SysEng 6542
Model Based Systems Engineering

Lecture - 2 MBSE Methodology

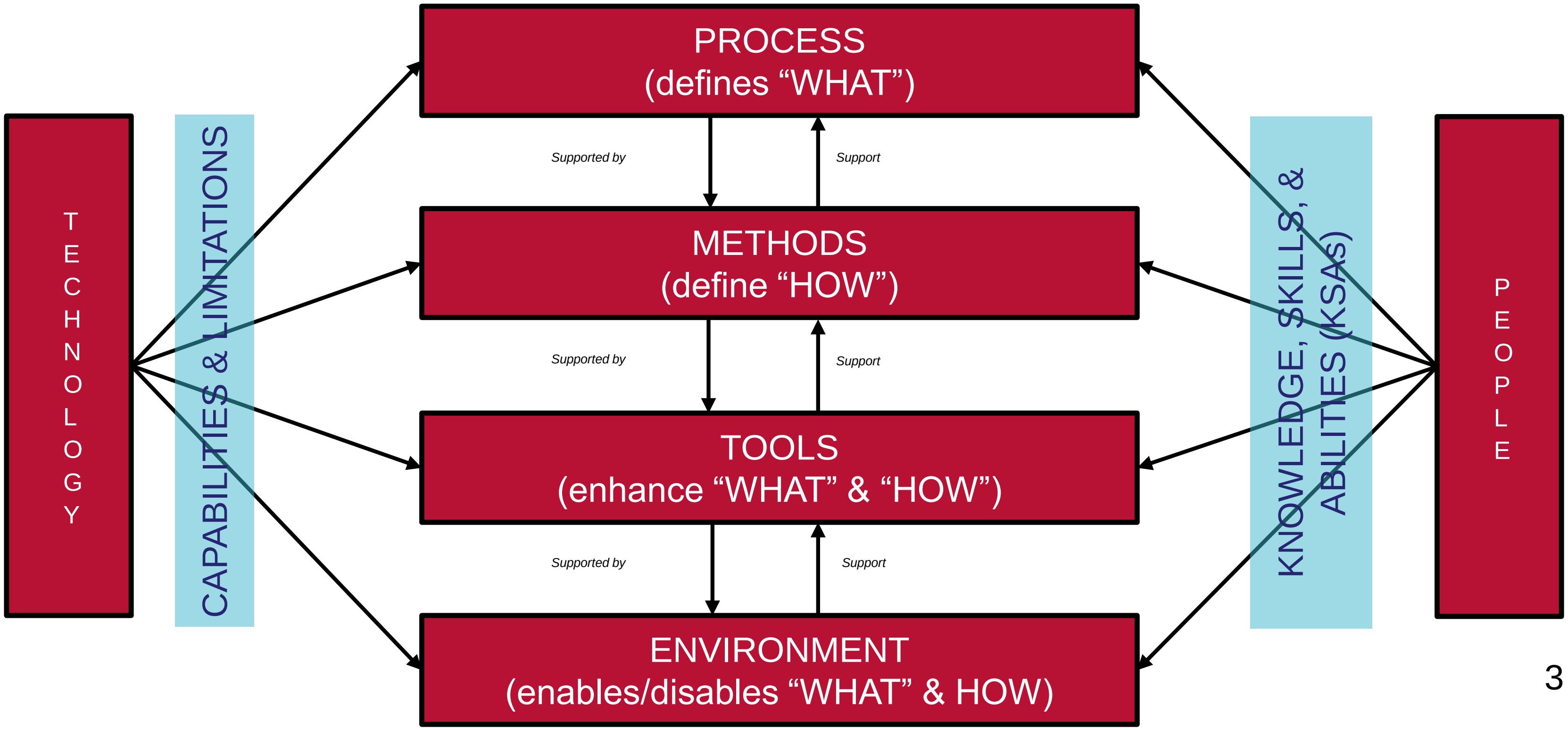
Dr Quoc Do

Leading MBSE Methodologies

- IBM Telelogic Harmony-SE
- INCOSE Object-Oriented Systems Engineering Method (OOSEM)
- JPL State Analysis
- IBM Rational Unified Process for Systems Engineering (RUB SE) for Model-Driven Systems Development (MDSD)
- Vitech Model MBSE

Key Terminologies

- The relationship between process, methods, tools and environment is depicted below (Martin, 1996)



Key Definitions

- ***A Process*** is a logical sequence of tasks performed to achieve a particular objective. A process defines *"What"* is to be done.
- ***A Method*** comprises of techniques for performing a task. It defines the *"how"* of each task.
- ***A Tool*** is an instrument or software suite, when applied to a particular method, can enhance a particular task.
- ***A Methodology*** is a collection of related processes, methods and tools. Essentially a *"recipe"* for solving a class of problems.

Essential Components in MBSE Methodology

- In order to shift from doc-centric to model-centric, it requires:
  - A language
  - A tool
  - A method

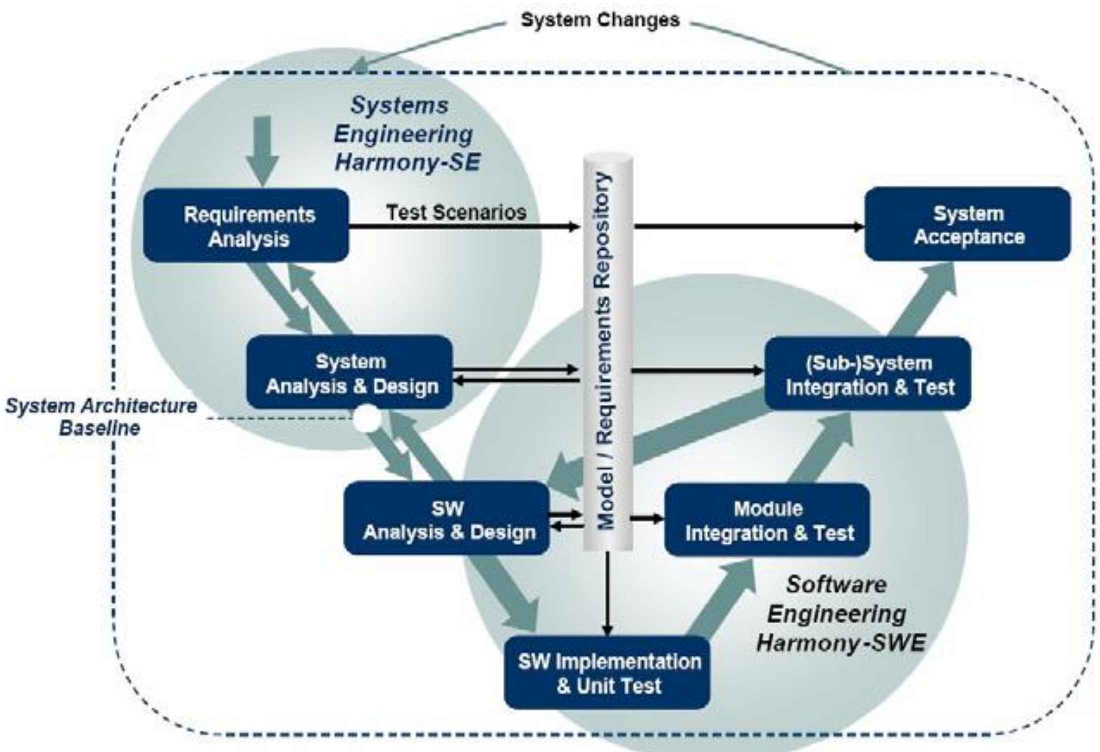
MBSE Language and Tool

- **SE Language**
  - **Systems Definition Language**
    - Unique to CORE and GENESYS from Vitech
  - **OMG SysML**
  - **Object Process Language (OPL)**
    - Unique to the Dori Object-Process Method (OPM)

- **Some SysML Tool Vendors:**
  - Integrity Modeler (PTC);
  - Cameo Systems Modeler (No Magic);
  - CORE and GENESYS (Vitech);
  - Enterprise Architect (Sparx Systems); and
  - Rational System Architect (IBM).

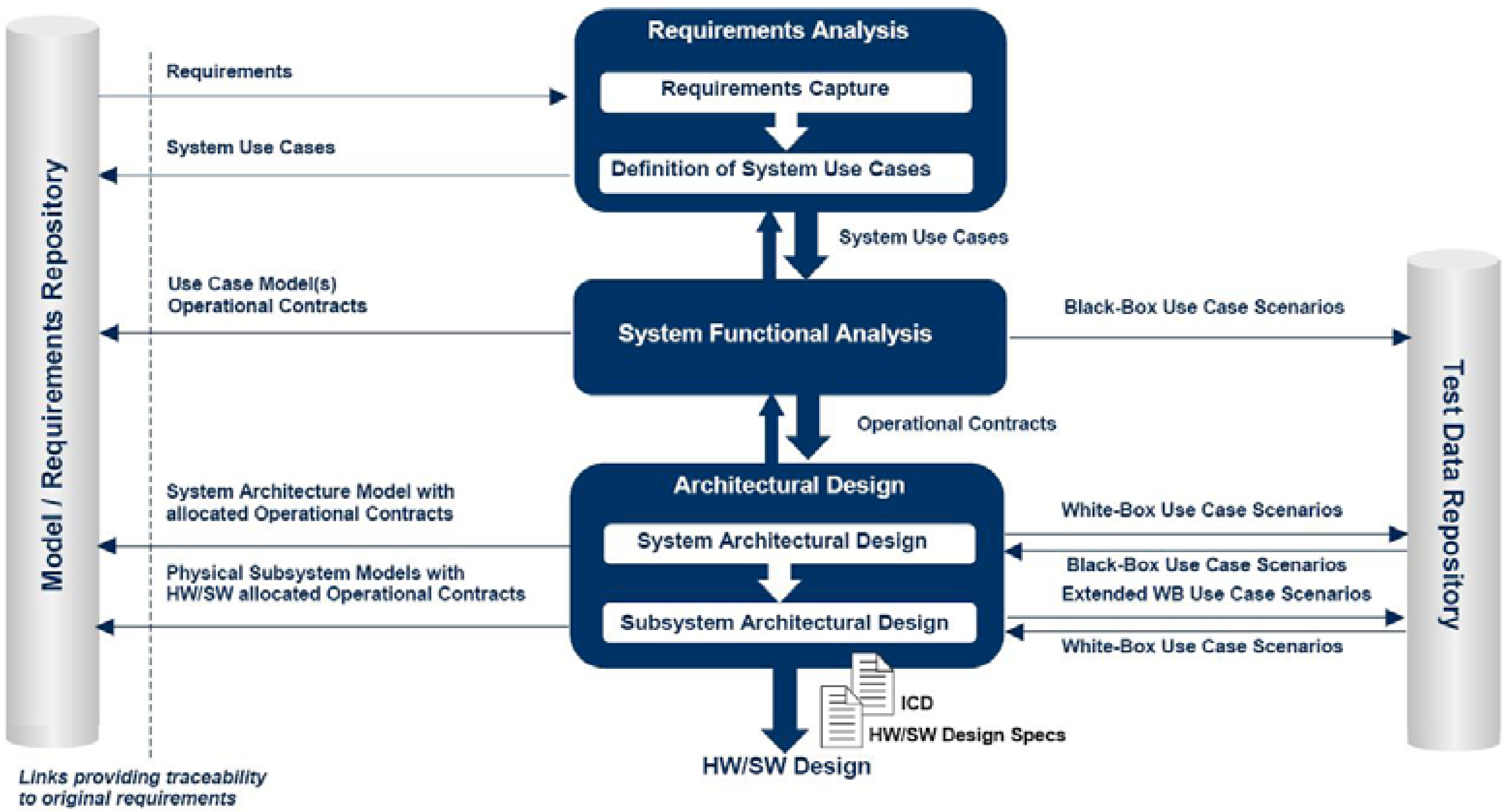
IBM Telelogic Harmony SE

- Harmony SE is a subset of a larger integrated system and software development process known as Harmony®.
- Harmony SE and Harmony were originally developed by I-Logic, Inc.

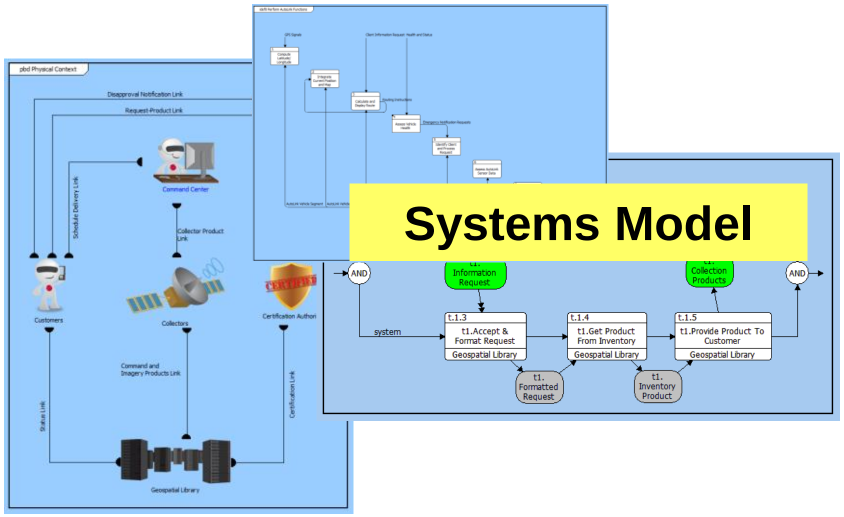


IBM Telelogic Harmony SE

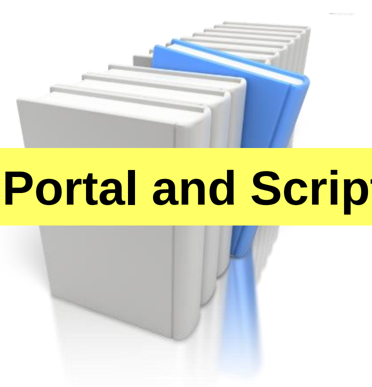
Harmony-SE Process Elements



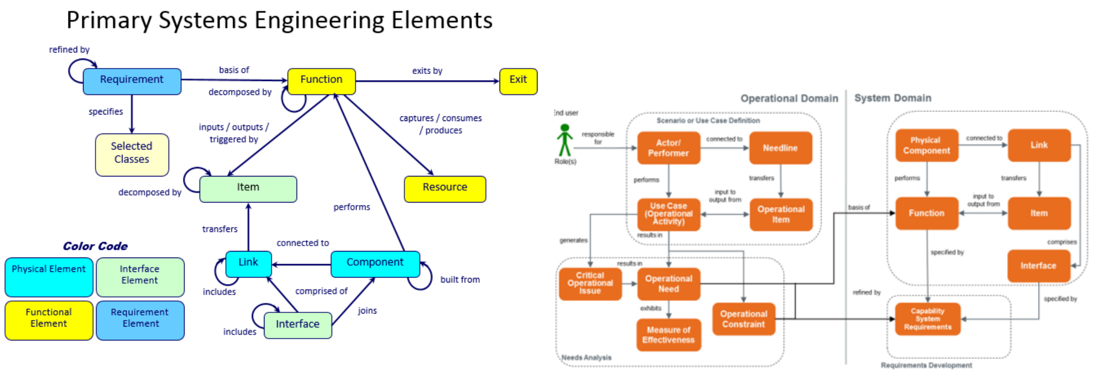
Vitech MBSE Methodology



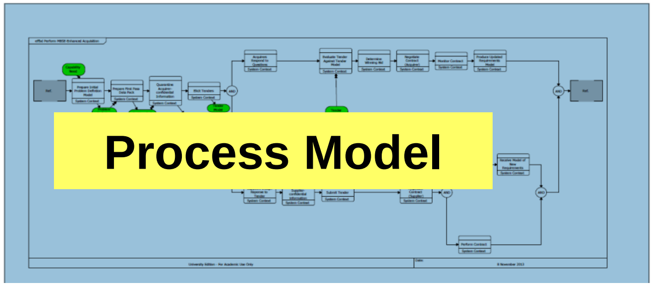
Systems Model



Portal and Scripts

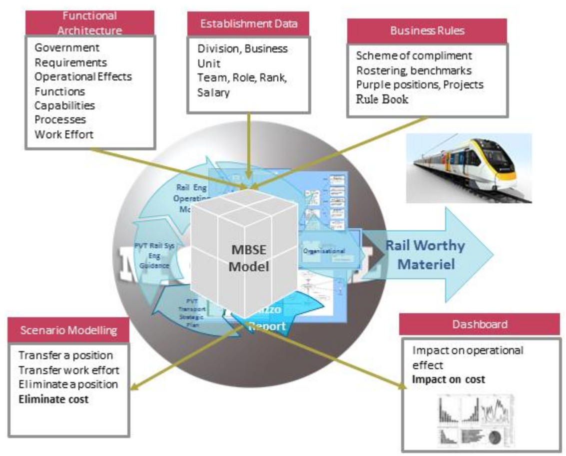


Schema/Metamodel



Process Model

Modeling Principles and Concepts



Modeling Principles Concepts

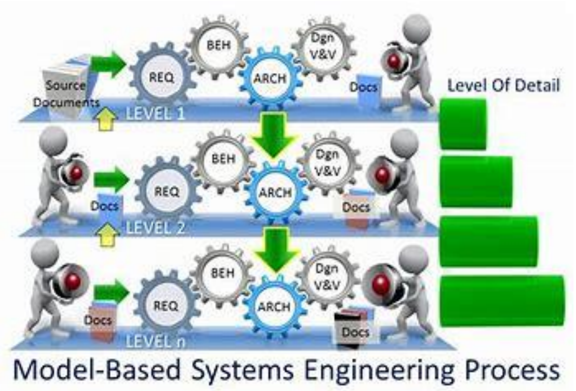
- Models, views, and diagrams
- General modeling concepts for managing complexity
- Architecture
- System Partitioning
- Contrasting Functional vs Object-Oriented System decomposition
- Use Case Flowdown

Models, Views, and Diagrams

- A system model provides a representation of the physical system and its environment
  - includes the semantics and notation
  - can be graphically represented by one or more diagrams
- A viewpoint is the perspective of a set of stakeholders that reflects the stakeholder concerns
- A view is intended to represent the model from a particular viewpoint
- Diagrams, tables, etc. can be used to describe a view

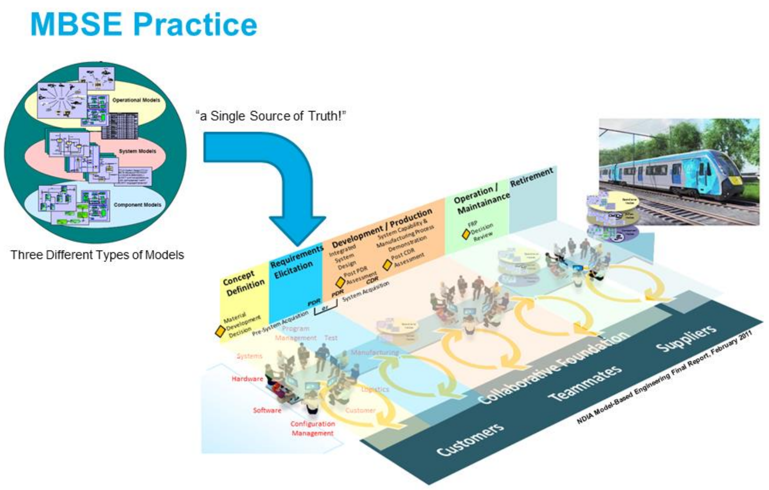
General Modeling Concepts For Managing System Complexity

- Separation of concerns
  - Avoid mixing of independent concerns
- Abstraction
  - Dealing with only what is of interest and deferring unnecessary detail
- Level of decomposition
  - focus on appropriate level of the system hierarchy (I.e. system, component)



General Modeling Concepts For Managing Complexity (cont.)

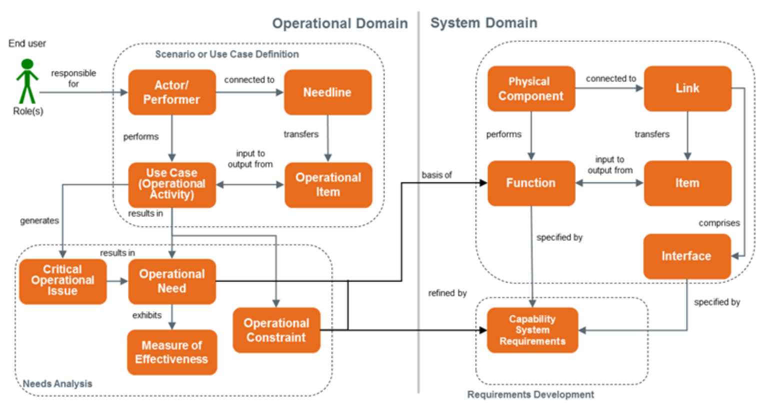
- Information hiding
  - Limiting visibility and access to the interfaces and hiding the detail
- Generalization/specialization
  - A taxonomy that specializes the elements by sharing common features and adding unique features
- Instantiation
  - Unique identification of a member of a class



Architecture

- The inter-relationship among the components of a system
  - Some definitions includes the guidelines for constructing the architecture

- Architecture views reflect different viewpoints (stakeholder perspectives)
  - Operational architecture
  - Functional architecture
  - Physical architecture
  - Software architecture
  - Data architecture
  - Security architecture
  - …



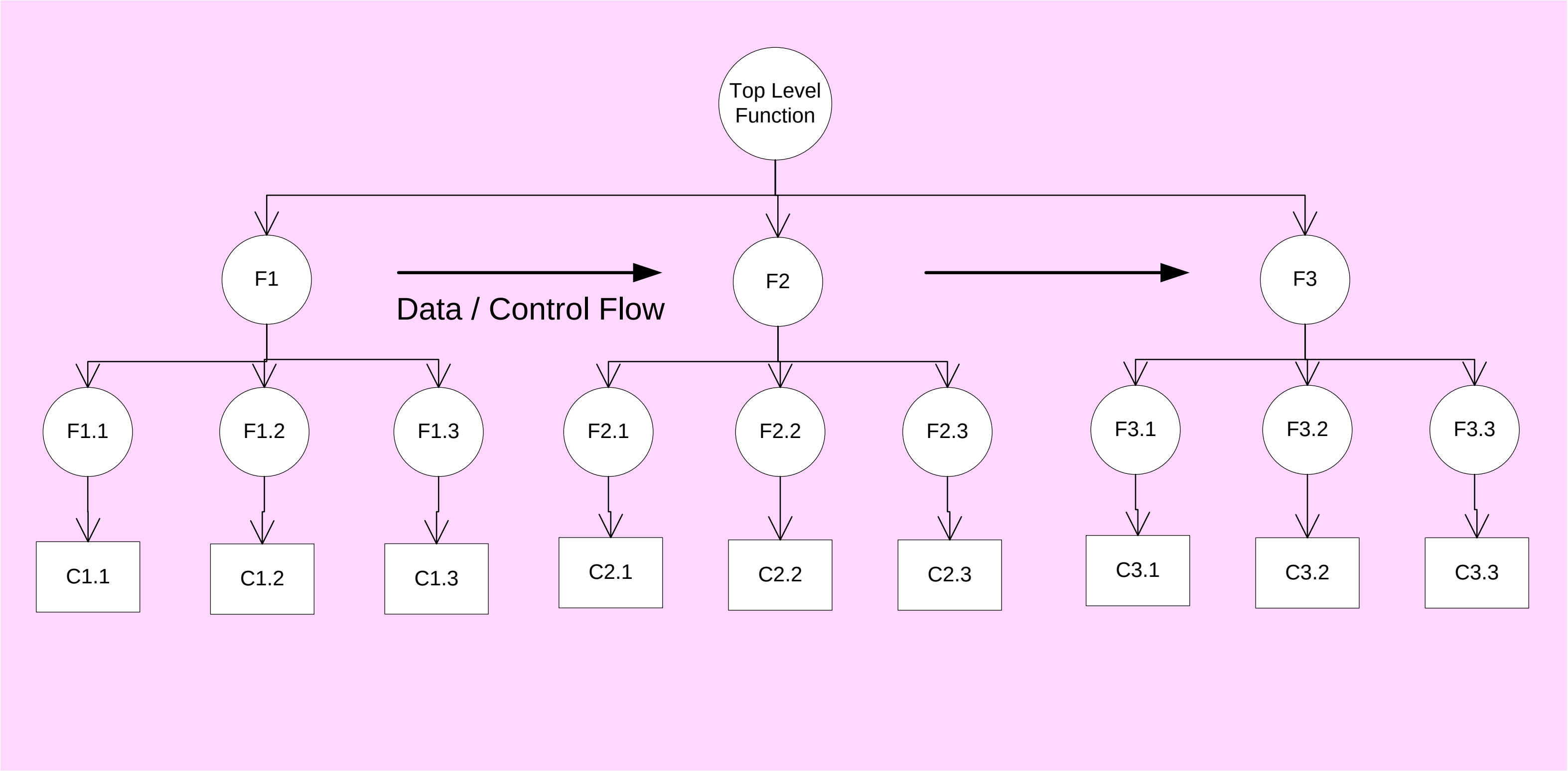
System Partitioning

- Critical aspect of architecture development
- Partition system into replaceable components
  - Logical components that are independent of technology and implementation
  - Physical components that address the functionality of the logical components and include implementation constraints
- Replaceable components should:
  - Include well defined interfaces
  - Be modular and cohesive
  - Can be further decomposable

System Partitioning (cont.)

- Repartitioning (aka refactoring) of functionality is done to separate concerns
- Architecture layers are a form of partitioning of services to minimize impact of changes

Traditional Functional Decomposition and Allocation to Components



Contrasting Functions, Logical, & Allocated Components

- Functions
  - Defines what a system/component does
  - Includes I/O and control
  - Decomposed into lower level functions
- Logical components
  - Derived from decomposition of system class
  - Performs a set of functions (operations) based on partitioning criteria
  - Includes state information
  - Technology/implementation independent
  - Abstraction of physical (allocated) component

Contrasting Functions, Logical, & Allocated Components (Cont.)

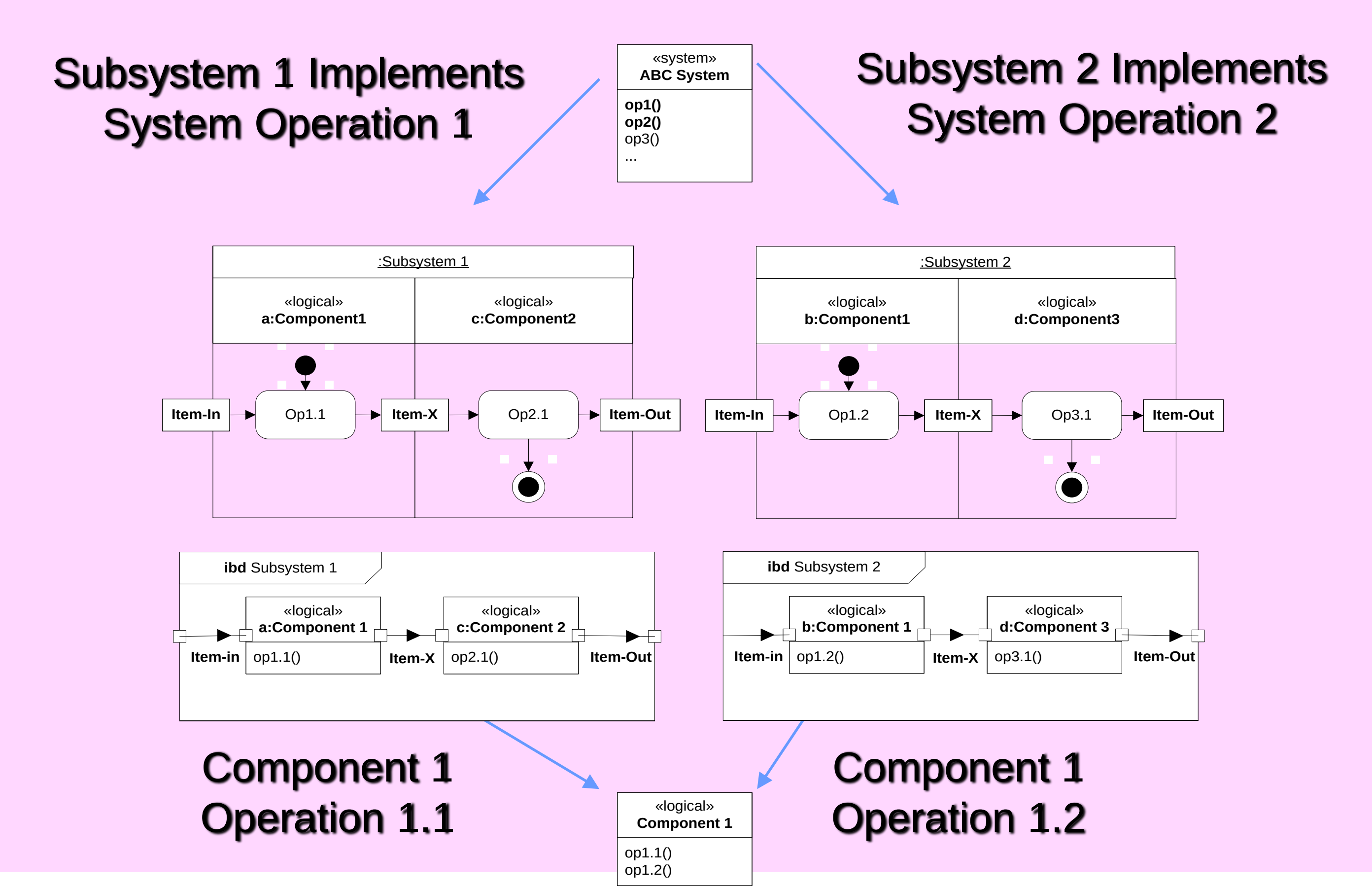
- Allocated components
  - Implement operations from one or more logical components
  - Includes physical/implementation constraints, such as weight, size, ..
  - Represents requirements for physical components

Use Case Flowdown

- Use cases at one level of system hierarchy correspond to goals or req'd capabilities for the next lower level to realize
  - Top level mission use cases are based on the desired mission capabilities
  - An enterprise class operation(s) is realized by the system and other external systems which are part of the enterprise
  - Similarly, an operation(s) of the system block is realized by its components

Flowdown of System Operations To Subsystems and Components

- Each Subsystem implements a single system operation
- A component can support multiple subsystems



Program Completed

Missouri University of Science & Technology