CPSC 310 – Software Engineering

Design

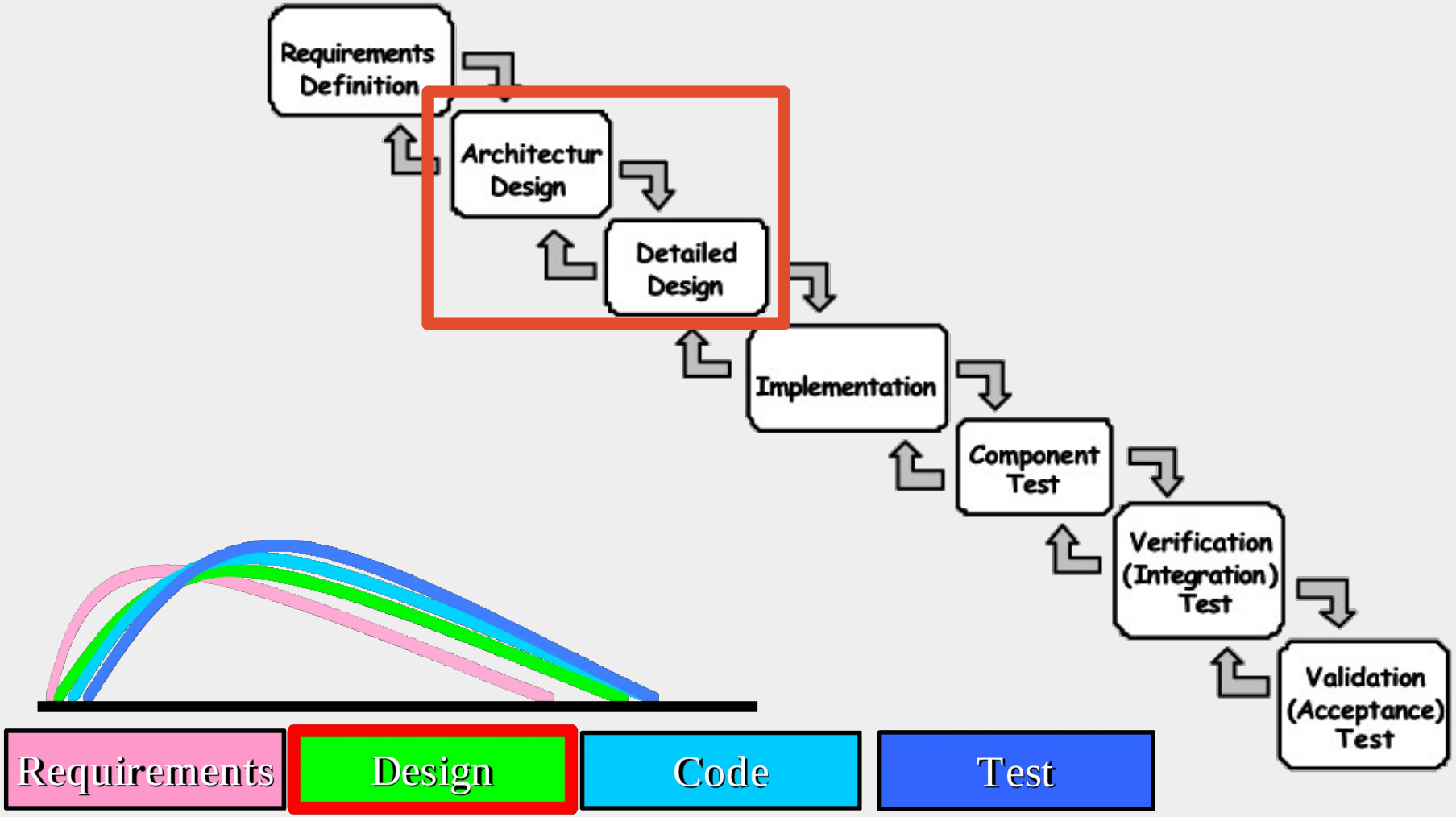
Introduction

Thanks to Krzysztof Czarnecki for some of the slides

Admin

- About last week:
  - Git lab
  - Requirements lecture
- About the lab this week: group project starts
  - Who did not find a group ?
  - There is a prerequisite for the lab

Where does it fit in the process?

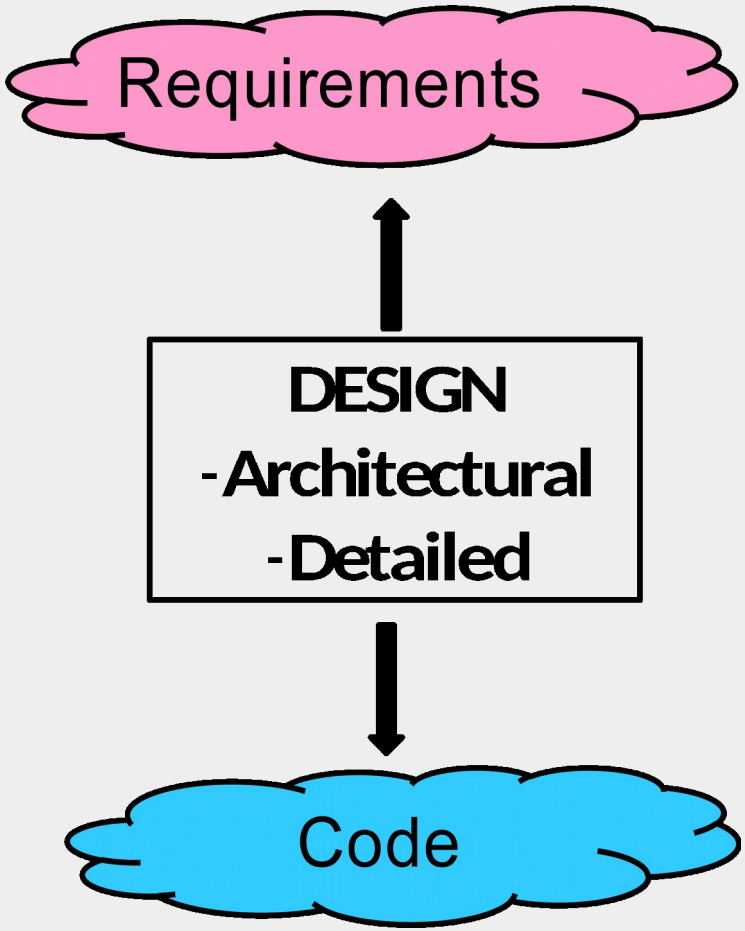


What is design ?

Requirements specification was about the WHAT the system will do

DESIGN is about the HOW the system will perform its functions

Design to Bridge the Gap



Why Design ?

Facilitates communication

Eases system understanding

Eases implementation

Helps discover problems early

Increases product quality

Reduces maintenance costs

Facilitates product upgrade

Cost of not planning…

Cost of not planning…

How to approach Design?

"Treat design as a wicked, sloppy, ***heuristic*** process. Don't settle for the first design that occurs to you. ***Collaborate***. Strive for ***simplicity***. Prototype when you need to. Iterate, iterate and ***iterate again***. You'll be happy with your designs."

McConnell, Steve. *Code Complete*. Ch. 5

How to approach Design?

- Study and understand the problem from different viewpoints
- Identify potential solutions and evaluate the tradeoffs
  - Design experience, reusable artifacts, simplicity of solutions
  - Sub-optimal, but familiar solutions often preferred
    advantages/disadvantages well known
  - Design is about making tradeoffs!
- Develop different models of system at different levels of abstraction and for different perspectives

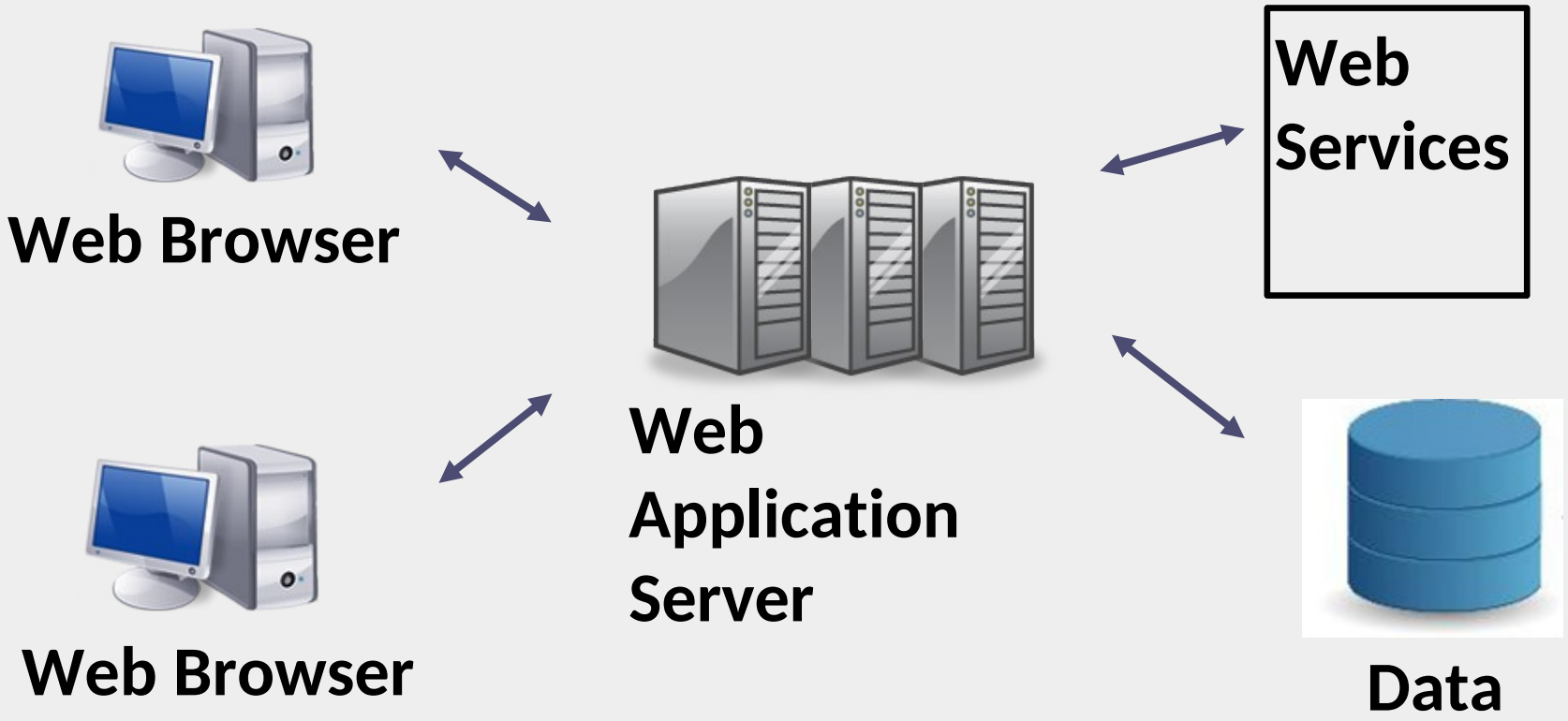
Levels of Design

- Architectural design (high level)
  - Overall structure: main components and their connections
  - Hard to change
  - Discussed more in CPSC 410
- Detailed design (low level)
  - Inner structure of the main components
  - May take the target programming language into account
  - Detailed enough to be implemented in the programming language

Architectural Design

- The architecture of a system describes its gross structure:
  - Main components and their behaviour (system level, sub-systems)
  - Connections between the components / communication (rough idea)
- Architectural Styles: data-flow, client/server,…

Web Architecture (Client / Server Style)

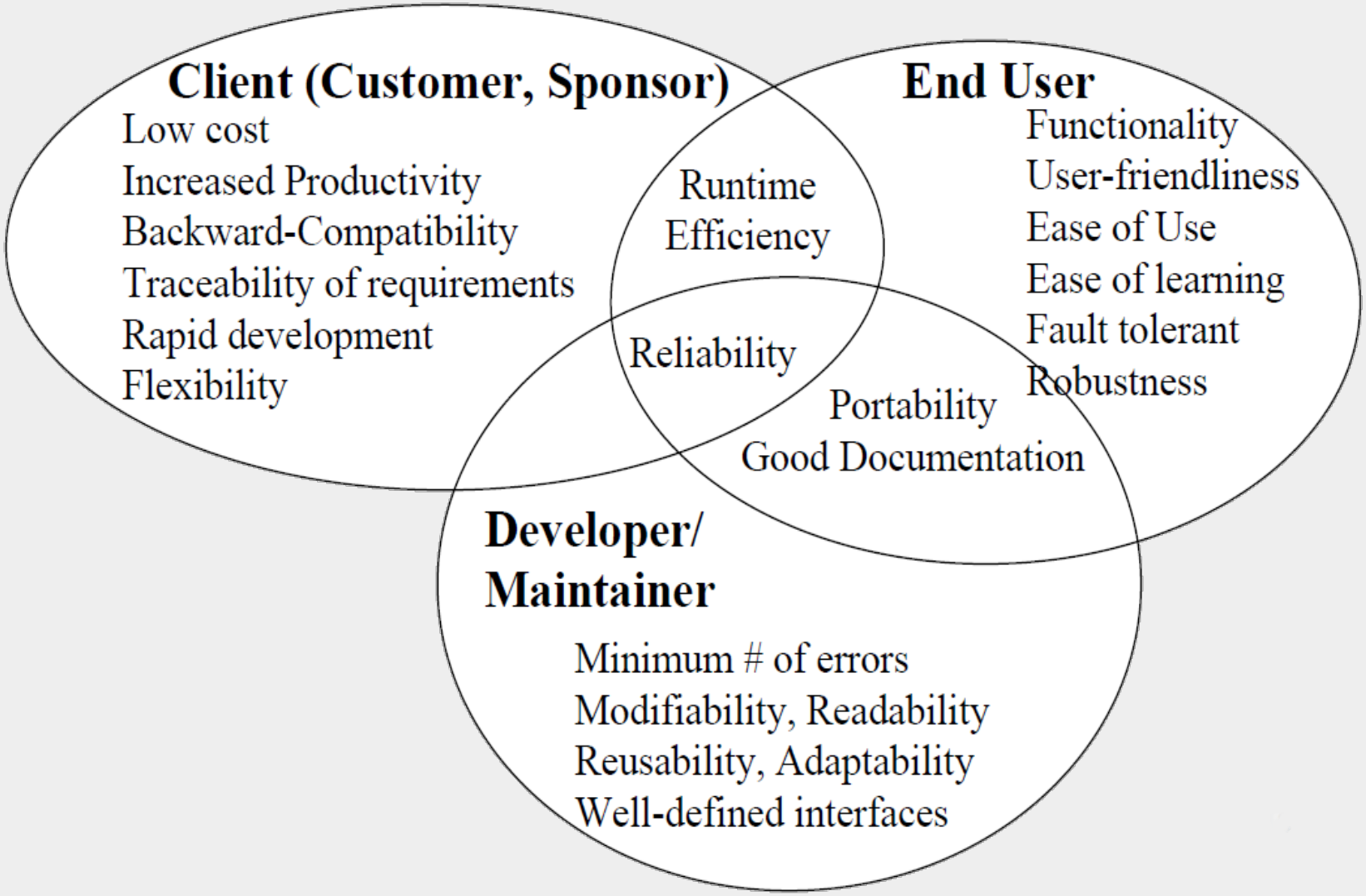


Detailed Design

- Concerned with programming concepts
- Functions, Modules, Classes, Packages
- Files
- Communication protocols
- Synchronization
- …

CPSC 310 Focus

Design Goals



Typical Design Trade-offs

Functionality vs. Usability

Cost vs. Robustness

Efficiency vs. Portability

Rapid development vs. Functionality

Cost vs. Reusability

Backward Compatibility vs. Readability

Challenges in Design

Complexity

Conformity

Changeability

Top-Down vs. Bottom-Up Design

- Top-Down
  - Recursively partition problem into smaller sub-problems
  - Continue until tractable solutions found
  - Note: Not practical for large system in its pure form
- Bottom-Up
  - Assemble, adapt, and extend existing solutions to fit the problem
- In practice: A combination of both
  - Decompose large problems into smaller, but using previous design knowledge
  - Use existing components and solutions
  - Perhaps tackle problematic portions first

Design Methods

- Design methods provide guidance
- Different flavors (more or less formal)
  - Heavyweight methods
    - Highly structured and documentation oriented methods
    - Usually generate mega amounts of graphical documentation
  - Agile methods
    - "Travel light"
  - Agile model-based methods
    - Best of both worlds
    - Still in early development

Design Methods

- Action oriented approach
  - e.g., data-flow design
  - favors the functional view
  - appropriate if actions are the main aspects of a system
- Data oriented approach
  - e.g., Jackson's design method
  - favors the data view
  - appropriate if data are the main aspects of a system
- OO approach
  - looks at both actions and data at the same time
  - system viewed as a collection of objects not functions
  - system state is decentralized – each object manages its own state information
  - objects have attributes defining state and operations which act on attributes
  - conceptually, objects communicate via messages
- Domain-specific approach (DSL, DSML)
  - A set of modeling views and concepts specifically developed for a class of problems

CPSC 310

Design Notation

- Abstraction → decision to select what is important, based on viewpoint.
- Four key viewpoints in software design:
  - Structural - the static properties of the software
  - Behavioral - cause and effect;
  - Functional - what tasks the software performs
  - Data modeling - the data objects used