Software Design and Architecture

Introduction

Things you need to know …

- Books:
  - Software Architecture and Design Illuminated
    - Kai Qian, Xiang Fu, Lixin Tao
  - Object Oriented Systems Analysis and Design using UML
    - Simon Bennett, Steve McRobb, Ray Farmer

Things you need to know …

- The lecture slides provide only the outline of the lecture.
- These outlines are not a substitute for class attendance and note taking.
- More importantly, these outlines are not a substitute for the text book.

Software Process

- A **software process** (also known as software methodology) is a set of related activities that leads to the production of the **software**.

- These activities may involve the **development** of the **software** from the scratch, or, modifying an existing system

Main Phases/Activities of Software Process

- Requirements Analysis (answers "WHAT?")
  - Specifying what the application must do, usually understandable to the layman
- Design (answers "HOW?")
  - Specifying what the parts will be, and how they will fit together
- Implementation
  - Writing the code
- Testing
  - Executing the application with test data for input
- Maintenance
  - Repairing defects and Enhancing capability

Software Process: Personal Finance Example

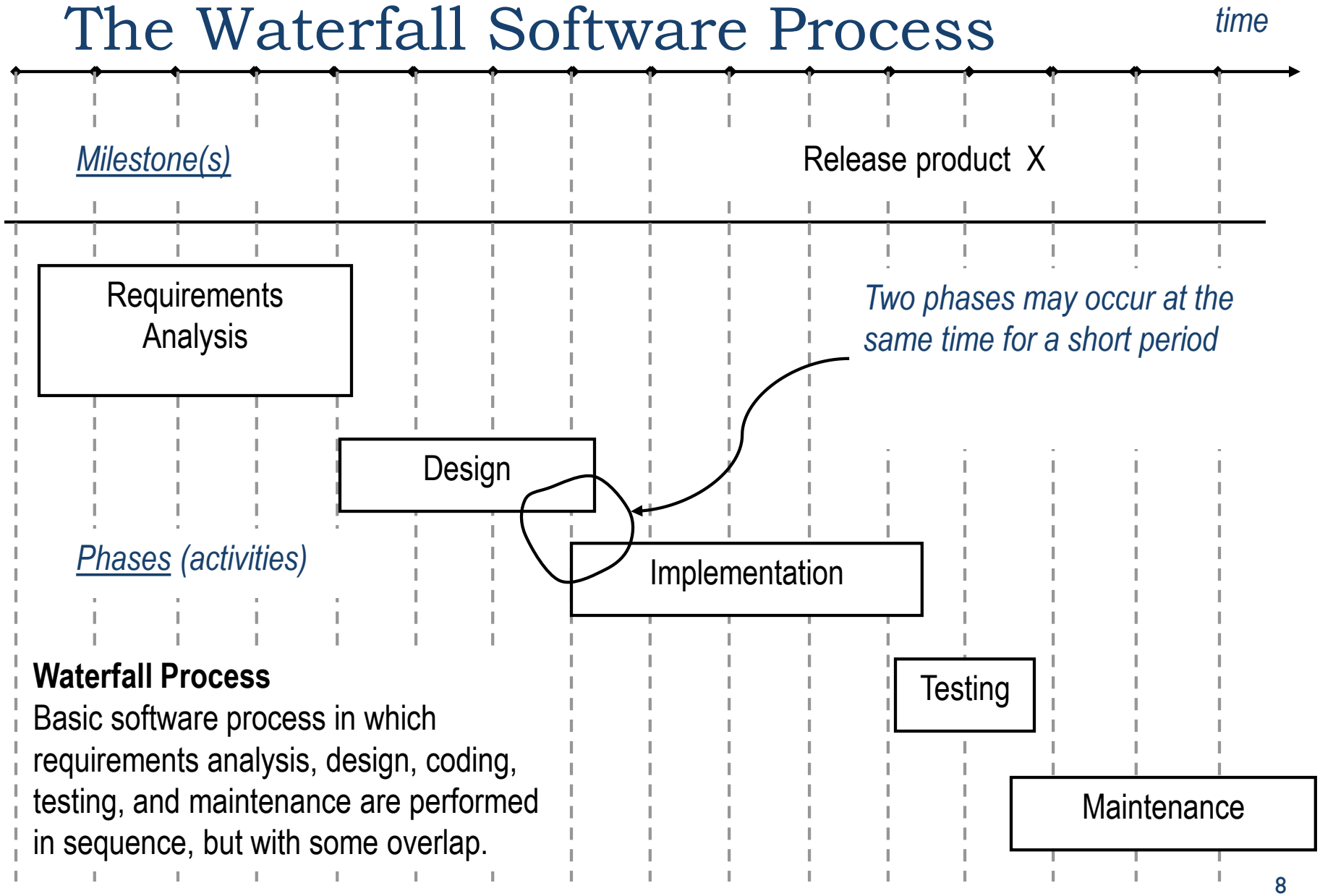
- Requirements Analysis: Text produced
  - e.g., " … The application shall display the balance in the user's bank account. …"
- Design: Diagrams and text
  - e.g., " … The design will consist of the classes CurrentAccount, SavingsAccount, …"
- Implementation: Source and object code
  - e.g., … class CurrentAccount{ double balance; … } …
- Testing: Test cases and test results
  - e.g., "… With test case: deposit $44.92 / deposit $32.00 / withdraw $101.45 / … the balance was $2938.22, which is correct. …"
- Maintenance: Modified design, code, and text
  - e.g., Defect repair: "Application crashes when balance is $0 and attempt is made to withdraw funds. …"
  - e.g., Enhancement: "Allow operation with Euros."

Software Process Model

- A software process model is a simplified representation of a software process. Each model represents a process from a specific perspective.

- These generic models are abstractions of the process that can be used to explain different approaches to the software development

The Waterfall Software Process

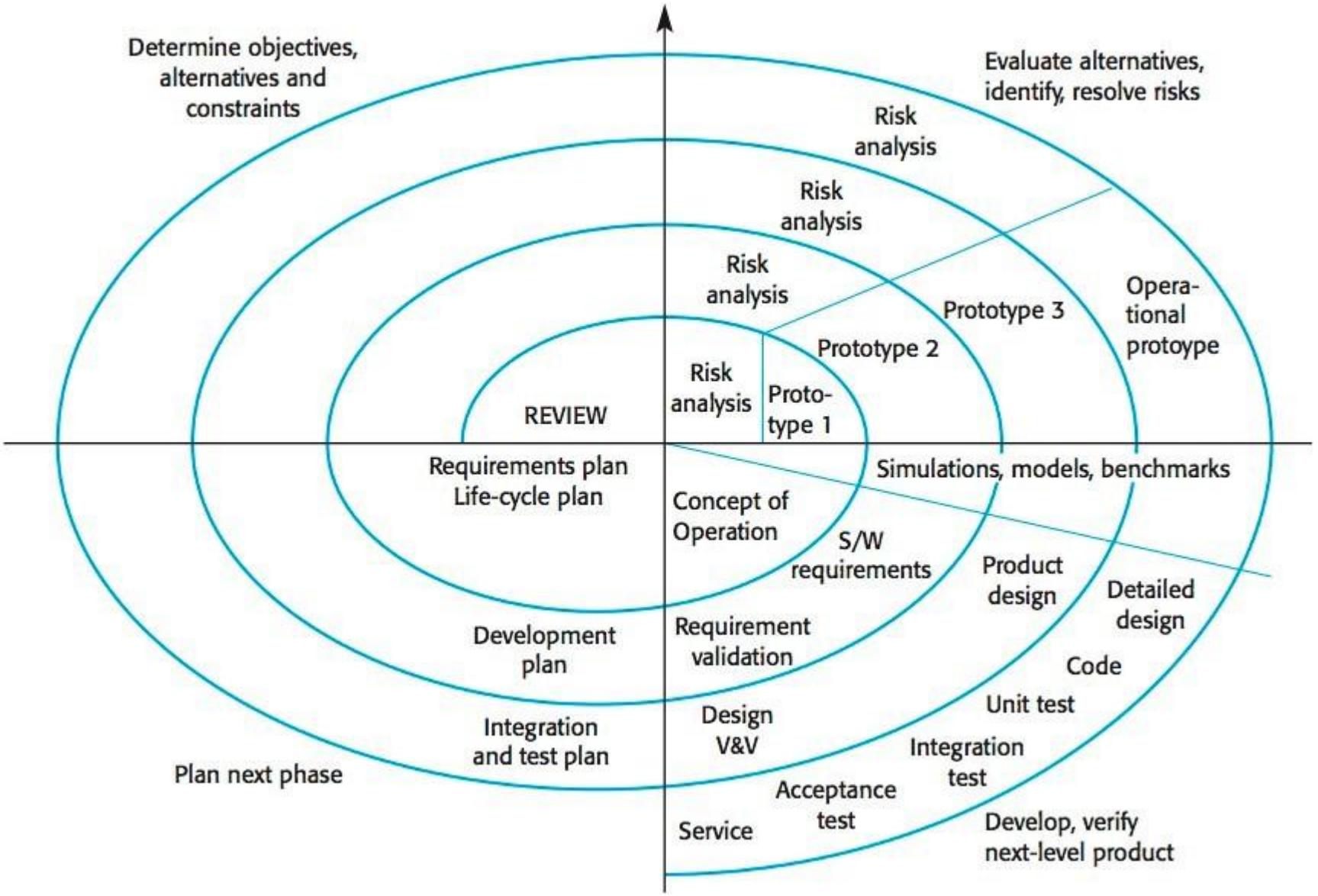


Waterfall Process
Basic software process in which requirements analysis, design, coding, testing, and maintenance are performed in sequence, but with some overlap.

Why a Pure Waterfall Process is Usually Not Practical

- Don’t know up front everything wanted and needed
  - Usually hard to visualize every detail in advance
- We can only estimate the costs of implementing requirements
  - To gain confidence in an estimate, we need to design and actually implement parts, especially the riskiest ones
  - We will probably need to modify requirements as a result
- We often need to execute intermediate builds
  - Stakeholders need to gain confidence
  - Designers and developers need confirmation they're building what’s needed and wanted
- Team members can't be idle while the requirements are being completed
  - Typically put people to work on several phases at once

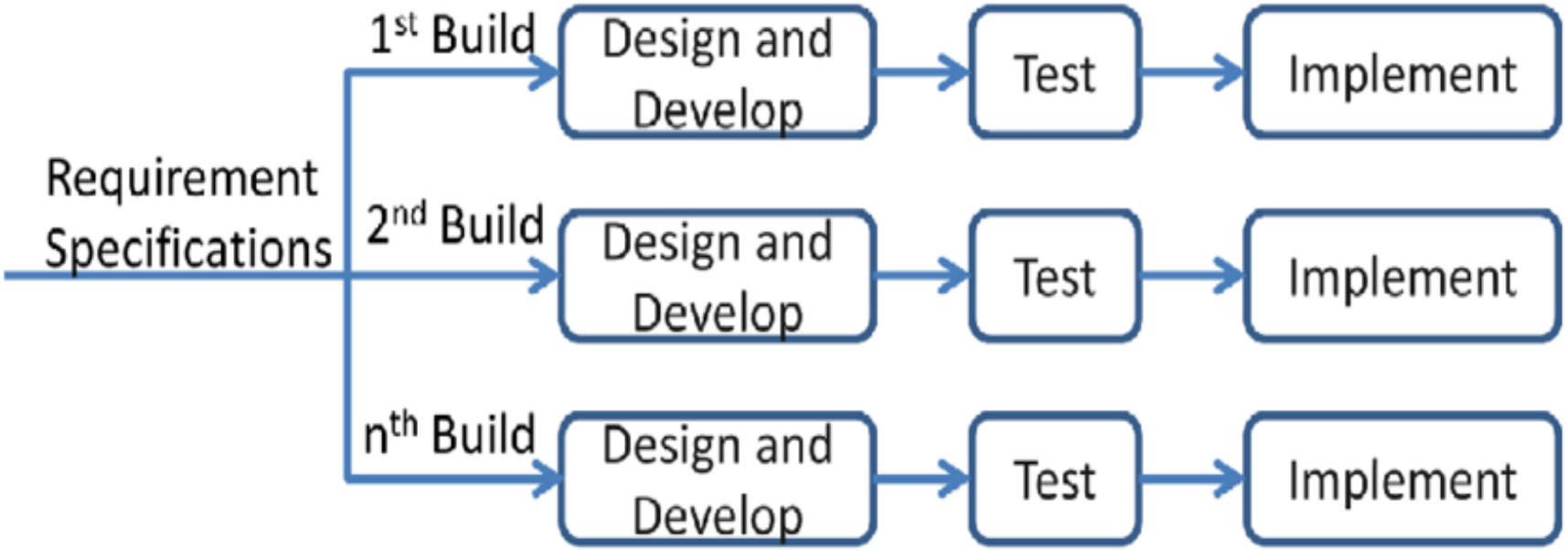
Spiral Process



The Iterative Process

- Iterative development model aims to develop a system through building small portions of all the features, across all components.
- We build a product which meets the initial scope and release it quickly for customer feedback. An early version with limited features important to establish market and get customer feedback.

Iterative Model

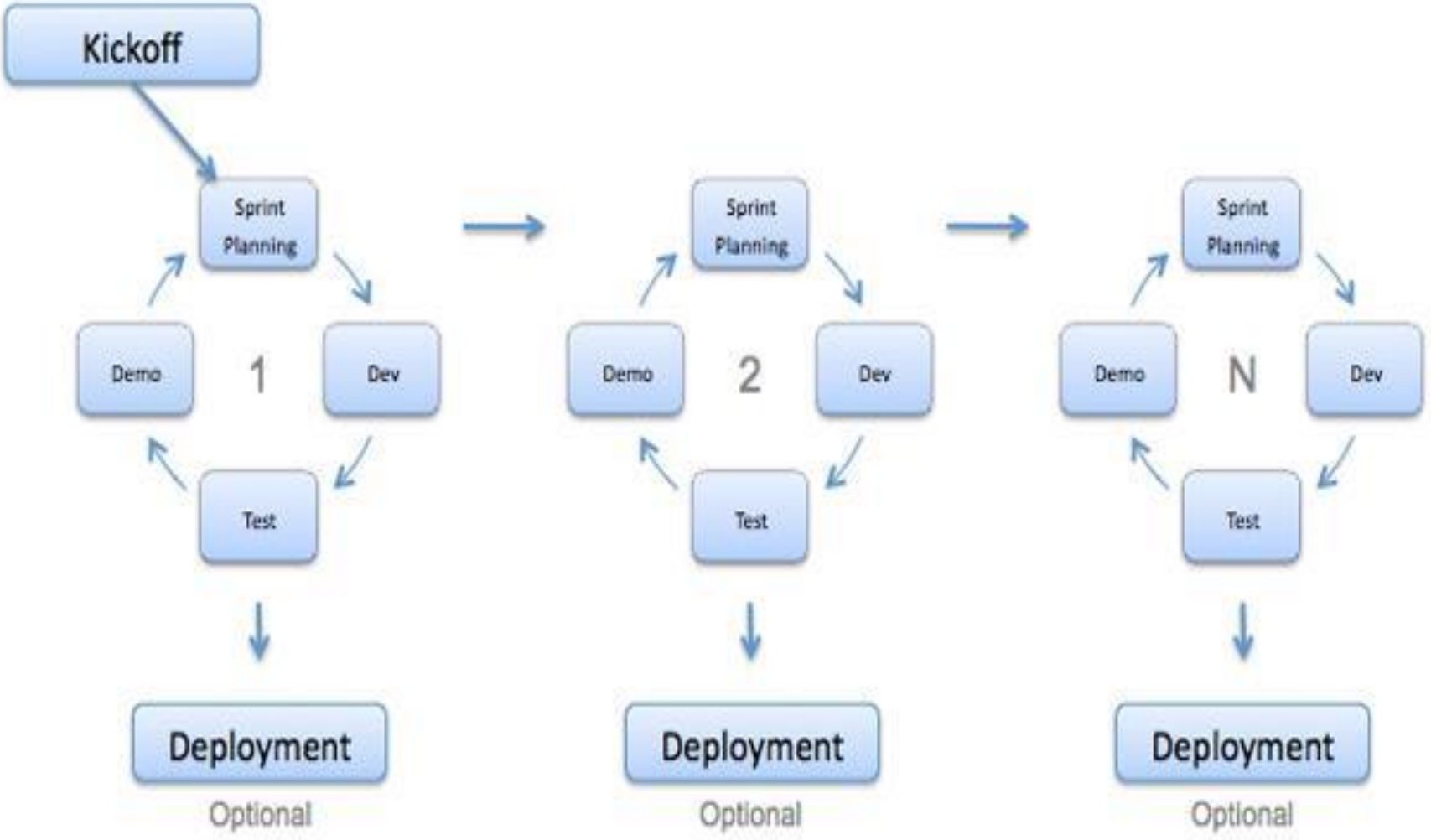


Agile Process

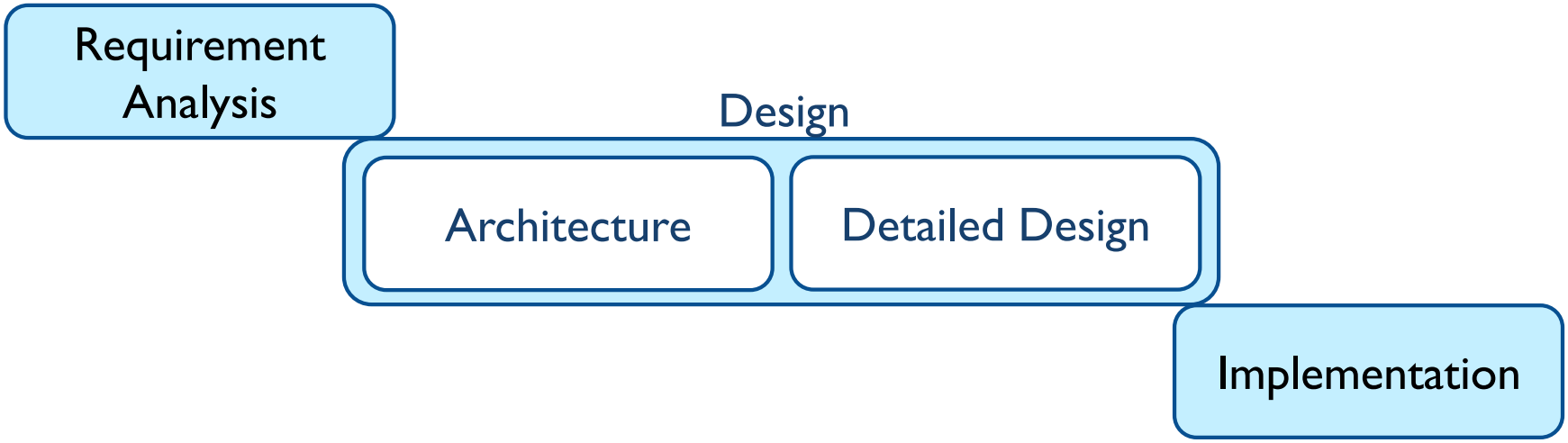
- The agile methods refers to a group of software development models based on the incremental and iterative approach, in which the increments are small and typically, new releases of the system are created and made available to customers every few weeks.

- Each release building on previous functionality. Each release is thoroughly tested to ensure software quality is maintained.

Agile Process



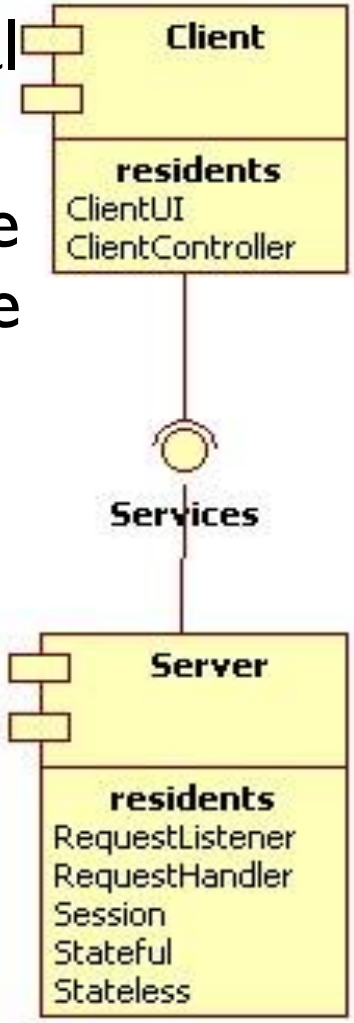
Software Development



- In Software Development, we start by thinking about Architecture, then Detailed Design that leads to Programming.

Software Architecture and Detailed Design

- Architecture addresses the structural aspects of the overall system
- Software Architecture focuses more on the interaction between the externally visible components of the system



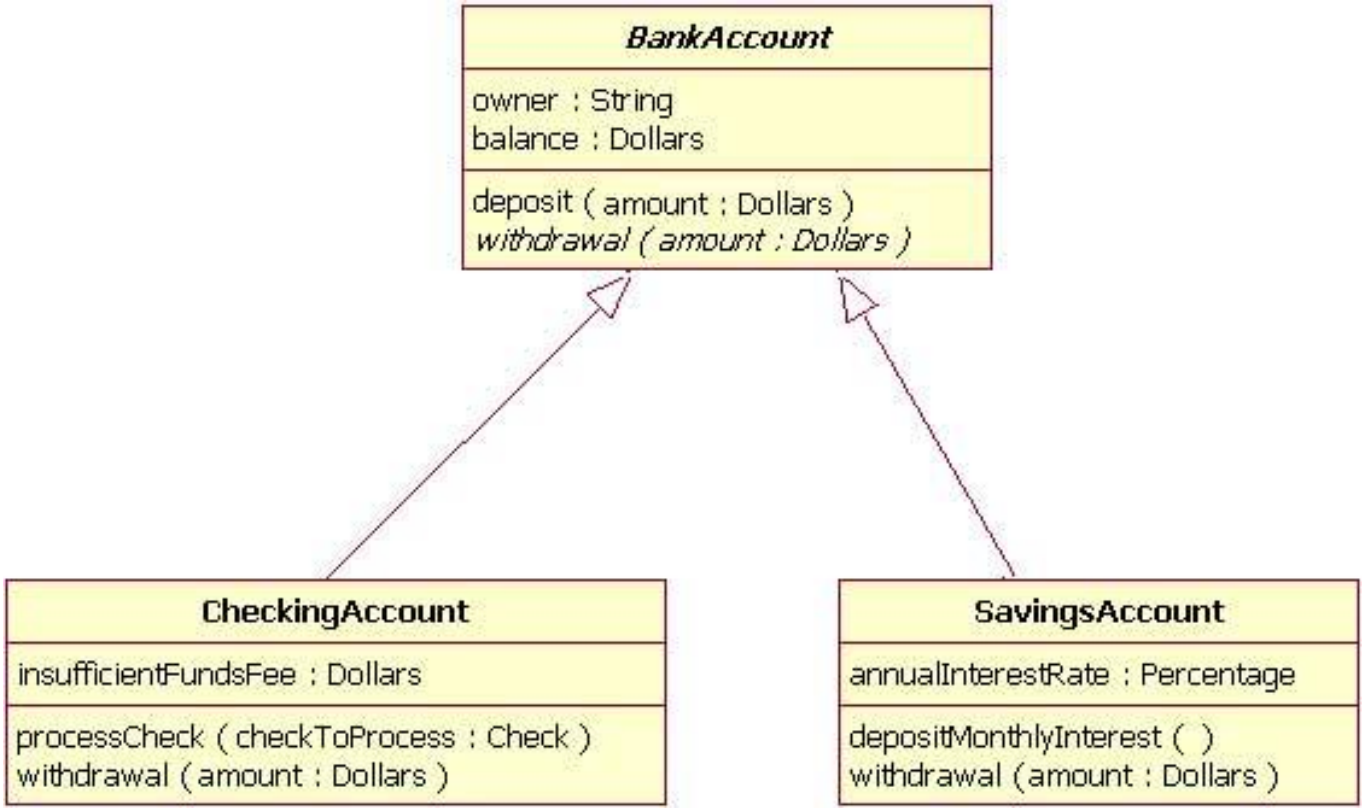
  - High-level design
  - Addresses structure of the sub-systems and communication between the sub-systems
  - What language is to be used? What kind of data storage is present, what recovery systems are in place?
  - Ex: Master-Slave or Client-Server Architectures

Software Design

- Software design is about how the internal components of the system interact with each other.
- Software design is about designing the individual modules / components.
- What are the responsibilities, functions, of module x or class Y?
- What can it do, and what not? What design patterns can be used? UML diagram/flow chart/simple wireframes (for UI) for a specific module/part of the system.

Software Detailed Design

- Detailed Design addresses the detailed working of the system



Summary

- Introduction
- Activities in software process
- Software process model
- Software development
- Software architecture
- Software design
- Software detailed design