Software Engineering
An Overview

Prof. David Bernstein
James Madison University

Computer Science Department

bernstdh@jmu.edu

What is Software?

- To the Hardware:

  A collection of instructions to be executed by a computer/processor

- To the Developer:

  A collection of human-readable statements (in a language) that can be converted to a collection of instructions to be executed by a computer/processor

Kinds of Software Products

- Bespoke:

  Software products that are developed (usually under contract) for a specific user/customer

- Generic:

  Software products that are developed (usually speculatively) and then sold in a market (either a mass market or a niche market)

Software Engineering

- Defined:

  The application of (scientific) theories, methods and tools to the specification, design, creation, verification/validation, deployment, operation, and maintenance of software products

- Scope:

  From specification to maintenance

  Involves theories/methods from psychology, mathematics/statistics, computer science, and management (of people and resources)

  Consists of science and art

Software Engineering vs. Computer Science/Engineering

- Computer Engineering:

  The application of theories (often from physics) to the creation of computational devices

  Usually thought of as a subset of electrical and electronic engineering

- Computer Science:

  The theories and methods that underly computation and the use of computational devices

Software Processes

- Definition:

  A set of activities/tasks (and corresponding inputs and outputs) that results in the specification, development, validation, and/or evolution of a software product

- Common Activities:

  Project planning, product design, engineering design, implementation, deployment, support/maintenance

- An Observation:

  Many of these activities involve "problem solving" and/or "design"

Generic Problem Solving/Design Activities

- The Parts of the Process:

  Identification of goals, objectives and constraints

  Generation of alternatives

  Evaluation of alternatives

  Selection of an alternative

- Categorizing the Parts of the Process:

  Analysis (to understand the problem)

  Resolution (to solve the problem)

Software Design

- Software Design:

  Software design is the process of specifying the nature and composition of a software system that satisfies client needs and desires, subject to constraints. (Fox, 2006)

- Software Product Design:

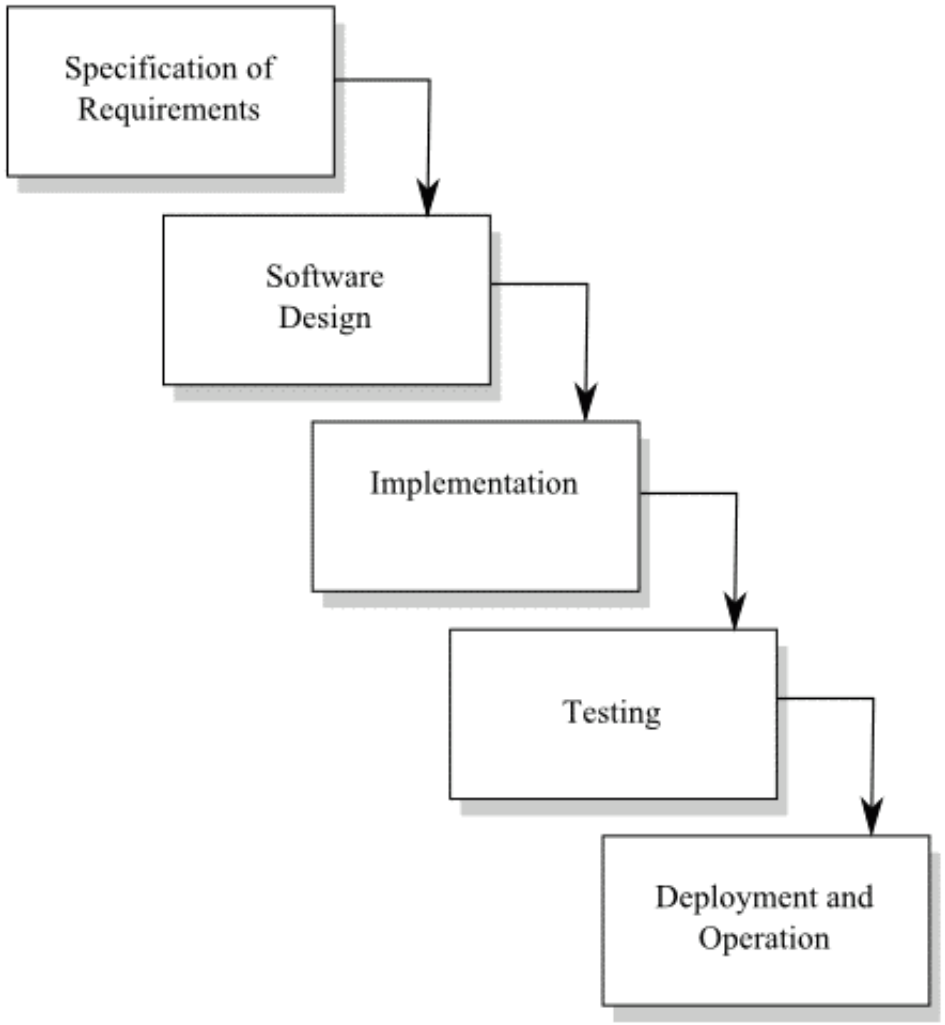
  Software product design is the process of specifying **software product features, capabilities and interfaces** to satisfy client needs and desires. (Fox, 2006)

  The analysis portion involves the creation of a needs list while the resolution portion involves the creation of a requirements specification

- Software Engineering Design:

  Software engineering design is the process of specifying **programs and sub-systems, and their constituent parts and workings**, to meet software product specifications. (Fox, 2006)

The Waterfall Process



The Scrum Process

- Sprints:

  Fixed duration development cycles (usually 1-4 weeks) that end on a specific date (whether the work has been completed or not)

- Scrum Roles:

  Product Owner (responsible for achieving maximum business value)

  Team (multi-functional; 5-10 people)

  ScrumMaster (helps the Team be successful; protects the Team)

- Process:

  Start of the Sprint: There is a Sprint Planning Meeting at which the Team selects items from a prioritized list of requirements/features and commits to completing them

Each Day: Members of the Team report (at a standup meeting of 15 minutes or less) on progress

End of the Sprint: The Team demonstrates what has been built (in a Sprint Review) and gets feedback for the next Sprint (in a Sprint Retrospective)

"Heavyweight" vs. "Agile"/"Lean" Methods

- A Common Categorization:

  Heavyweight: Waterfall, Incremental Delivery, Spiral

  Agile/Lean: RUP, XP, Scrum, FDD

- One View of the Difference:

  The different process involve the same activities but vary dramatically in the cycle times

Becoming a Better Product Designer

- Needs Elicitation Techniques:

  Interviews

  Observations

  Focus Groups

  Document Studies

  Competitive Product Studies

  Prototype Demonstrations (and the resulting feedback)

- Requirements Generation/Documentation Techniques:

  Brainstorming and classification trees

  Atomization, verifiability and uniformity

- User Interface Design Techniques

  Consistency/Stability ("Principle of Least Astonishment")

  Simplicity/Clarity

  Familiarity

  Availability (key features should be readily available)

  Forgiveness/Discoverability

Becoming a Better Engineering Designer

- Architectural Styles:

  A method of characterizing a software product

- Design Patterns:

  A general, reusable solution to a common problem

Becoming a Better Implementer

- Tools:

  Integrated development environments

  Debuggers

  Unit testing frameworks

  System testing frameworks

  Profilers

  Coverage analysis tools

  Static analysis tools

- Idioms:

  A means of expressing a recurring construct in a particular programming language