CS 3141:
Team Software Project –
Introduction

Ali Ebnenasir
Department of Computer Science
Michigan Technological University

Acknowledgement

• Betty H.C. Cheng

Software Engineering

• Systematic approach for developing software
• Methods and techniques to develop and maintain quality software to solve problems.
  (Software Engineering: Methods and Management, Pfleeger, 1990)
• Study of the principles and methodologies for developing and maintaining software systems.
  ("Perspectives on Software Engineering," Zelkowitz, 1978)
Software Engineering

- Practical application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate, and maintain them. ("Software Engineering," Boehm, 1976)
- Deals with establishment of sound engineering principles and methods in order to economically obtain software that is reliable and works on real machines. ("Software Engineering," Bauer, 1972)

Why Engineer Software?

- Software plays a role in almost every aspect of our lives
- Software failure could lead to loss of life, property, and could damage critical infrastructures
  - August 2003 black out
  - Automatic Luggage system – Denver Airport
  - Medical devices failures – Therac 25
- Need systematic (and possibly automatic) approaches for software development

Why is it difficult?

- Developers lack domain knowledge (business logic)
- Communication difficulties:
  - Amongst developers
  - Between developers and users
- Incremental nature of user requirements
- Crosscutting nature of non-functional concerns
  - Security
  - Reliability
What is important?

• Problem abstraction:
  – Figure out what the problem is in the application domain
  – Inputs, outputs and a set of constraints that must be met

• Modeling:
  – Model different entities involved in the problem
  – Balance the level of abstraction with reality
  – Use a modeling language to specify the problems

• Problem solving:
  – Focus on solving the problem in the context of your modeling language

• Conceptual model:
  – A set of artifacts representing the problem along with its solution

Example: Functional Decomposition

• Decompose the problem into a set of independent sub-problems
  – Each sub-problem may represent one functionality of the system

• Solve the sub-problems using necessary abstractions

Analysis Views

• Functional
  – Top-down hierarchy of functions

• Control flow
  – Control flow graphs (CFGs)

• Data flow
  – Data flow diagrams (DFDs)
  – Entity relationship diagrams (ERDs)

• Object-Oriented
  – Data and control encapsulated as an object
Data Flow Diagram

Check for overdue books → Patron ID

Patron ID

Check-out books

Read Patron record → Patron record

Patron ID

Change book status → Book ID

Read book record → Book record

Write book record

Legend:

Process

Data Flow

Entity Relationship Diagram

Library Patron

Has-a

Address

Requests

Checks out

1:M

Cardinality

Book

Has-a

Publisher

Software Engineering Phases and Process Models
Software Engineering Phases

- Definition: What?
- Development: How?
- Maintenance: Managing change
- Umbrella Activities: Throughout lifecycle

Definition

- Requirements definition and analysis
  - Developer must understand
    - Application domain
    - Required functionality
    - Required performance
    - User interface

Definition (cont.)

- Project planning
  - Allocate resources
  - Estimate costs
  - Define work tasks
  - Define schedule

- System analysis
  - Allocate system resources to
    - Hardware
    - Software
    - Users
Development

- Software design
  - User interface design
  - High-level design
    - Define modular components
    - Define major data structures
  - Detailed design
    - Define algorithms and procedural detail

Development (cont.)

- Coding
  - Develop code for each module
  - Unit testing
- Integration
  - Combine modules
  - System testing

Maintenance

- Correction - Fix software defects
- Adaptation - Accommodate changes
  - New hardware
  - New company policies
- Enhancement - Add functionality
- Prevention - make more maintainable
### Umbrella Activities

- Reviews - assure quality
- Documentation - improve maintainability
- Version control - track changes
- Configuration management - integrity of collection of components

### Development Process

- Step-by-step procedure to develop software
- Typically involves the major phases:
  - analysis
  - design
  - coding
  - testing

### Process Model / Methodology

- Set of activities, notations, tools, in defined sequence
- Goal: Order, predictability, quality, cost control
- Follows requirements=>design=>coding, etc. sequence (usually)
- Usually defines phases or steps
- Often has notations
- Sometimes has tools
When to use prototyping?

- Help the customer pin down the requirements
  - Concrete model to “test out”
  - Often done via the user interface
- Explore alternative solutions to a troublesome component
  - e.g., determine if an approach gives acceptable performance
- Improve morale
  - Partially running system provides visibility into a project

*Never press down a prototype to final product!*
Spiral Process Model

Planning → Risk Analysis
Customer Evaluation → Engineering

The V-Model

- Requirement Analysis
- System Design
- Architecture Design
- Module Design
- Coding
- Unit Test Design
- Integration Test Design
- System Test Design
- Acceptance Test Design
- Acceptance Testing
- System Testing
- Integration Testing
- Acceptance Test

*An extension of the waterfall model
*Focus on testing

Process Models

- Idealized view of the development process
- Different models are often used for different subprocesses
  - may use spiral model for overall development
    - prototyping for a particularly complex component
    - waterfall model for other components
CMM: Capability Maturity Model

A process meta-model that defines a set of capabilities that should be achieved by software development organizations/companies.

- **Level 1: Initial**
  - ad hoc
  - success depends on people

- **Level 2: Repeatable**
  - track cost, schedule, functionality

- **Level 3: Defined**
  - use standardized processes

- **Level 4: Managed**
  - collect detailed metrics

- **Level 5: Optimizing**
  - continuous process improvement
  - "built-in" process improvement

Software Engineering Institute: http://www.sei.cmu.edu/cmm/

Why is software development so difficult?

- **Communication**
  - Between customer and developer
  - Poor problem definition is largest cause of failed software projects
  - Within development team
  - More people = more communication
  - New programmers need training

- **Project characteristics**
  - Novelty
  - Changing requirements
  - 5 x cost during development
  - up to 100 x cost during maintenance
  - Hardware/software configuration
  - Real time requirements

Why is software development difficult? (cont.)

- **Personnel characteristics**
  - Ability
  - Prior experience
  - Communication skills
  - Team cooperation
  - Training

- **Facilities and resources**
  - Identification
  - Acquisition

- **Management issues**
  - Realistic goals
  - Cost estimation
  - Scheduling
  - Resource allocation
  - Quality assurance
  - Version control
  - Contracts

- **Crosscutting concerns**
  - Availability requirements
  - Security requirements
  - Reliability requirements
  - Correctness/High assurance
Summary

- Software lifecycle consists of
  - Definition (what)
  - Development (how)
  - Maintenance (change)
- Different process models concentrate on different aspects
  - Waterfall model: maintainability
  - Prototype model: clarifying requirements
  - Spiral model: identifying risk
  - V model: extensive testing/verification
- Maintenance costs much more than development

Bottom Line

- Software is a major part of our societal infrastructure
  - Costs upwards of $200 billion/year
- Need to
  - Improve software quality
  - Reduce software costs/risks

Impact of SE Research

- Impact of research is measured over time and utility
- Who do you think has had the most impact on the field of software engineering?
- Why?
People you should know…

- **David Parnas**: modularity, reuse, abstraction
- **Edsger Dijkstra**: structured programming
- **Harlan Mills**: clean room programming
- **Mark Weiser**: ubiquitous computing

Software vs. Hardware Engineering

Hardware and Hardware Engineering

- Characteristics:
  - Components are packaged as individual building blocks
  - Standardized interfaces among components
  - Large number of off-the-shelf components
  - Performance, cost, and availability easily determined/measured

- Hardware configuration built from a hierarchy of "building blocks"
### Hardware Engineering

- **Phases to system engineering of hardware:**
  - Development Planning and requirements analysis:
    - best classes of hardware for problem
    - availability of hardware
    - type of interface required
    - identification of what needs to be designed and built
  - Establish a Plan or "road map" for design implementation
    - May involve a hardware specification
      - Use Hardware Description Languages (HDL)
      - Use CAE/CAD to develop a prototype (breadboard)
      - Develop printed circuit (PC) boards
      - Manufacturing of boards

### Software and Software Engineering

- Function may be the implementation of a sequential procedure for data manipulation
- Performance may not be explicitly defined (exception in real-time systems)
- Software element of computer-based system consists of two classes of programs, data, and documentation
  - **Application Software:**
    - implements the procedure that is required to accommodate information processing functions
  - **System Software:**
    - implements control functions that enable application software to interface with other system elements

### Three high-level phases of Software Engineering

- **Definition phase:**
  - Software planning step → **Software Project Plan**
    - scope of project
    - risk
    - resource identification
    - cost and schedule estimates
  - **Software Requirements Analysis → Requirements Specification**
    - System element allocated to software is defined in detail
    - Formal information domain analysis to establish models of information flow and structure (expand to produce specification)
    - Prototype of software is built and evaluated by customer
    - Performance requirements or resource limits defined in terms of software characteristics
  - Definition and Requirements must be performed in cooperation
Development Phase

- Development Phase:
  - Translate set of requirements into an operational system element
    - Design → Design Specification
    - Coding (appropriate programming language or CASE tool)
  - Should be able to directly trace detail design descriptions from code: i.e., traceability

- Verification, release, and maintenance phase:
  - Testing software → Testing Plan
    - to find maximum number of errors before shipping
    - Testing shows software has errors, it does not show the lack of errors!
  - Prepare software for release → Quality Assurance
  - Maintain software throughout its lifetime