

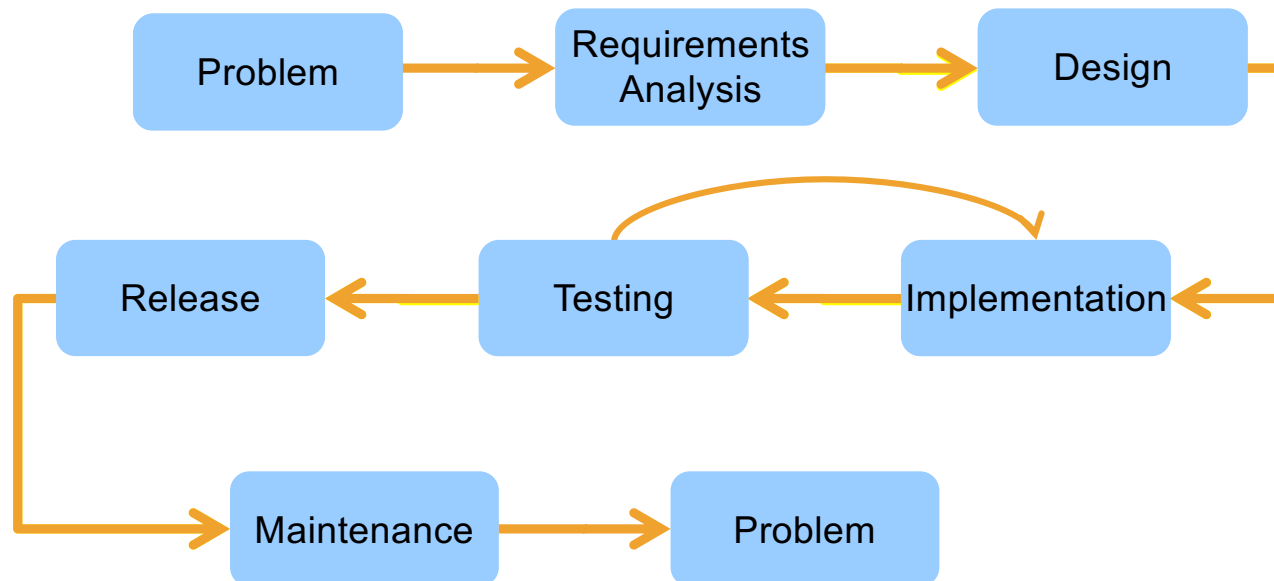
# CS 4320 / 7320

## Software Engineering

Module 5 – Design Basics

# What is the SDLC?

## Where does Design fit?



# Software Design Process

## Two-step Process:

1. **Architectural design** describes how software is organized into components
2. **Detailed design** describes the desired **behavior** of these components

# Architectural Design

1. System Architecture
  1. High level physical systems involved
  2. Connections between systems
  3. Identify dependencies
2. Data Architecture
  1. High level (Entity) descriptions of data
  2. Show the data structure (conceptual) at the entity level
  3. Illustrate critical data flows
3. Process Architecture
  1. Integrate business or user process practices into the architecture
  2. Show the touchpoints between process and technology
4. Software Architecture
  1. Software components
  2. Connections between components
  3. Key Software Subsystems (i.e., messaging hubs, databases, software components)
  4. Some of these components may also show up in the system architecture

# Detailed Design

1. Software components
2. Connections between them
3. Describe behavior down to methods
4. Possibly stub methods out
5. Could include writing test cases for methods

# Software Design Principles

## Key notions

that are the basis for **many different**  
software design approaches and concepts.

# Software Design Principles

## 1. Abstraction

By Parameterization

By Specification

*Procedural Abstraction*

*Data Abstraction*

*Control (Iteration) Abstraction*

# Software Design Principles

## 2. Coupling and Cohesion

*Aiming for **Appropriate Coupling** and **High Cohesion***

## 3. Decomposition and Modularization

*Separate **functionalities** and **responsibilities***

*Well defined **interfaces***



# Software Design Principles

## 4. Encapsulation and Information Hiding

Packaging implementation details **together**

**Restricting direct access** to a component's details

## 5. Separation of Interface and Implementation

**Public interface** separate from implementation details

# Software Design Principles

## 6. Sufficiency, Completeness, and Primitiveness

All **important** abstraction characteristics  
but **nothing more**

Design based on **patterns** that are easy to implement

## 7. Separation of Concerns

Architectural **views specific to a group** of stakeholders

# Key Issues in Software Design

*Important issues that cut across the whole system*

## 1. Concurrency

Think about order, sequence of actions

Concerns: efficiency, atomicity,  
synchronization, scheduling.

# Key Issues in Software Design

## 2. Control and Handling of Events

Organize data and control flow

Handle reactive and temporal events

## 3. Data Persistence

How to handle long-lived data

# Key Issues in Software Design

## 4. Distribution of Components

How to distribute software **across hardware**  
(computer and network)

How components **communicate**

How middleware deals with **heterogeneous software**

# Key Issues in Software Design

## 5. Error and Exception Handling and Fault Tolerance

Prevent, tolerate, and process errors

Deal with exceptional conditions

## 6. Interaction and Presentation

Structure and organize user interaction

How to present information

# Key Issues in Software Design

## 7. Security

Prevent **unauthorized** disclosure, creation, change, deletion of information

Avoid **denial of access** to authorized users

How to **respond to attacks**

***Access control** and proper use of **cryptography***

# References

P. Bourque and R. E. Fairley, Eds., SWEBOK v3.0: Guide to the Software Engineering Body of Knowledge, IEEE, 2014.