**Basis for Software Engineering**

- Software Engineering is based on a collection of fundamental principles
- These principles guide the development of all aspects of software development
  - Languages
  - Methods
  - Tools
  - Process
  - Project Management
**Fundamental Principles**

- Rigor and Formality
- Separation of Concerns
- Modularity
- Abstraction
- Anticipation of Change
- Generality
- Incrementality

**How it Really Works**

<table>
<thead>
<tr>
<th>Time</th>
<th>Failure Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ideal Curve</td>
</tr>
<tr>
<td></td>
<td>Actual Curve</td>
</tr>
</tbody>
</table>

Problems added through side-effects

Bug-fix
Part 1, Overview

**Waterfall Model**

- Requirements Definition
- Systems and Software Design
- Implementation and Unit Testing
- Integration and System Testing
- Operation and Maintenance

**Evolutionary Development**

- Concurrent Activities
  - Specification
  - Development
  - Validation
- Outline Description
- Initial Version
- Intermediate Versions
- Final Version

October 02 MSSE, DR, Mats Heimdahl
What We Covered

- Software engineering is concerned with the theories, methods and tools for developing, managing and evolving software products
- Software products consist of programs and documentation
  - Product attributes are maintainability, dependability, efficiency and usability
- Seven fundamental principles of software development
  - Keep these in mind in the MSSE program and at work
More We Have Covered

- The software process consists of those activities involved in software development
- The waterfall model considers each process activity as a discrete phase
- Evolutionary development considers process activities as concurrent
- The spiral process model is risk-driven
- Process visibility involves the creation of deliverables from activities
- Software engineers have ethical, social and professional responsibilities

Requirements Engineering

Establishing what the customer requires from a software system
Key Points

• The structure of the requirements document is of critical importance
  • Use a template to always get it right
    • Developed in house to fit your needs

• Individual requirements contain more information than you may think
  • Use templates to remember it all
  • The requirement must still be well written

Key Points

• Use a standard document structure and forms for the individual requirements
• Use checklists to make sure the individual requirements are “good”
• Use checklists to make sure the document is “good”
• Keep an eye towards the future – things will change
  • What requirements are likely to change
    • Structure the your requirements accordingly
  • Provide rationale and traceability
• Make sure all stakeholders agree on the requirements document
Key Points

- Do yourself and the testing group a favor—
  Develop Test Cases for Each Requirement

- If the requirement cannot be tested, you most likely have a bad requirement
  - Rewrite so it is testable
  - Remove the requirement
  - Point out why this is an untestable requirement

- Your requirements and testing effort will be greatly improved

Key Points

- Deploy use-cases to identify the user functions
  - Ask the customer “What do you want to accomplish?”
- Consider all stakeholders to complete the non-functional requirements
- Always have a heavy customer involvement
- Use checklists with known problem areas to avoid forgetting things
  - Learn from other’s (and your own) mistakes
**Question**

- Is it fair to say that a preliminary user’s manual is a form of prototype? Explain your answer.

- The Preliminary User Manual is a form of paper prototype for the software that is to be built. It provides the reader with a characterization of the software taken from the user’s point of view.

**Question**

- Why is it so important that the requirements document is both clear and readable as well as unambiguous and precise?
  - Customer has to read and understand the document to validate the specifications. Must be clear and readable
  - Developer has to design the system based on requirements. Must be unambiguous and precise
  - Requirements may be a basis for a contract – Must be unambiguous and precise to avoid problems
  - Quality of product is directly dependent on requirements – errors in requirements are difficult and costly to correct at later stages and increases defect rate in the end product
**Question**

- Explain why it is very difficult to produce a complete and consistent set of requirements. Mention and explain at least 5 reasons.
  - Difficult to identify all stakeholders or end-users
  - Difficult for customers to articulate their requirements
  - Different people express requirements in different ways using different notations
  - Assumptions about system environment are valid only at the time when they are made
  - The sheer size of the requirement document may make it difficult to produce complete and consistent requirements
  - Often requirements are not properly validated due to time and money constraints

**Question**

- The following statements (that may appear in a requirements specification) are unclear and ambiguous. Please rewrite the statements so that they can be objectively evaluated (that is, we can test an implementation and determine if the statements have been met). You may make any reasonable assumptions about the requirements.

  - After a high temperature is detected and alarm must be raised quickly.

  - Novice users should be able to learn the interface quickly.
**Key Points**

- A prototype can be used to give end-users a concrete impression of the system’s capabilities
- Two types of prototyping
  - Evolutionary prototyping
  - Throw-away prototyping
- Rapid development is essential for prototype systems
- Prototype structures become corrupted by constant change
  - Long-term evolution is difficult

**Key Points**

- In a throw-away prototype start with the least well-understood parts
- In an evolutionary prototype, start with the best understood parts
- Prototyping is essential for parts of the system such as the user interface which cannot be effectively pre-specified
Object Modeling Summary

- Classes
  - Name
  - Attributes
  - Operations
- Associations
  - Roles
  - Link attributes
- Aggregation
- Inheritance

In Class Assignment

- Develop a first draft class diagram of the FLMS
- Pens and transparencies are up at the podium
- Meet back here at: