CS 320
Fundamentals of Software Engineering

Lecture 2: Software Processes (1)
Software Process

- A structured set of **activities** required to develop a software system

- Many different software processes but all involve:
  - Specification - defining what the system should do
  - Design and implementation - defining the organization of the system and implement the system
  - Validation - checking that it does what the customer wants
  - Evolution - changing the system in response to changing customer needs
Plan-Driven and Agile Processes

* Plan-driven processes are processes where all of the process activities are planned in advance and progress is measured against this plan.

* In agile processes, planning is incremental and it is easier to change the process to reflect changing customer requirements.

* In practice, most practical processes include elements of both plan-driven and agile approaches.

* There are no right or wrong software processes.
Software Process Model

- An abstract representation of a software process
- Presents a description of a process from some particular perspective
- General approach for organizing a project into activities
  - Help the project manager and his/her team to decide:
    - What work should be done
    - In what sequence to perform the work
  - Aids to thinking, **NOT** rigid prescriptions of the way to do things
When there is no process model:

- OK for small, informal projects
- Inappropriate for complex software
- Inappropriate for professional environments where on-time delivery and high quality are expected
  - Requirements and design not worked out before implementation
  - The design of software deteriorates faster if it is not well designed
  - Since there are no plans, there is nothing to aim towards
  - No explicit recognition of the need for systematic testing
  - The above problems make the cost of developing and maintaining software very high

The Opportunistic Approach
Software Process Models

- 3 generic models
  - Waterfall model
  - Incremental development
  - Reuse-oriented software engineering

- In practice, most large systems are developed using a process that incorporates elements from all of these models
The Waterfall Model

Plan-driven

Problem: difficult to accommodate change after the process is underway.
When Should We Use Waterfall Model

- Is appropriate when the requirements are well-understood and changes will be fairly limited during the design process.

- However, few business systems have stable requirements.

- Mostly used for large systems engineering projects where a system is developed at several sites.

- Plan-driven nature of the model helps coordinate the work.
Incremental Development

Outline description → Specification → Development → Validation

Concurrent activities

Initial version
Intermediate versions
Final version
Incremental Development Benefits

• The cost of accommodating changing customer requirements is reduced.
  • The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
• It is easier to get customer feedback on the development work that has been done.
  • Customers can comment on demonstrations of the software and see how much has been implemented.
• More rapid delivery and deployment of useful software to the customer is possible.
  • Customers are able to use and gain value from the software earlier than is possible with a waterfall process.
Incremental Development
Problems

✦ The process is not visible.

✦ Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.

✦ System structure tends to degrade as new increments are added.

✦ Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.
Reuse-oriented Software Engineering

- Requirements specification
- Component analysis
- Requirements modification
- System design with reuse
- Development and integration
- System validation
Reuse-oriented Software Engineering

- **Benefits**
  - Reduce the amount of software to be developed, and the cost
  - Faster delivery of software

- **Problems**
  - Requirement compromises