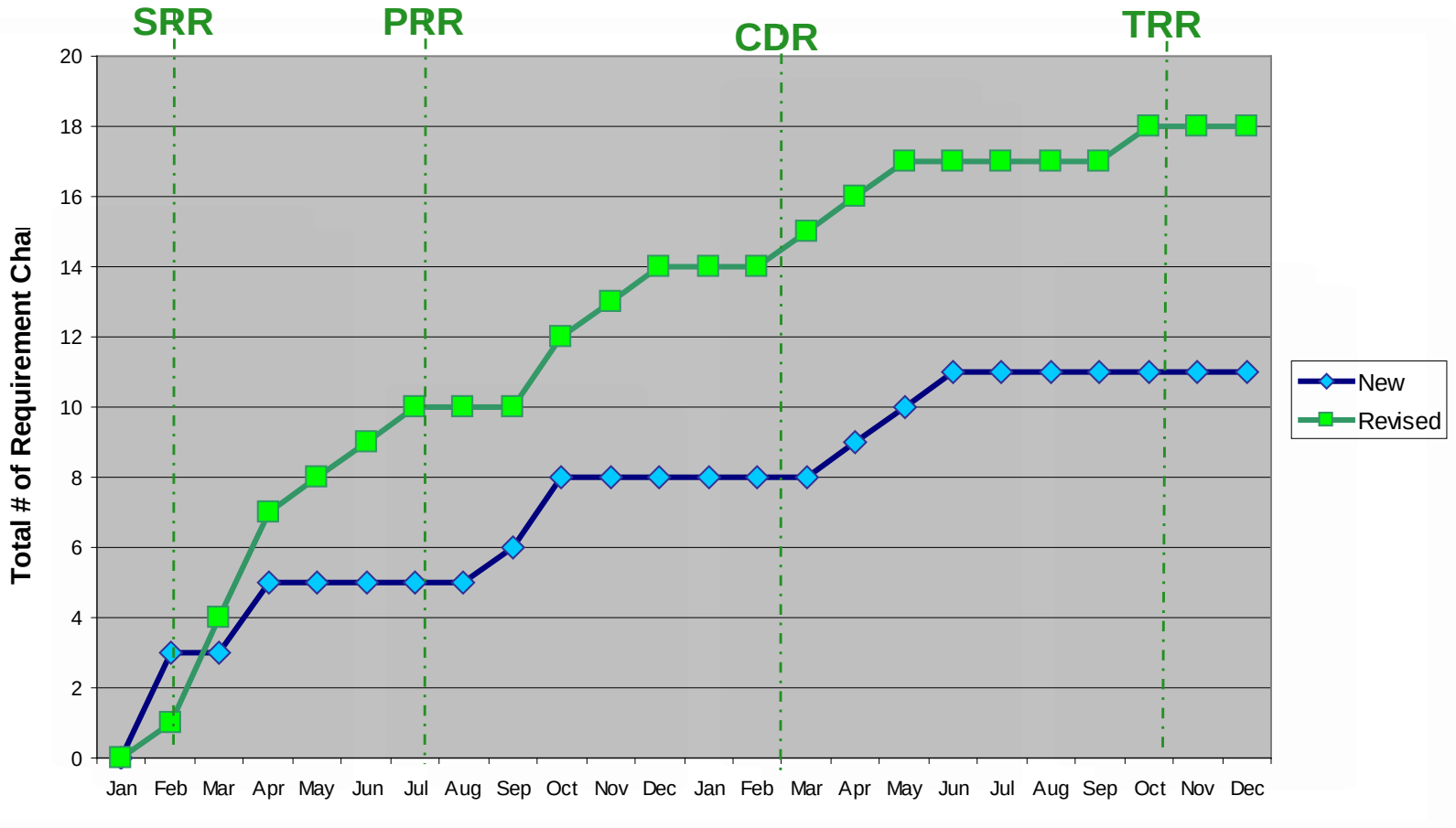
SPR Severity Level Compared to Historical Data



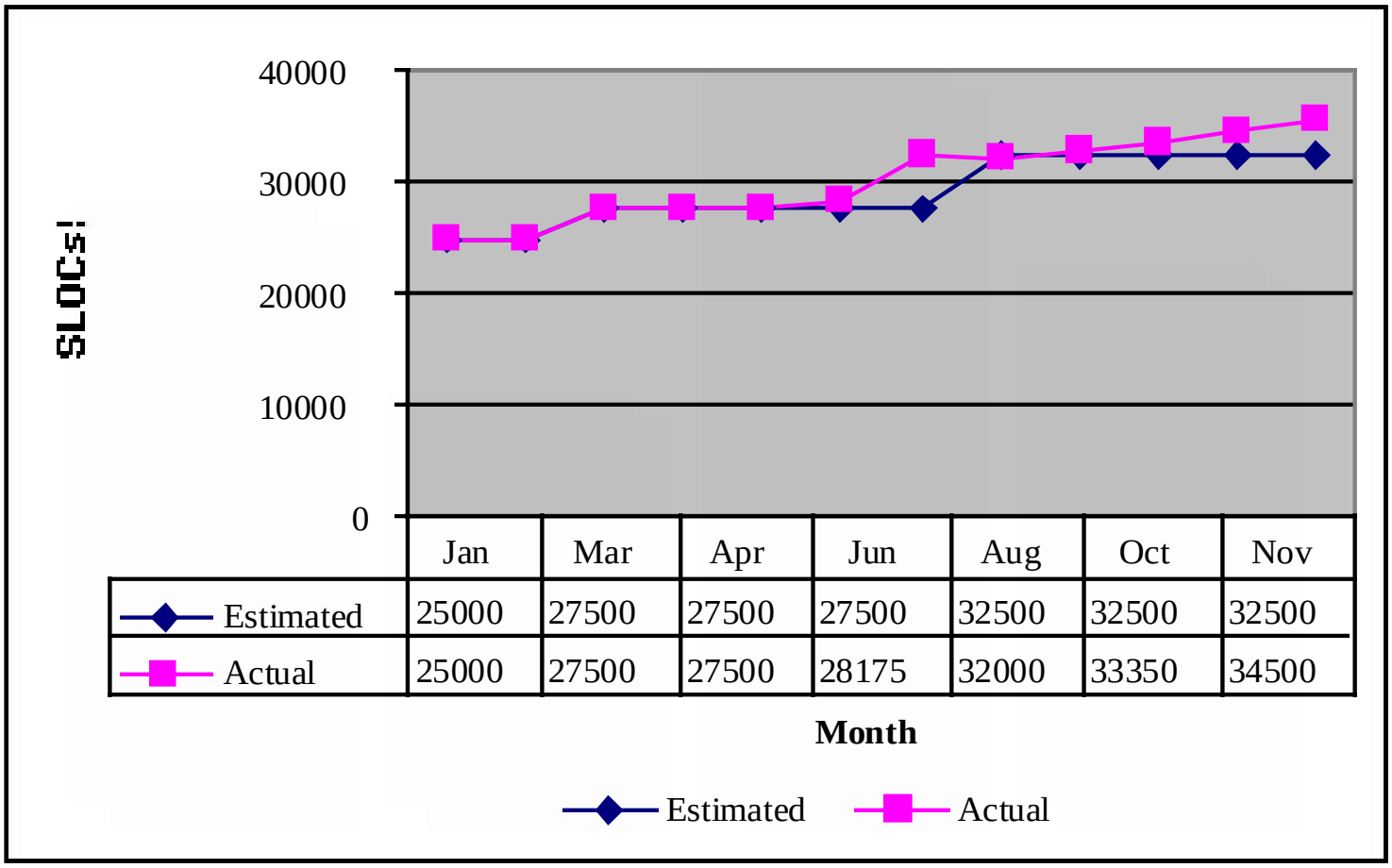
Defect Removal Tracking Using A Statistical Model



Requirements Stability



Software Size Metric



Session 5 – System Integration and Test

Lessons

1. **Program Overview Update**
2. **Process Overviews**
3. **Integration, Planning, Management and Readiness**
4. **System Test Methods**
5. **Test Measurement Techniques and Metrics (if time permits)**

Appendices*

K. **Critical Systems Validation**

L. **Security Testing**

- Relationships with SE Lifecycle
- V&V Processes
- System Interfaces
- Technical Management and Test Processes

- Integration and Test Planning
- Requirements Management
- Interface Management
- Technical Readiness

- White Box Testing
- Black Box Testing
- Integration Testing

- TPM Metrics
- Quality Metrics
- Planning and Control Metrics

*Appendices A through J were contained in earlier sessions of the course; Appendix D has a complete course bibliography. Appendices will not be formally covered.

Metrics Taxonomy

- **Three types of metrics exist**
  - Technical Performance Measures (TPMs)
  - Quality Metrics
  - **Planning and Control Metrics**
    - **Resource utilization**
    - **Schedule**
    - **Staffing**

Resource Utilization

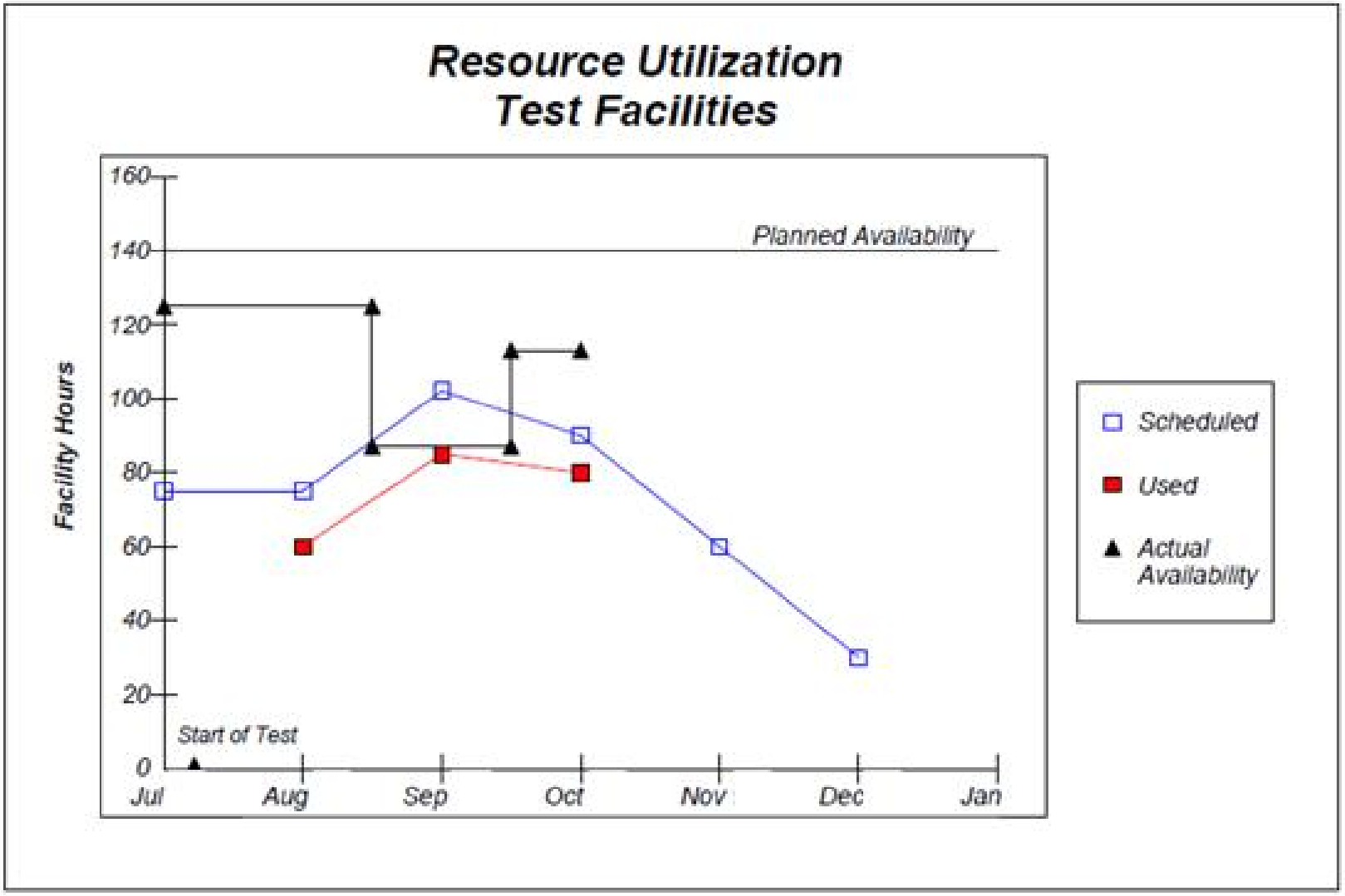
- **Resources include:**
  - **Memory utilization (disk and solid state)**
  - **I/O utilization**
  - **Process throughput**

- **As integration of a system progresses evidence of resource consumption in run-time operations can be assessed and extrapolated to predict future problems**

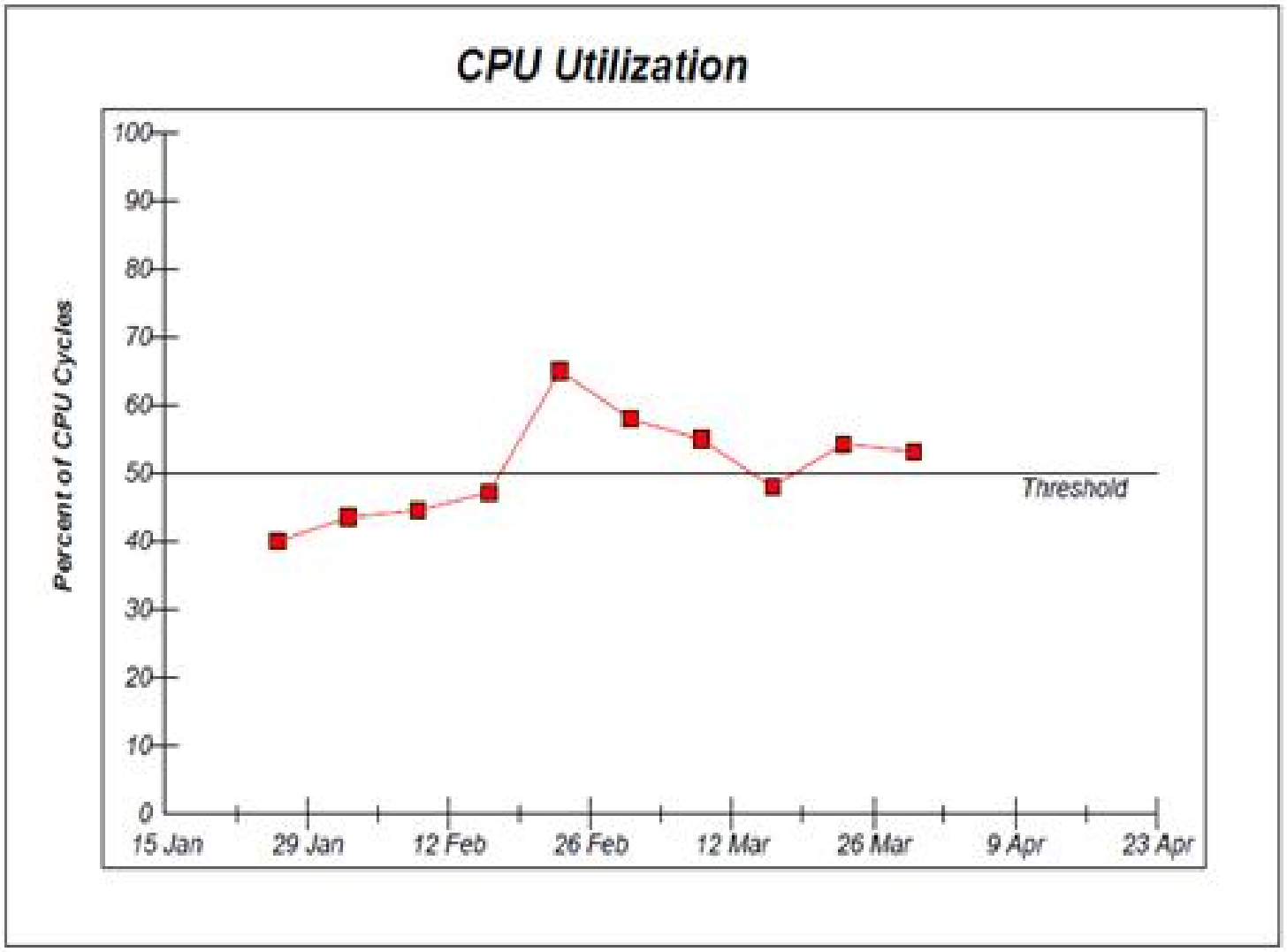
- **Typically a defined reserve for each resource is required at intermediate development milestones and upon delivery of the system**

- **It is critical to track resource utilization to find problems earlier when there is time to respond without delaying system delivery**

Test Facility Utilization



CPU Utilization

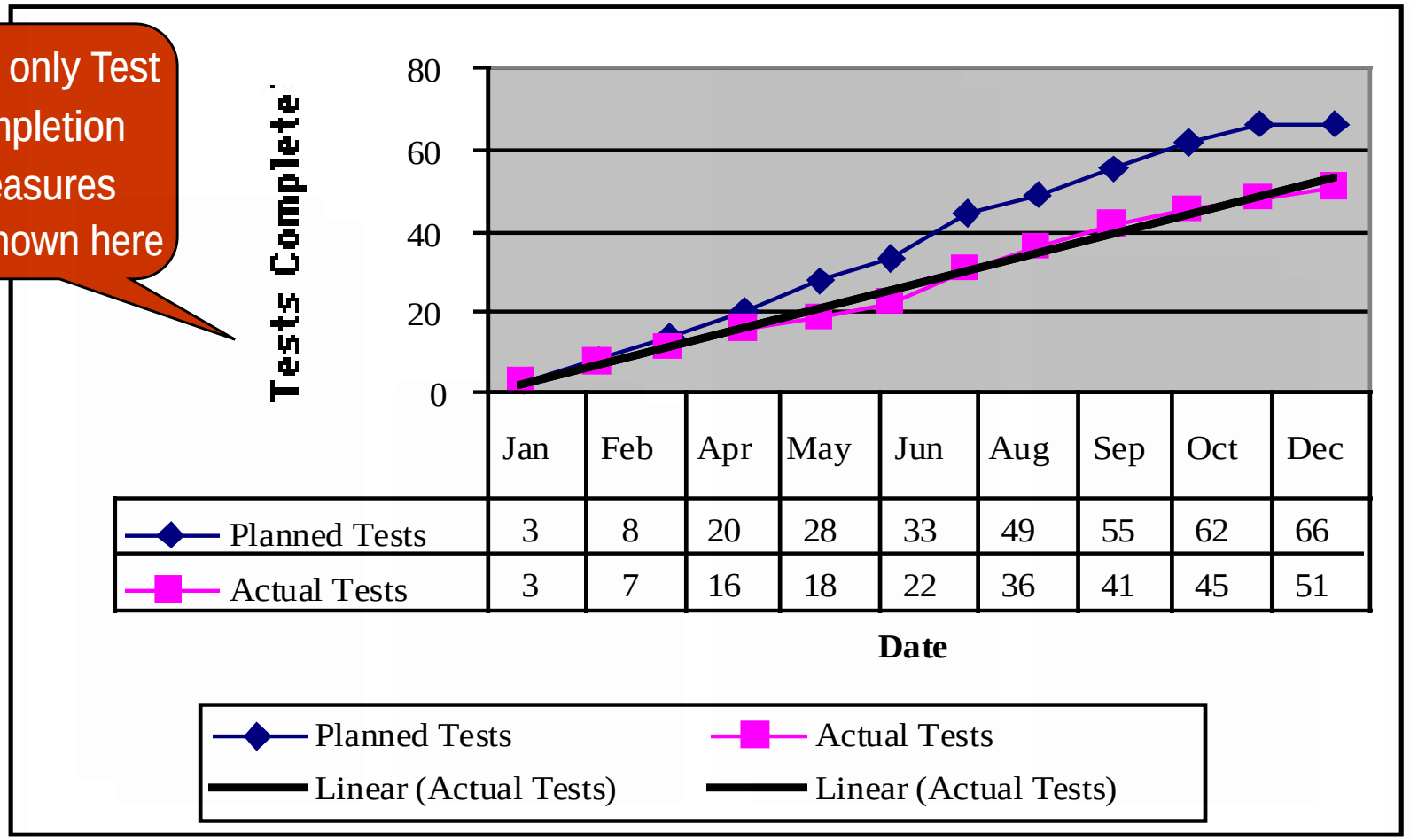


Schedule Progress Example

- **Can be used for any type of scheduled and tracked test event**
  - **Unit test completion**
  - **Integration completion (at any or all levels)**
  - **Demonstration preparation task completion**
  - **Qualification test dry run completion**
  - **Test definition progress**

Schedule Progress Example

Note: only Test Completion Measures are shown here

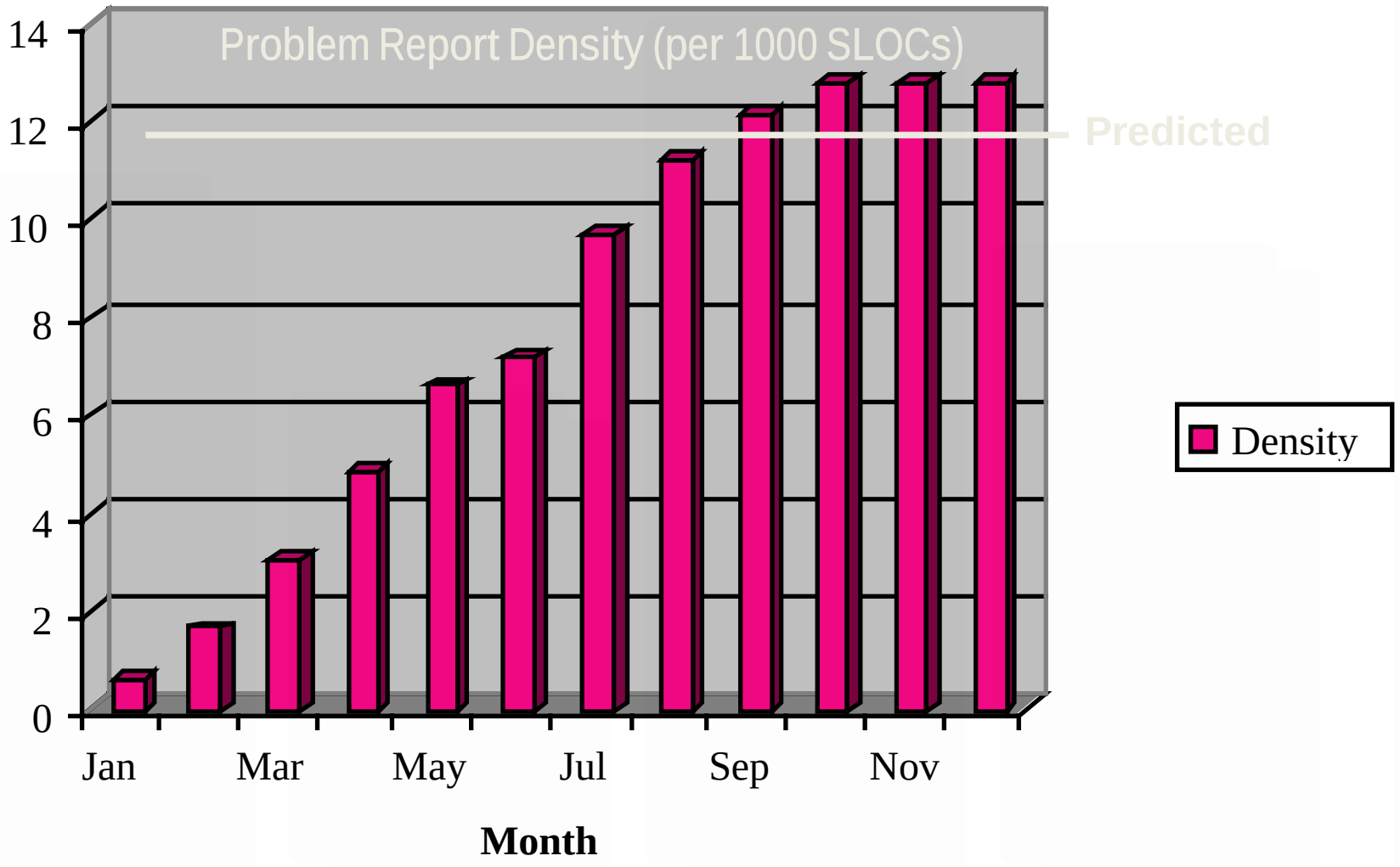


Metrics Analysis

- **Metrics are not solely for static analysis**
  - **Show trends**
  - **Help predict long term performance and future problems**

- **Metric analysis is accomplished by -**
  - **Stand alone assessment - interpret and evaluate the collected data on face value**
  - **Correlation - changes in one metric imply directly a problem with another is forthcoming**
  - **Extrapolation - metrics may extrapolated to predict future performance**

Stand-alone Example

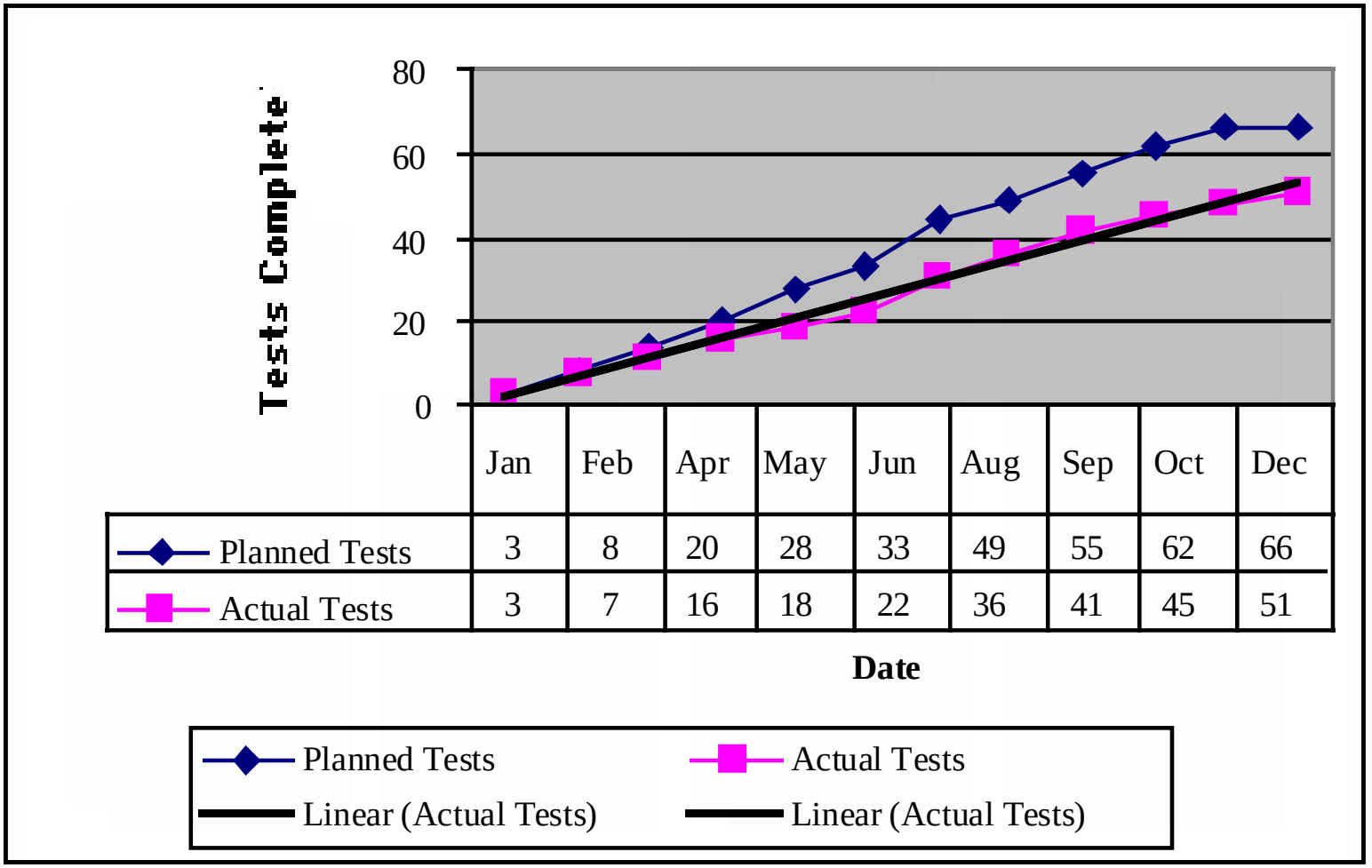


Correlation Example

- A lack of requirements stability can imply potential problems in:
  - Schedule completion
  - Defect detection at a given phase

- A longer than estimated time to resolve System Problem Reports can imply:
  - Schedule problems
  - Design problems
  - Staffing problems
  - Unit test thoroughness problems
  - Insufficient tools or test environment

Extrapolation Example



Long Term Metric Use

- Collect consistent set of metrics over time and across numerous programs
- Normalize the metrics to make them comparable and also provide indicators of future program performance
- Use the metrics as a basis for analyzing performance on current programs
- Use the metrics as a basis for estimating performance on future programs
- Use metrics data for defect prevention

Summary

- Metrics are of value to the systems engineer in assessing and predicting many aspects of system quality, risk, performance, and overall integrity

- Metrics collected, analyzed, and archived for many programs provide valuable comparison data for new programs and support process improvement and defect prevention activities

References

- **Metrics Guidebook for Integrated Systems and Product Development; INCOSE; July 1995**

- **Systems Engineering Measurement Primer; INCOSE; Ver 1.0; March 2008**



Systems Engineering Technical Short Course

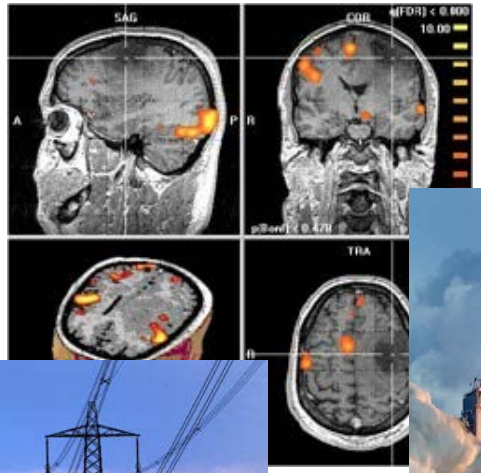
(Session 5: System Integration and Test,

AKA: Verification, Validation and Test - VVT)

Appendix K:
Critical Systems Validation

Critical Systems















Critical System Essentials

Safety

The system should not harm people or the system's environment

Reliability

The system must operate without serious failures

Availability

The system must be available to deliver services when requested to do so

Security

The system must be able to protect itself and its data from malicious use

Classes of Critical Systems

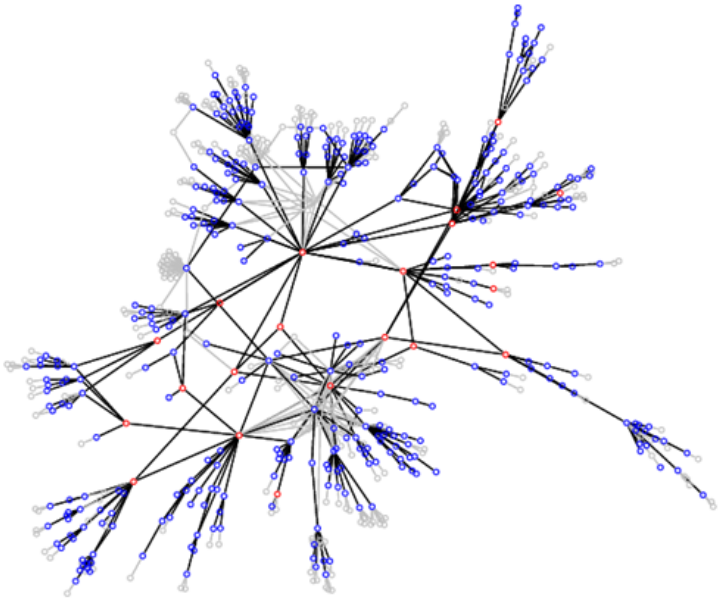
- Safety-critical systems
  - Failure results in loss of life, injury or damage to the environment e.g. chemical plant protection system;
- Mission-critical systems
  - Failure results in failure of some goal-directed activity e.g. spacecraft navigation system;
- Business-critical systems
  - Failure results in high economic losses e.g. customer accounting system in a bank;
- Infrastructure systems
  - Failure results in a loss of infrastructure capability e.g. power distribution control system, broadband communications, etc.

System Dependencies

- Independent critical systems
  - Infrastructure/hardware is part of the system
  - System operation is not dependent on external systems
  - Embedded control systems such as those in medical devices
- Critical software systems
  - Usually rely on commodity hardware/OS
  - System operation is dependent on external infrastructure provision
  - Hospital appointments system

Systems of Systems

- A critical system is rarely a single system but is a network of several software-intensive systems as well as infrastructure systems
- Systems that support organizational needs (e.g. an inter-bank payments system) have to be designed to be robust so that they can cope with failures and unavailability in the other systems on which they depend



Systems of Systems

- Systems of systems (SoS) are complex socio-technical systems with
  - Different owners and management policies
  - Distributed operation
  - Heterogeneous hardware and software
- Individual systems may be part of several SoS so
  - Conflicting requirements from different uses of the system
  - Complex negotiations may be required when system changes are to be made

Socio-Technical Systems



- Socio-technical systems include IT systems and the social and organizational environment in which these systems are used
- Key influences are human behavior, organizational processes and policies, regulations, culture

Regulation

- Regulators are government-appointed bodies whose job is to ensure that companies and other bodies conform to national and international laws.
- This normally involves interpreting the law and government policy and establishing standards and regulations that must be followed by industry.
- Examples of regulators
  - FCC
  - FAA
  - FDA
  - DOE



Regulators and Critical Systems

- Some critical systems may have to be certified by regulators before they are put into use. This is particularly true for safety-critical systems.
- This means that the regulators check that the system is conformant to current regulations and standards.
  - This normally involves the system developers producing evidence (a safety case or a dependability case e.g.) that demonstrates that the system is dependable.
- Examples
  - FAA – Avionics
  - FDA – Medical devices



System Criticality

- Primary critical systems
  - Systems where system failure leads directly to an incident that has an associated loss of some kind
  - Typically, these are control systems or systems that are closely associated with a control system
  - Example – failure of engine management system in a car causes engine to cut out while driving
- Secondary critical systems
  - Systems whose failure may (but need not) lead to failure in an associated system that then leads to loss of some kind
  - Example – medical information system that maintains incorrect information about treatment

Critical Systems Engineering

- Focus is on the use of techniques and methods to develop dependable and secure systems.
- The costs of critical system failure are so high that development methods may be used that are not cost-effective for other types of system.
- An important aim for many critical systems is certification and the development process has to be geared to achieving such certification.
- Certification costs can exceed development costs.

Software Engineering for Critical Systems

- Formal methods for systems specification and analysis.
- Use of specialized tools such as model checkers and static analyzers.
- Risk-driven approach to system specification and management.
- Systems to support the development of dependability cases.
- Disciplined configuration management of all software and hardware.
- Detailed process record keeping.

System Dependability

- For many computer-based systems, the most important system property is the dependability of the system.
- It reflects the extent of the user's confidence that it will operate as users expect and that it will not 'fail' in normal use.
- So, you can think of a dependable system as a system that does not fail.

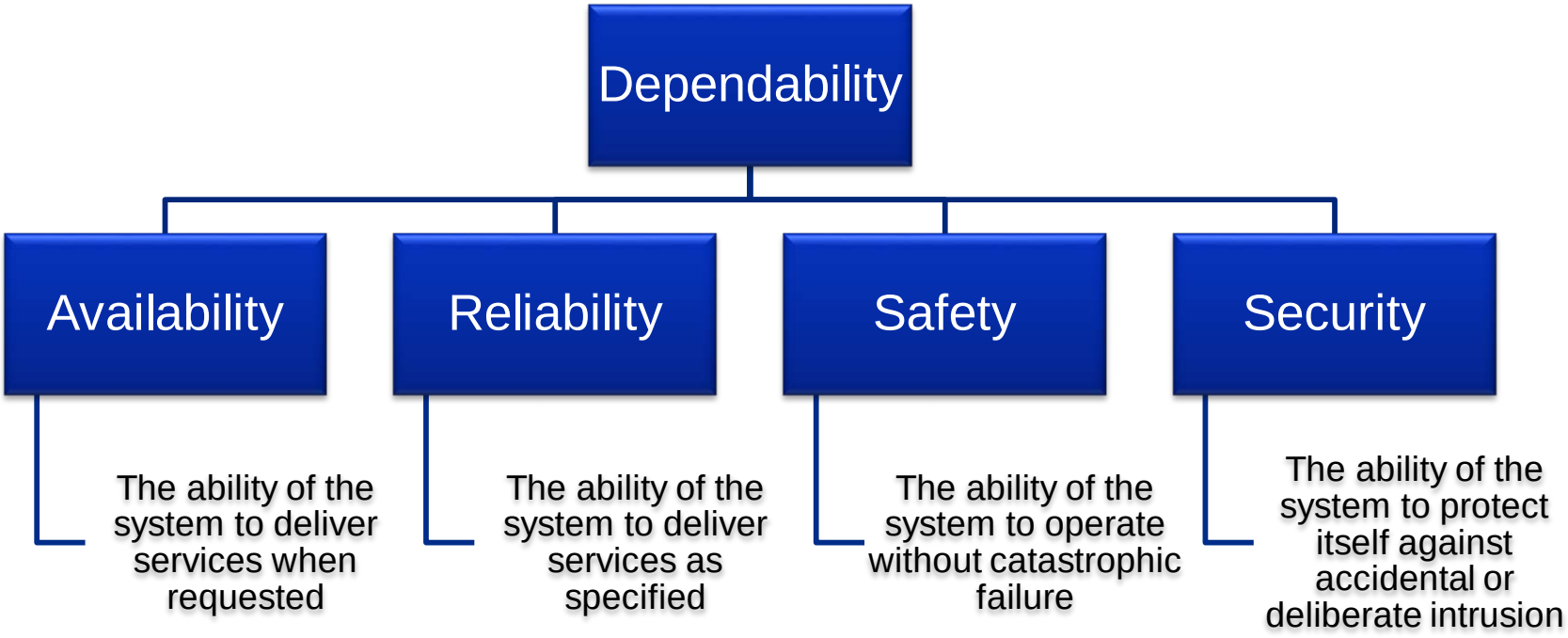
Importance of Dependability

- System failures may have widespread effects with large numbers of people affected by the failure.
- Systems that are not dependable and are unreliable, unsafe or insecure may be rejected by their users.
- The costs of system failure may be very high if the failure leads to economic losses or physical damage.
- Undependable systems may cause information loss with a high consequent recovery cost.

Dependability is Subjective

- The dependability of a system is not absolute but depends on the judgement of a system stakeholder. What seems to be a system failure to one stakeholder is normal behaviour to another.
- The dependability of a system reflects the user's degree of trust in that system.

Principal Dependability Properties



Dependability Attribute Dependencies

- Safe system operation depends on the system being available and operating reliably.
- A system may be unreliable because its data has been corrupted by an external attack.
- Denial of service attacks on a system are intended to make it unavailable.
- If a system is infected with a virus, you cannot be confident in its reliability or safety.

Validation of Critical Systems

- The verification and validation costs for critical systems involves additional validation processes and analysis than for non-critical systems:
  - The costs and consequences of failure are high so it is cheaper to find and remove faults than to pay for system failure;
  - May have to make a formal case to customers or to a regulator that the system meets its dependability requirements.
  - This dependability case may require specific V&V activities to be carried out.
  - V&V costs can make up more than 50% of the total system development costs

Validation Perspectives

- Reliability validation
  - Does measured system reliability meet its specification?
  - Is system reliability good enough to satisfy users?
- Safety validation
  - Does system operate so that accidents do not occur?
  - Are accident consequences minimized?
- Security validation
  - Is system secure against external attack?

Validation Techniques

- Static techniques
  - design reviews and program inspections
  - mathematical arguments and proof
- Dynamic techniques
  - statistical testing
  - scenario-based testing
  - run-time checking
- Process validation
  - SE processes should minimize the chances of introducing system defects

Static Validation Techniques

- Concerned with analysis of documentation
- Focus is on finding system errors and identifying potential problems that may arise during system operation
- Documents may be prepared to support static validation
  - structured arguments
  - mathematical proofs

Formal Methods and Validation

- Specification validation
  - developing a formal model of a system specification often reveals errors and omissions
  - mathematical analysis of a formal specification is another way to discover specification problems
- Formal verification
  - mathematical arguments are used to demonstrate that a program or design is consistent with its formal specification

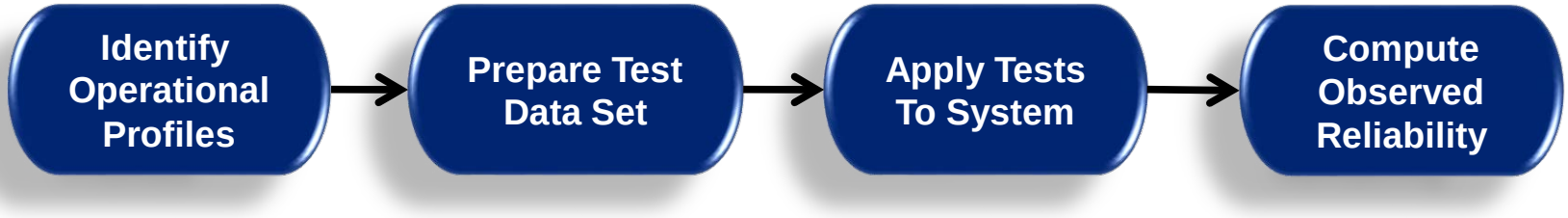
Formal Validation Problems

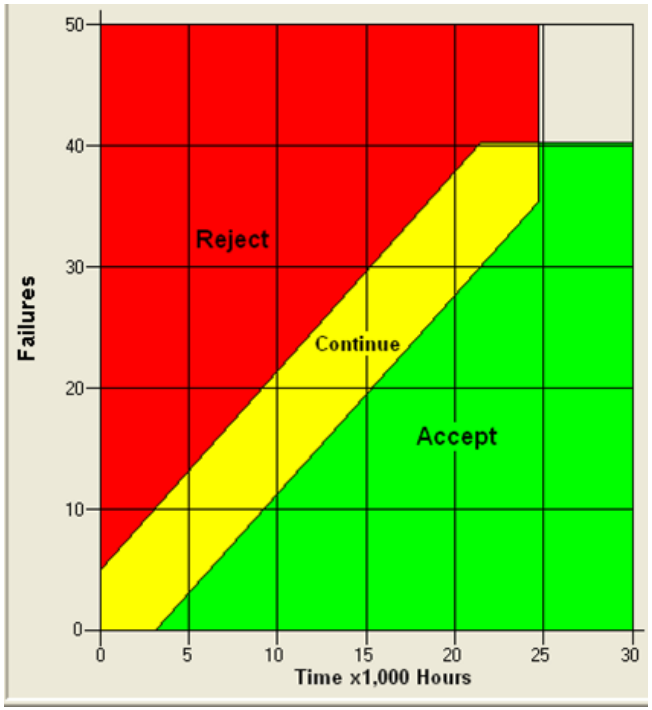
- The formal model of the specification is not likely to be understood by the domain expert
  - this makes it hard to check that the formal model is an accurate representation of the system specification
  - a consistently wrong specification is useless
- Verification does not scale-up
  - verification is a complex, error-prone process
  - the cost of verification increases exponentially with system size

Reliability Validation

- Reliability validation involves exercising the program to assess whether or not it has reached the required level of reliability
- This cannot normally be included as part of a normal defect testing process because data for defect testing is (usually) atypical of actual usage data
- Reliability measurement therefore requires a specially designed data set that replicates the pattern of inputs to be processed by the system
  - Operational Profiles

The Reliability Measurement Process





- Establish the operational profile for the system
- Construct test data reflecting the operational profile
- Test the system and observe the number of failures and the times of these failures
- Compute the reliability after a statistically significant number of failures have been observed

Reliability Measurement Problems

- Operational profile uncertainty
  - The operational profile may not be an accurate reflection of the real use of the system.
- High costs of test data generation
  - Costs can be very high if the test data for the system cannot be generated automatically.
- Statistical uncertainty
  - You need a statistically significant number of failures to compute the reliability but highly reliable systems will rarely fail.

Operational Profiles

- An operational profile is a set of test data whose frequency matches the actual frequency of these inputs from 'normal' usage of the system.
- A close match with actual usage is necessary otherwise the measured reliability will not be reflected in the actual usage of the system.
- Can be generated from real data collected from an existing system or (more often) depends on assumptions made about the pattern of usage of a system.

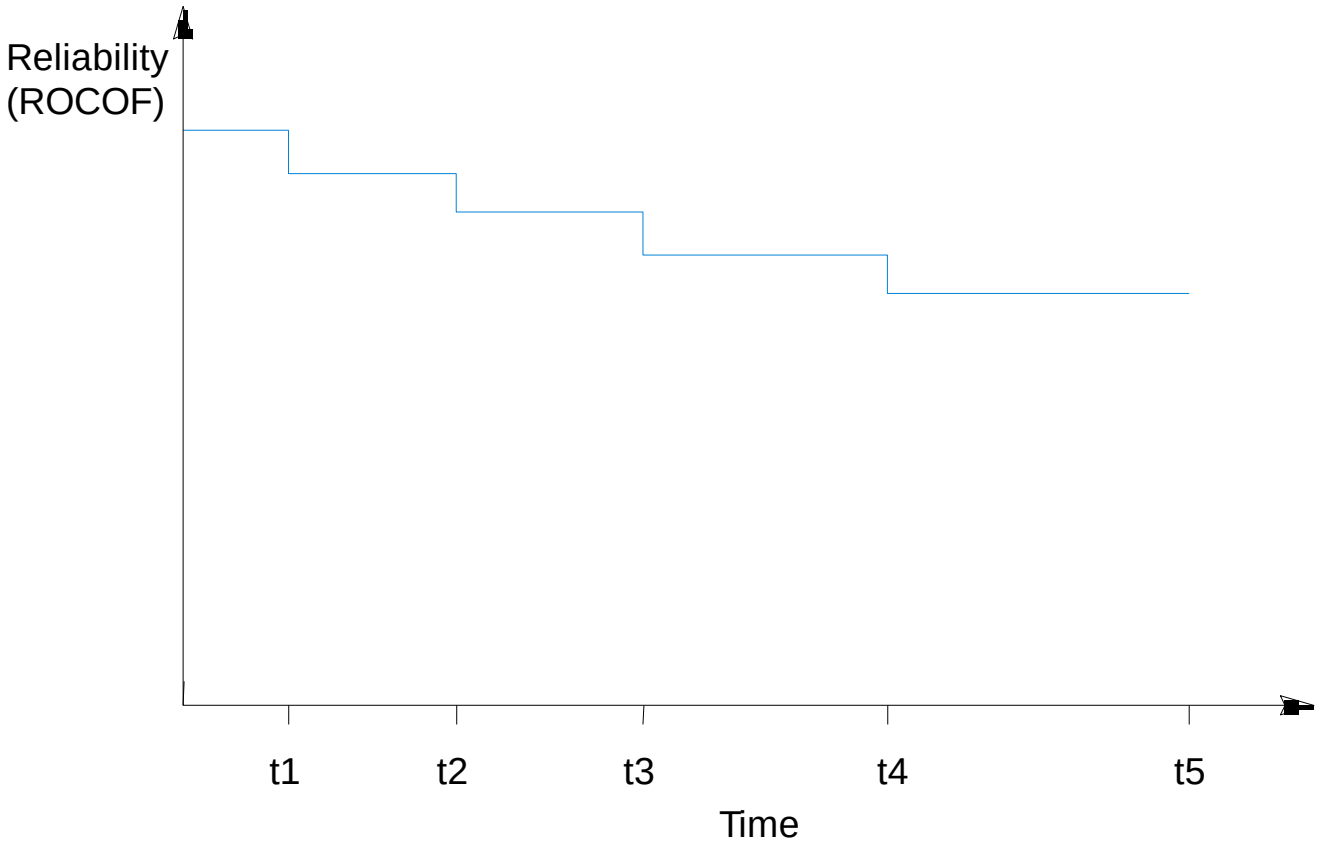
Reliability Prediction

- A reliability growth model is a mathematical model of the system reliability change as it is tested and faults are removed.

- It is used as a means of reliability prediction by extrapolating from current data
  - Simplifies test planning and customer negotiations.
  - You can predict when testing will be completed and demonstrate to customers whether or not the reliability growth will ever be achieved.

- Prediction depends on the use of statistical testing to measure the reliability of a system version.

Equal-step Reliability Growth

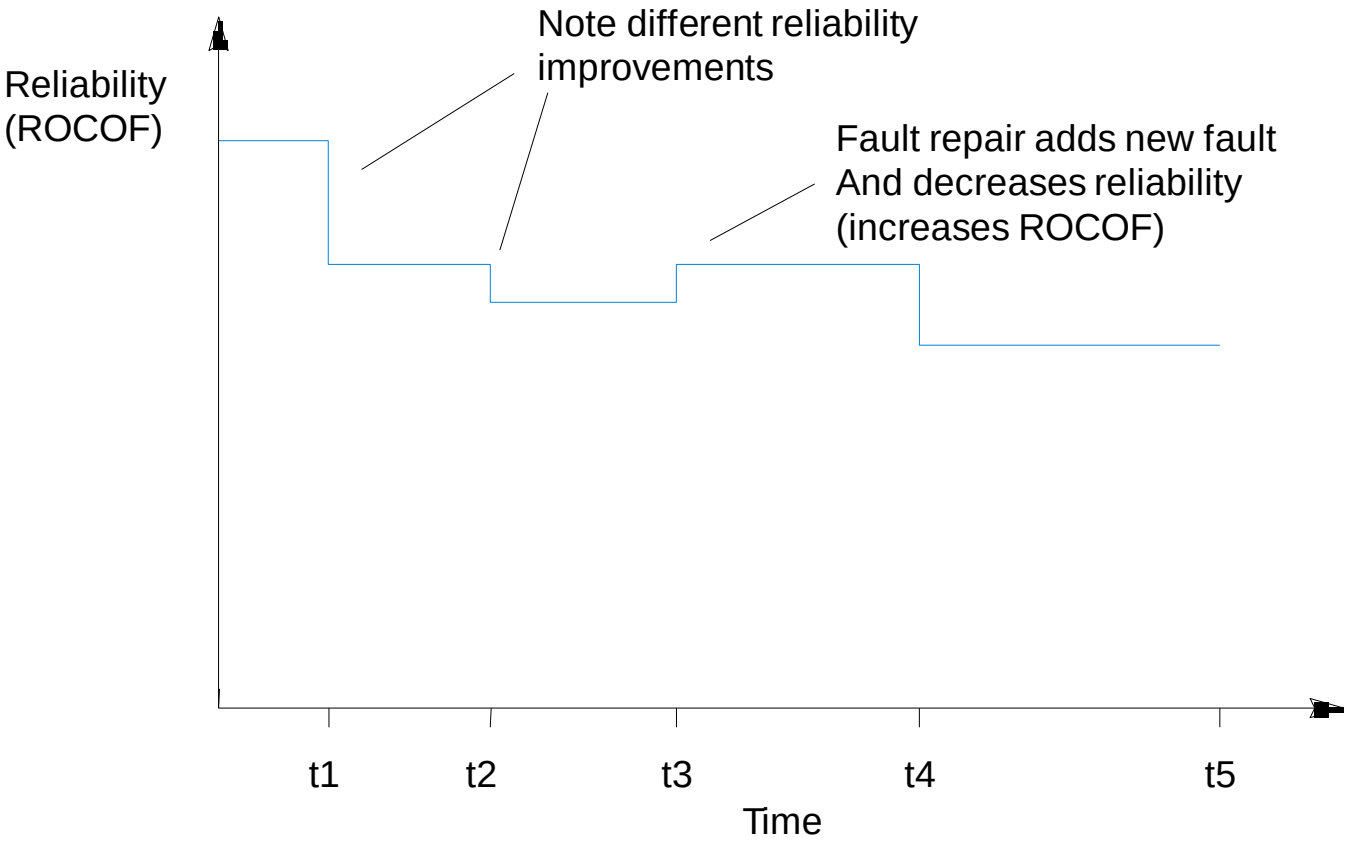


ROCOF - Rate Of Occurrence Of Failures

Observed Reliability Growth

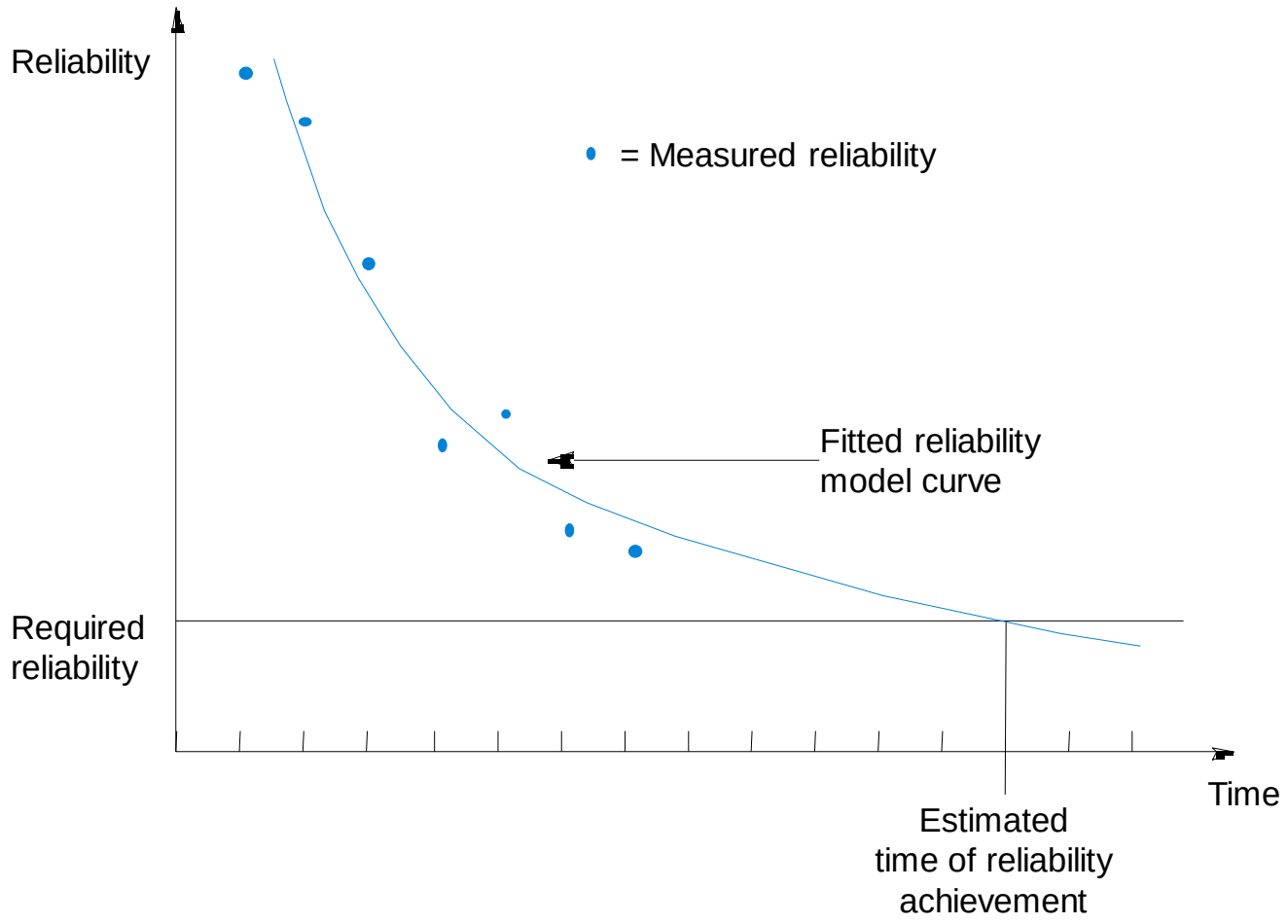
- The equal-step growth model is simple but it does not normally reflect reality.
- Reliability does not necessarily increase with change as the change can introduce new faults.
- The rate of reliability growth tends to slow down with time as frequently occurring faults are discovered and removed from the software.
- A random-growth model where reliability changes fluctuate may be a more accurate reflection of real changes to reliability.

Random-step Reliability Growth



ROCOF - Rate Of Occurrence Of Failures

Reliability Prediction



Safety

- Safety is a property of a system that reflects the system's ability to operate, normally or abnormally, without danger of causing human injury or death and without damage to the system's environment.

- It is important to consider software safety as most devices whose failure is critical now incorporate software-based control systems.

- Safety requirements are often exclusive requirements i.e. they exclude undesirable situations rather than specify required system services. These generate functional safety requirements.

Safety Assurance

- Safety assurance and reliability measurement are quite different:
  - Within the limits of measurement error, you know whether or not a required level of reliability has been achieved;
  - However, quantitative measurement of safety is impossible.
  - Safety assurance is concerned with establishing a confidence level in the system.
  - A reliable system can be unsafe if the requirements are incorrect or incomplete



Unsafe Reliable Systems

- There may be dormant faults in a system that are undetected for many years and only rarely arise.
- Specification errors
  - If the system specification is incorrect then the system can behave as specified but still cause an accident.
- Hardware failures generating spurious inputs
  - Hard to anticipate in the specification.
- Context-sensitive commands i.e. issuing the right command at the wrong time
  - Often the result of operator error.



Safety Confidence

- Confidence in the safety of a system can vary from very low to very high.

- Confidence is developed through:
  - Past experience with the company developing the software;
  - The use of dependable processes and process activities geared to safety;
  - Extensive V&V including both static and dynamic validation techniques.

Safety Achievement

- Hazard avoidance
  - The system is designed so that some classes of hazard simply cannot arise.
- Hazard detection and removal
  - The system is designed so that hazards are detected and removed before they result in an accident.
- Damage limitation
  - The system includes protection features that minimise the damage that may result from an accident.

Normal Accidents

- Accidents in complex systems rarely have a single cause as these systems are designed to be resilient to a single point of failure
  - Designing systems so that a single point of failure does not cause an accident is a fundamental principle of safe systems design.

- Almost all accidents are a result of combinations of malfunctions rather than single failures.

- It is probably the case that anticipating all problem combinations, especially, in software controlled systems is impossible so achieving complete safety is impossible. Accidents are inevitable.

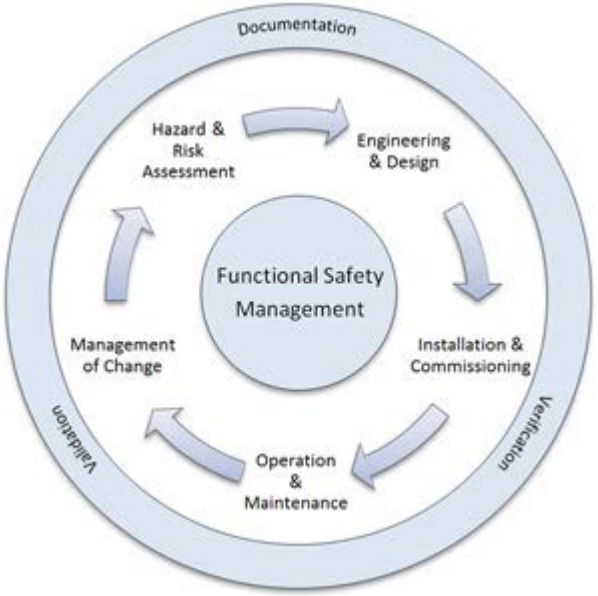
Process Assurance

- Process assurance involves defining a dependable process and ensuring that this process is followed during the system development.

- The use of a safe process is a mechanism for reducing the chances that errors are introduced into a system.
  - Accidents are rare events so testing may not find all problems;
  - Safety requirements are sometimes 'shall not' requirements so cannot be demonstrated through testing.

Safety Related Process Activities

- Creation of a hazard logging and monitoring system.
- Appointment of project safety engineers.
- Extensive use of safety reviews.
- Creation of a safety certification system.
- Detailed configuration management.



Hazard Analysis

- Hazard analysis involves identifying hazards and their root causes.
- There should be clear traceability from identified hazards through their analysis to the actions taken during the process to ensure that these hazards have been covered.
- A hazard log may be used to track hazards throughout the process.
- Safety can be assured by
  - hazard avoidance
  - accident avoidance
  - protection systems



Safety Arguments

- Safety arguments are intended to show that the system cannot reach in unsafe state.
- These are weaker than correctness arguments which must show that the system code conforms to its specification.
- They are generally based on proof by contradiction
  - Assume that an unsafe state can be reached;
  - Show that this is contradicted by the program code.
- A graphical model of the safety argument may be developed.

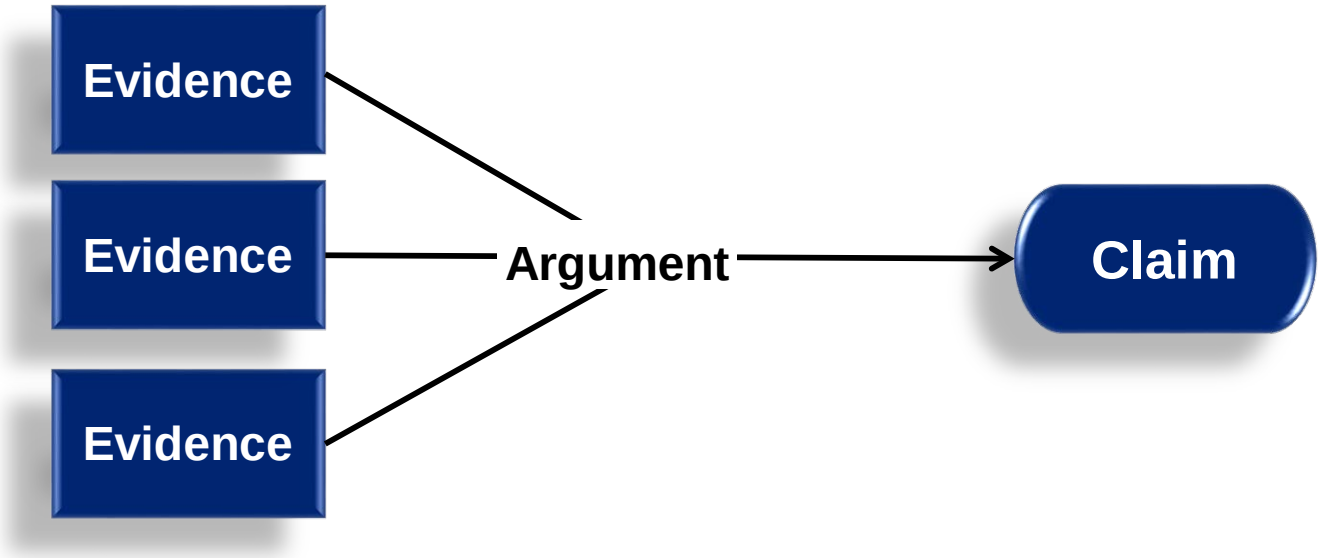
The System Safety Case

- It is now normal practice for a formal safety case to be required for all safety-critical computer-based systems e.g. railway signalling, air traffic control, etc.
- A safety case is:
  - *A documented body of evidence that provides a convincing and valid argument that a system is adequately safe for a given application in a given environment*.
- Arguments in a safety or dependability case can be based on formal proof, design rationale, safety proofs, etc. Process factors may also be included.

Components of a Safety Case

Component	Description
System description	An overview of the system and a description of its critical components.
Safety requirements	The safety requirements abstracted from the system requirements specification.
Hazard and risk analysis	Documents describing the hazards and risks that have been identified and the measures taken to reduce risk.
Design analysis	A set of structured arguments that justify why the design is safe.
Verification and validation	A description of the V & V procedures used and, where appropriate, the test plans for the system. Results of system V &V.
Review reports	Records of all design and safety reviews.
Team competences	Evidence of the competence of all of the team involved in safety-related systems development and validation.
Process QA	Records of the quality assurance processes carried out during system development.
Change management processes	Records of all changes proposed, actions taken and, where appropriate, justification of the safety of these changes.
Associated safety cases	References to other safety cases that may impact on this safety case.

Argument Structure



- Much like a legal case presented in a courtroom, an assurance case requires arguments linking evidence with claims of conformance to dependability-related requirements.
- A claim embodies what we want to show.
- An argument tells us why we believe a claim has been met.
- Evidence supporting the argument that may take many forms, including test results, formal analyses, simulation results, fault-tree analyses, hazard analyses, modeling, and inspections

Security Assurance

- Vulnerability avoidance
  - The system is designed so that vulnerabilities do not occur. For example, if there is no external network connection then external attack is impossible
- Attack detection and elimination
  - The system is designed so that attacks on vulnerabilities are detected and neutralised before they result in an exposure. For example, virus checkers find and remove viruses before they infect a system
- Exposure limitation and recovery
  - The system is designed so that the adverse consequences of a successful attack are minimised. For example, a backup policy allows damaged information to be restored

Security Assessment

- Security assessment has something in common with safety assessment.

- It is intended to demonstrate that the system cannot enter some state (an unsafe or an insecure state) rather than to demonstrate that the system can do something.

- However, there are differences
  - Safety problems are accidental; security problems are deliberate;
  - Security problems are more generic - many systems suffer from the same problems; Safety problems are mostly related to the application domain

Security Validation

- Experience-based validation
  - The system is reviewed and analysed against the types of attack that are known to the validation team.
- Tool-based validation
  - Various security tools such as password checkers are used to analyse the system in operation.
- Tiger teams
  - A team is established whose goal is to breach the security of the system by simulating attacks on the system.
- Formal verification
  - The system is verified against a formal security specification.

Security Testing

- Testing the extent to which the system can protect itself from external attacks.
- Problems with security testing
  - Security requirements are ‘shall not’ requirements i.e. they specify what should not happen. It is not usually possible to define security requirements as simple constraints that can be checked by the system.
  - The people attacking a system are intelligent and look for vulnerabilities. They can experiment to discover weaknesses and loopholes in the system.
- Static analysis may be used to guide the testing team to areas of the program that may include errors and vulnerabilities.

Summary

- Reliability measurement relies on exercising the system using an operational profile - a simulated input set which matches the actual usage of the system.
- Reliability growth modelling is concerned with modelling how the reliability of a software system improves as it is tested and faults are removed.
- Safety arguments or proofs are a way of demonstrating that a hazardous condition can never occur.
- It is important to have a dependable process for safety-critical systems development. The process should include hazard identification and monitoring activities.
- Security validation may involve experience-based analysis, tool-based analysis or the use of 'tiger teams' to attack the system.
- Safety cases collect together the evidence that a system is safe.

In Closing


WALLY WRITES THE CRITICAL CODE FOR OUR NATION'S NEW AIR TRAFFIC CONTROL SYSTEM.
THE CROWD IS SILENT.
S. Adams E-mail: SCOTTADAMS@AOL.COM
SUDDENLY THE GIFTED PROGRAMMER EMPLOYS A RARELY SEEN STRATEGY OF "CODE REUSE."
THE CROWD GOES WILD.
1/31/96 © 1998 United Feature Syndicate, Inc.(NYC)
SO YOU USED CODE FROM THE PAYROLL SYSTEM?
HERE'S A TIP: DON'T FLY ON PAY DAY.

References

- *Software Engineering*; Ian Sommerville; Addison Wesley; 7th edition ; 2004

- Ian Sommerville; *Critical Systems Engineering* lesson; 2013



Systems Engineering Technical Short Course

(Session 5: System Integration and Test,

AKA: Verification, Validation and Test - VVT)

Appendix L:
Security Engineering,
Evaluation and Assurance

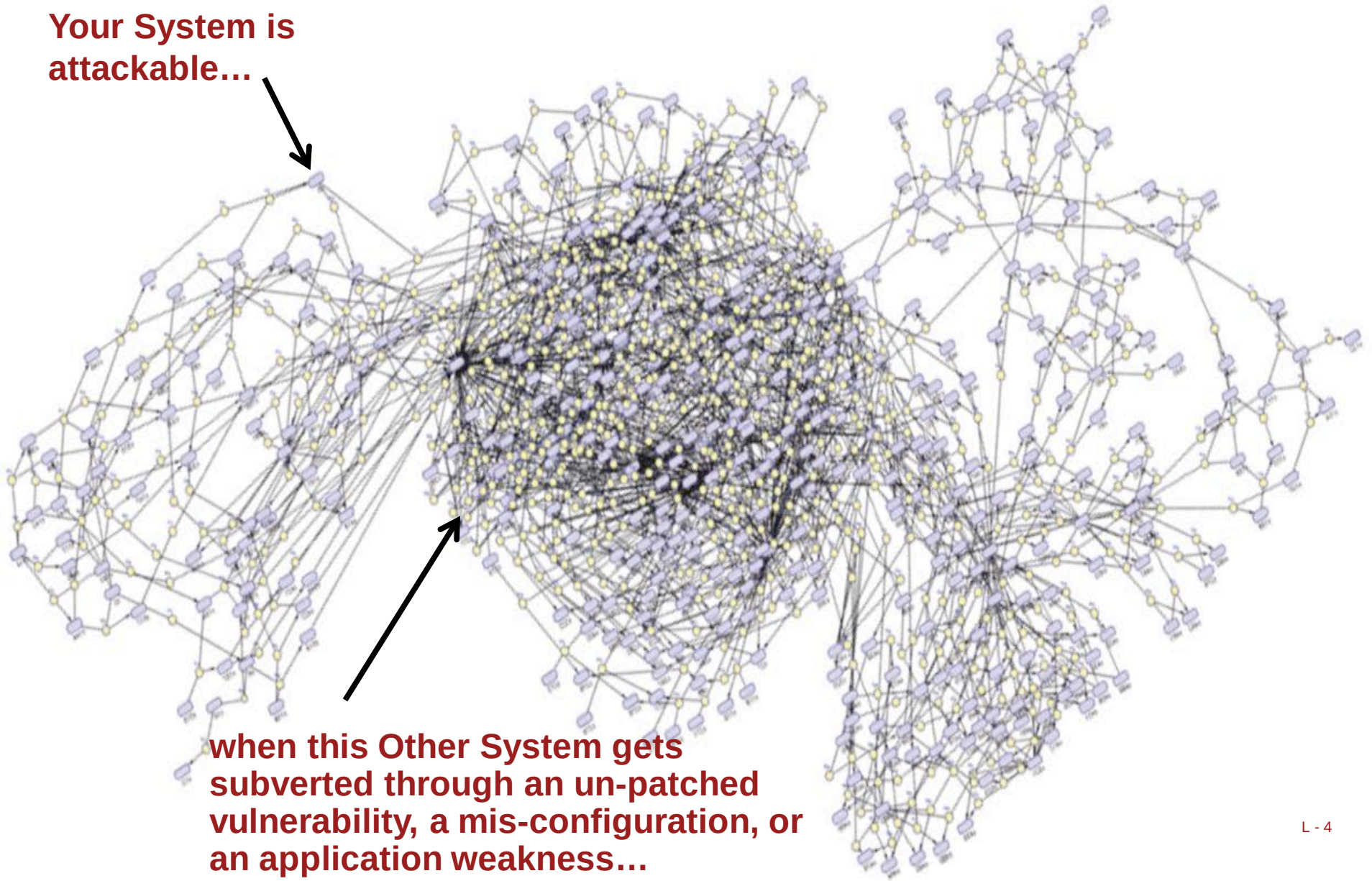
If it’s provably secure, it probably isn’t.

LARS KNUDSEN
Deputy Director
Department of Mathematics
Technical University of Denmark

A Sad State of Affairs

- Terrorism
- Cyber-crime
- Home break-in's
- Just about all complex systems require some level of security
- Determining the right mix can be tough
- A program could spend ALL of its budget on just security
- A mix of conflicting rules, regulations and standards make the problem even more complex but…one thing is certain … once you build it you need to test it!

Everything is Connected!



Systems Security

- Malicious behavior can include
  - Fraud/theft – unauthorized access to money, goods or services
  - Vandalism – causing damage for personal reasons (frustration, envy, revenge, curiosity, self esteem, peer recognition, . . .)
  - Terrorism – causing damage, disruption and fear to intimidate
  - Warfare – damaging military assets to overthrow a government
  - Espionage – stealing information to gain competitive advantage
  - Sabotage – causing damage to gain competitive advantage
  - "Spam" – unsolicited marketing wasting time/resources
  - Illegal content – child pornography, . . .

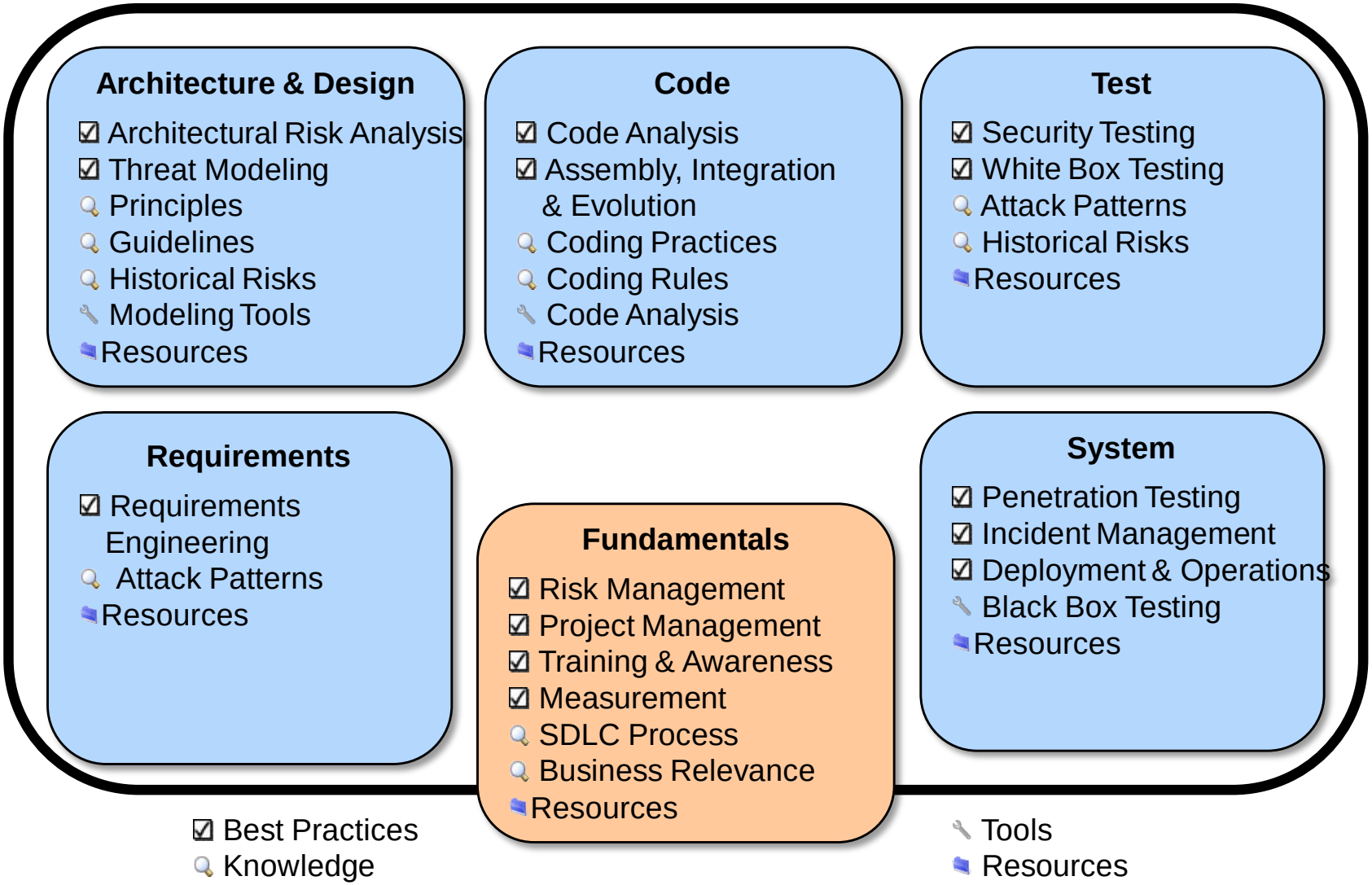
- Security vs. safety engineering
  - Focus on intentional rather than accidental behavior
  - Assumes presence of intelligent adversary

Basic Security Concepts

- Confidentiality: Disclosure of information to only intended parties
- Integrity: Determine whether the information is correct or not
- Authentication: Access to authorized persons
- Authorization: The process of determining that a requester is allowed to receive a service or perform an operation
- Availability: Assuring information and communications services will be ready for use when expected
- Non-repudiation: A measure intended to prevent the later denial that an action happened, or a communication took place, etc.



Security Life Cycle Activities



Why Security Test?

- Any piece of software containing a security flaw can make a secure environment vulnerable to an attack

- Increasing exploits of vulnerabilities in systems show that there are growing needs to develop more secure software



Security Engineering

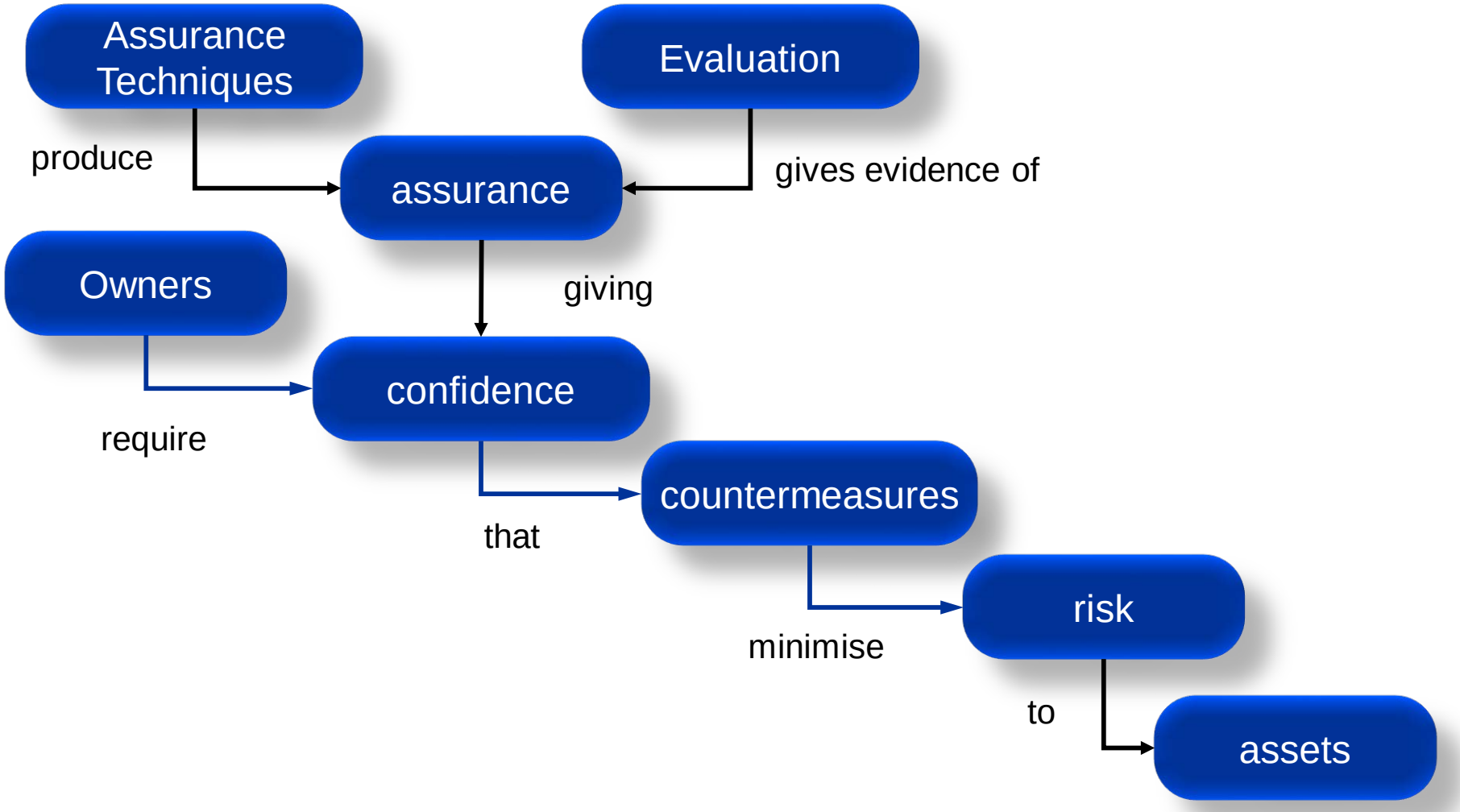
- Security engineering is about building systems that remain dependable in the face of malice, error, or mischance
- As a discipline, it focuses on tools, processes, and methods needed to design, implement, and test complete systems
- Also focuses on adaptating existing systems as their environments evolve
- Security engineering requires cross-disciplinary expertise
  - Cryptography, computer security, hardware tamper-resistance, knowledge of applied psychology, organizational/audit methods and the law
- System engineering skills, from business process analysis through software engineering to T&E, are also important; but they are not sufficient, as they deal only with error and mischance rather than malice

System Evaluation & Assurance

- Assurance – will the system work as designed
- Evaluation – the methodology use to convince others the system works
- Assurance is hard!
  - How do you define the system?
  - How do you deal with people who protect the wrong thing, because their model of the requirements is out-of-date or plain wrong?
  - How do you allow for human failures?
- Evaluation is even harder!
  - How do you prove to that the system does work?



Assurance / Evaluation



Assurance

- A working definition "our estimate of the likelihood that a system will not fail in some particular way.*"

- This can be based on a number of factors, such as:
  - The process used to develop the system
  - The identity of the person or team who developed it
  - Particular technical assessments, such as the use of formal methods or the deliberate introduction of a number of bugs to see how many of them are caught by the testing team
  - Experience – which ultimately depends on having a model of how reliability grows (or decays) over time as a system is subjected to testing, use, and maintenance

* Ross Anderson; *Security Engineering*; Second Edition, Wiley, 2008

Security Testing

- Testing centers around a 'White Box" approach looking at source code, hardware vulnerabilities and documentation

- Architectural flaws
  - Does the system use guessable or too-persistent session identifiers?
  - Is there a way to inject code?
  - Is the security policy coherent, or are there gaps in between the assumptions?

- Implementation flaws
  - Are there stack and integer overflows?

- Less common flaws
  - Weak cypto keys or keys set to a constant value
  - Poor random number generators
  - Weakness in trust assumptions between components

Security Testing Techniques

- Vulnerability Scanning
- Penetration testing
- Ethical Hacking
- Security Risk Assessment
- Security Auditing
- Password cracking
- Port Scanning and Service Mapping
- Firewall Rule Testing
- SQL Injection
- Operating System (OS) Hardening
- IP Spoofing
- Packet Sniffing
- Virtual Private Network Testing
- Social Engineering
- Network Scanning
- File Integrity Testing
- Cross Side Scripting
- Parameter Manipulation
- Denial of Service Testing
- Command Injection
- War Dialing
- Wireless LAN Testing
- Buffer Overflow Testing
- Format String Testing
- Random Data Testing
- Random Mutation Testing
- Session Hijacking
- Phishing
- URL Manipulation

Vulnerability Scanning

- Involves scanning application for all known vulnerabilities
- Scanner is designed to assess computers, computer systems, networks or applications for weaknesses
- Scanning can also be done on code during software development
- Generally done through various vulnerability scanning software
  - Nessus, Klocwork, Security Auditor's Research Assistant (SARA), Tripwire



Penetration Testing

- Practice of simulating real-world attacks by discovering and exploiting software, system, and network vulnerabilities
- Normally performed by a specialized team of experts
- Penetration Testing Motivation
  - Allow an organization to assess its security posture
  - Allow vulnerabilities to be addressed before they are exploited
  - Marketing tool
  - Can be used to promote awareness amongst non-technical executives
  - Can be used to test computer emergency response teams and Intrusion Detection Systems



Ethical Hacking

- Hacking performed to help a company or individual identify potential threats on the computer or network
- An ethical hacker attempts to hack their way past the system security, finding any weak points in the security that could be exploited by other hackers
- The organization uses what the ethical hacker finds to improve the system security, in an effort to minimize, if not eliminate, any potential hacker attacks
- In order for hacking to be deemed ethical, the hacker must obey the below rules.
  - Have permission to probe the network and attempt to identify potential security risks
  - Respect the individual's or company's privacy and only go looking for security issues
  - Report all security vulnerabilities you detect to the company, not leaving anything open for you or someone else to come in at a later time
  - Let the software developer or hardware manufacturer know of any security vulnerabilities you locate in their software or hardware if not already known by the company

Security Risk Assessment



- Risk assessment looking at risks that depend on the loss and the possibility/probability of data/information loss occurrence
- Can be carried out in the form of interviews, discussions, surveys and analysis
- Quantitative Risk Analysis
  - Two fundamental elements; the probability of an event occurring and the likely loss should it occur
  - Calculate the Annual Loss Expectancy (ALE) or the Estimated Annual Cost (EAC)
  - Rank events in order of risk and make appropriate decisions
  - But…data may be unreliable or inaccurate and probability can rarely be precise
- Qualitative Risk Analysis
  - Widely used
  - Assesses threats, vulnerabilities and controls

Security Auditing

- Can be done manually or automatically
- Manual
  - A manual or systematic measurable technical assessment of a system or application.
  - Include interviewing staff, performing security vulnerability scans, reviewing application and operating system access controls, and analyzing physical access to the systems
- Automated
  - Include system generated audit reports or using software to monitor and report changes to files and settings on a system
  - Systems can include personal computers, servers, mainframes, network routers, switches
- To be effective needs to be a continuous process

Password Cracking

- The process of recovering passwords from data that have been stored in or transmitted by a computer system
- Can use Brute Force and Dictionary Attacks
- ID weak passwords
- Tools readily available
  - Cain and Abel; John the Ripper, Hashcat, Hydra, DaveGrohl … to name a few
- As testers we look at whether a system uses strong passwords and has a password policy appropriate to that system and organization



Port Scanning and Service Mapping

- Identification and locating of Open Ports
  - Often used by administrators to verify security policies of their networks and by attackers to identify running services on a host with the view to compromise it

- Identification of Running Services



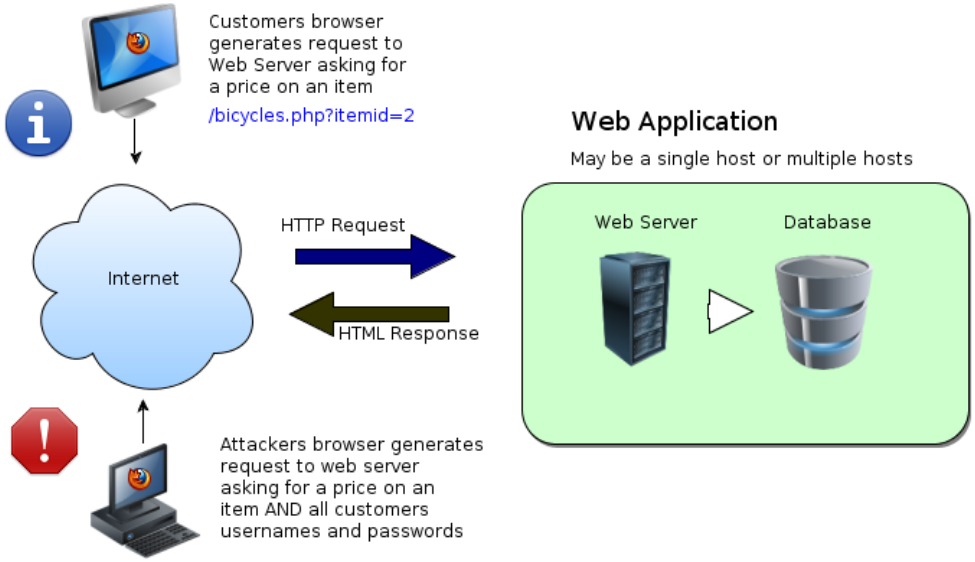
Firewall Rule Testing

- Identify inappropriate or conflicting rules
- Appropriate placement of vulnerable system behind a firewall
- Discovering administrative backdoors or tunnels



SQL Injection

- Code injection technique, used to attack data driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker)
- SQL injection must exploit a security vulnerability in an application's software
- SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database



/bicycles.php?itemid=2 union all select customer.username,customer.password,3,4,5 --

Operating System Hardening

- Process of securing a system by reducing its surface of vulnerability.
- A system has a larger vulnerability surface the more functions it fulfills; in principle a single-function system is more secure than a multipurpose one
- Reducing available vectors of attack typically includes the removal of unnecessary software, unnecessary usernames or logins and the disabling or removal of unnecessary services
  - Configure and apply patches
  - Updating the operating system
  - Disable or restrict unwanted services and ports
  - Manage the log files
  - Protect from malware
  - Protect from internet misuse
  - Implementation of cyber-safe policies



Security Testing Techniques (continued)

- IP Spoofing
  - Creating Internet Protocol (IP) packets with a forged source IP address
- Packing Sniffing
  - Capture and analyze all of the Network traffic
- Virtual Private Network Testing
  - Penetration testing
- Social Engineering
  - Psychological manipulation of people
  - Divulging confidential information
- Network Scanning
  - ID active hosts
  - Collect IP addresses, OS details, system architecture and running services, network user and group names, routing tables and Simple Network Management Protocol (SNMP) data



Security Testing Techniques (continued)

- File Integrity Testing
  - Verifying file integrity against corruption using a checksum
- Cross Side Scripting
  - Injecting malicious client side script into Web pages
- Parameter Manipulation
  - Cookies, Form Field, URL or HTTP Header manipulation
- Denial of Service Testing
  - Flooding a target machine with enough traffic to make it incapable of functioning
- Command Injection
  - Inject and execute commands by the attacker
  - Execute system level commands through a vulnerable application



Security Testing Techniques (continued)

- War Dialing
  - Using a modem to dial a list of phone numbers
  - Searching for computers and fax machines
- Wireless LAN Testing
  - Searching for existing WLAN and logging wireless access points
- Buffer Overflow Testing
  - Overwriting of memory fragments of the process, buffer or char type
- Format String Testing
  - Supplying format type specifiers in the application input
- Random Data Testing
  - Random data inputs by a program
  - Encoded random data included as parameters
  - Crashing built-in code assertions



Security Testing Techniques (continued)



- Random Mutation Testing
  - Bit flipping of known legitimate data
  - Byte stream sliding within known limits of legitimate data
- Session Hacking
  - Exploitation of a valid computer session
  - Exploitation of the Web session control mechanism
  - Gain unauthorized access to the Web Server
- Phishing
  - Masquerading as a trustworthy entity in an electronic communication
  - Acquiring usernames, passwords and credit card details
- URL Manipulation
  - Make a web server deliver inaccessible web pages
  - URL rewriting

Testing Models

- Zero-knowledge attack
  - performed by testers who have no real information about the target environment
  - designed to provide the most realistic penetration test possible

- Full-knowledge attack
  - performed with the tester having as much information about the target environment as possible
  - designed to simulate an attacker who has intimate knowledge of the target organization's systems – such as a real employee

Potential Threats

- Script Kiddie
  - Has limited or no knowledge of how computer systems work
  - Rely on pre-written exploits and vulnerability scanners to find and exploit vulnerabilities
- Master Cracker
  - Has intimate knowledge of IT technology and system code
  - Find original vulnerabilities, write customized exploits and spend much of their time learning and finding flaws in new technology
- Malicious Insider
  - Doesn't necessarily know much about IT systems but does know a lot about YOUR system
  - This enables them to attack a system at its most vulnerable point
- Naive employee
  - Generally damages IT systems through an inability to correctly operate even the most simplest applications





Things To Be Considered

- What will be done with information after test has been performed?
- Methods used to limit unintentional damage to networks and/or systems
- Internal vs. External testing team
- Attack profile of potential threat sources



Evaluation

- The process of assembling evidence that a system meets, or fails to meet, a prescribed assurance target
  - Needed to reassure principals who will rely on system
- Tension can arise when the party who implements protections is different than those of the party the relies on those protections
- Shortfalls
  - "Assurance" and "Evaluation" often are interpreted to apply to only the technical aspects of a system
  - Often ignore system issues like usability, organizational issues and corporate governance
  - Company directors want assurance…
    - Procedures are followed
    - No material errors in accounts
    - Laws are complied with
  - But evaluation schemes often ignore the organizational and human elements of a system



Security Guidelines (some, there are many more!)

- Federal Information Processing Standards (FIPS)
  (http://csrc.nist.gov/publications/PubsFIPS.html)
  - FIPS Publications were issued by the National Institute of Science and Technology (NIST) as a result of the Information Technology Reform Act of 1996 and the Federal Information Security Management Act of 2002
  - With the passage of the Federal Information Security Management Act of 2002, there is no longer a statutory provision to allow for agencies to waive mandatory FIPS
- Common Criteria for Information Technology Security
  (http://www.commoncriteriaportal.org/)
  - Abbreviated as Common Criteria or CC. An international standard (ISO/IEC 15408) for computer security certification
  - The CC is useful as a guide for the development of products or systems with IT security functions and for the procurement of commercial products and systems with such functions
  - The CC concentrates on threats to that information arising from human activities, whether malicious or otherwise, but may be applicable to some non- human threats as well

Resources (partial list)

- Common Weakness Enumeration (CWE) - http://cwe.mitre.org/
- Open Web Application Security Project (OWASP) – https://www.owasp.org/
- Web Application Security Consortium (WASC) – http://www.webappsec.org/
- National Vulnerability Database (NVD) – http://nvd.nist.gov/
- Common Vulnerabilities and Exposures (CVE) - http://cve.mitre.org/
- SEI CERT - http://www.cert.org/
- DHS SW Assurance Common Body of Knowledge - https://buildsecurityin.us-cert.gov/bsi/dhs.html
- SANS – Training & Certification - http://www.sans.org/

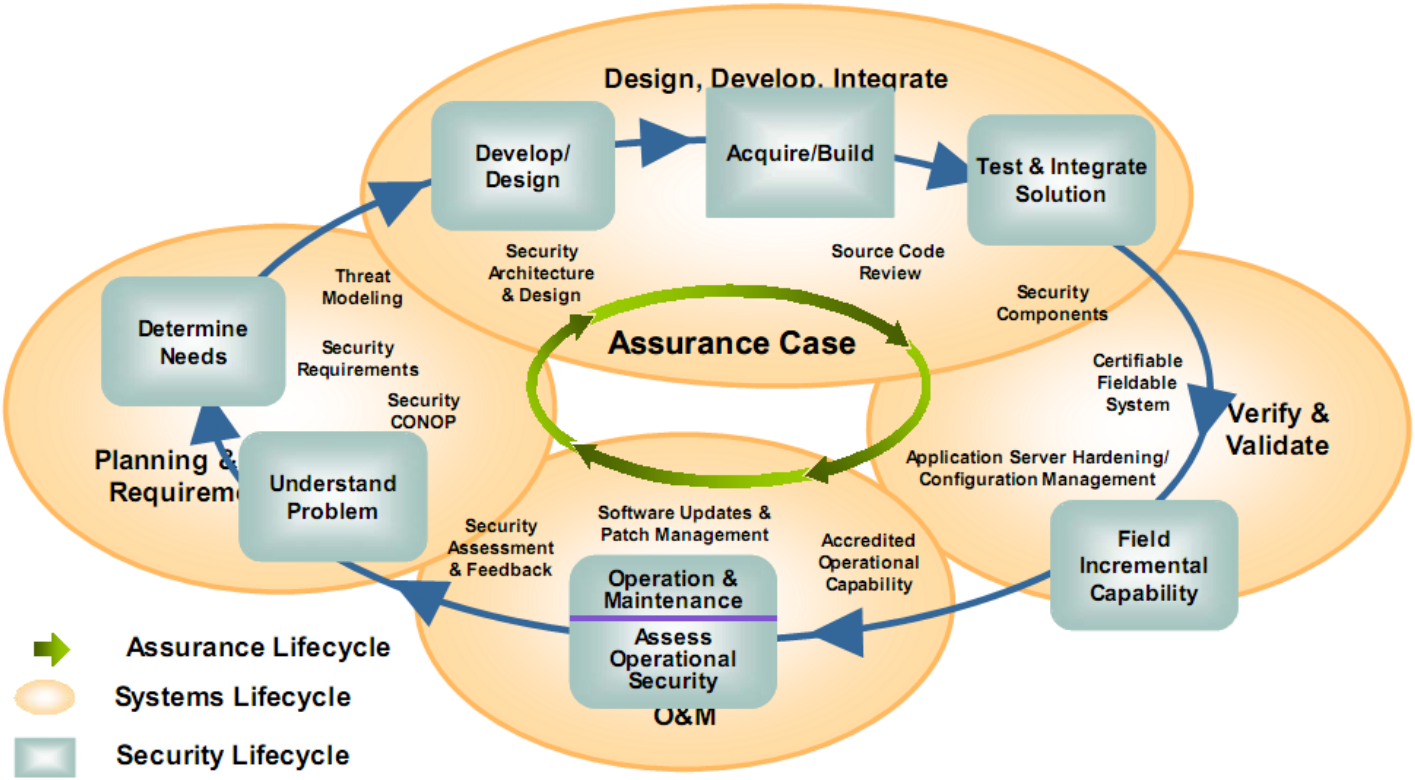






ISO/IEC 21827

- Systems Security Engineering Capability Maturity Model® (SSE-CMM®)
- A content-independent standard, which facilitates implementation of a good process for ANY set of security practices



Model Process Areas

Security Engineering Process Areas	# of Base Practices
Administer Security Controls	4
Assess Impact	6
Assess Security Risk	6
Assess Threat	6
Assess Vulnerability	5
Build Assurance Argument	5
Coordinate Security	4
Monitor Security Posture	7
Provide Security Input	6
Specify Security Needs	7
Verify and Validate Security	5

Project and Organizational Process Areas	# of Base Practices
Ensure Quality	8
Manage Configuration	5
Manage Project Risk	6
Monitor and Control Technical Effort	6
Plan Technical Effort	10
Define Organization's Security Engineering Process	4
Improve Organization's Security Engineering Process	4
Manage Product Line Evolution	5
Manage Systems Engineering Support Environment	7
Provide Ongoing Skills and Knowledge	8
Coordinate with Suppliers	5

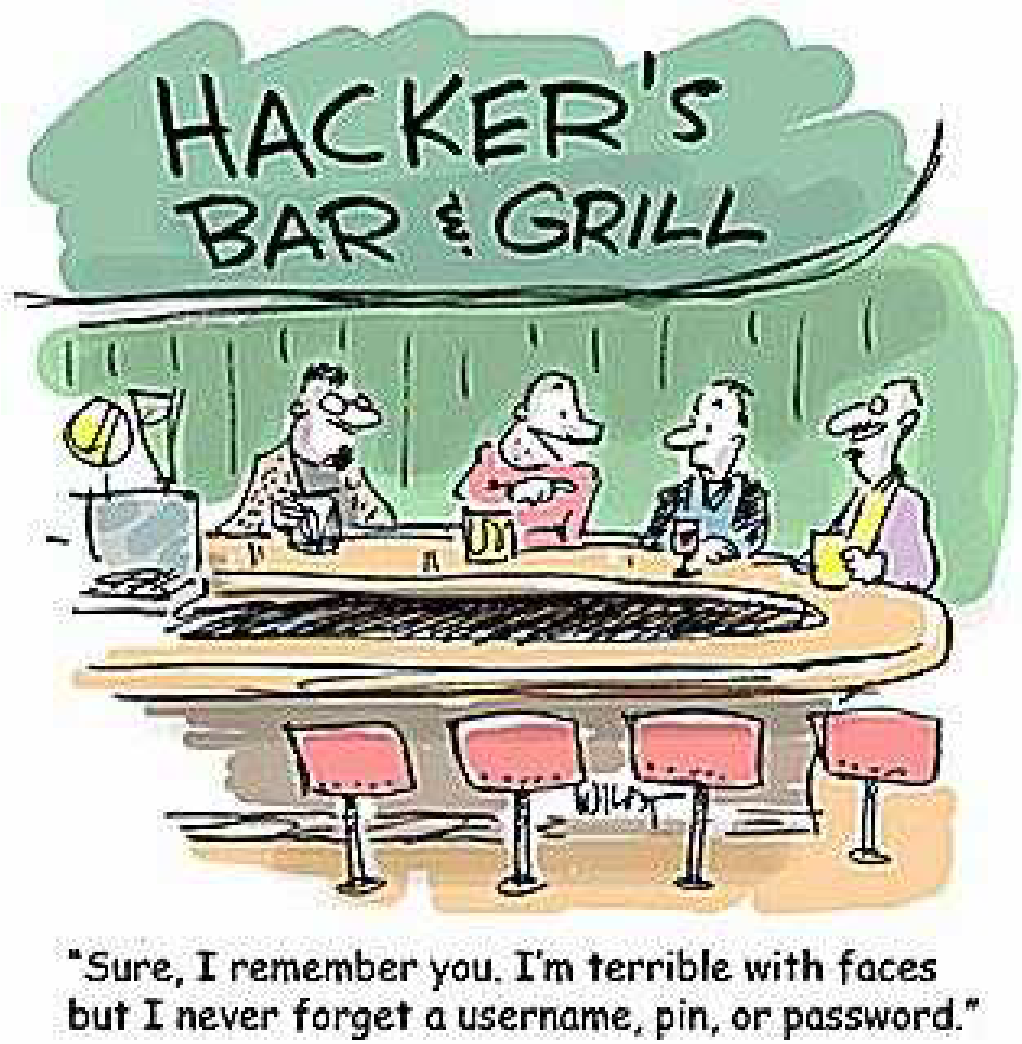
ISO/IEC 21827 Appraisal

- Evidence based appraisal method
  - Provides “As Is” picture
  - Tailored to the organization
  - Supports business/ mission objectives
  - Identifies areas for Improvement

Measuring Assurance

- ISO/IEC 21827 compliant processes can be measured and managed to
  - Tie security practice performance to business and security goals
  - Quantify compliance with standards
  - Measure effectiveness and efficiency of security implementation
  - Identify data used for measurement

- Measures allow us to…
  - Repeat measurement and provide relevant performance trends over time
  - Identify opportunities for corrective actions and formulate action plans
  - Support security improvement and budget recommendations
  - Produce evidence that substantiates assurance cases


HACKER'S
BAR & GRILL
"Sure, I remember you. I'm terrible with faces
but I never forget a username, pin, or password."

Summary

- Security testing is based on requirements and threats
- It is risk based
- IT is the focus of much security testing but…there are physical aspects that needs to be addressed also
- The hardest part about security is understanding the myriad of rules, regulations and standards
- It is not always about the technical aspects…human behavior, organization, society, etc., play a role in defining the threats and thus the requirements