COLLEGE of CHARLESTON

1770

Systems *Engineering: Design and Development*

ENGR 387

Agenda

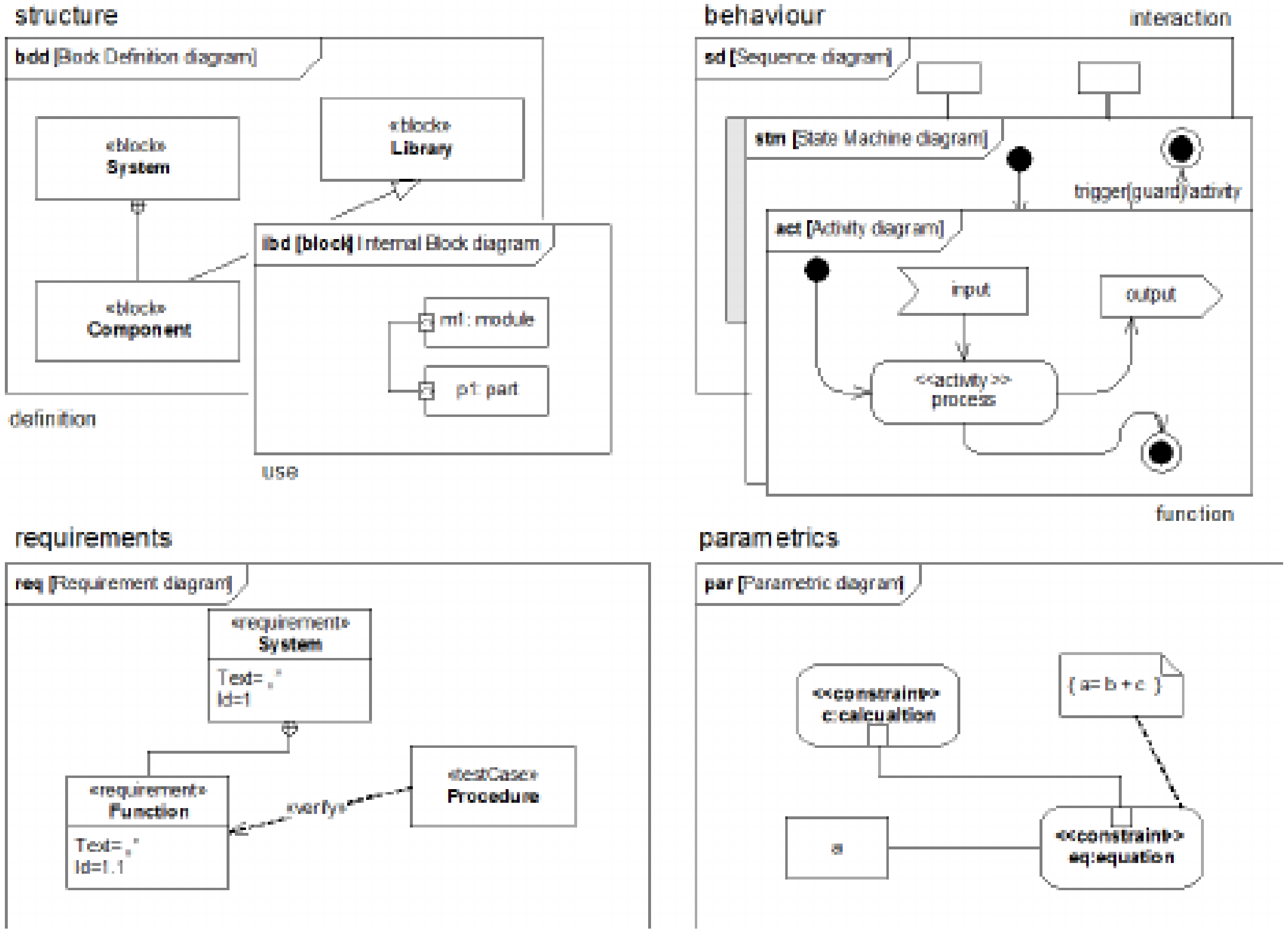
- OOSEM motivation and background
- An introduction to the steps in the OOSEM method
- Tailoring the OOSEM method
- SysML Diagram Taxonomy

OOSEM Introduction

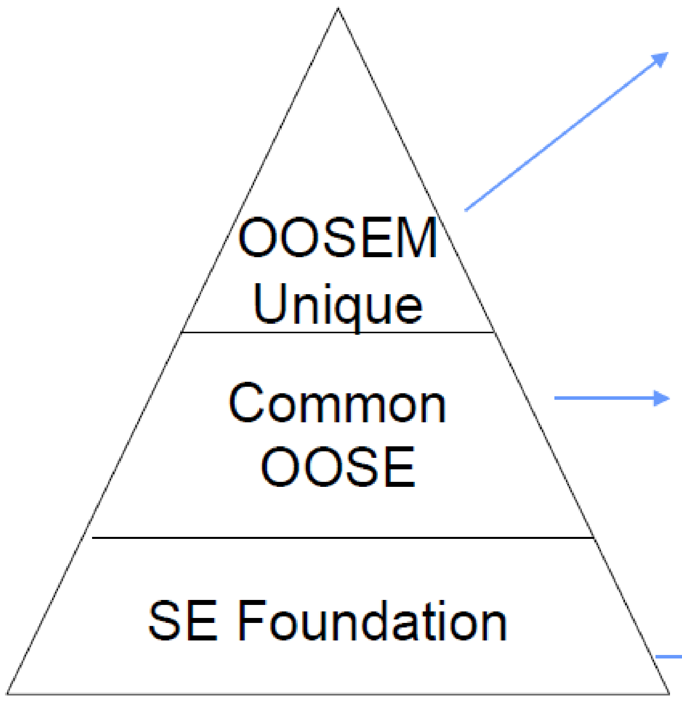
- Motivation
  - Provide a SE method that leverages object-oriented (O-O) concepts, traditional top-down SE approach, and SysML to facilitate:
  - Capture and analysis of requirements and design information to specify complex systems using SysML
  - Integration with O-O software development, hardware development, and test processesFlexibility to accommodate changing requirements and design evolution

- Background
  - Top-down Systems Engineering Approach
  - Scenario-driven process
  - Uses SysML to analyze, specify, design, and verify a system

SysML – 4 Pillars



OOSEM Features Build on OO & SE Foundation



Causal analysis
Enterprise model
Elaborated context
Requirements variation analysis
System/logical decomposition
Partitioning criteria
Node allocation
….

Top down SE approach
Recursive SE process
Use case/scenario driven (reqt's – test)
Black box/white box
OO concepts
UML/SysML

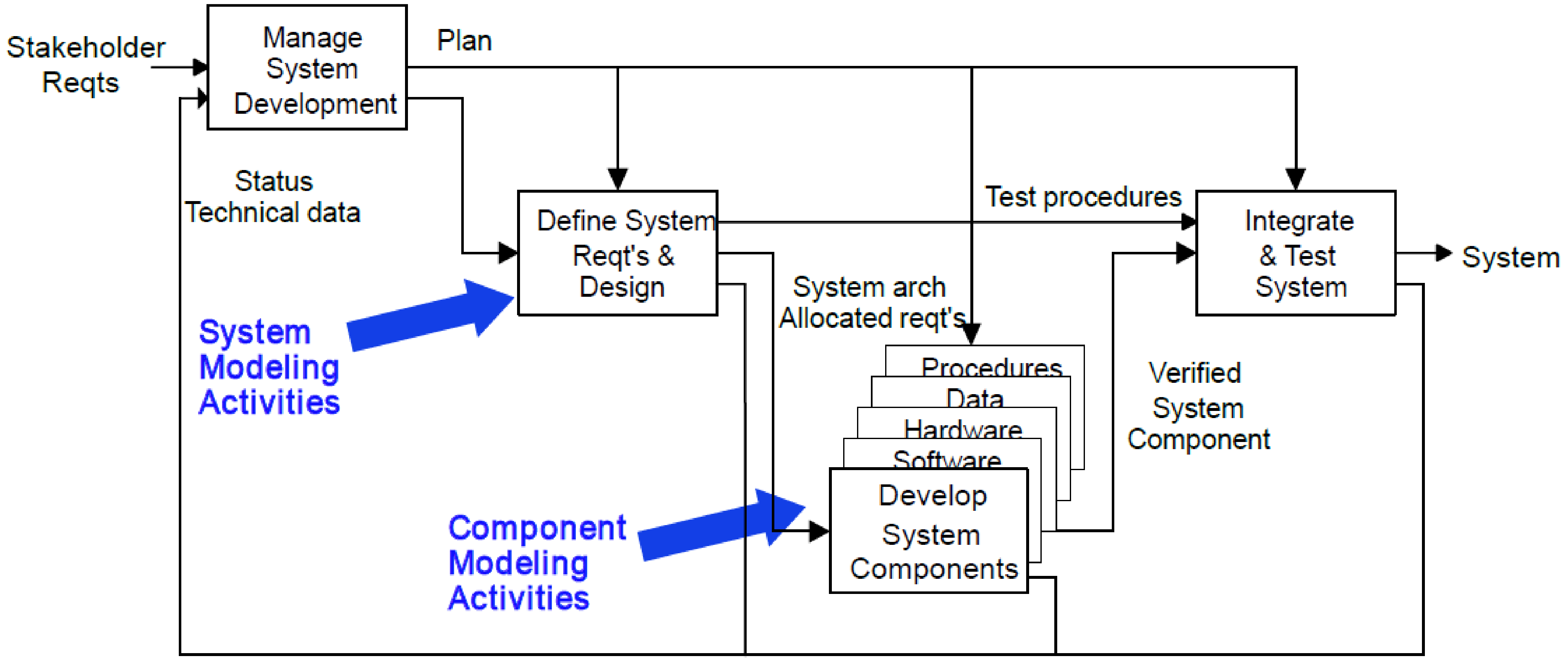
SE Process
Reqts, Trades, ..

What exactly is OO?

- What makes it "*OO*"SEM?
- What OO concepts are leveraged in OOSEM?
    - Classes, attributes, and operations (i.e. blocks)
    - Generalization/Specialization
    - Encapsulation
    - Use Cases

- Note: OO is different when applied to systems vs software

System Development Process


Stakeholder Reqts
Manage System Development
Plan
Status Technical data
Define System Reqt's & Design
Test procedures
Integrate & Test System
System
System arch Allocated reqt's
System Modeling Activities
Procedures
Data
Hardware
Software
Develop System Components
Verified System Component
Component Modeling Activities

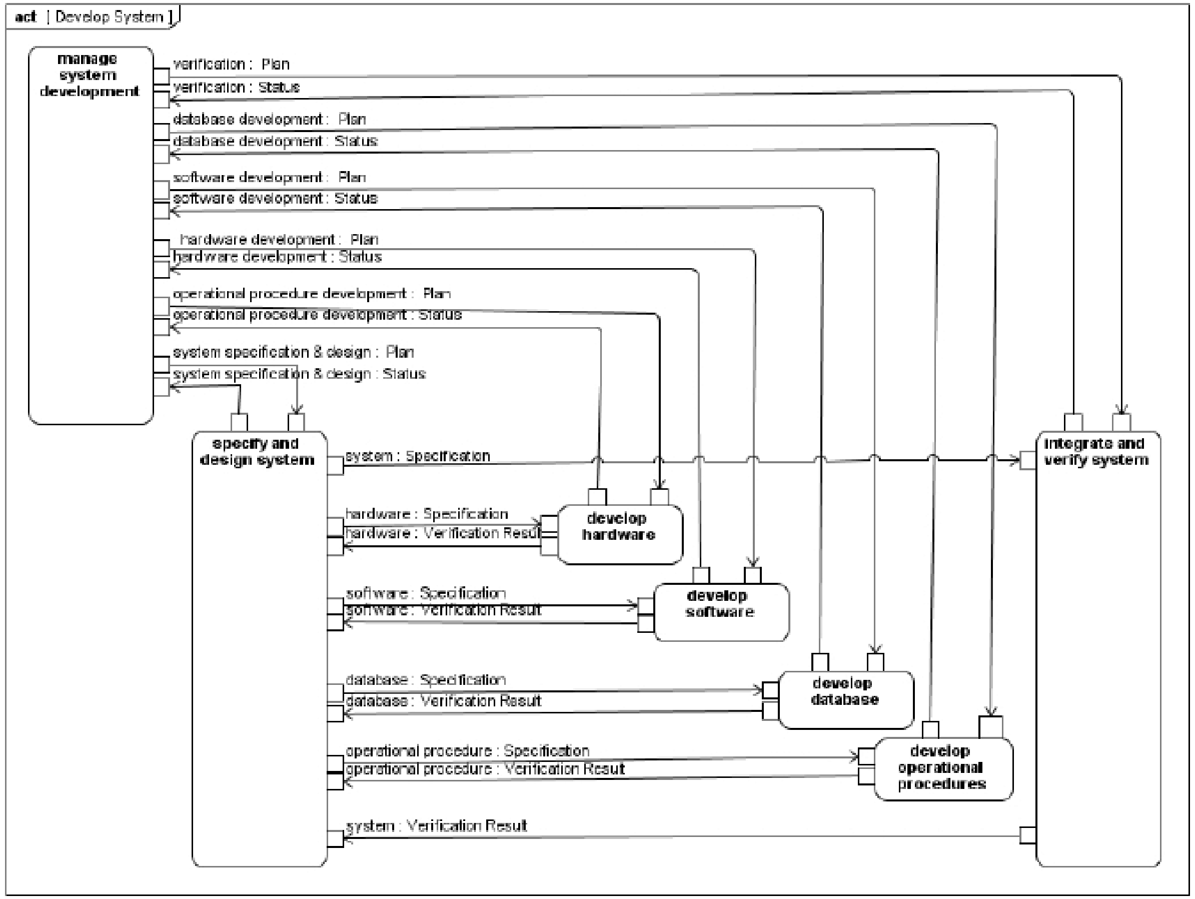
Integrated Product Development (IPD) is essential to improve communications

A Recursive V process that can be applied to multiple levels of the system hirearchy

System Development Process Overview

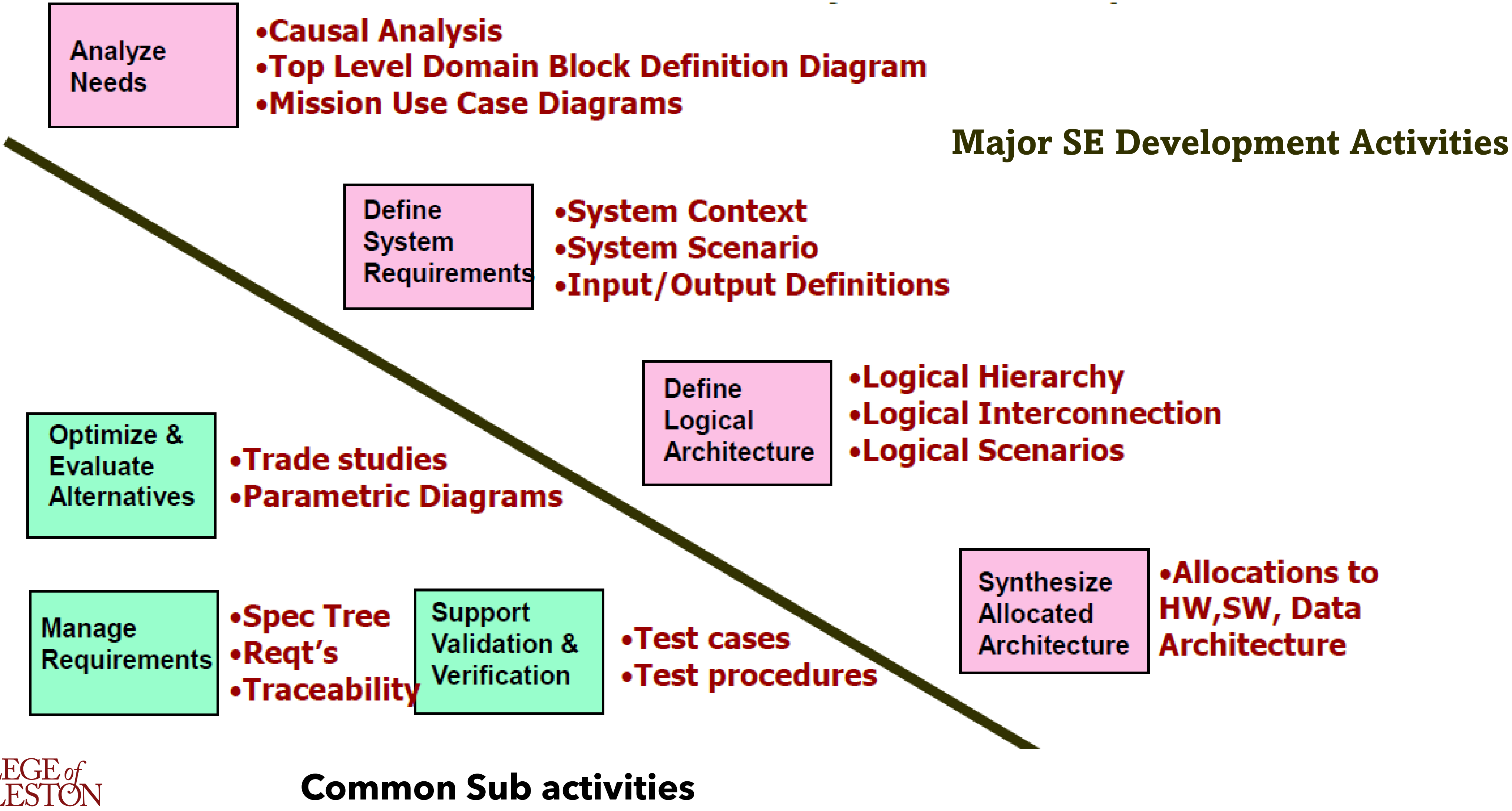
- Includes:
  - Management Process
  - System Specification and Design
  - ‘Next-level’ Development Process
  - System Integration and Verification

- Can be applied recursively to multiple levels of the systems hierarchySimilar to the SE ‘Vee’
- Majority of this course will focus on the ‘Specify and Design System’ activity


act [Develop System]
manage system development
verification : Plan
verification : Status
database development : Plan
database development : Status
software development : Plan
software development : Status
hardware development : Plan
hardware development : Status
operational procedure development : Plan
operational procedure development : Status
system specification & design : Plan
system specification & design : Status
specify and design system
system : Specification
hardware : Specification
hardware : Verification Resul
develop hardware
sofware : Specification
software : Verification Result
develop software
database : Specification
database : Verification Result
develop database
operational procedure : Specification
operational procedure : Verification Result
develop operational procedures
system : Verification Result
integrate and verify system

© 2011 Elsevier, Inc.: A Practical Guide to SysML

MBSE Process OOSEM Approach – Selected Artifacts



Major SE Development Activities

Analyze Needs

- •Causal Analysis
- •Top Level Domain Block Definition Diagram
- •Mission Use Case Diagrams

Define System Requirements

- •System Context
- •System Scenario
- •Input/Output Definitions

Define Logical Architecture

- •Logical Hierarchy
- •Logical Interconnection
- •Logical Scenarios

Synthesize Allocated Architecture

- •Allocations to HW,SW, Data Architecture

Common Sub activities

Optimize & Evaluate Alternatives

- •Trade studies
- •Parametric Diagrams

Manage Requirements

- •Spec Tree
- •Reqt's
- •Traceability

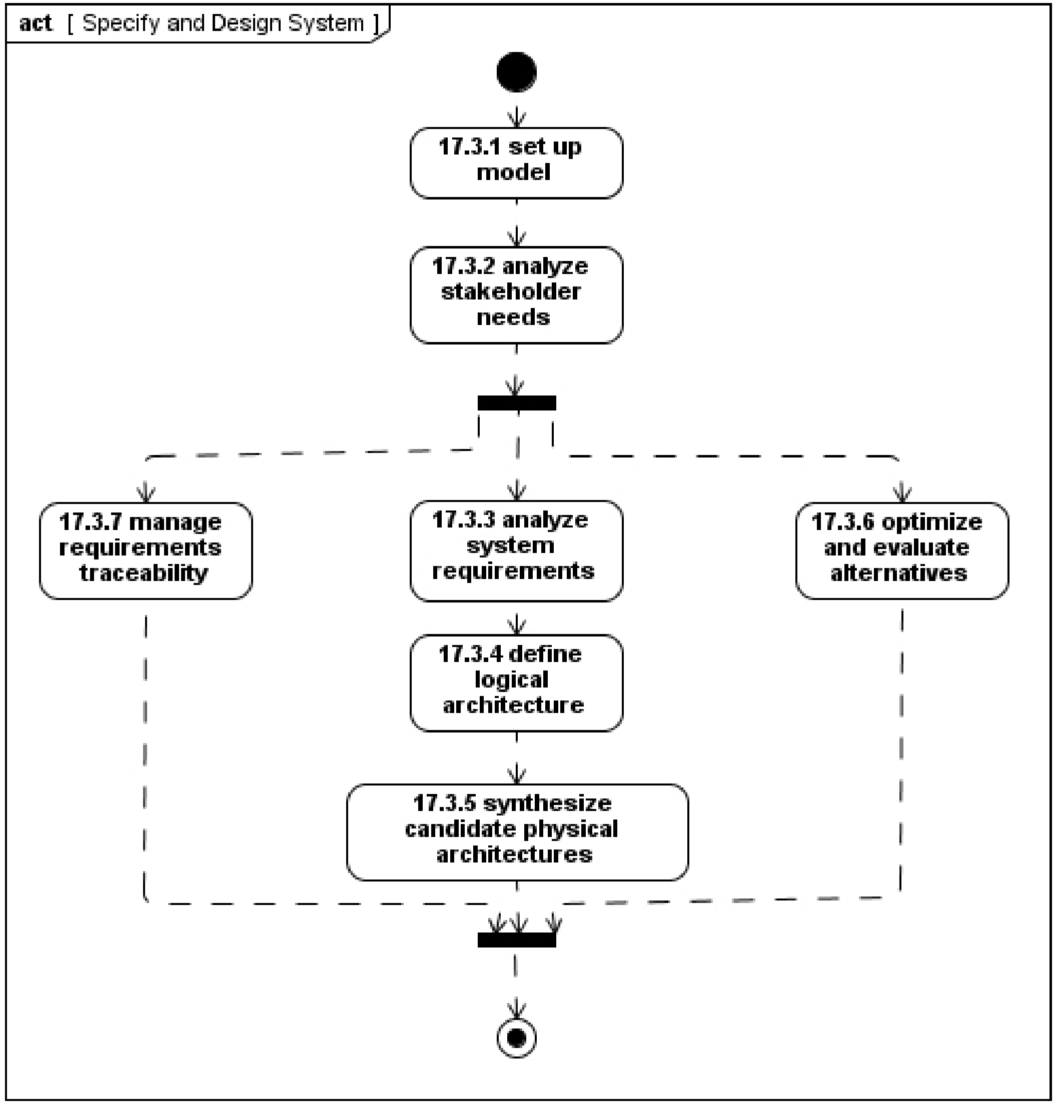
Support Validation & Verification

- •Test cases
- •Test procedures

Specify and Design System

- Purpose:
  - Develops the specification of elements at the next level of the system hierarchy
    - Example: applying the process at the system level, results in the specification of the system elements

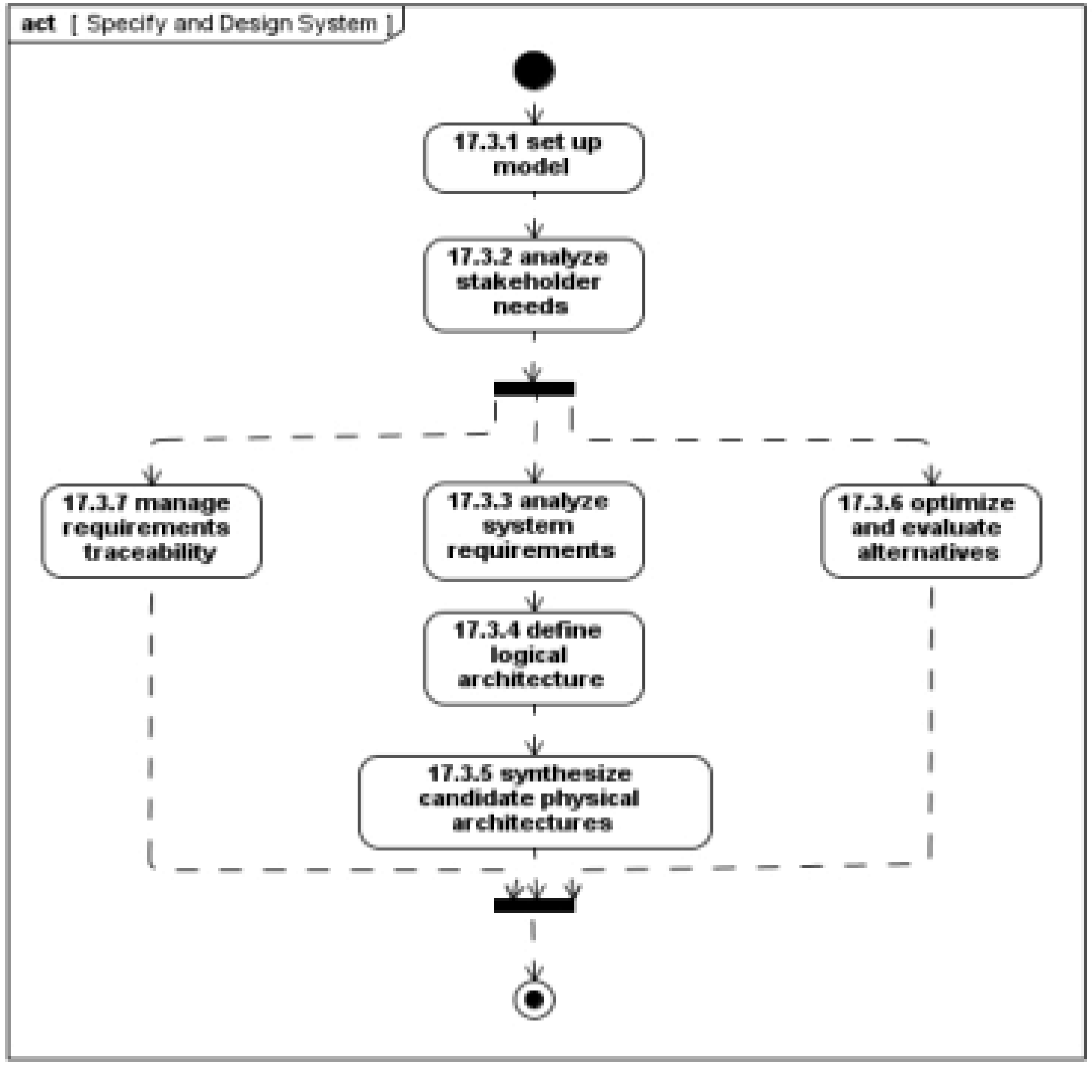
- Includes activities to:
  - Analyze the requirements
  - Define the architecture
  - Allocate the requirements to the next level of design


act [Specify and Design System]
17.3.1 set up model
17.3.2 analyze stakeholder needs
17.3.7 manage requirements traceability
17.3.3 analyze system requirements
17.3.6 optimize and evaluate alternatives
17.3.4 define logical architecture
17.3.5 synthesize candidate physical architectures

Set Up Model

- Establishes the modeling conventions and standards
  - Ensures consistent representation and style across the model

- Establishes the model organization

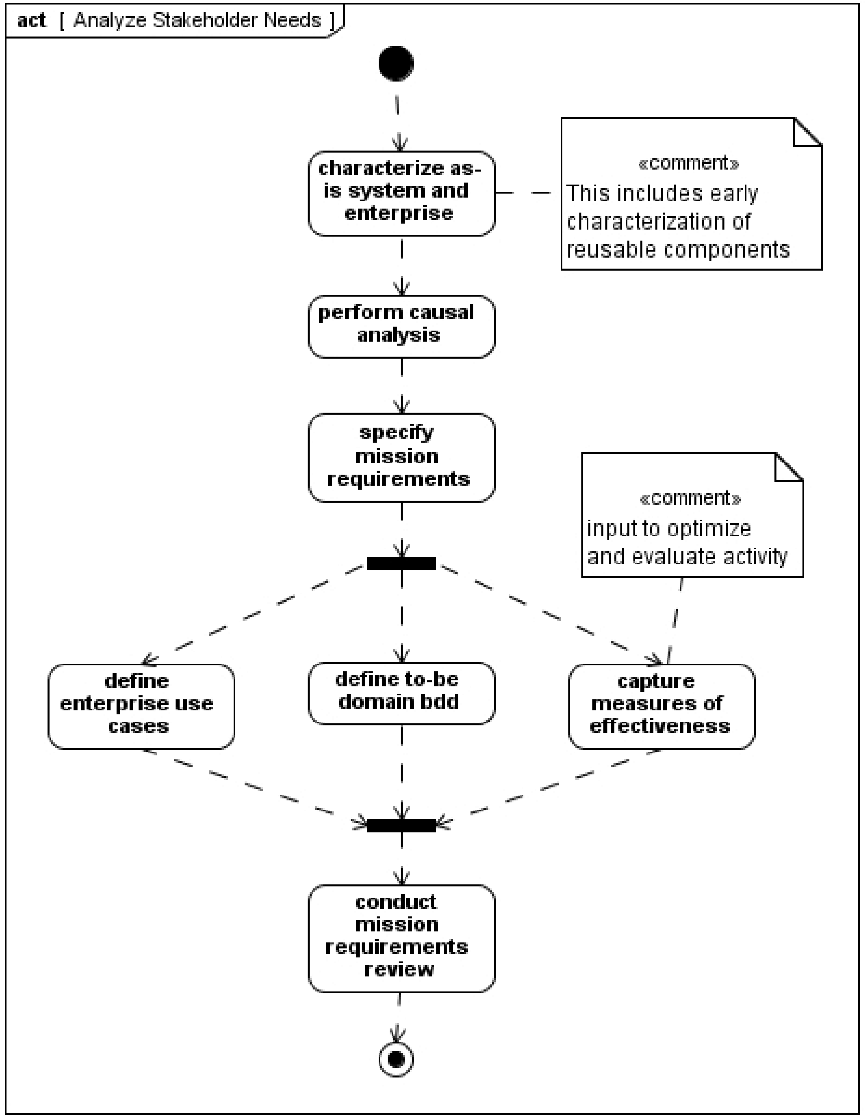


act [Specify and Design System]
17.3.1 set up model
17.3.2 analyze stakeholder needs
17.3.7 manage requirements traceability
17.3.3 analyze system requirements
17.3.6 optimize and evaluate alternatives
17.3.4 define logical architecture
17.3.5 synthesize candidate physical architectures

Analyze Stakeholder Needs

- Characterizes the as-is system
  - Limitations
  - Potential improvements

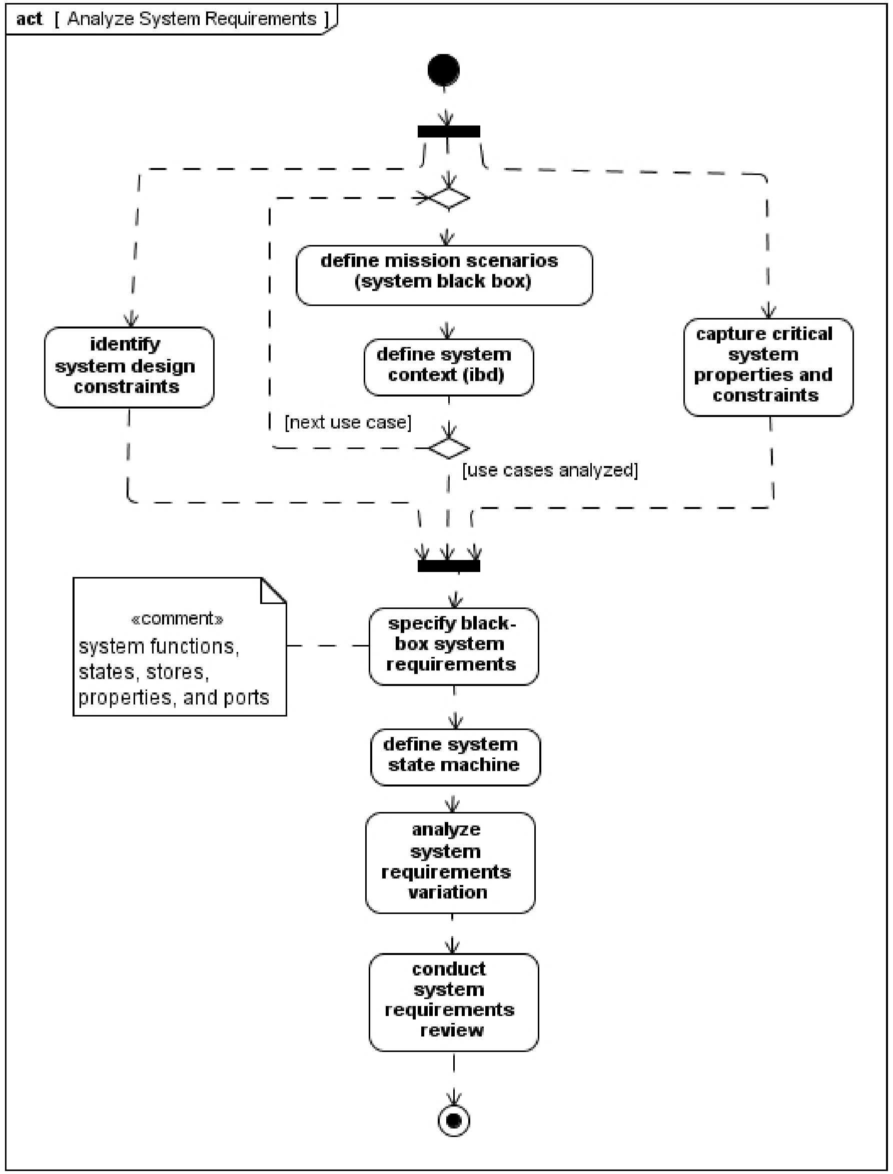
- Specifies mission requirements for the to-be system



act [Analyze Stakeholder Needs]
characterize as-is system and enterprise
«comment»
This includes early characterization of reusable components
perform causal analysis
specify mission requirements
«comment»
input to optimize and evaluate activity
define enterprise use cases
define to-be domain bdd
capture measures of effectiveness
conduct mission requirements review

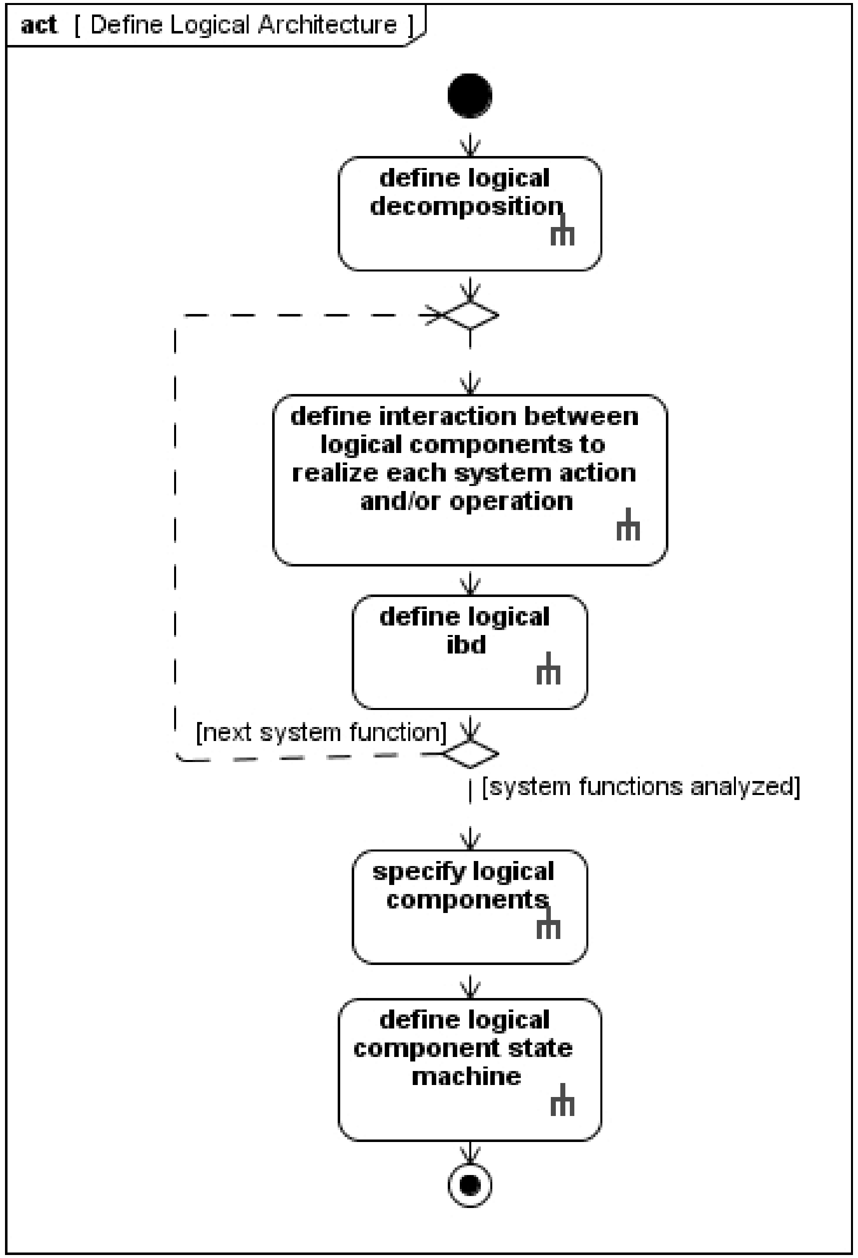
Analyze Stakeholder Requirements

- Specifies the system requirements in terms of its input and output responses and other black box characteristics


act [Analyze System Requirements]
identify system design constraints
define mission scenarios (system black box)
define system context (ibd)
[next use case]
[use cases analyzed]
capture critical system properties and constraints
«comment»
system functions, states, stores, properties, and ports
specify black-box system requirements
define system state machine
analyze system requirements variation
conduct system requirements review

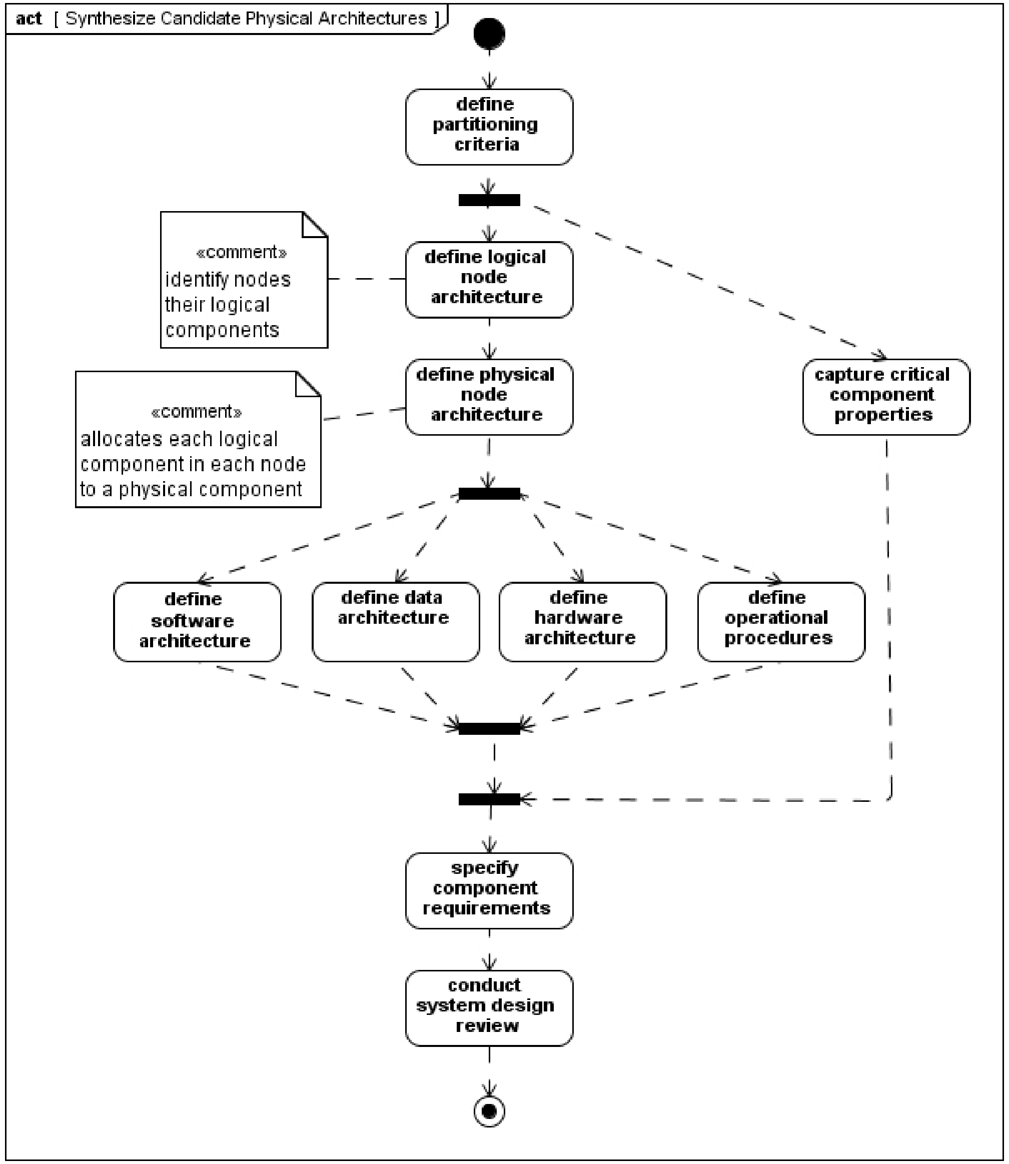
Define Logical Architecture

- Decomposes the system into logical components and defines how the logical components interact


act [Define Logical Architecture]
define logical decomposition
define interaction between logical components to realize each system action and/or operation
define logical ibd
[next system function]
[system functions analyzed]
specify logical components
define logical component state machine

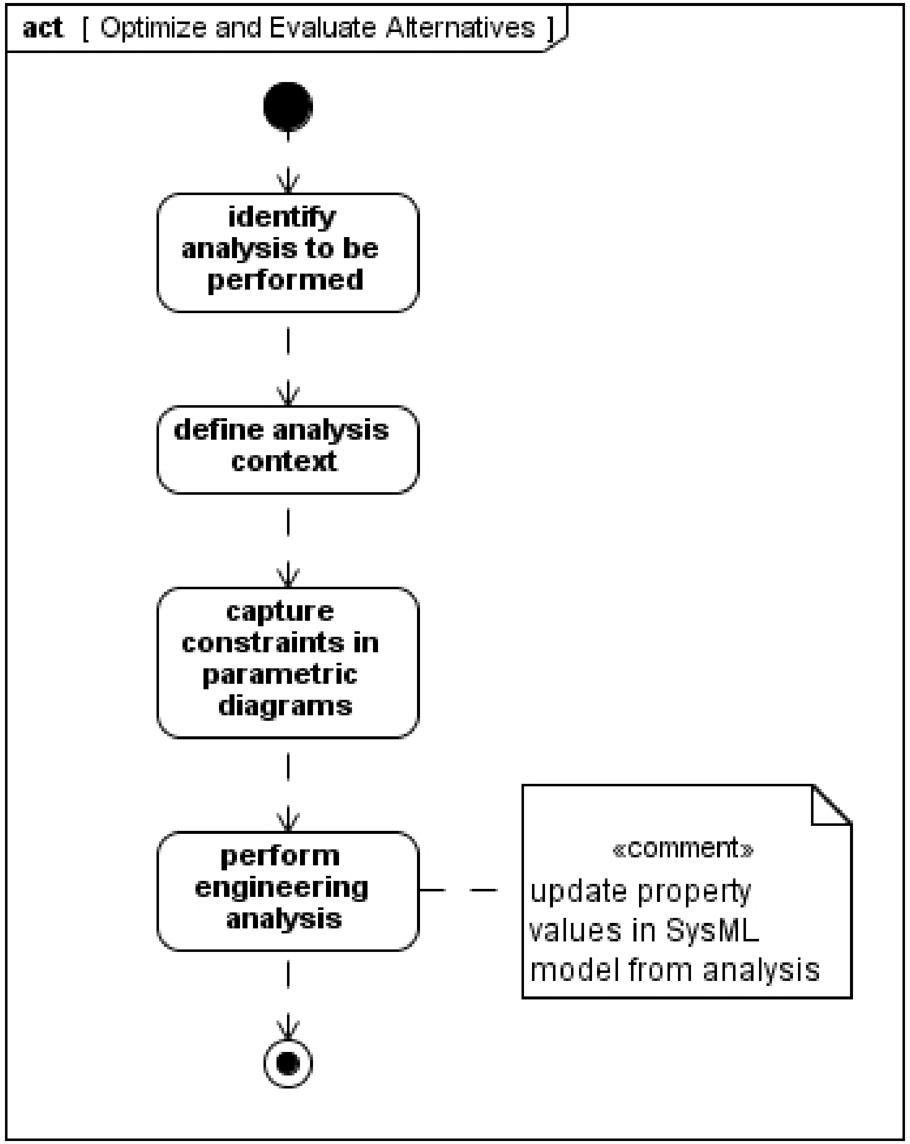
Synthesize Candidate Physical Architectures

- Allocates the logical components to physical components that are implemented in hardware, software, data, and procedures


act [Synthesize Candidate Physical Architectures]
define partitioning criteria
«comment» identify nodes their logical components
define logical node architecture
define physical node architecture
«comment» allocates each logical component in each node to a physical component
capture critical component properties
define software architecture
define data architecture
define hardware architecture
define operational procedures
specify component requirements
conduct system design review

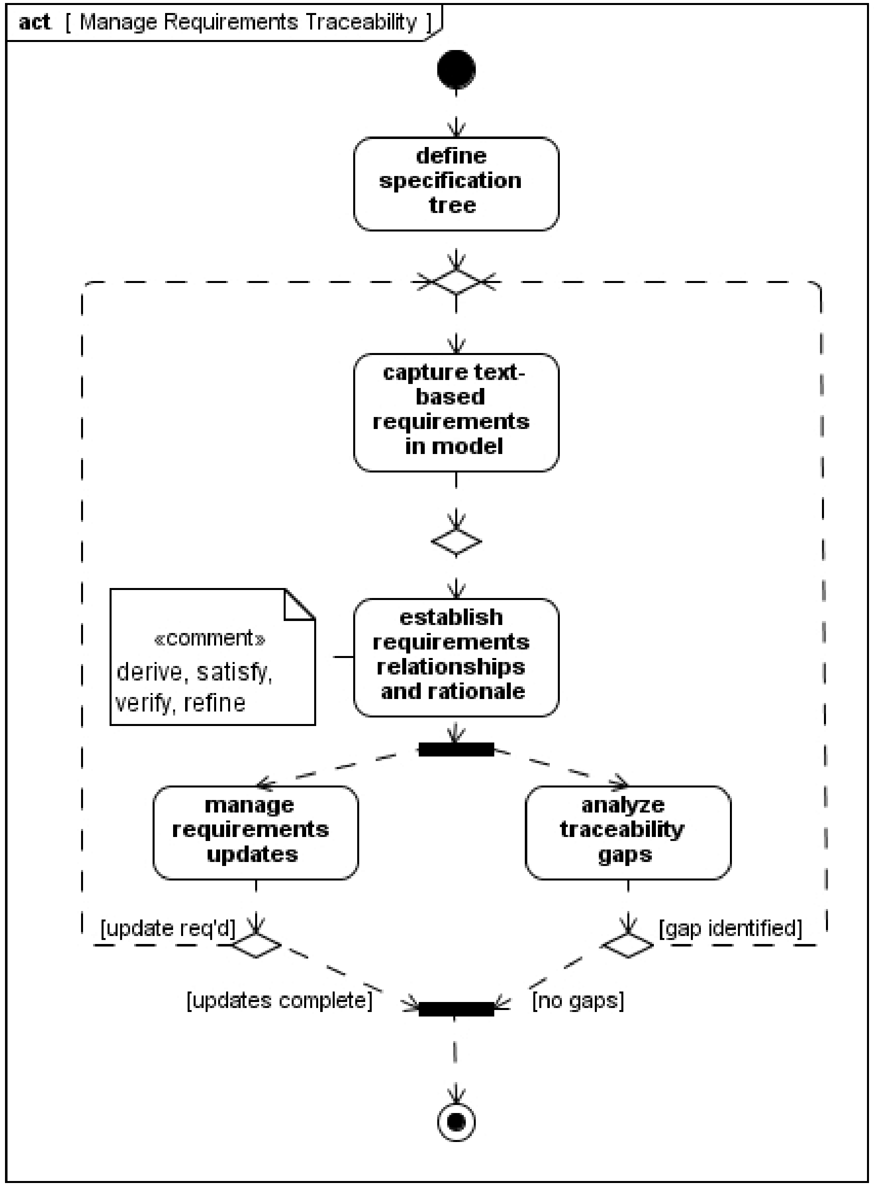
Optimize and Evaluate Alternatives

- Invoked throughout the process
    - Performs engineering analysis that supports system design trade studies and design optimization


act [Optimize and Evaluate Alternatives]
identify analysis to be performed
define analysis context
capture constraints in parametric diagrams
perform engineering analysis
«comment»
update property values in SysML model from analysis

Manage Requirements Traceability

- Invoked throughout the process
  - Manage traceability from the mission-level requirements to the component requirements


act [Manage Requirements Traceability]
define specification tree
capture text-based requirements in model
«comment»
derive, satisfy, verify, refine
establish requirements relationships and rationale
manage requirements updates
analyze traceability gaps
[update req'd]
[gap identified]
[updates complete]
[no gaps]

Tailoring the OOSEM method

- OOSEM can be tailored to meet project objectives and constraints (i.e. time and money available)

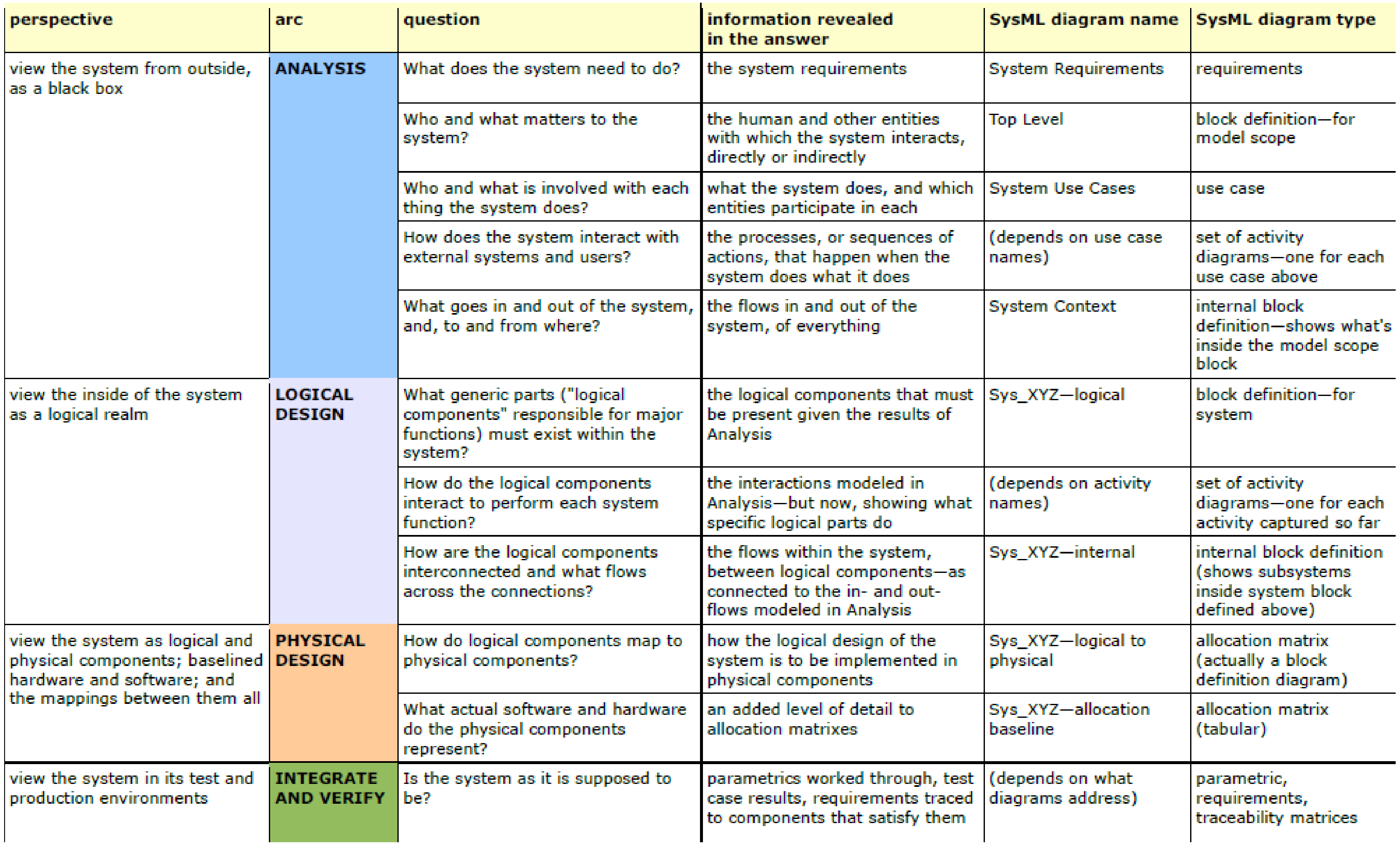
- Tailoring involves selecting the appropriate level of rigor to apply to each of the OOSEM activities
    - Selecting the activities and their ordering
    - Selecting the artifacts to generate
    - Selecting the level of detail

- INCOSE OOSEM Working Group is currently working to define a scaled down version of OOSEM (OOSEM-Lite)
    - Example: SysML Storybooks

SysML Storybooks

- Concept developed by Abe Raher, as part of the INCOSE OOSEM Working Group
- Storybooks:
  - Communicate with pictures as much as with words
  - Are short enough to finish in one sitting
  - Maintain a clear enough thread that we don't get lost
  - Are simple and vivid enough that we remember what happened and why
  - Prepare us to understand more complicated stories
- Thus storybooks make their content easier to grasp
- Introducing these concepts into the realm of system modeling:
  - Ask a question in everyday, non-technical language
  - Answer it with a picture
  - Use the picture as the basis for exploring the next question

Storybook Questions and the Corresponding SysML Diagram



perspective	arc	question	information revealed in the answer	SysML diagram name	SysML diagram type
view the system from outside, as a black box	**ANALYSIS**	What does the system need to do?	the system requirements	System Requirements	requirements
		Who and what matters to the system?	the human and other entities with which the system interacts, directly or indirectly	Top Level	block definition—for model scope
		Who and what is involved with each thing the system does?	what the system does, and which entities participate in each	System Use Cases	use case
		How does the system interact with external systems and users?	the processes, or sequences of actions, that happen when the system does what it does	(depends on use case names)	set of activity diagrams—one for each use case above
		What goes in and out of the system, and, to and from where?	the flows in and out of the system, of everything	System Context	internal block definition—shows what's inside the model scope block
view the inside of the system as a logical realm	**LOGICAL DESIGN**	What generic parts ("logical components" responsible for major functions) must exist within the system?	the logical components that must be present given the results of Analysis	Sys_XYZ—logical	block definition—for system
		How do the logical components interact to perform each system function?	the interactions modeled in Analysis—but now, showing what specific logical parts do	(depends on activity names)	set of activity diagrams—one for each activity captured so far
		How are the logical components interconnected and what flows across the connections?	the flows within the system, between logical components—as connected to the in- and out-flows modeled in Analysis	Sys_XYZ—internal	internal block definition (shows subsystems inside system block defined above)
view the system as logical and physical components; baselined hardware and software; and the mappings between them all	**PHYSICAL DESIGN**	How do logical components map to physical components?	how the logical design of the system is to be implemented in physical components	Sys_XYZ—logical to physical	allocation matrix (actually a block definition diagram)
		What actual software and hardware do the physical components represent?	an added level of detail to allocation matrixes	Sys_XYZ—allocation baseline	allocation matrix (tabular)
view the system in its test and production environments	**INTEGRATE AND VERIFY**	Is the system as it is supposed to be?	parametrics worked through, test case results, requirements traced to components that satisfy them	(depends on what diagrams address)	parametric, requirements, traceability matrices

SysML Diagram Taxonomy

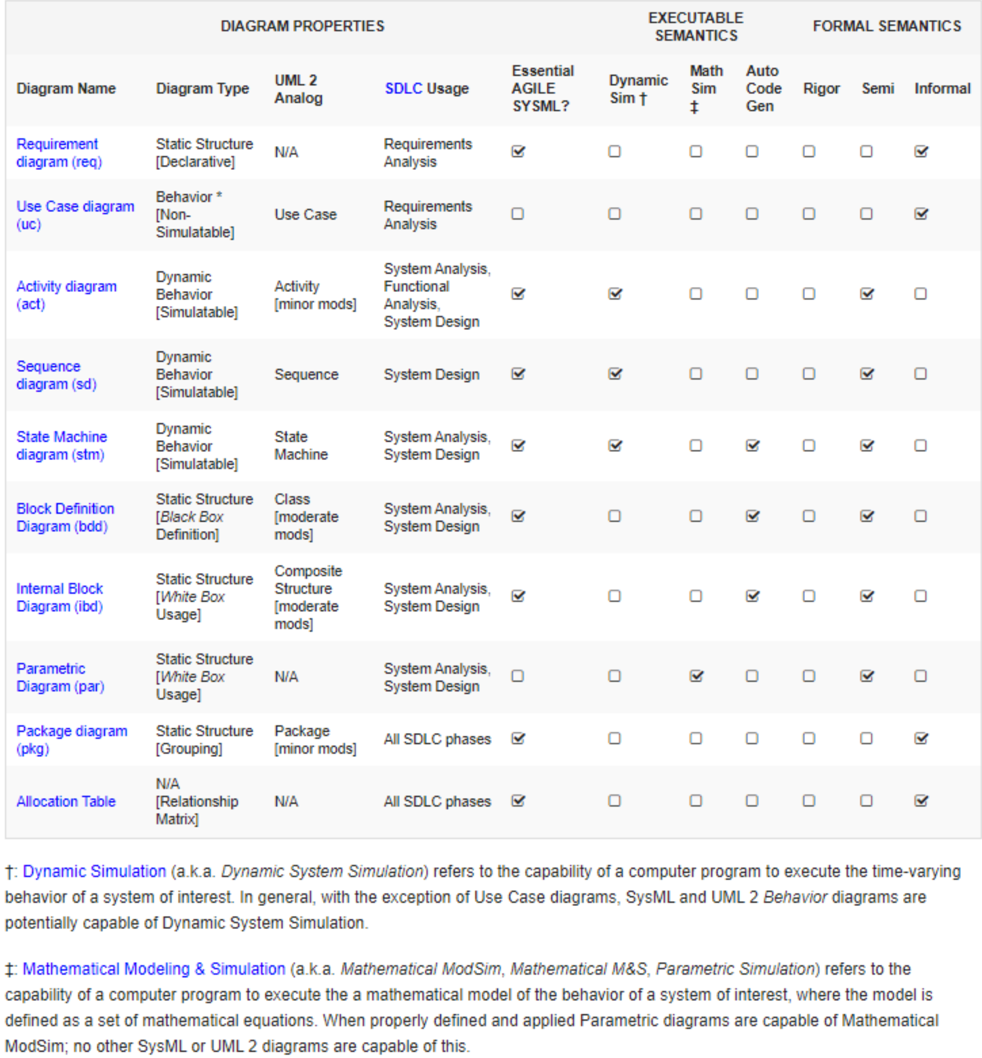
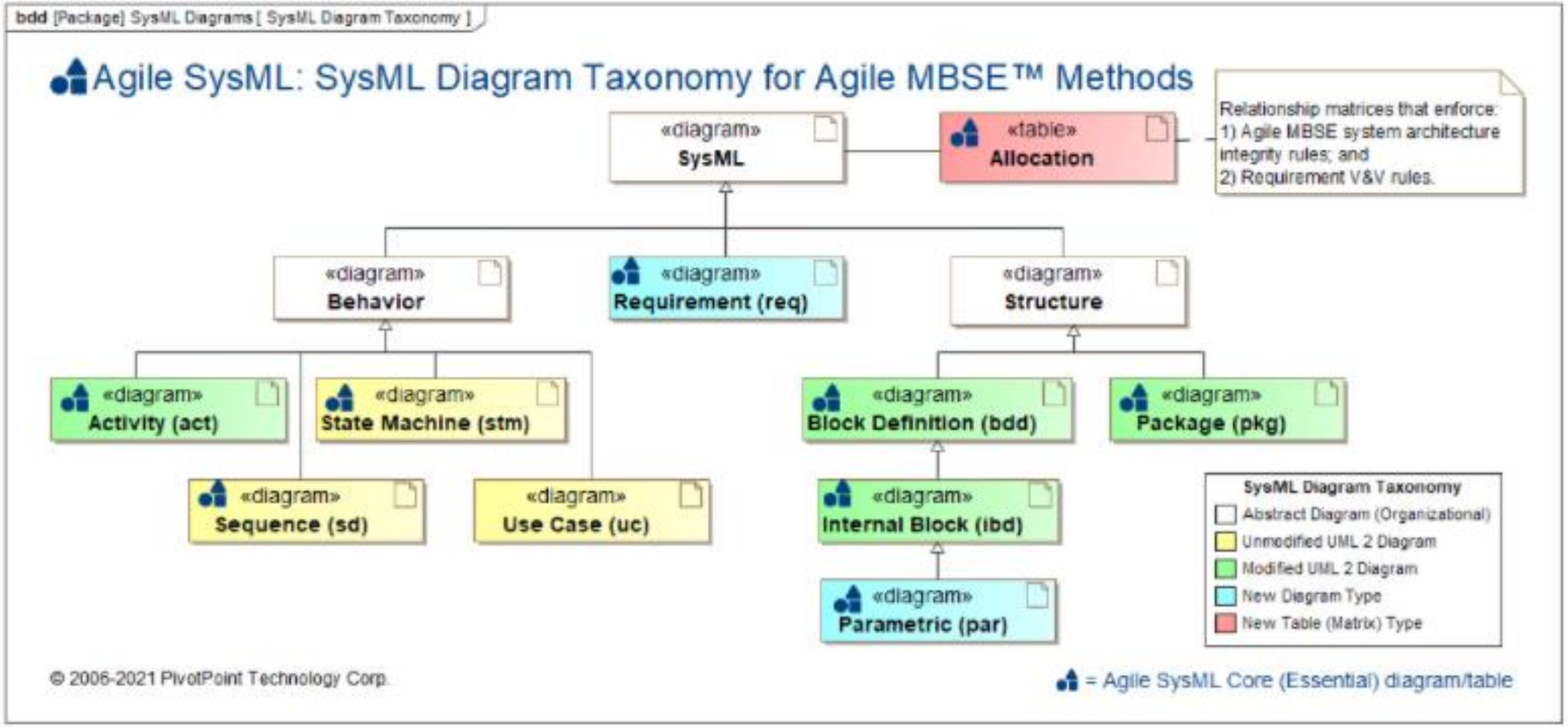


DIAGRAM PROPERTIES					EXECUTABLE SEMANTICS			FORMAL SEMANTICS		
Diagram Name	Diagram Type	UML 2 Analog	SDLC Usage	Essential AGILE SYSML?	Dynamic Sim †	Math Sim ‡	Auto Code Gen	Rigor	Semi	Informal
Requirement diagram (req)	Static Structure [Declarative]	N/A	Requirements Analysis	☑	□	□	□	□	□	☑
Use Case diagram (uc)	Behavior * [Non-Simulatable]	Use Case	Requirements Analysis	□	□	□	□	□	□	☑
Activity diagram (act)	Dynamic Behavior [Simulatable]	Activity [minor mods]	System Analysis, Functional Analysis, System Design	☑	☑	□	□	□	☑	□
Sequence diagram (sd)	Dynamic Behavior [Simulatable]	Sequence	System Design	☑	☑	□	□	□	☑	□
State Machine diagram (stm)	Dynamic Behavior [Simulatable]	State Machine	System Analysis, System Design	☑	☑	□	☑	□	☑	□
Block Definition Diagram (bdd)	Static Structure [*Black Box* Definition]	Class [moderate mods]	System Analysis, System Design	☑	□	□	☑	□	☑	□
Internal Block Diagram (ibd)	Static Structure [*White Box* Usage]	Composite Structure [moderate mods]	System Analysis, System Design	☑	□	□	☑	□	☑	□
Parametric Diagram (par)	Static Structure [*White Box* Usage]	N/A	System Analysis, System Design	□	□	☑	□	□	☑	□
Package diagram (pkg)	Static Structure [Grouping]	Package [minor mods]	All SDLC phases	☑	□	□	□	□	□	☑
Allocation Table	N/A [Relationship Matrix]	N/A	All SDLC phases	☑	□	□	□	□	□	☑

†: Dynamic Simulation (a.k.a. *Dynamic System Simulation*) refers to the capability of a computer program to execute the time-varying behavior of a system of interest. In general, with the exception of Use Case diagrams, SysML and UML 2 *Behavior* diagrams are potentially capable of Dynamic System Simulation.

‡: Mathematical Modeling & Simulation (a.k.a. *Mathematical ModSim*, *Mathematical M&S*, *Parametric Simulation*) refers to the capability of a computer program to execute the a mathematical model of the behavior of a system of interest, where the model is defined as a set of mathematical equations. When properly defined and applied Parametric diagrams are capable of Mathematical ModSim; no other SysML or UML 2 diagrams are capable of this.

SysML Diagram Taxonomy


bdd [Package] SysML Diagrams [SysML Diagram Taxonomy]
Agile SysML: SysML Diagram Taxonomy for Agile MBSE™ Methods
«diagram» SysML
«table» Allocation
«diagram» Behavior
«diagram» Requirement (req)
«diagram» Structure
«diagram» Activity (act)
«diagram» State Machine (stm)
«diagram» Sequence (sd)
«diagram» Use Case (uc)
«diagram» Block Definition (bdd)
«diagram» Package (pkg)
«diagram» Internal Block (ibd)
«diagram» Parametric (par)
SysML Diagram Taxonomy
□ Abstract Diagram (Organizational)
🟨 Unmodified UML 2 Diagram
🟩 Modified UML 2 Diagram
🟦 New Diagram Type
🟥 New Table (Matrix) Type
= Agile SysML Core (Essential) diagram/table

Relationship matrices that enforce:
1) Agile MBSE system architecture integrity rules; and
2) Requirement V&V rules.

© 2006-2021 PivotPoint Technology Corp.

SysML Diagram Taxonomy for Agile MBSE™

Summary

- OOSEM uses SysML and Object-Oriented concepts (e.g. Use Cases) to provide a top-down, scenario-driven process to analyze, specify, design, and verify a system
- This method flows requirements down from stakeholder needs to component-level specifications
- Specify and Design System' activity includes:
  - Analysis of stakeholder needs
  - Analysis of system requirements
  - Definition of the logical architecture
  - Synthesis of candidate physical architectures
  - Optimizing and evaluating alternatives
  - Managing requirements traceability

References

Lykins, Friedenthal, Meilich, Adapting UML for an Object-Oriented Systems Engineering Method (OOSEM), INCOSE Symposium, Minneapolis, USA, 2000

"A Practical Guide to SysML: The Systems Modeling Language"; Friedenthal, Moore, and Steiner; 2009, Elsevier, Inc., (Chapter 16)

Estefan, Jeff A. "Survey of Model-Based Systems Engineering (MBSE) Methodologies" INCOSE-TD-2007-003-02, 10 June 2008