Software Development Process

Lecture 1

Introduction

Outline of the Course

- Objective: This course will discuss several **Software Development Processes**
- Outline:
  - Introduction to Software Development Process
  - Software Development Process Models
    - **Sequential Models**: Waterfall Model, V-Model
    - **Iterative and Incremental Models**: Spiral Model, Unified Process
    - **Agile Processes**: Extreme Programming, Scrum, Rapid Application Development (RAD), etc.
    - **Open Source Process**
  - Software Process Improvement

Building a Software

- Think about your favorite software
  - Applications / Utilities
  - Games
  - Mobile apps
  - Operating Systems
  - Web servers
- They are built for some specific purposes, i.e., to solve problems
- Software development is **problem solving**

Software Development

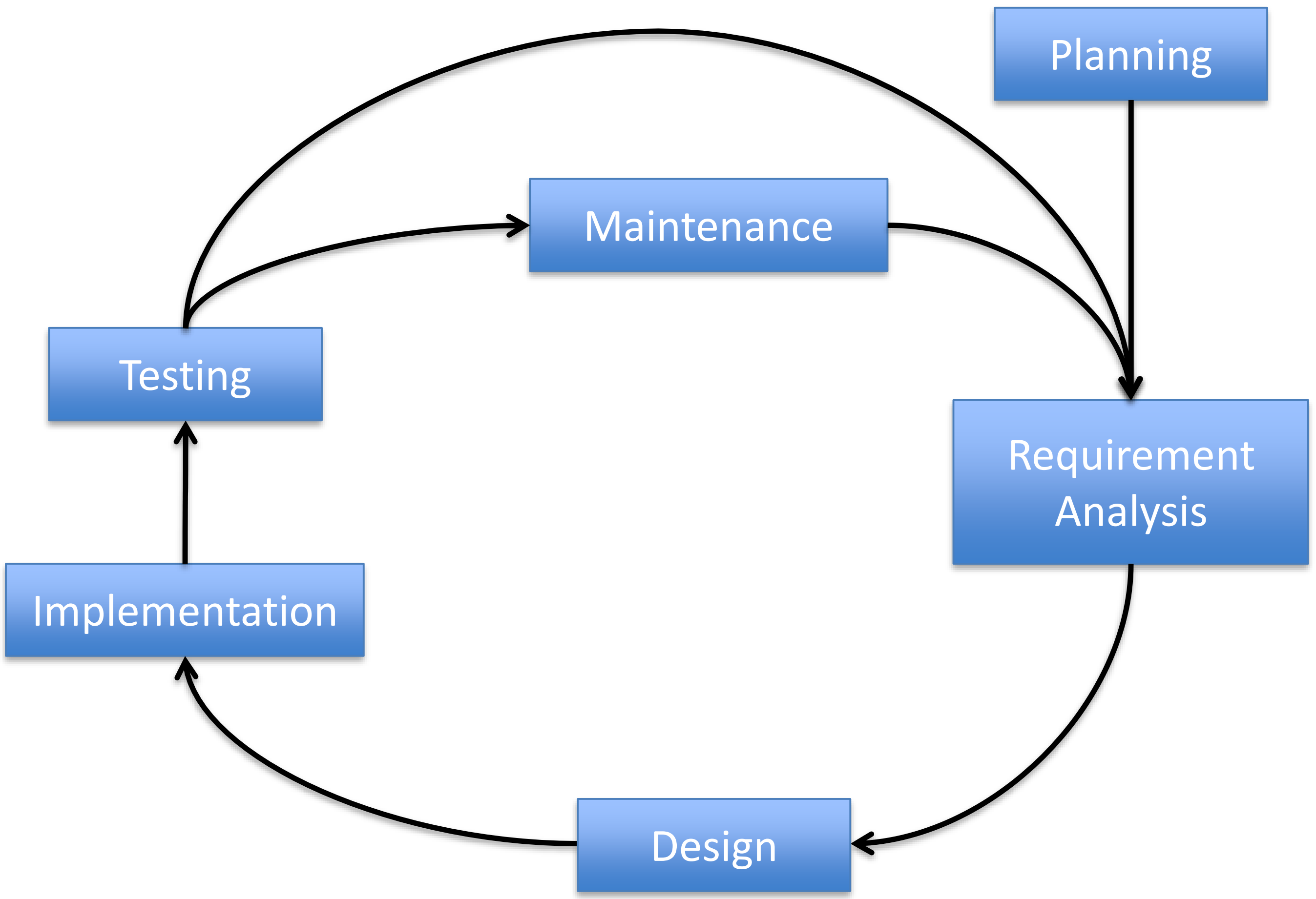
- **It is not just coding!**
- It is problem solving
  - Understanding a problem
  - Proposing a solution and plan
  - Engineering a system based on the proposed solution using a **good** plan
- It is about dealing with complexity
  - Creating abstractions and models
  - Notations for abstractions
- It is knowledge management
  - Elicitation, analysis, design, validation of the system and the solution process
- It is tools making
  - Implement the solution according to the plan
  - Maintain and improve the tools

Software as a Solution

- To solve a "problem" these questions have to be answered:

Requirement Analysis	What is the problem?	Application Domain
System Design	What is the solution?	
Detailed Design	What are the best mechanism to implement the solution?	Solution Domain
Implementation	How is the solution constructed?	
Testing	Is the problem solved?	
	Can the customer use the solution?	
Maintenance	Are enhancements needed?	

Software Development Phases



Software Development Phases (2)

- **Planning**: Define initial idea/concept of the software product and determine rough schedule, resources and costs.
- **Requirement Analysis**: Specify what the application must do; answers "**what?**"
- **Design**: Specify the components (subsystems) and how they fit; answers "**how?**"
- **Implementation**: Write the code
- **Testing**: Execute the application with test data
- **Maintenance**: Repair defects and add capability
- Virtually all software involve these phases
- The question is how to execute those phases effectively

Planning

- **Inception**: Formulate the product idea
  - "What are we going to do?"
  - Very high-level
  - E.g. chatting app, photography
- **Project planning**: After the high-level idea is conceived, a work plan is developed
  - Identify high-level activities, work items, schedule, available resources and cost
  - "What do we have to do and what do we have?"
  - Result: a Software Project Management Plan (SPMP)

Requirement Analysis

- Obtain detailed product information
  - Customer's wants and needs
  - The problems that the software is intended to solve
- Specific product features and functionalities and also performance, reliability and usability are determined
- "**What**" the software is supposed to do
- Result: Software Requirement Specification (SRS) or Requirement Analysis Document (RAD)

System Design

- Determine "**how**" to construct the software
- Categorized into two levels:
  - Architecture design
    - Overall, high-level design
    - How the software are divided into subsystems
    - How the subsystems relate to each other
  - Detailed design
    - How each subsystem works
    - How do they communicate with each other
    - Specific algorithms, data structure, interfaces, etc.
    - User interface and database design
- Result: Software Design Document (SDD)

Implementation

- Coding: Translate the software design to a programming language
  - Subsystem implementation
  - Subsystems integration
- Result: Source code and the object code that is ready to be tested

Testing

- Test the implemented code for correctness
- Testing can be divided into three levels:
  - **Unit test**: Conducted by developers
  - **Integration test**: Subsystems are integrated and tested together to see if they interface properly
  - **System test**: All subsystems are integrated and the entire system is tested to ensure that it meets the user requirements
- System testing typically follows by **beta testing** and **acceptance testing**
- Acceptance testing is conducted by the customer on the **final release** of the software

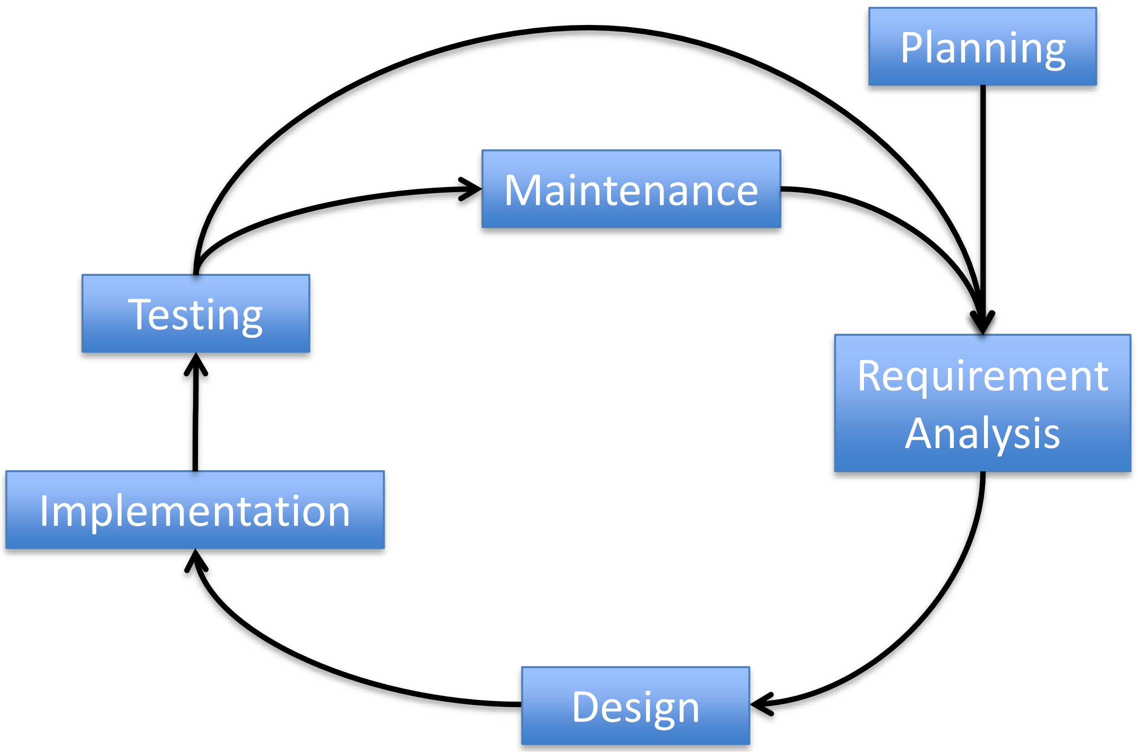
Testing (2)

- Testing is carried out to ensure that the software product does what it is intended to
- The more closely software product meets its requirements and the requirements meet customer needs, the higher the **software quality**
- A deviation from what the software is required to do is called a **defect**

Maintenance

- Maintenance phase takes place after the final release
- Maintenance involves:
  – Repair software defects
  – Additional features and functionalities
  – Improve attributes of the system such as performance or reliability

Typical Software Project Phases



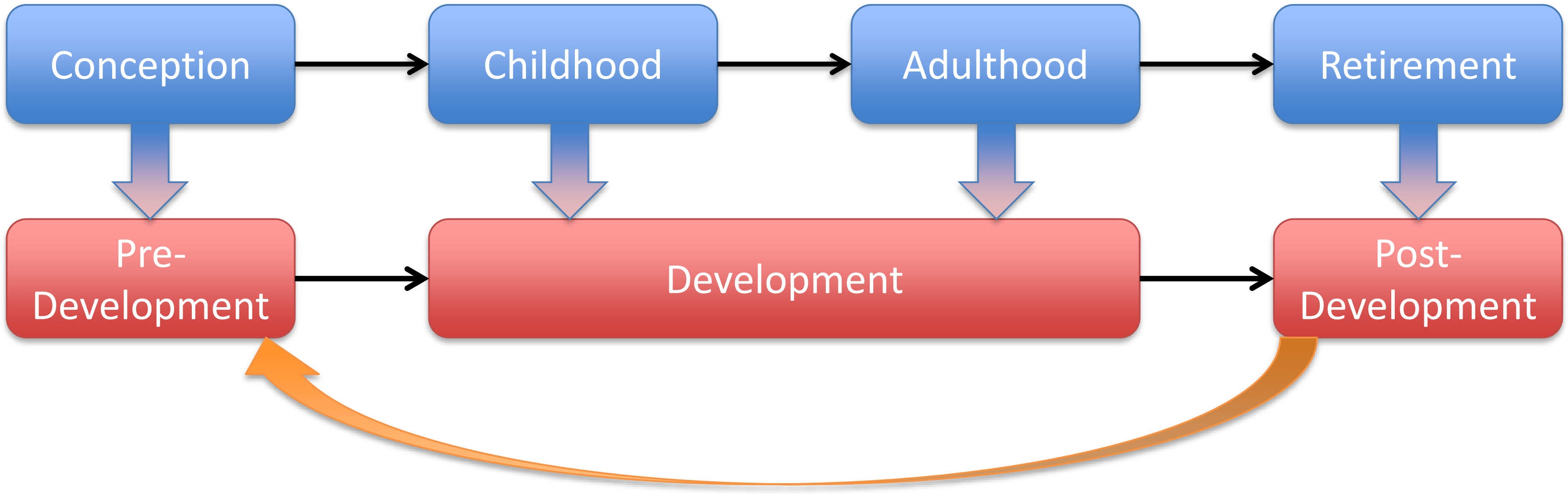
- These are typical software project phases or activities
- Question: How should we schedule these phases?

Software Development Process

- **Software Development Process**
  - Aka Software Process or **Software Development Life Cycle** (**SDLC**)
  - Framework that define the order and frequency of software development phases (activities)
  - Provide guideline for the people who involved in a software project
- **Software Development Process Model**: A specific software process implementation

Software Life Cycle

- The term "lifecycle" is based on the metaphor of the life of a person



Current Software Development Process Models

- Currently, several software development process models are proposed
- Example: Waterfall, spiral, agile process, etc.
- Each of these processes are designed for different purposes and project attributes (e.g., project size, time, domain knowledge)
- There might be no “right” method
- Before we discuss further in software development process, let’s see who/what are involved with a software development

Four P's in Software Development

- **People**: Group of people that are involved with the project; **stakeholders**
- **Product**: The software product and the associated documents
- **Project**: Activities that carried out to produce the product
- **Process**: Framework that the team used to conduct the development activities. That is, the software development process

People

- Those who are involved with and have a stake in the project outcome or **stakeholders**
  - **Business management**: Responsible for the business side of the software company. May not have technical knowledge
  - **Project manager (PM)**: Planning and tracking a project. Keep the project on schedule and within budget. May have only brief technical knowledge
  - **Dev team**: Developing and maintaining a software. Strong technical skill.
  - **Customers**: The one who pays for the project. May not use the software.
  - **End users**: Well, the users.

Product

- Products of software project includes:
  - Source code
  - Object code
  - Project documentation
  - Customer documentation
- Software products are usually called "**artifacts**"

Project and Process

- All software **projects** consist of similar phases or activities to carry out a software
  - Planning, requirement analysis, design, implementation, testing, maintenance
  - The differences between projects are, e.g.:
    - Development paradigm, techniques and tools
- **Process**: Framework that define the order and frequency of software development phases (activities)

Software Development Process Models (1)

- **Sequential Models**
  - Phases are executed in sequential manner
  - Phase 2 can be executed only when Phase 1 is finished
  - Example: Waterfall Model, V-Model

Software Development Process Models (2)

- **Iterative and Incremental Models**
  - Start with small portions of a software project
  - Repeatedly add portions into the projects
  - Example: Spiral Model, Unified Process, Prototyping
- Iterative vs. incremental
  - **Incremental** fundamentally means "**add onto something**"
    - Incremental development helps you improve your process
  - **Iterative** fundamentally means "**re-do**"
    - Iterative development helps you improve your product

Software Development Process Models (3)

- **Agile Processes**:
  - Evolved from iterative and incremental process
  - Intended to speed up software development and respond to ***change***
  - Prefer code, person knowledge and customer involvement over documentation and contracts
- Example: Extreme Programming, Scrum, Rapid Application Development (RAD), etc.

Software Development Process Models (4)

- **Open Source Process**: The process used to conduct open-source software development
- Open source software projects are unlike ordinary (paid, closed source) software projects
  - The developers are attracted to the projects only after early software versions
  - Thus, requirements are rarely gathered
  - They select their own roles (not assigned by others)
  - Some projects may depend or even merged into other projects
  - Users are sometimes the coders
- That is, open source projects are more self-organized compared to well-planned closed source ones

Why Do We Need “Software” Process?

- Unlike other engineering/development problems
  - Buildings and constructions
  - Car making, city planning, gardening, etc.
- ..in Software development, **changes** occur constantly
  - Requirements may be changed or added anytime
- Frequent changes are difficult to manage
  - Planning and cost estimation are difficult
- There could be more than one software system to consider
  - System under development (new versions)
  - Released systems (current version)

Software Development Process Improvement

- Assume that we have a software development process model in-place
- We might have the following questions:
  - Is our model is the most suitable for our development? Is it effective?
  - Do our development teams follow the model correctly?
  - If our software or organization are changed, would our current model still work?
  - Can we improve our process?
- These questions are addressed by **Software Process Improvement** (**SPI**) frameworks

Tentative Schedule

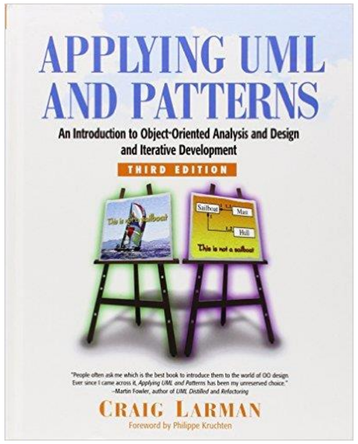
- Week 1: Introduction
- Week 2: Sequential and Iterative and Incremental Models
- Week 4: Unified Process
- Week 5: Introduction to Agile Process
- Week 6: Extreme Programming and Scrum
- Week 7: Test Driven Development
- Week 8: Cost Estimation and Planning Poker
- Week 9: Continuous Integration
- Week 10-15: Advanced topics in Software Development Process

Evaluation

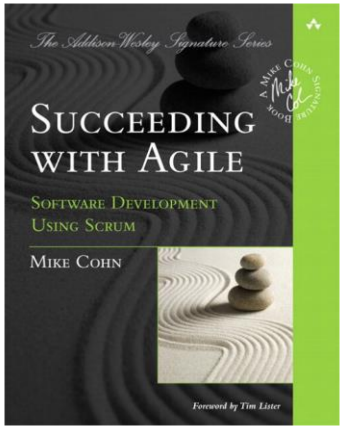
• Midterm exam		30%
• Final exam	50	60%
• Participation	20	10%

• Midterm exam		0%
• Final exam	80	100%
• Participation	20	N/A

Textbooks

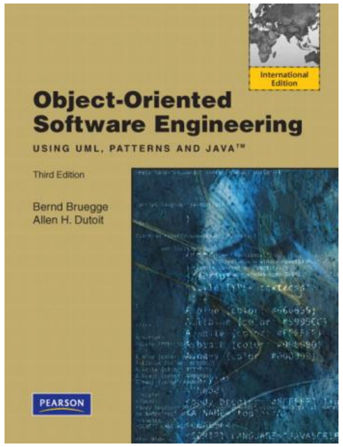

APPLYING UML AND PATTERNS
An Introduction to Object-Oriented Analysis and Design and Iterative Development
THIRD EDITION
CRAIG LARMAN
Foreword by Philippe Kruchten

- Craig Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd* Edition, 2004

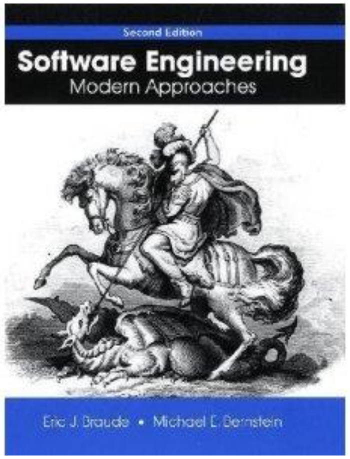

The Addison-Wesley Signature Series
SUCCEEDING WITH AGILE
SOFTWARE DEVELOPMENT USING SCRUM
MIKE COHN
Foreword by Tim Lister

- Mike Cohn, *Succeeding with Agile: Software Development Using Scrum, Addison-Wesley Professional*, 2009

Textbooks



- Bernd Bruegge and Allen H. Dutoit, *Object-oriented Software Engineering: Using UML, Patterns, and Java, 3rd Edition*, Prentice Hall, 2009



- Eric J. Braude and Michael E. Bernstein, *Software Engineering: Modern Approaches, 2nd Edition*, Wiley, 2010