INCOSE-SD MBSE Tutorial

MBSE/SysML Tutorial

Rick Hefner, Caltech, rhefner@caltech.edu

NoMagic Tool Workshop

Jason Wilson, No Magic, jason.WILSON@3ds.com

Schedule

8:00 am	Registration
9:00 am	MBSE
10:15 am	SysML Overview
11:30 am	MBSE Adoption
12:00 am	Lunch
12:45 pm	NoMagic Tool Workshop
4:00 pm	End

What do you hope to learn today?

Model-Based Systems Engineering (MBSE)

MBSE Principles

SysML

SysML Diagrams

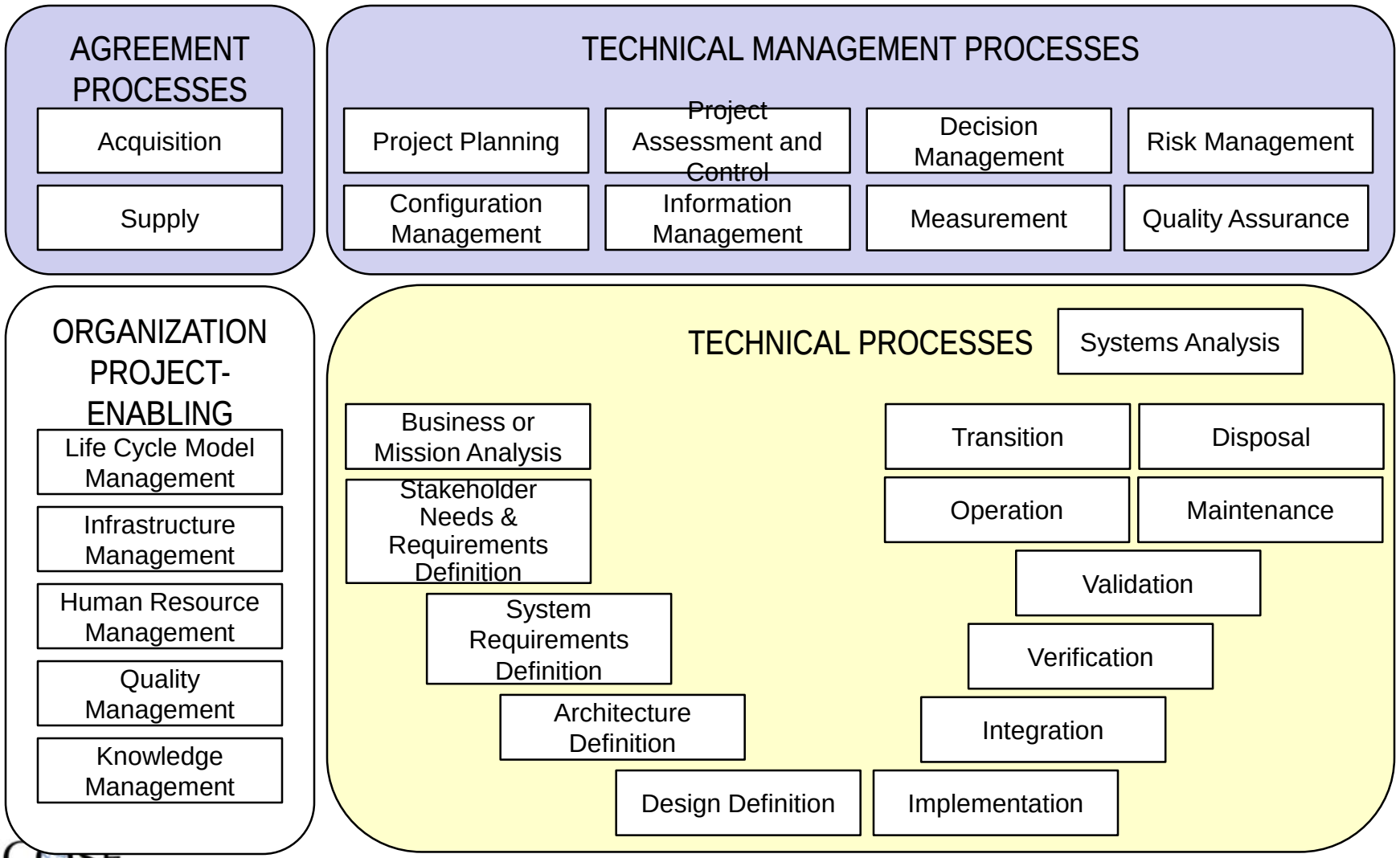
Model-Based Systems Engineering (MBSE)

The formalized application of modeling *to support system requirements, design, analysis, verification and validation activities* beginning in the conceptual design phase and continuing throughout development and later life cycle phases.

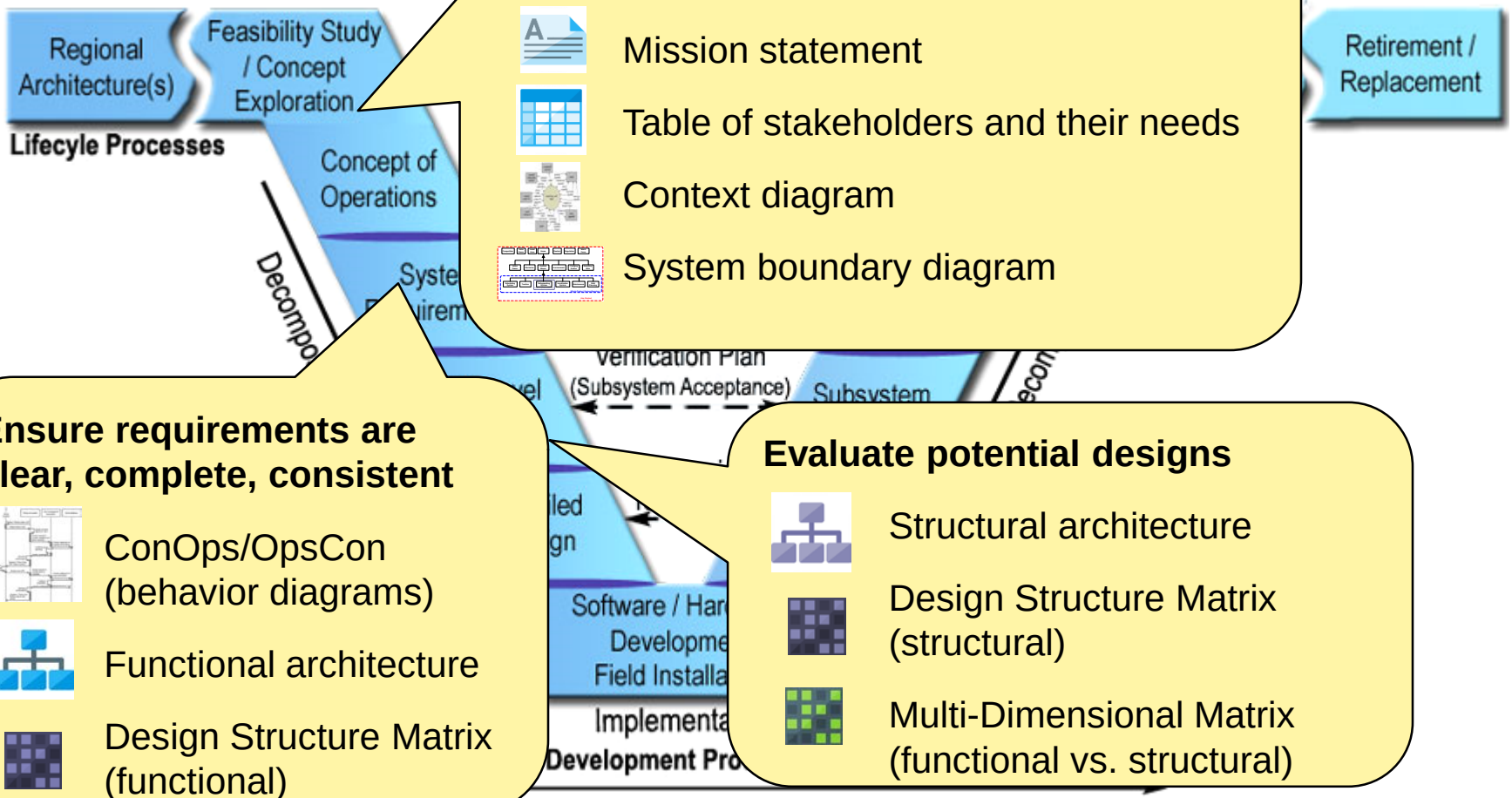
- INCOSE SE Vision 2020 (INCOSE-TP-2004-004-02), Sept 2007

System Engineering Process

INCOSE SE Handbook - ISO/IEC/IEEE 15288:2015



Key SE Representations and Their Purpose



Understand the problem context

- Mission statement
- Table of stakeholders and their needs
- Context diagram
- System boundary diagram

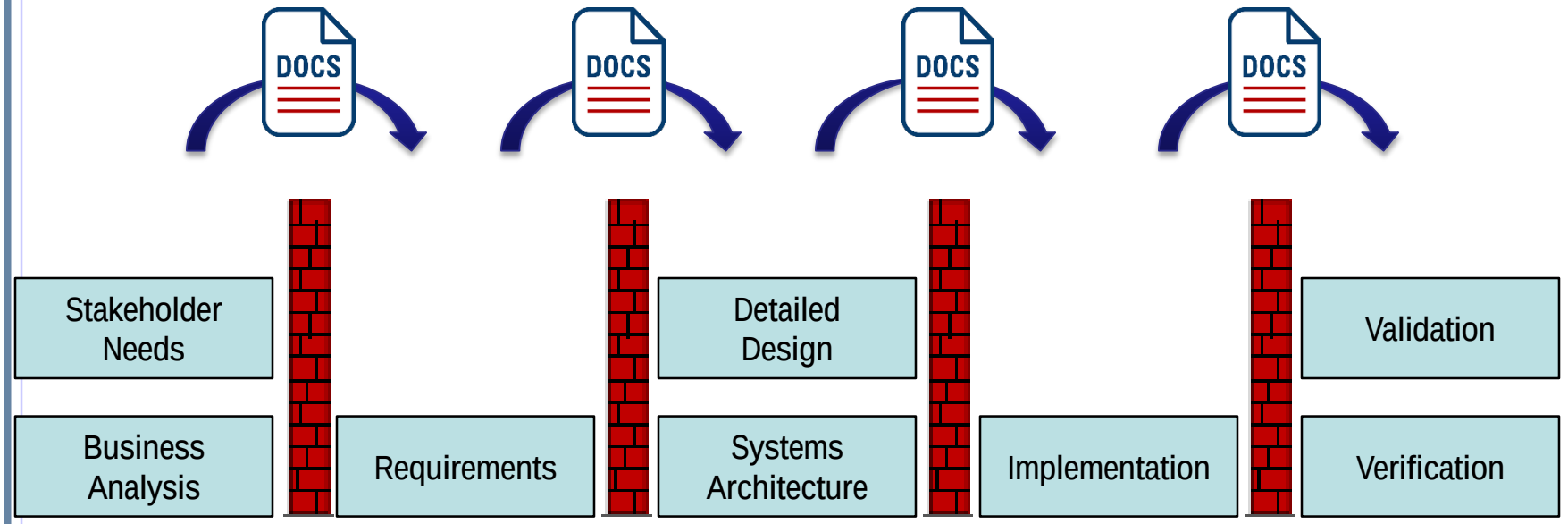
Ensure requirements are clear, complete, consistent

- ConOps/OpsCon (behavior diagrams)
- Functional architecture
- Design Structure Matrix (functional)

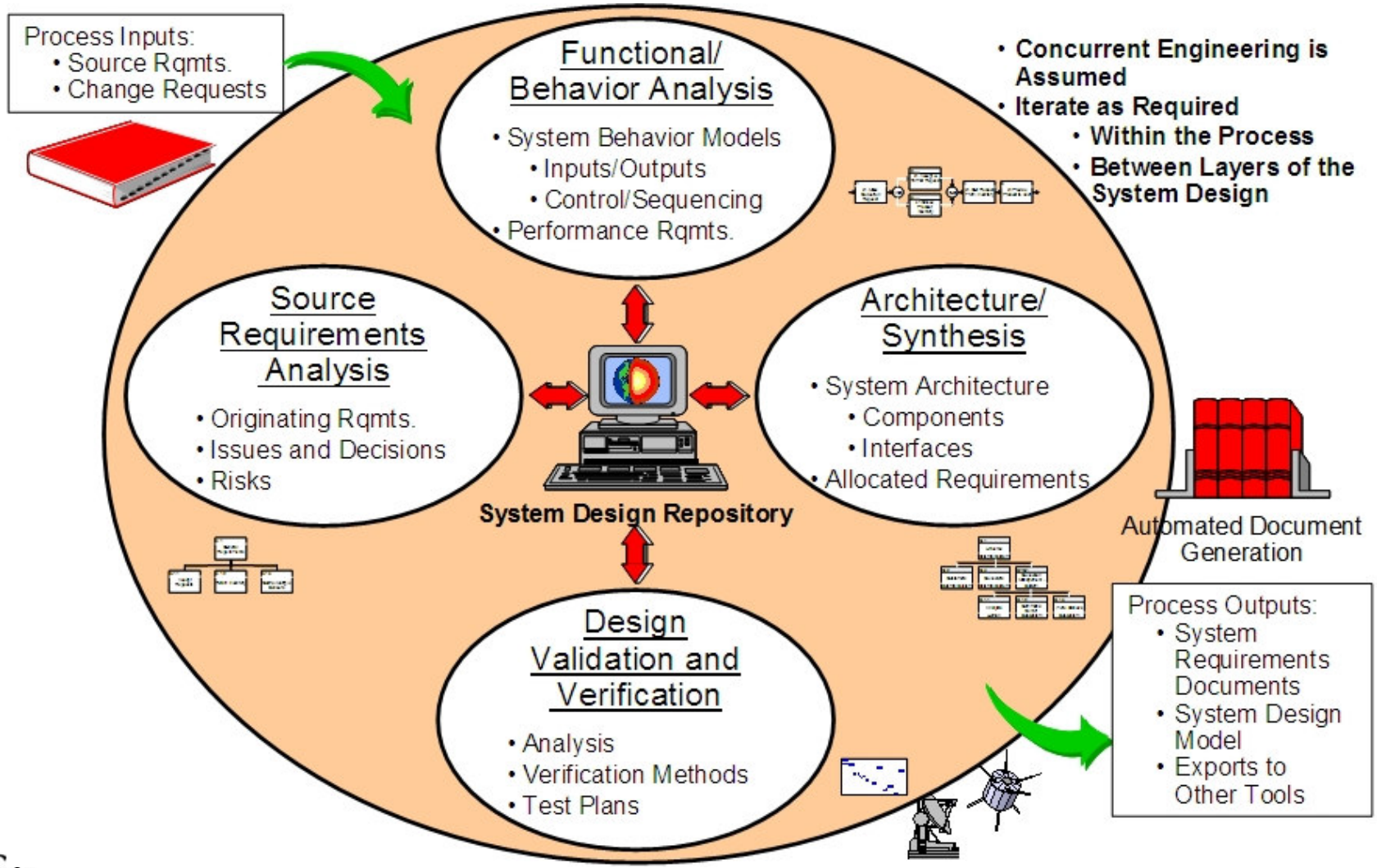
Evaluate potential designs

- Structural architecture
- Design Structure Matrix (structural)
- Multi-Dimensional Matrix (functional vs. structural)

Traditional Systems Engineering

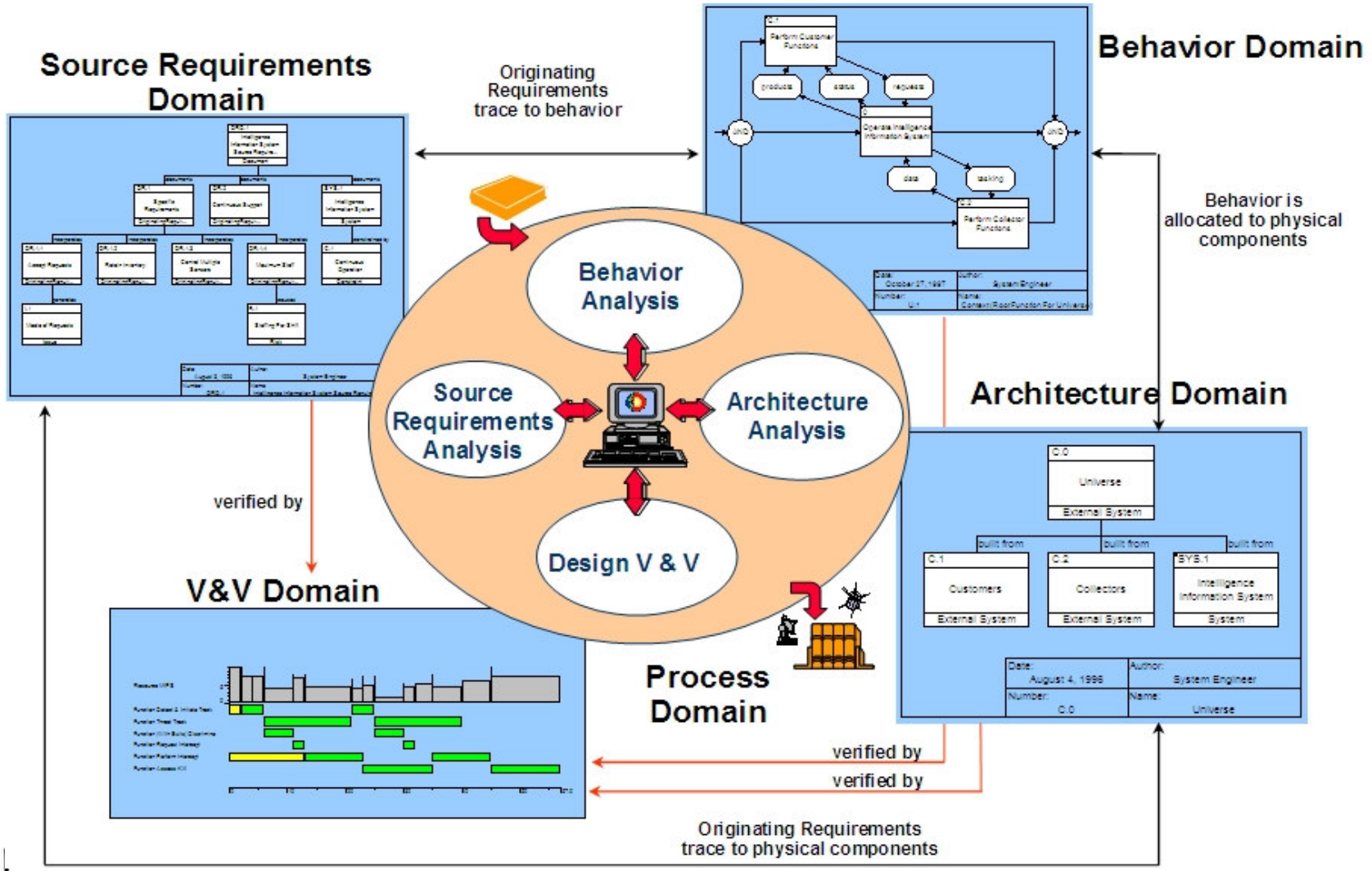


Model-Based Approach



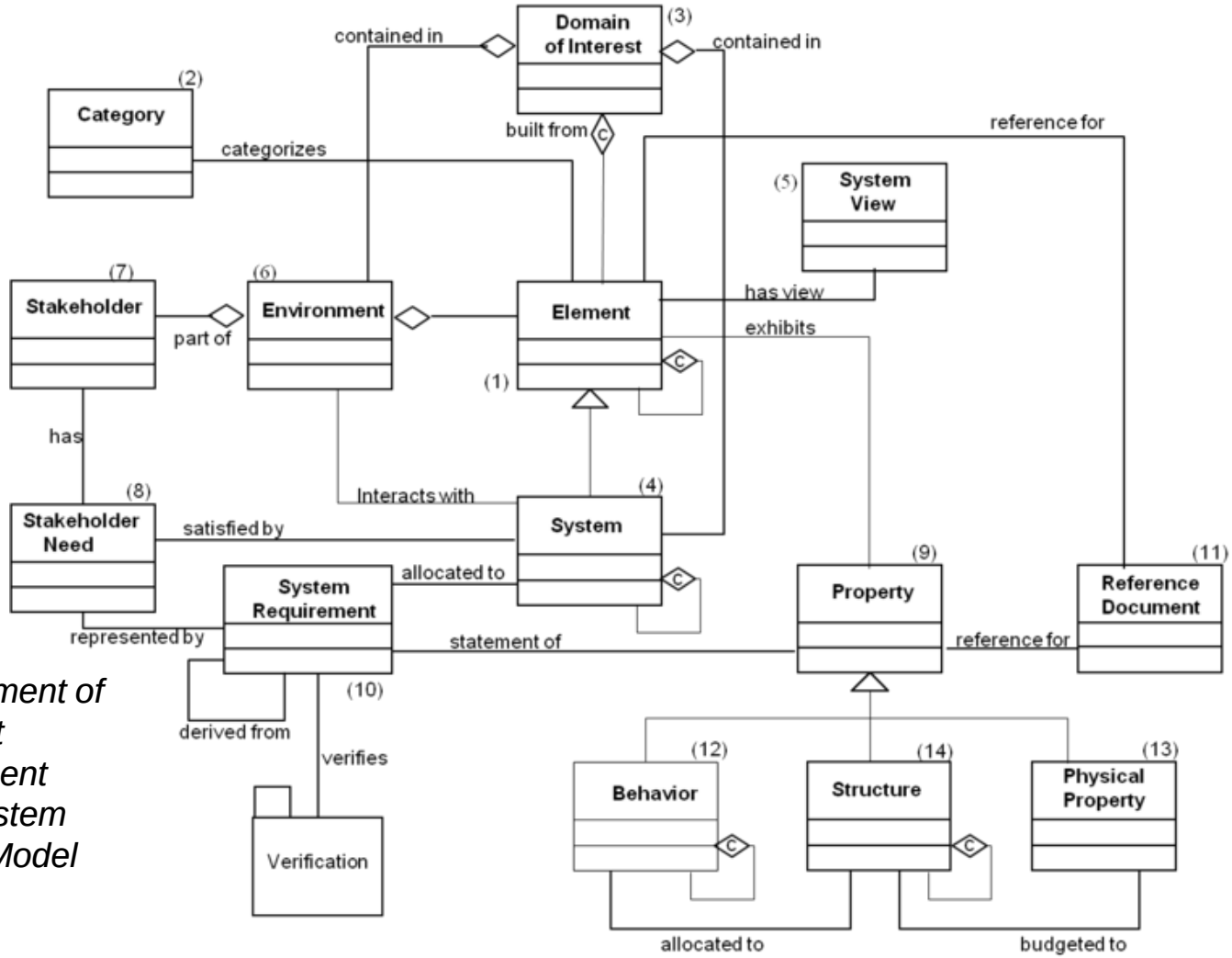
Ref: Vitech

MBSE Domains



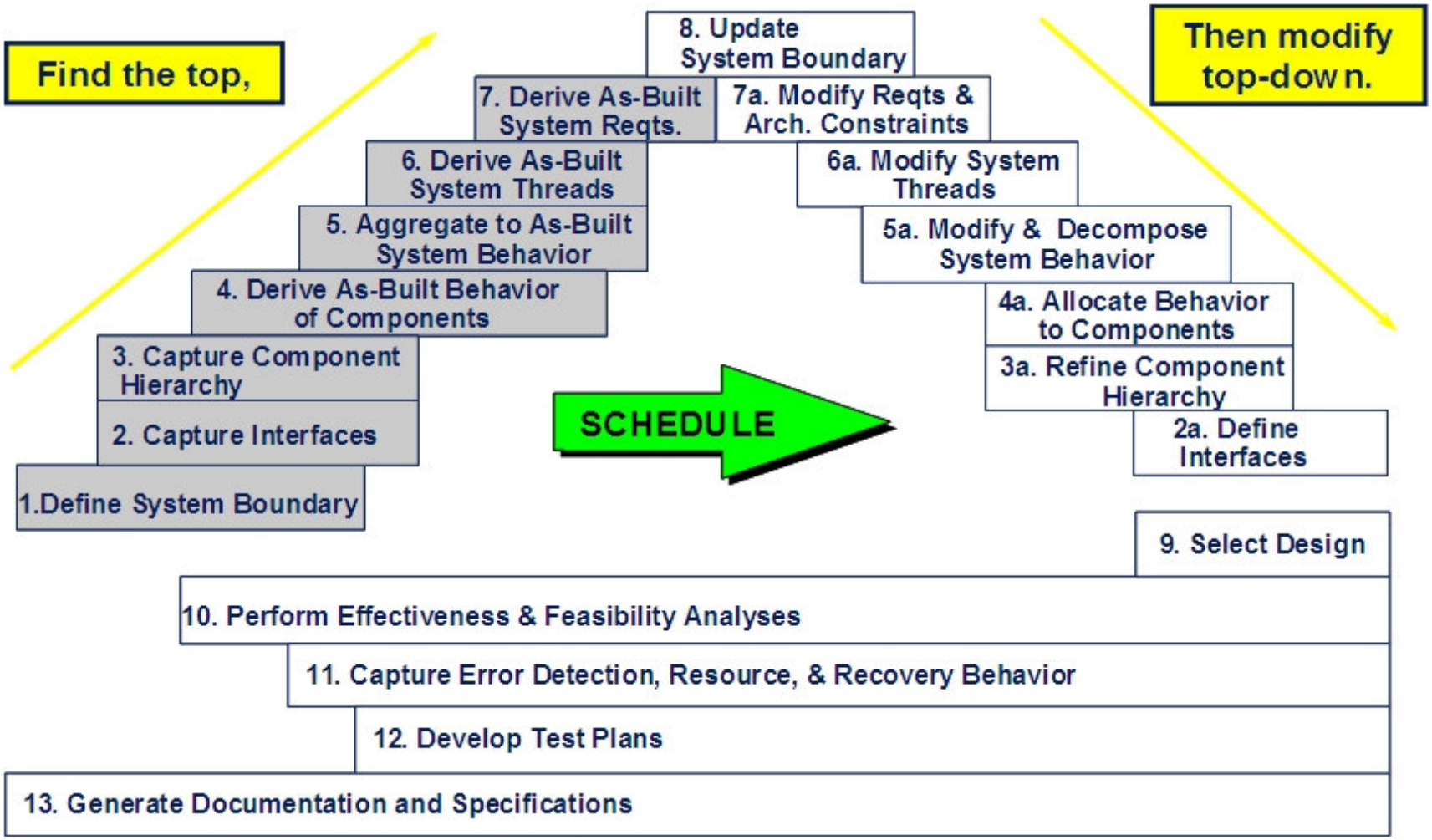
Ref: Vitech

MBSE Model Terminology



Ref: Fragment of the Object Management Group System Concept Model

MBSE Activities



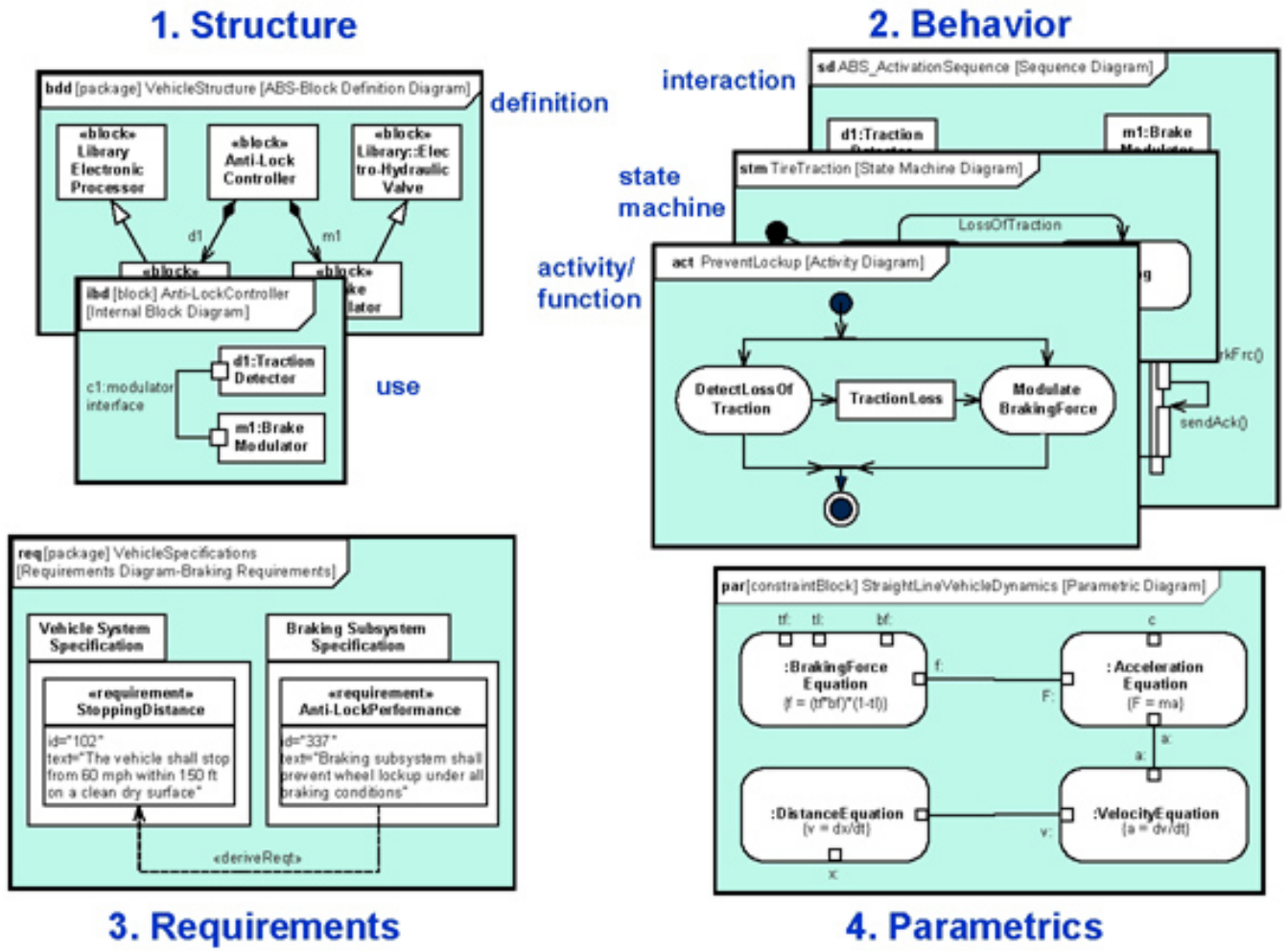
Ref: Vitech

Systems Modeling Language (SysML)

- A general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems
- Provides graphical representations with a *semantic foundation* for modeling system requirements, behavior, structure, and parametrics
- Developed by the Object Management Group (OMG)

Four Pillars of SysML

http://www.omgsysml.org



Note that the Package and Use Case diagrams are not shown in this example, but are respectively part of the structure and behavior pillars

Multiple System Views to Communicate Requirements and Design

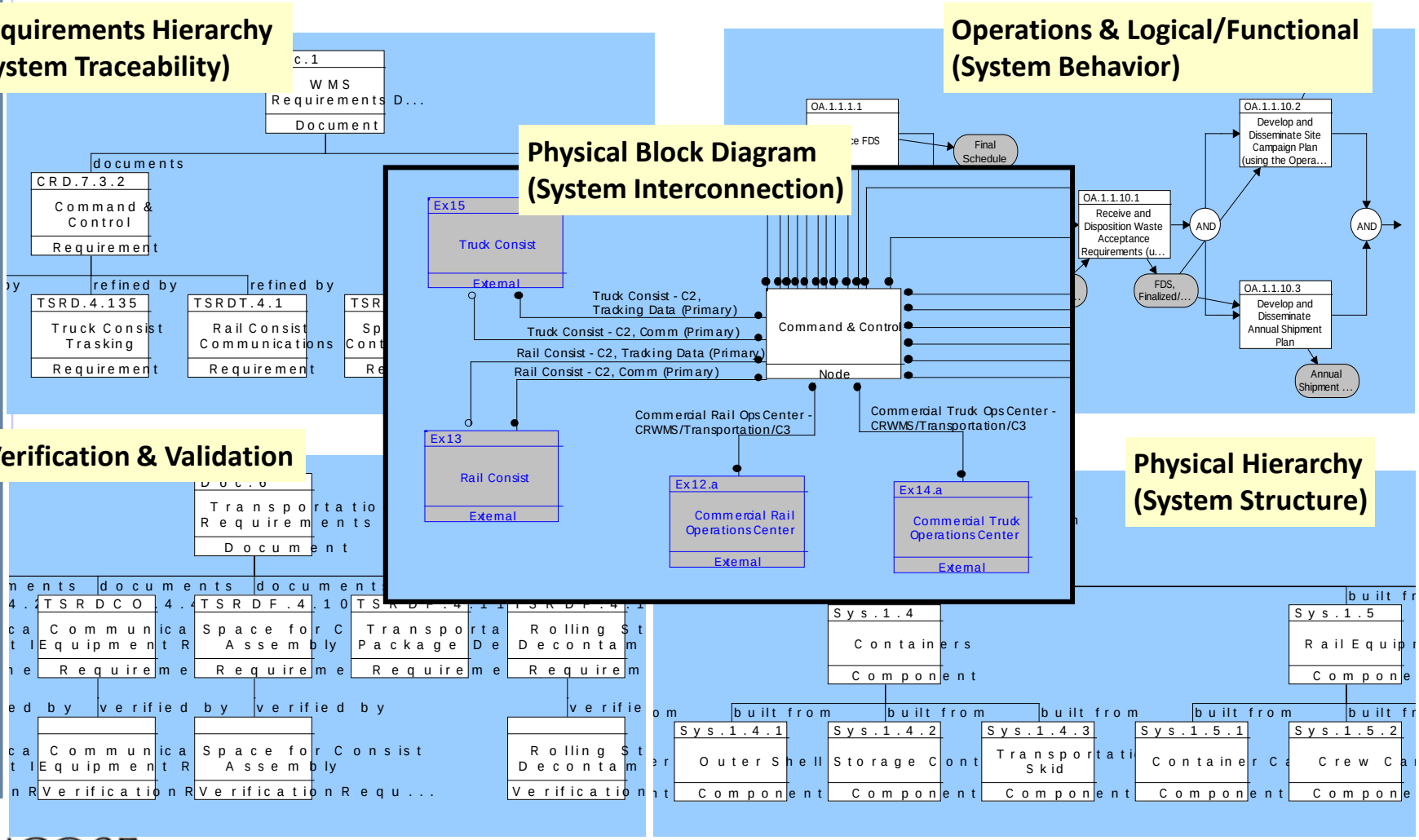
Requirements Hierarchy (System Traceability)

Operations & Logical/Functional (System Behavior)

Physical Block Diagram (System Interconnection)

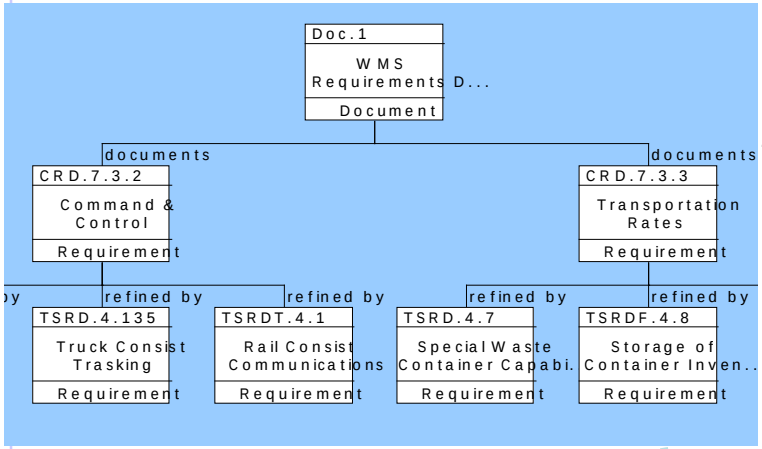
Verification & Validation

Physical Hierarchy (System Structure)

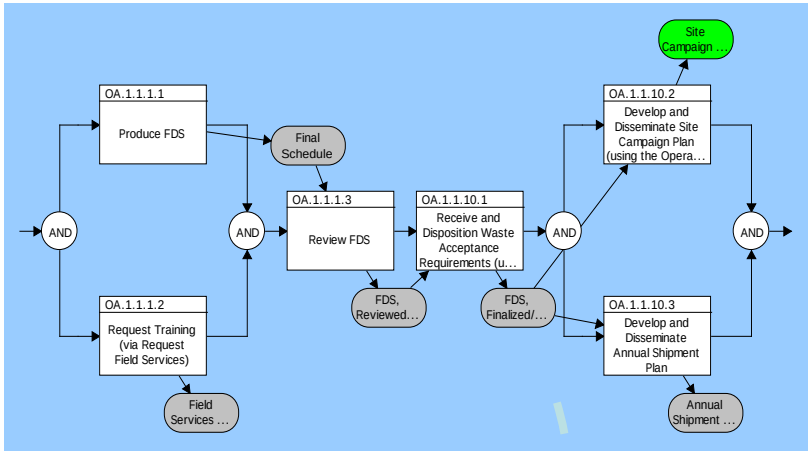


Views produced by CORE

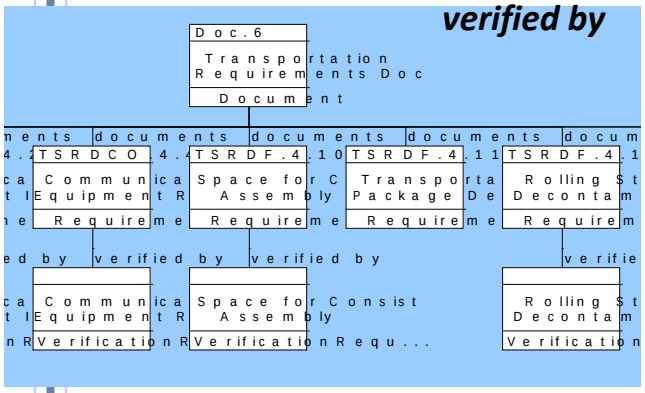
Integrated System Model



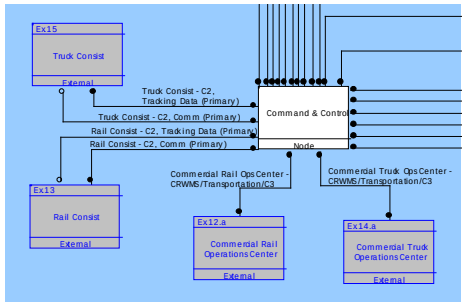
trace to



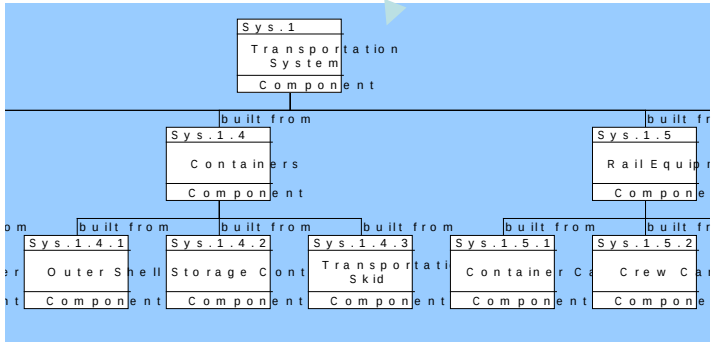
verified by



functional I/O implemented by

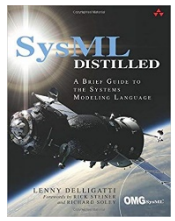


allocated to



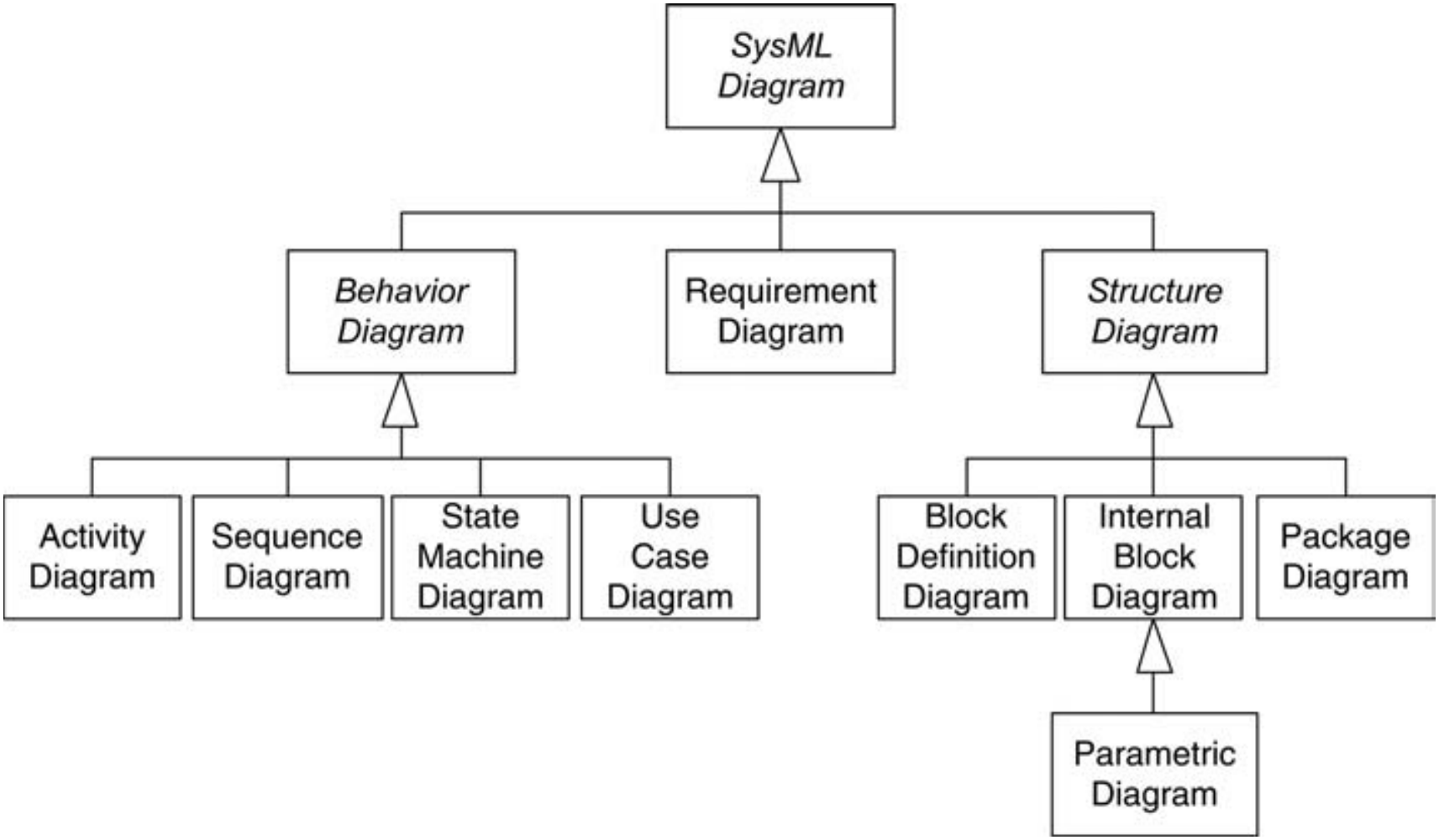
SysML Diagrams

Block Definition Diagram • Internal Block Diagram
Use Case Diagram • Activity Diagram
Sequence Diagrams • State Machine Diagram Parametric Diagram • Package Diagram
Requirement Diagram • Allocations



All diagrams are from SysML Distilled by Lenny Delligatti

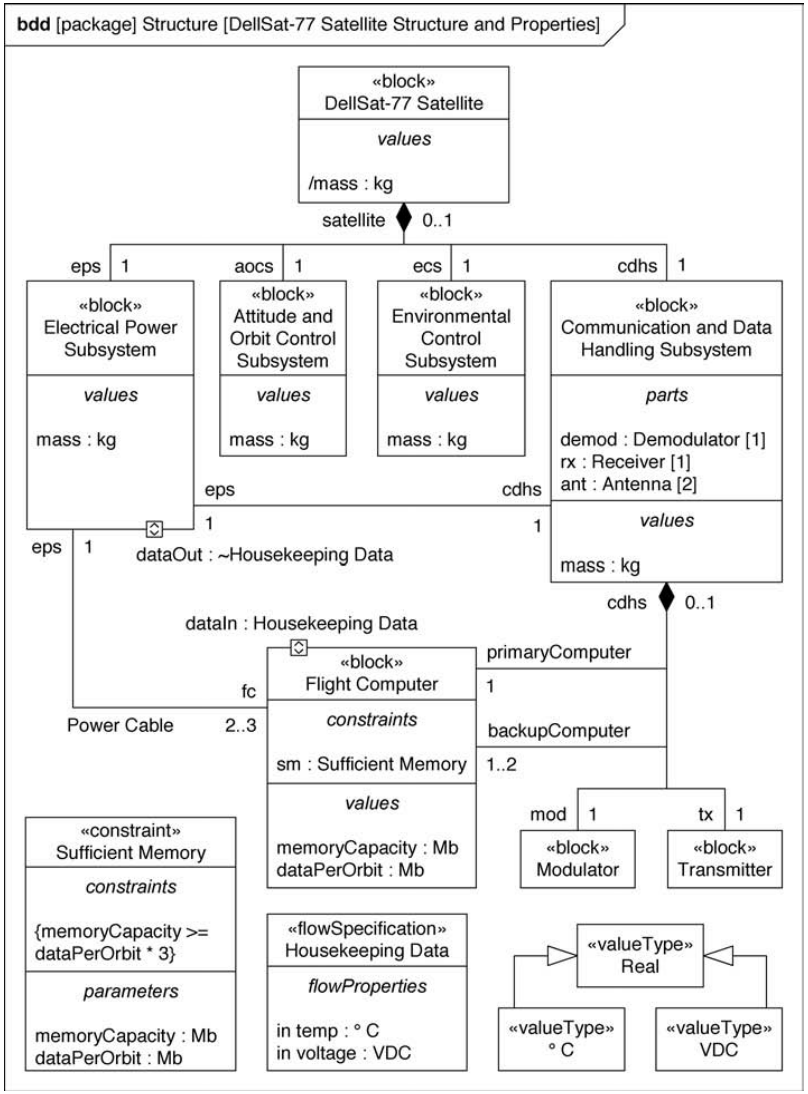
SysML Diagram Taxonomy



Block Definition Diagram

Used to display elements such as blocks and value types (elements that define the types of things that can exist in an operational system) and the relationships between those elements

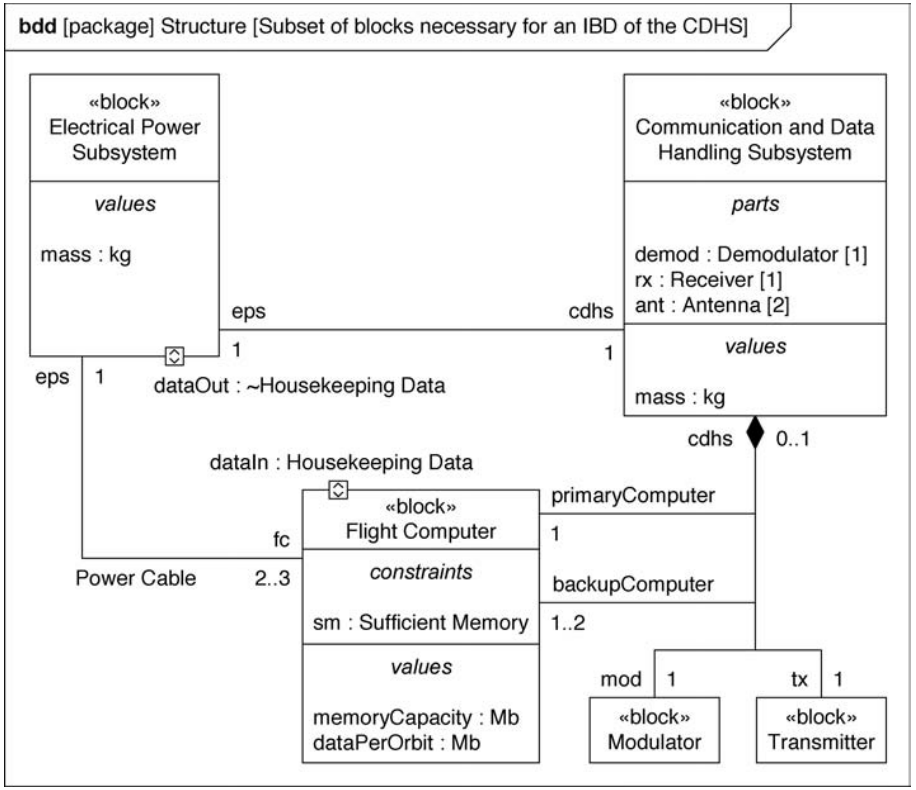
- *Parts*
- *References*
- *Values*
- *Constraints*
- *Operations*
- *Receptions*
- *Standard ports* (in SysML v1.2 and earlier)
- *Flow ports* (in SysML v1.2 and earlier)
- *Full ports* (in SysML v1.3)
- *Proxy ports* (in SysML v1.3)
- *Flow properties* (in SysML v1.3)
- *Structure*



Internal Block Diagram

Used to specify the internal structure of a single block

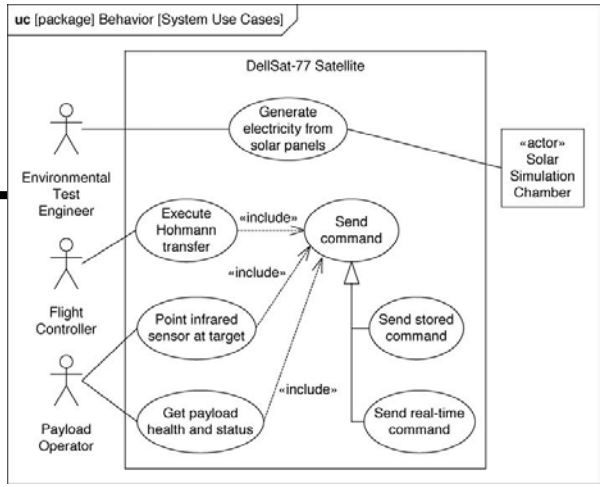
- Shows the connections between the internal parts of a block and the interfaces between them
- A BDD defines the block and its's properties; the IBD displays a valid configuration of that block



Use Case Diagram

Used to convey the scenarios a system performs and the actors that invoke and participate in them

- A black-box view of the services that a system performs in collaboration with its actors



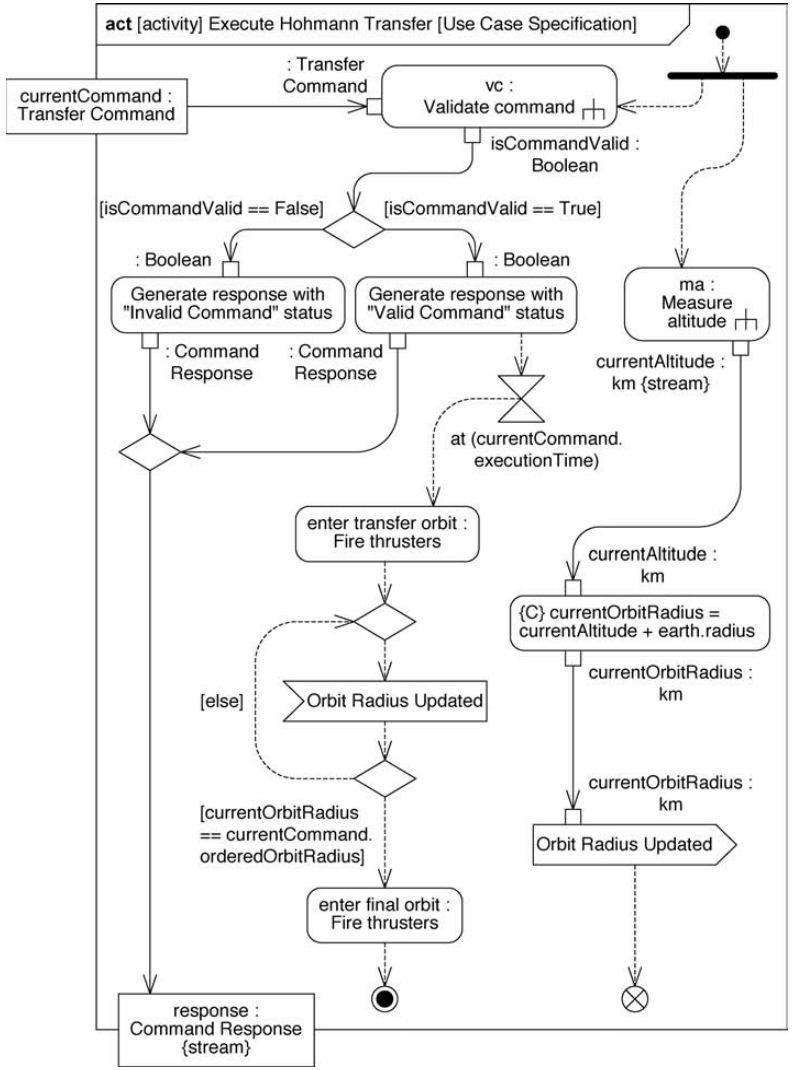
- **Use case name:** a verb phrase
- **Scope:** the entity that owns (provides) the use case (for example, the name of an organization, system, subsystem, or component)
- **Primary actor:** the actor that invokes the use case (the actor whose goal the use case represents)
- **Supporting (secondary) actors:** actors that provide a service to the system (participate in the use case by performing actions)
- **Stakeholder:** someone or something with a vested interest in the behavior of the system

- **Preconditions:** the conditions that must be true for this use case to begin
- **Guarantees (postconditions):** the conditions that must be true at the end of the use case
- **Trigger:** the event that gets the use case started
- **Main success scenario:** the scenario (the sequence of steps) in which nothing goes wrong
- **Extensions (alternative branches):** alternative sequences of steps branching off of the main success scenario
- **Related information:** whatever your project needs for additional information

Activity Diagram

Used to specify a behavior, with a focus on the flow of control and the transformation of inputs into outputs through a sequence of actions

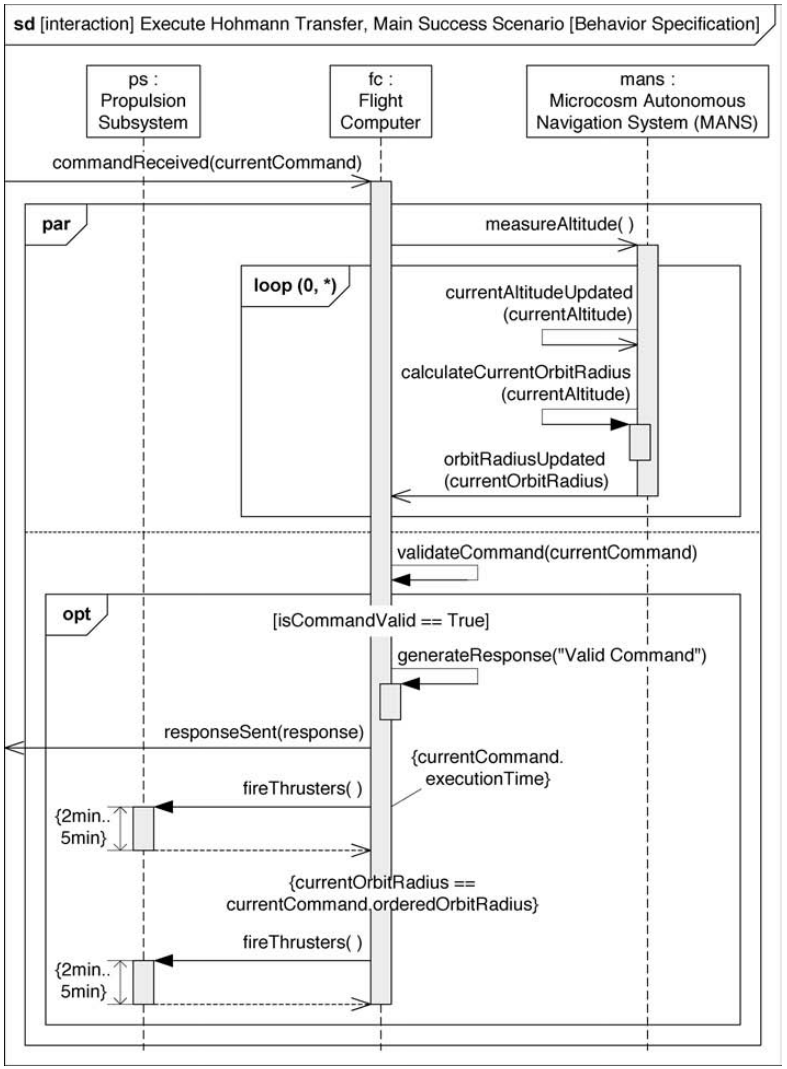
- Commonly used as an analysis tool to understand and express the desired behavior of a system



Sequence Diagram

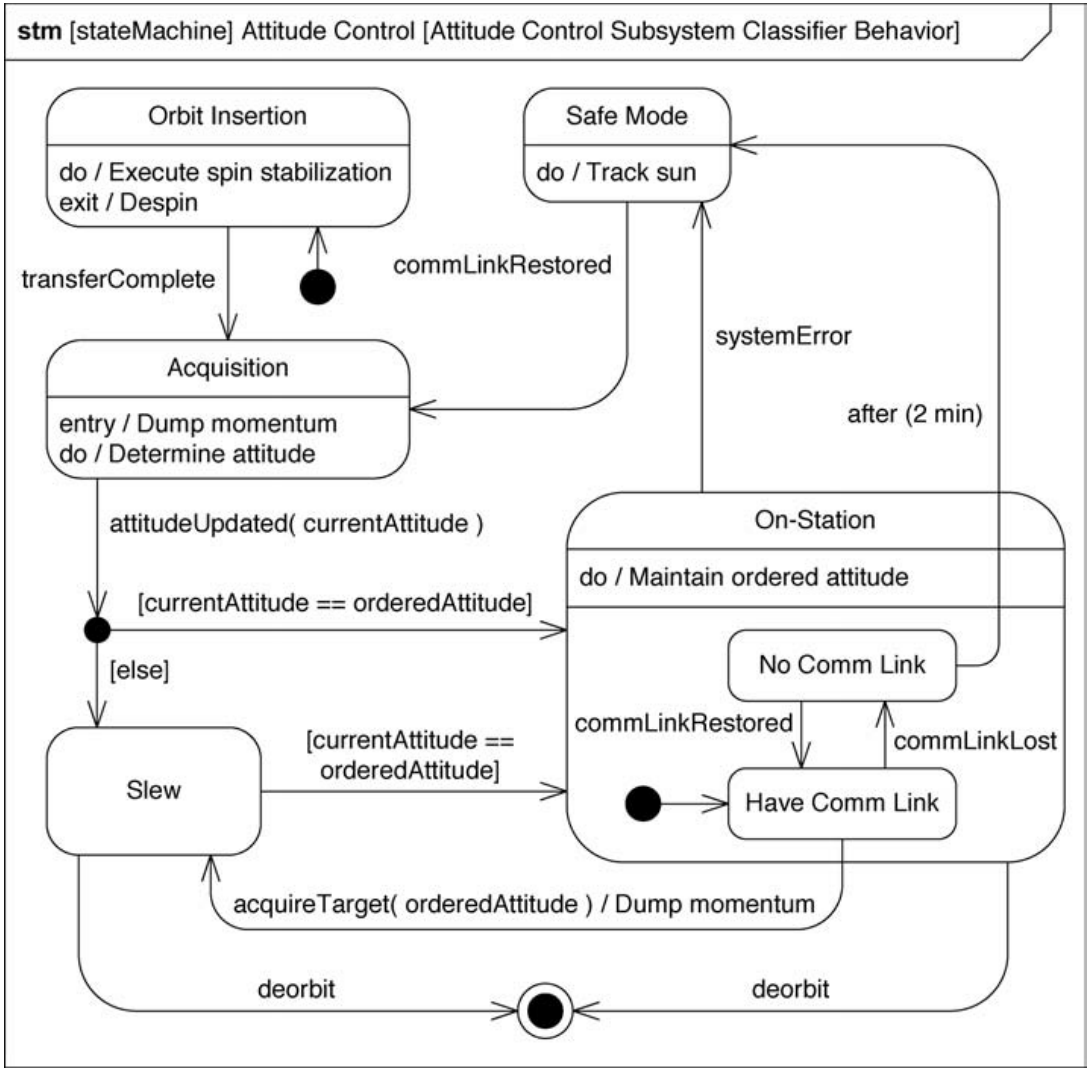
Used to specify a behavior, with a focus on how the parts of a block interact with one another via operation calls and asynchronous signals

- Commonly used as a detailed design tool to precisely specify a behavior as an input to the development stage of the life cycle
- An excellent mechanism for specifying test cases
- Allows the focus to be on how the parts of a block interact with one another via operation calls and asynchronous signals to produce an emergent behavior (**interaction**)



State Machine Diagram

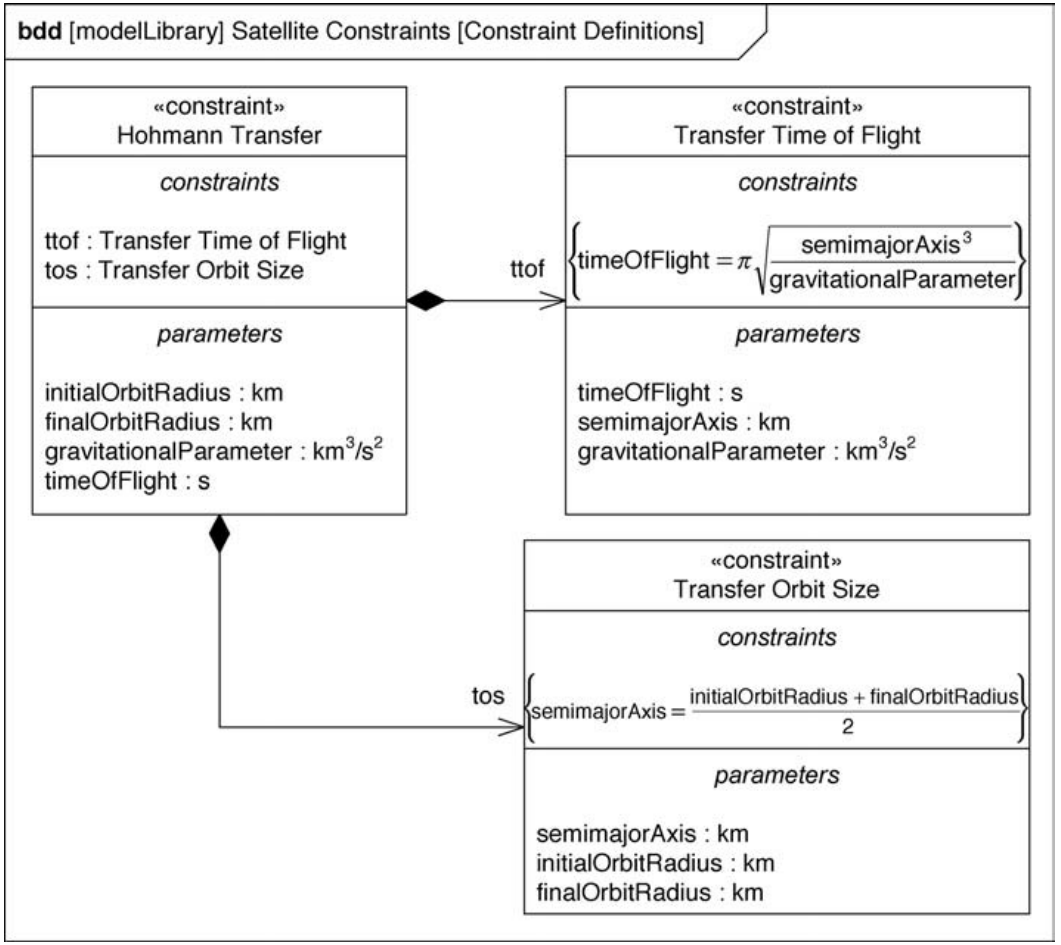
Used to specify a behavior, with a focus on the set of states of a block and the possible transitions between those states in response to event occurrences



Parametric Diagram

Used to express how one or more constraints (specifically, equations and inequalities) are bound to the properties of a system

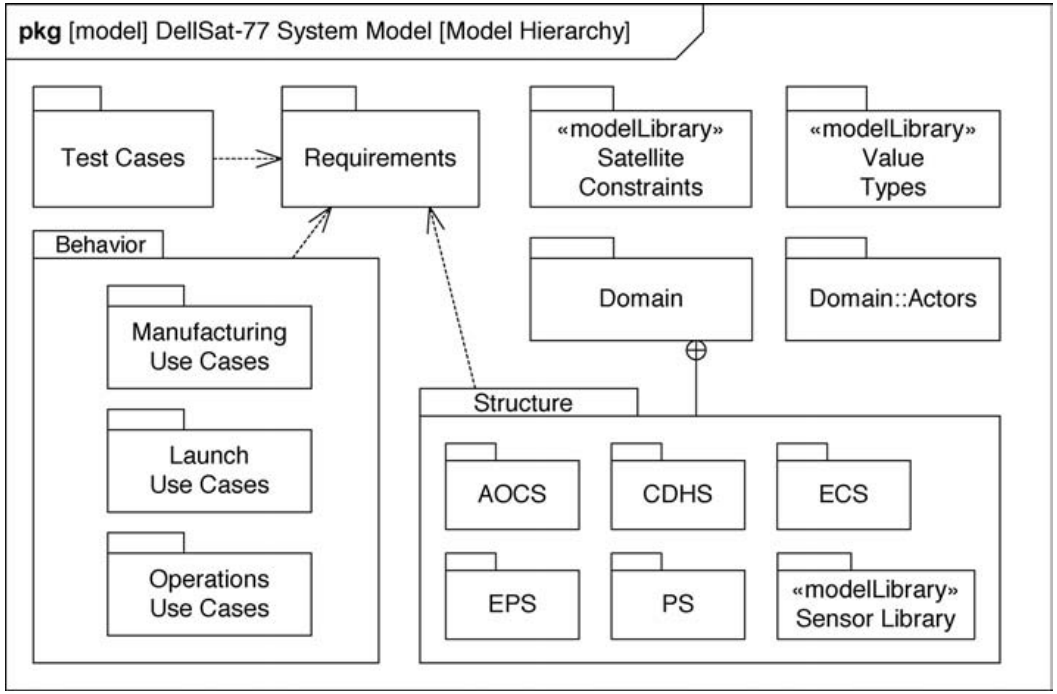
- To specify assertions about valid system values within an operational system (and therefore detect exceptional conditions when they occur)
- To use the blocks in your system model to provide the inputs for (and capture the outputs of) engineering analyses and simulations during the design stage



Package Diagram

Used to display the way a model is organized in the form of a package containment hierarchy

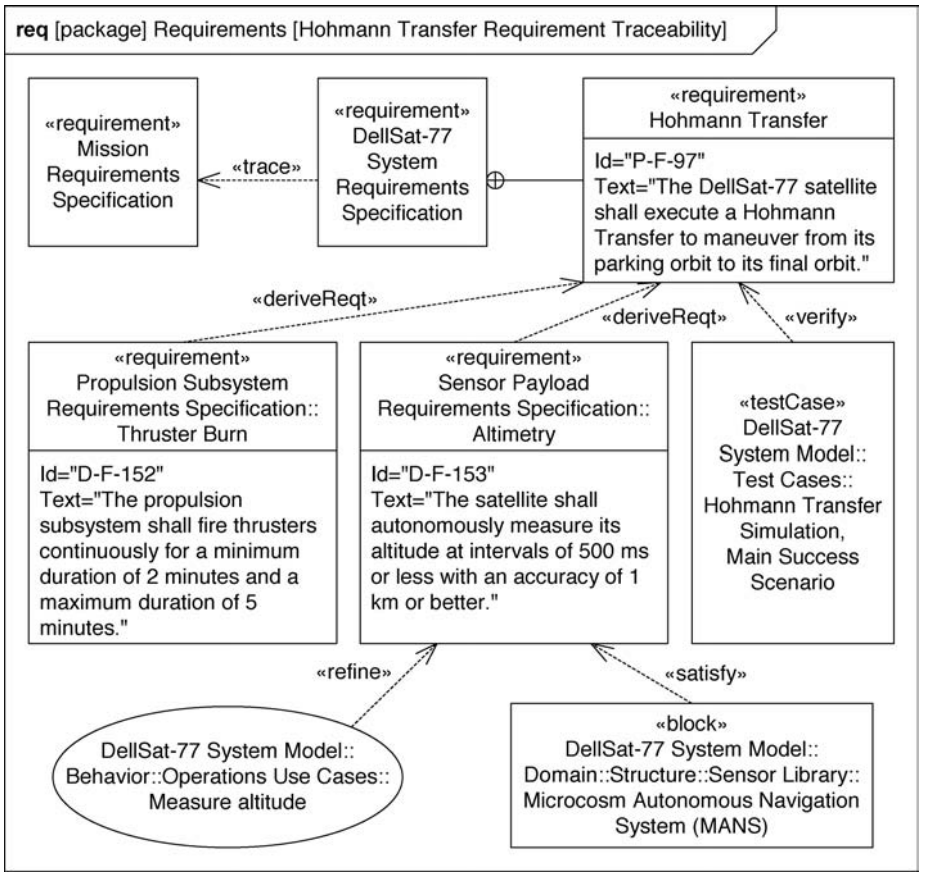
- May also show the model elements that packages contain and the dependencies between packages and their contained model elements
  - *Models*
  - *Model libraries*
  - *Profiles*
  - *Views*



Requirements Diagram

Used to display text-based requirements, the relationships between requirements, and the relationships between requirements and other model elements

- **Trace** - A modification to the supplier element (↑) *may* result in the need to modify the client element (tail end)
- **Derive** – Client requirement is derived from the supplier requirement
- **Refine** – Client element is more concrete (i.e., less abstract) than the supplier element
- **Satisfy** – Client requirement is fulfilled by the supplier element
- **Verify** – Client requirement is verified by the supplier test case



Allocations: Cross-Cutting Relationships Between Model Elements

Behavioral (functional) allocation – Allocate a behavioral element to a structural element

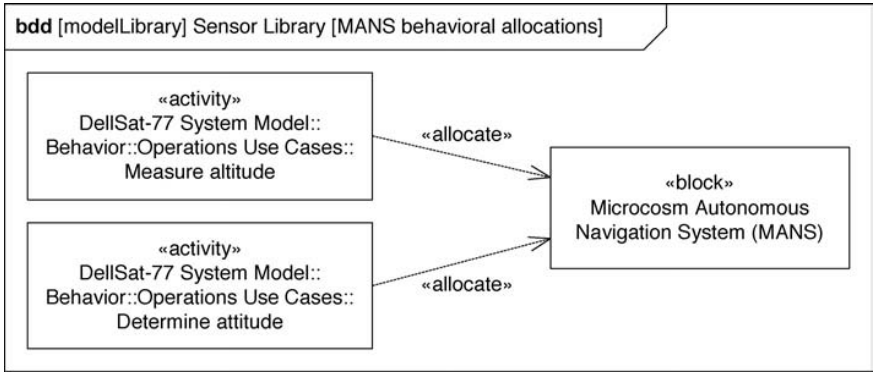
- Allocating an activity, an interaction, or a state machine behavior to a block
- Allocating an action (in an activity) to a part property (owned by a block)

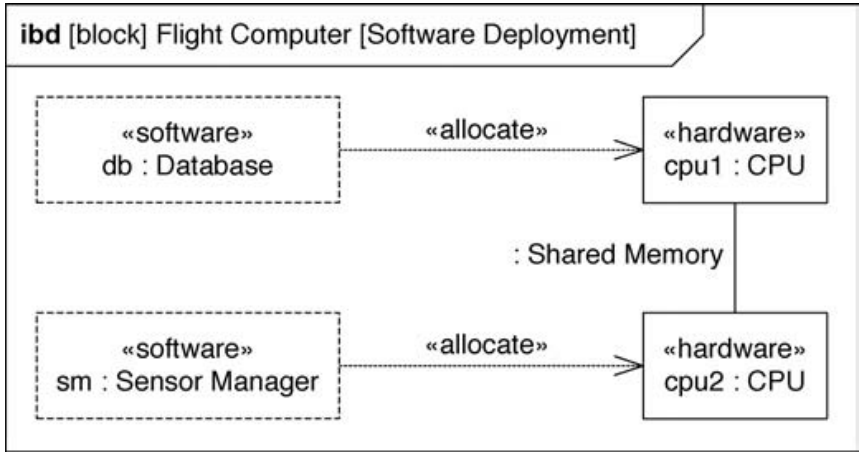
Structural allocation - Allocate a structural element to another structural element

- Allocating a logical block to a physical block
- Allocating a software property to a hardware property

Requirements allocation

- Uses “satisfy” relationship in the Requirements Diagram





MBSE Adoption

Introduction

- MBSE offers a powerful new tool for improving the quality and efficiency of product development
- Organizations must overcome significant barriers in adopting MBSE across their projects
- The presentation will highlight some of the key challenges and offer some solutions for overcoming them

MBSE Challenges

- Justifying the Investment
- Defining the Scope and Strategy
- Developing the Proper Technical Skills
- Developing the Proper Management Skills
- Standardization and Reuse

Justifying the Investment

Enterprise-wide MBSE adoption requires substantial upfront investment and risk-taking

- Investment is driven by the adoption strategy and timeline
- Great uncertainty in the estimation of costs and return-on-investment
- Unquantified pre-conditions and progress measures
- Few studies exist showing clear return-on-investment
- Returns take significant time to materialize

Strategies

- Start with small, targeted, high ROI wins
- Gather data that qualitatively demonstrates value (e.g., reduction of rework)

Benefits
(INCOSE *MBSE Adoption Guide*)

- Increases early rigour, identifies gaps and inconsistencies, and helps to eliminate errors.
- Helps to facilitate checking across domains and disciplines, and increases the degree to which requirements, design and V&V information can be checked by software, complementing the capabilities and skills of engineers
- Increases the potential for design re-use

Defining the Adoption Scope and Strategy

- MBSE is typically adopted *off-cycle* (a sandbox environment) or
  on-cycle (development projects)
  - Off-cycle projects may lack the rigor and realism of development projects
  - On-cycle projects present significant risks
- Need for modeling rules, guidelines, tool customizations, training materials, etc.
- Strategies
  - Review case studies and adoption guide for situational awareness
  - Accelerate adoption by using real or realistic case studies and examples during training

Developing the Proper Technical Skills

- MBSE requires a new set of technical skills across a wide spectrum of roles
  - SysML models and notations
  - Tool usage (mechanical tasks inhibit systems thinking)
  - Use of models to support engineering analyses
- Strategies
  - Start with a strong base of systems thinking and systems engineering skills
  - Don’t worry about selecting a enterprise-wide tool/approach until you gain some experience
  - Select tools used in training to minimize mechanics

Developing the Proper Management Skills

- The impacts of MBSE on project management are largely unknown
  - Unknown methods for planning, estimation, measurement, and control
  - Greater need for collaboration across the enterprise
  - New mechanisms for interfacing with customers and their staff
- Strategies
  - Adapt existing project management mechanisms and infrastructure
  - Project managers should attend technical training

Standardization and Reuse

- Although standardization and re-use offer significant benefits, it is difficult to determine an appropriate strategy
  - Dependent on commonality of applications across the enterprise, tool selection, product development methodology, existing reuse strategy, …
  - Typical “cut and paste” approach leads to poor engineering
- Strategies
  - Delay decisions until you gain experience
  - Focus initially on common system components