ENES 489P Hands-On Systems Engineering Projects

Strategies of Systems Engineering Development

Mark Austin

E-mail: austin@isr.umd.edu

Institute for Systems Research, University of Maryland, College Park

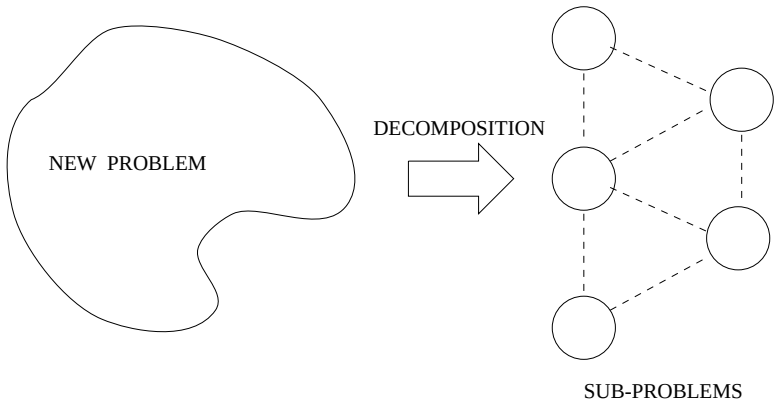
Topic 3: Systems Engineering Development

Topics:

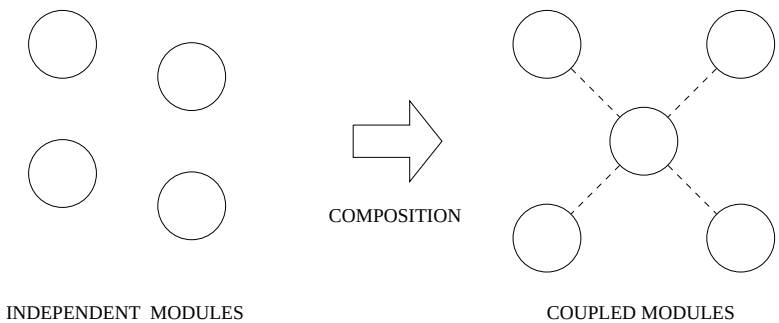
1. Top-down and bottom-up development.
2. Guidelines for design decomposition.
3. Established strategies of development.
4. Course Preview

Top-Down and Bottom-Up Development

Top-down design (decomposition)



Bottom-up development (synthesis)



Top-Down and Bottom-Up Development

Advantages/Disadvantages of Top-Down Decomposition

- Can customize a design to provide what is needed and no more.
- Decomposition simplifies development – lower-level (sub-system) development may only require input from a single discipline.
- Start from scratch implies slow time-to-market.

Advantages/Disadvantages of Bottom-up Development

- Reuse of components enables fast time-to-market.
- Reuse of components improves quality because components will have already been tested.
- Design may contain (many) features that are not needed.

Example of Top-Down Development

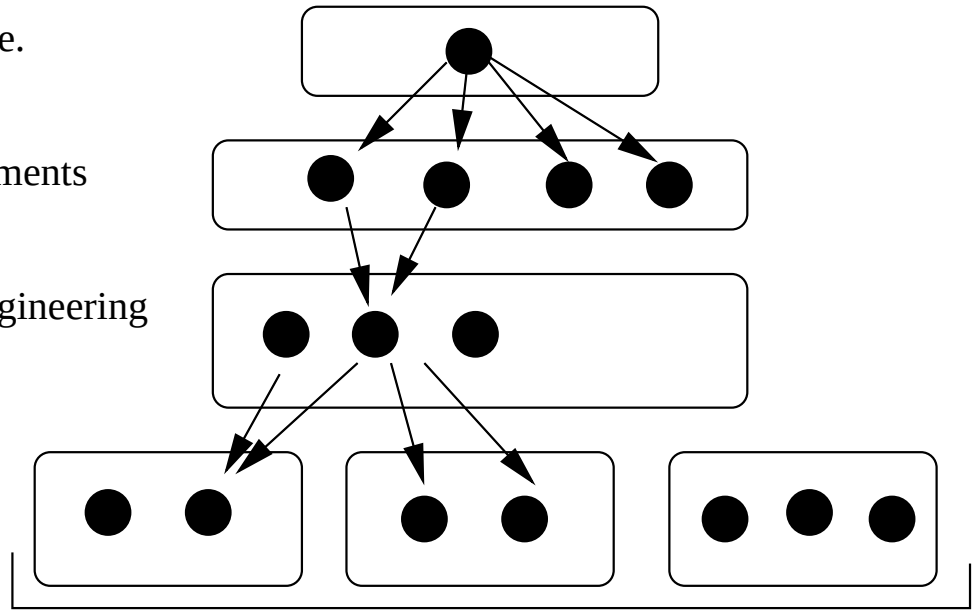
Example 1. Layered development and organization of requirements at NASA Goddard.

Level 0 −− Mission Objective.

Level 1 −− Science Requirements

Level 2 −− System−level engineering requirements

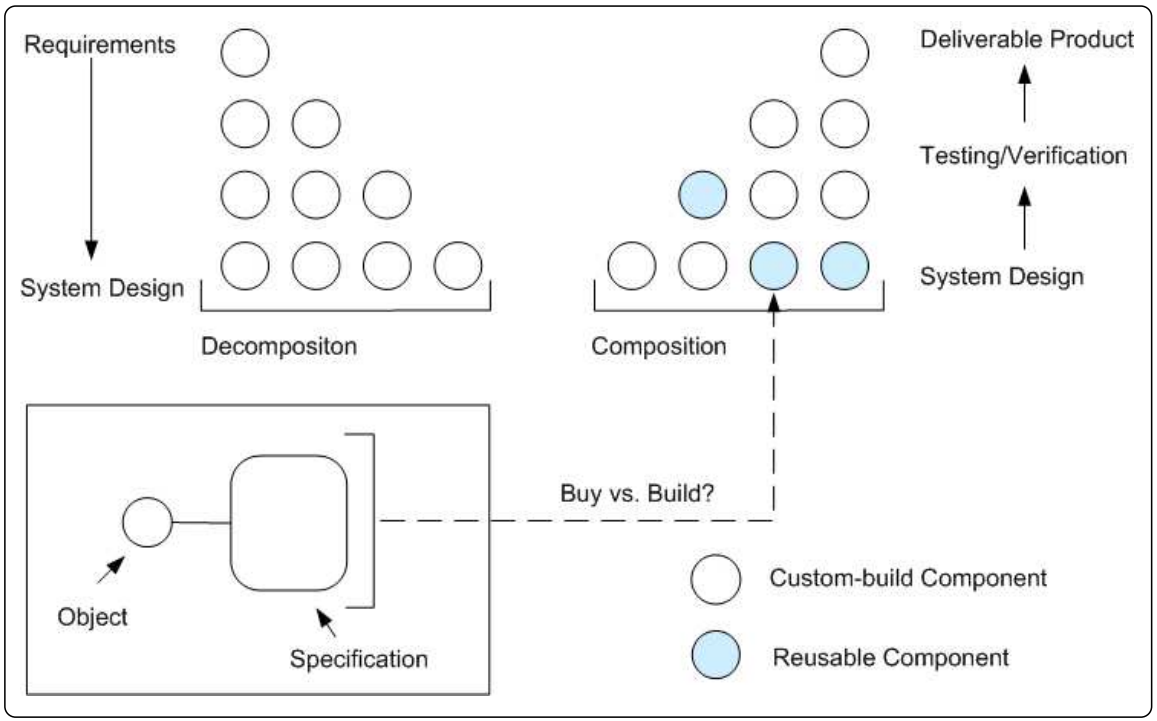
Level 3 −− Sub−system requirements



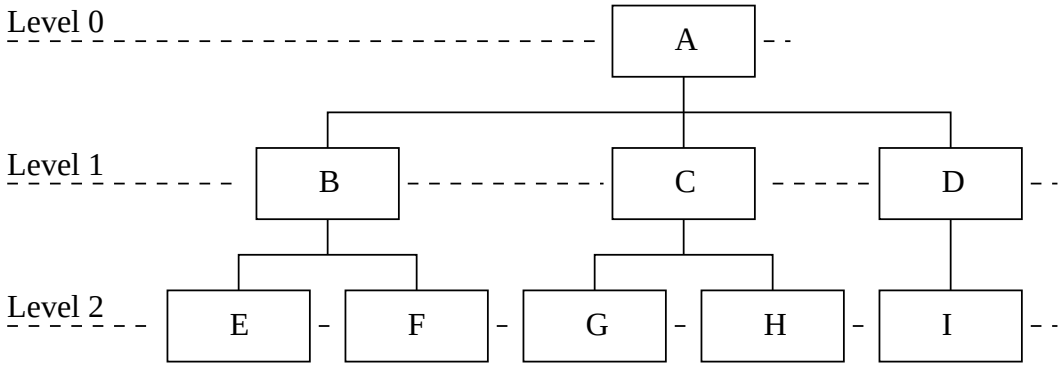
Requirements are organized into clusters for team development.

Example of Top-Down Development

Example 2. Top-down decomposition and bottom-up synthesis coupled to reuse of objects/sub-systems.



Guidelines for Design Decomposition



Guidelines for the design of modules are:

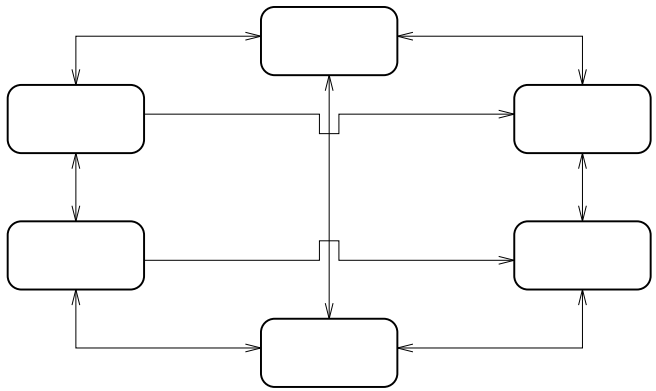
1. One module should have no more than seven subordinate modules.
2. There should be separation between the controller modules and the worker modules.
3. Every module must perform a task appropriate to its place in the hierarchy.
4. Modules should only receive as much information as they need.

The motivation for following these guidelines is modules that will be functional, easy to understand, testable, and reusable.

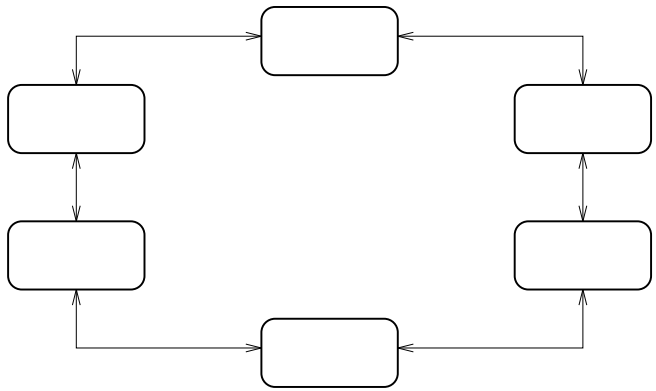
Guidelines for Design Decomposition

Coupling

Coupling is a measure of the interface complexity (or degree of interdependence) between modules.



HIGHLY - COUPLED SYSTEM



LOOSELY - COUPLED SYSTEM

In design, we should:

1. Keep the interfaces as minimal as possible;
2. Keep the interfaces as simple as possible, when in fact, one must exist.

Guidelines for Design Decomposition

Cohesion

Cohesion is a measure of how well the components of a module are related to one another (put another way, cohesion is a measure of the functional association of the element within an element).

In design, we should:

1. Keep related functions together;
2. Keep unrelated functions apart.

Coupling and cohesion work together

Generally speaking, ...

> **...modules with components that are well related will have the capability of plugging into loosely coupled systems.**

Guidelines for Design Decomposition

Module Complexity

Modules should be kept as simple as possible, and hide the details of implementation from the outside environment.

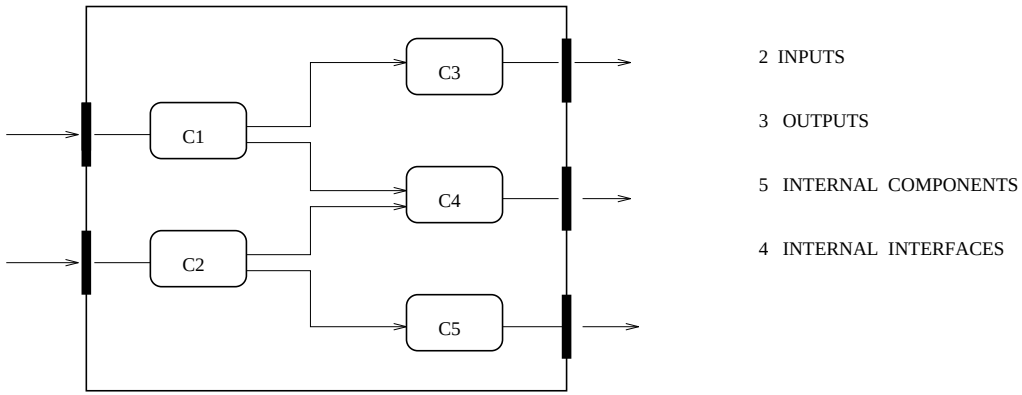


Figure 1: Elements of module complexity.

Three factors that contribute to module complexity are: (a) its size, (b) the number of internal functions and connections within the modules, and (c) the number of interfaces to the modules.

Established Strategies of Development

Strategy 1. Simplify Design through Separation of Concerns

Complex systems are often characterized by ...

> **... many components, intertwined network structures, concurrent behaviors, and complicated communications and interactions among subsystems and components.**

To facilitate understanding of these design issues/concerns, we aim to

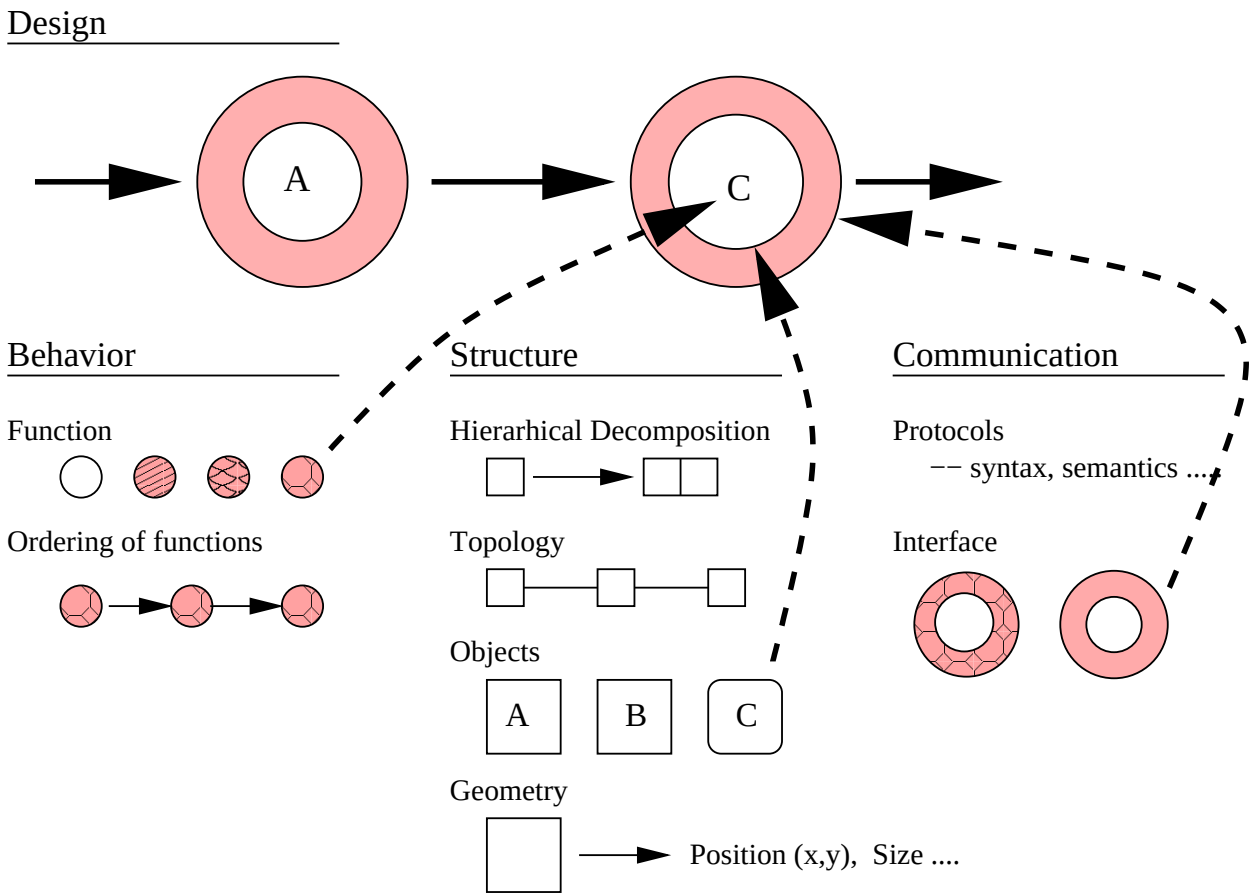
> **... pull a design apart and examine it from perspectives (or ”facets” or viewpoints) that are almost orthogonal, thereby factoring out so-called cross-cutting concerns.**

Achieving (almost) orthogonality of concerns is important because ...

> **... it means we can explore options in one viewpoint (or dimension of design) without affecting other concerns.**

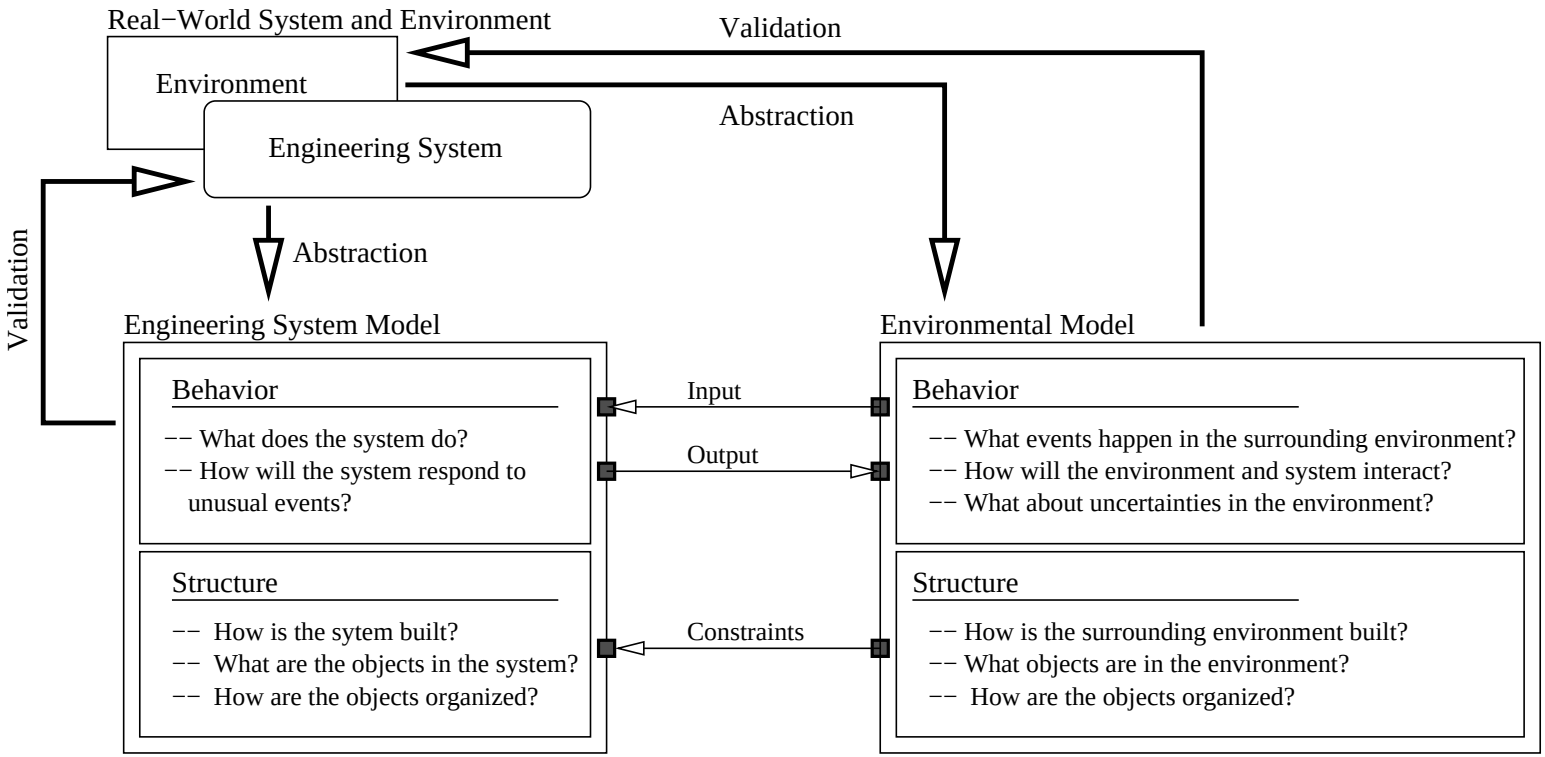
Established Strategies of Development

Example 1. Separation of concerns (e.g., structure, behavior, communication) in simple network.



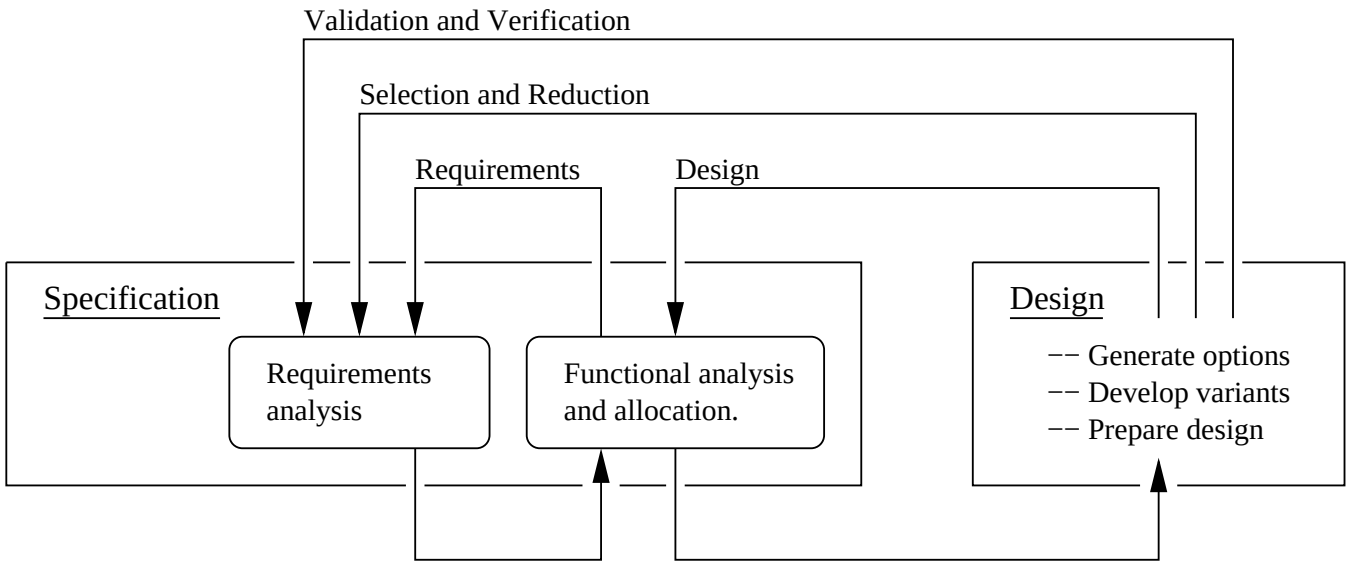
Established Strategies of Development

Example 2. Synthesis of models for engineering system and surrounding environment.



Established Strategies of Development

Example 3. Separation of SE activities/products – requirements, design and validation results.



Established Strategies of Development

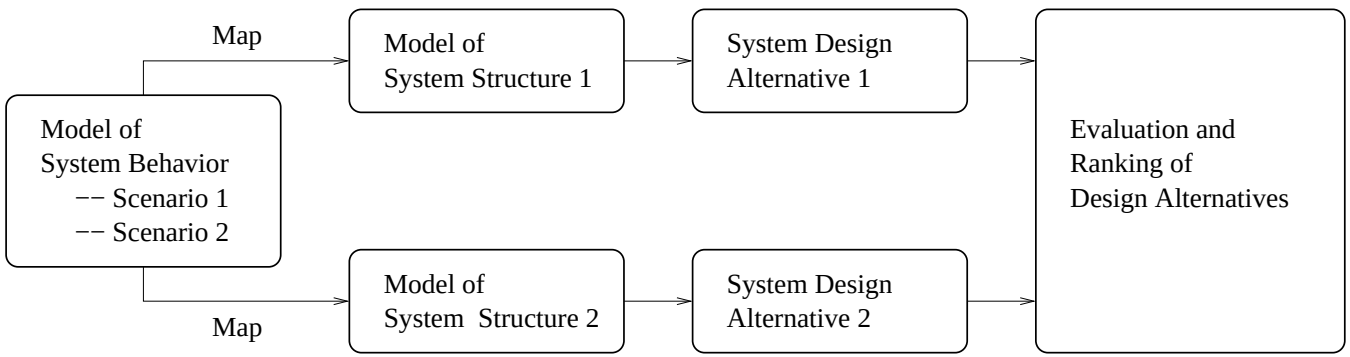
Strategy 2. Function before Physical

We promote the description of systems in two orthogonal ways:

- The function that the systems is intended to provide,
- Candidate architectures for realizing the functionality.

Function-Architecture Co-Design

Map models of system behavior onto system structure alternatives.



Identify measures of effectiveness. Then evaluate and rank design alternatives.

Established Strategies of Development

Benefits of Function-Architecture Co-Design

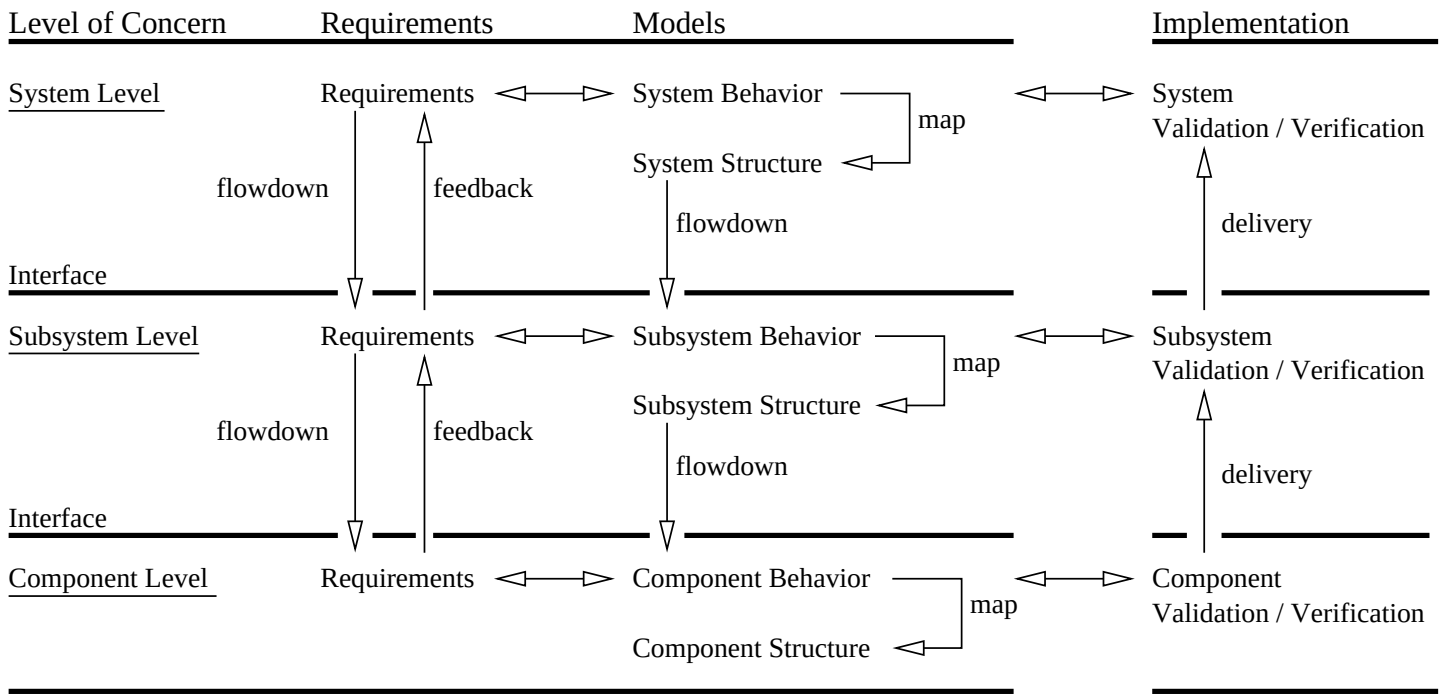
Lessons learned in industry indicate that a clean separation of system functionality and system architecture enables:

1. Much easier modifications of the design at the system level,
2. More effective exploration of alternative solutions.
3. Reuse of major components.

Established Strategies of Development

Strategy 3. Layered Approach to Development

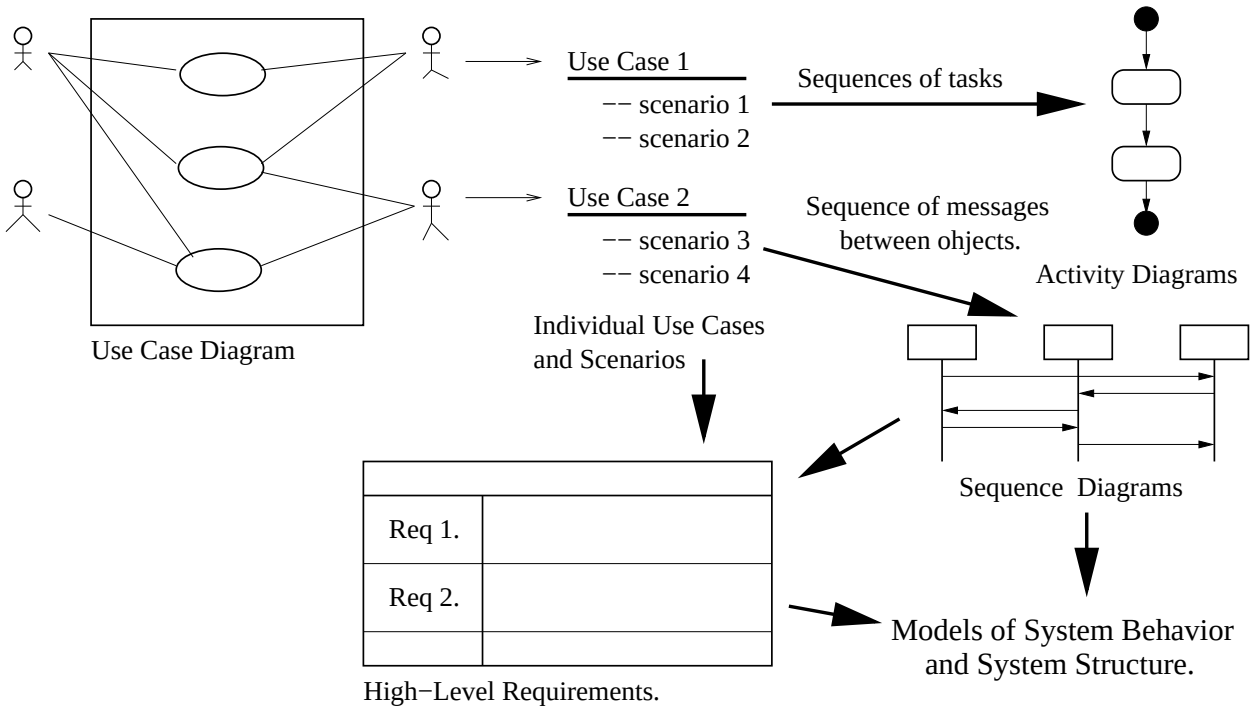
The tenet of “breadth before depth” leads to a layered approach to development.



Course Preview

Problem Definition. Development of an Operations Concept.

Pathway from goals and scenarios to simplified models of behavior and requirements.



Course Preview

Key Points:

- The functional description dictates what the system must do.

  Here, we employ a combination of use cases (and use case diagrams), textual scenarios, and activity and sequence diagrams to elicit and represent the required system functionality.

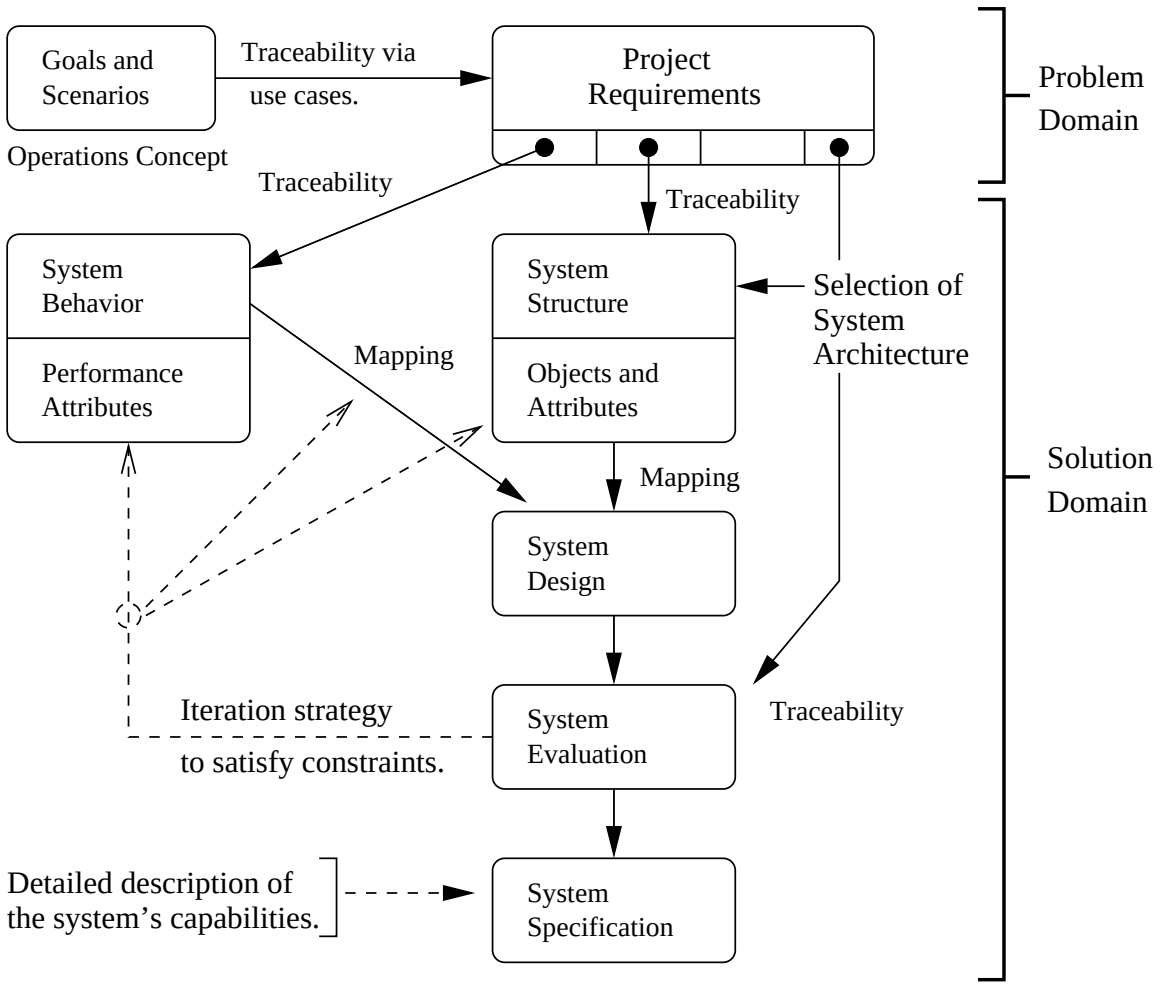
- A complete system description will also include statements on minimum levels of acceptable performance and maximum cost.

  Since a system does not actually exist at this point, these aspects of the problem description will be written as design requirements/constraints.

- Further design requirements/constraints will be obtained from the structure and communication of objects in the models for system functionality (e.g., required system interfaces).

Course Preview

Problem Solution. Pathway from Requirements to Models of System Behavior/Structure and System Design



Course Preview

Key Points:

- Requirements are organized according to the role they will play in the system-level design.
- Models of behavior specify **what** the system will actually do.
- Models of structure specify **how** the system will accomplish its purpose.
- The nature of each object/subsystem will be captured by its attributes. Attributes includes:
  - The attributes of the physical structure of the design,
  - The attributes of the environmental elements that will interact the the system.
  - Attributes of the system inputs and system outputs
- We create the system-level design by mapping fragments of system functionality/behavior onto specific subsystems/objects in the system structure.