Requirements → Analysis → System Design → Object Design → Implementation → Testing
System Design – System Decomposition

• Migration from Analysis to System Design.
• Define *Design Goals*
• Design Initial *Sub System* Decomposition.
• *Refine* the Sub System to address the Design Goals.
System

System Design

Object Design

Implementation
Activities of System Design

This phase will produce the following

1. Design Goals
2. Software Architecture
3. Boundary Use Cases, Exceptions, hardware configurations
Design Goal:

• Design Goals come from Non-Functional Requirements.
• Trade Off decisions made.
• Sub System Decomposition is the bulk of the System Design.
• Developers divide the system into manageable parts to reduce complexity.
• Each Sub-System is assigned to a smaller teams.
# Mapping Architecture to SW Engineering

<table>
<thead>
<tr>
<th>Item</th>
<th>Architectural Concept</th>
<th>SW Engineering Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>Components</td>
<td>Room</td>
<td>Subsystem</td>
</tr>
<tr>
<td>Interfaces</td>
<td>Doors</td>
<td>Services</td>
</tr>
<tr>
<td>Non-Functional</td>
<td>Living Area</td>
<td>Response time</td>
</tr>
<tr>
<td>Requirements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functional</td>
<td>Residential House</td>
<td>Use Cases</td>
</tr>
<tr>
<td>Requirements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Costly Rework</td>
<td>Moving Walls</td>
<td>Change Subsystem Interfaces</td>
</tr>
</tbody>
</table>
Design Goals Continued

- Finalize the strategies for
  - Hardware / Software Strategy
  - Persistent Data Management
  - Control Flow
  - Access Control Policy
  - Handling Boundary Conditions
What is a Subsystem?

• **A subsystem** is a **replaceable** part of the system with **well-defined interfaces** that encapsulates the state and behavior of its **contained classes**.

• To reduce the complexity of the **Solution Domain**, a System is divided into **smaller systems** called Subsystem.

• Subsystems are also Called **Categories**
• **Service**: Is a related operations that share a common purpose.

• Define the Subsystem in terms of the services they provide.

• During this phase we define the Services and in the Object Design phase we develop the operations it will provide.
Subsystem Interface:

• A set of operations of a subsystem that are available for other subsystems.

• Subsystem Interface includes
  – Name of the operation
  – Pass through Parameters and data type
  – Return values
  – High level behavior

• Object Design focus on the API (Application Programmer Interface)

• Minimize the information provided about the implementation.
  – Should not reference about the internal data structure.
• Coupling
  – Number of dependencies between two systems.
  – Loosely coupled – relatively independent.
  – Impact analysis.
  – Assembly connectors or Ball-and-Socket Connector

• Cohesion
  – Number of dependencies within a subsystem.
  – Unrelated objects – low cohesion.
Coupling Example

Student Information Display

Class Registration

Graduation Tracker

Student Database
Modified Design to show loosely coupled systems

Student Information Display

Class Registration

Graduation Tracker

Request Translator

Student Database
Cohesion Example

Source: OOSE-Bernd Bruegge & Allen H. Dutoit
Refined Cohesion

Source: OOSE-Bernd Bruegge & Allen H. Dutoit
Questions?
References

• Use Cases Combined with BOOCH, OMT UML Process and Products
  - Putnam P Texel, Charles B Williams

• Object-Oriented Software Engineering Using UML Patterns, and JAVA
  - Bernd Bruegge & Allen H. Dutoit

• Software Engineering 9th Edition
  - Ian Sommerville