Lecture 3: Software Engineering

CS190: Software System Design

January 28, 2002
Steven P. Reiss

I. Today's Class

A. Overview of Software Engineering

1. Purpose
2. Models of the process
3. How this relates to CS190

B. Team assignments

1. My assignment based on your handins
2. Initial group meetings
3. Stop me at 10:30

II. Whither Software Engineering

A. Is there a software crisis

B. We need to understand how to build software systems

C. Is software engineering successful

III. Problem Solving

A. Six phases of problem solving

1. Problem formulation
2. Problem analysis -- define the specific problem to solve
3. Search/decision -- finding the right solution
4. Specification -- detailing the solution
5. Implementation
6. Maintenance

B. Does this apply to software?

IV. Software Development

A. Requirements Analysis

1. Define the problem from the user's view

2. **Determine outlines of the best solution**
3. **Determine what is required and what is optional**
4. **Determine limitations on resources**
5. **Determine acceptance criteria**

B. Specifications

1. **Detail the problem -- what will the program do**
   a) From the programmers point of view
2. **Define the inputs and outputs**
3. **Define interfaces to existing systems**
4. **Give a precise statement of what will be done**
5. **How does this meet requirements**
6. **Develop testing and acceptance plan**

C. Design

1. **Design in general**
   a) Develop data structures and algorithms
   b) Problem analysis
   c) Define the solution down to the level where it can be easily implemented
2. **Top-Level Design**
   a) High-level data structures
   b) Classes and their interfaces
3. **Detailed Design**
   a) Details of class implementation
   b) Helper classes, methods, etc.
4. **Prototyping as a design alternative**
   a) Where implications are not well understood
   b) Risk management

D. Coding

1. **Easiest part of the process**
2. **Emphasis on programming style**
3. **Emphasis on defensive programming**

E. Testing

1. Module testing -- small portions (class testing)
2. Integration testing -- putting classes together
3. System testing -- testing the whole thing
4. Acceptance testing -- by the user

F. Operation and Maintenance

1. Most costly and longest (hopefully) phase
2. Effect of changes typically is a cascade
3. Often done by new people

V. Models of Software Development

A. Waterfall model

1. Adding feedback
2. Adding prototyping

B. Spiral model

C. Extreme programming model