EECS 22L: Software Engineering Project in C Language

Lecture 2

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 2: Overview

- Software Development Process
  - Application specification
  - Software architecture design and specification
    - Software layers and modules
  - Implementation, testing, and debugging
  - Software release
- Source Code Management
  - Collaborative software development
  - Version trees
  - Example: Concurrent Versions System (CVS)

Software Development Process

- EECS 22L Software Development Process
  1. Application specification
     - User's perspective (aka. client or customer)
     - Documentation
  2. Software architecture design and specification
     - Developer's perspective
     - Documentation
  3. Implementation, testing, and debugging
     - Unit testing
     - System testing
  4. Software release
     - Binary program and documentation
     - Source code and documentation

Software Development Process

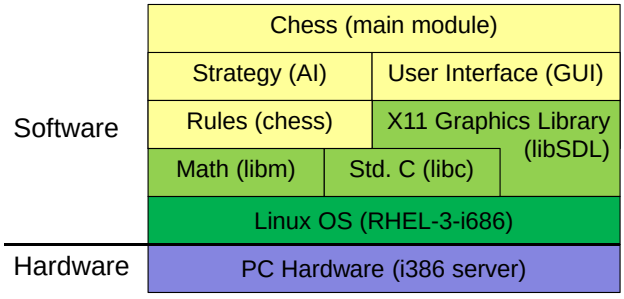
1. Application Specification
   - Goal: Specify the user experience!
     - What does the user (aka. customer, client) want?
     - What does he need to provide? What does he get?
     - What does the software do? What features does it have?
   - Deliverable: Application Specification Document
     - Input data including options and parameters
       - What? In which format? In which order? From which device? …
     - Processing
       - What? (*not* how!) What happens? What is presented?
     - Output
       - What? In which format? In which order? To which device? …
   - ➢ Some features may be intentionally left "unspecified"
   - ➢ Specification can be the starting point of the final documentation!

Software Development Process

2. Software Architecture Design and Specification
   - Goal: Specify the developer's perspective!
     - What modules is the program composed of? Dependencies?
     - How do the modules interact? What functions and parameters?
     - What data structures are used? What algorithms?
   - Deliverable: Software Specification Document
     - ➢ *Detailed plan to develop and implement the software!*
     - Software layers and modules
       - Software architecture with layers of modules and libraries
       - Application Procedural Interface (API) of modules (tentative headers!)
     - Data structures and algorithms
       - How is data organized?
       - How is data processed?
     - Implementation plan
       - Project timeline
       - Tasks and team member assignments

Software Development Process

- Example: Software Layers and Modules
  - Stack of all components in the software architecture
    - Hardware infrastructure
    - Operating system (OS) infrastructure
    - OS and third-party libraries
    - Application modules



Software Development Process

3. Implementation, Testing, and Debugging (part 1)
   - Goal: Build the software!
     - Implement the modules and integrate them
     - Unit testing
     - System testing
   - Deliverable 1: Alpha Version of Software Package
     - ➢ *Demonstrate feasibility to the user!*
     - Software program
       - Binary executable program
       - Debugging support enabled (console output, flow and limitations)
     - Initial data set (if applicable)
       - Initial set of input data demonstrating alpha status
     - Documentation
       - Prerelease of user manual
       - Release notes with known limitations

Software Development Process

3. Implementation, Testing, and Debugging (part 2)
   - Goal: Build the software!
     - Implement the modules and integrate them
     - Unit testing
     - System testing
   - Deliverable 2: Beta Version of Software Package
     - ➢ *Preview software to the user!*
     - Software program
       - Binary executable program
       - Assertions enabled (locate crashes!)
     - Testing data set (!)
       - Input data and automated test scripts
     - Documentation
       - User manual
       - Release notes with known limitations

Software Development Process

4. Software Release (part 1)
   - Goal: Release the software to the user!
     - Complete implementation
     - Complete documentation
     - Complete testing
   - Deliverable 1: Binary program and documentation
     - ➢ *Everything needed for a user (client, customer) to install, learn, and use the software!*
     - Software program
       - Binary executable program (no debugging support, optimized)
     - Tutorial data set (if applicable)
       - Initial set of input data demonstrating tool usage to the user
     - Documentation
       - User manual
       - Release and installation notes

Software Development Process

4. Software Release (part 2)
   - Goal: Release the software to the user!
     - Complete implementation
     - Complete documentation
     - Complete testing
   - Deliverable 2: Source code and documentation
     - ➢ *Everything needed for a developer to install, maintain, and upgrade the software!*
     - Software source code
       - Header and module files
       - Development setup (Makefile, scripts, etc.)
     - Testing data set
       - Input data and automated scripts for regression tests
     - Documentation
       - Software architecture manual
       - Release and installation notes

Source Code Management

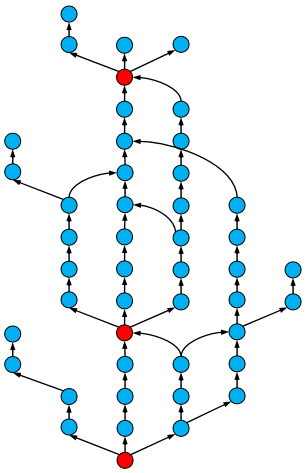
- Source Code Management
  - Also known as *Version Control*
  - or *Configuration Management*
- Purpose and Goals
  - Team-based, *concurrent* software development
  - Access control
  - Archive for software development and versions
  - Common data base with records of
    - Source code, documentation, and other build files
    - Versions and revisions
    - Branches and merges
    - History and log information
  - Efficient storage space usage with remote access

Source Code Management

- Collaborative Software Development
  - ➢ *Shared but dependent source code files!*
  - Two options:
    - Single modifications with file *locking*
      - Ensures that no two developers modify the same file
      - But has drawbacks:
        - » Locking may be forgotten
        - » Locking may lead to deadlocks (!)
        - » Locally modified files may lead to mismatches with locked ones…
    - Multiple modifications with *merging*
      - Multiple developers work on the same files at the same time
        - » Multiple modifications are allowed, different versions exist!
      - Files are *merged* when inserted into the common code base (“merge and commit to the repository”)
      - ➢ Merging can often be performed automatically!

Source Code Management

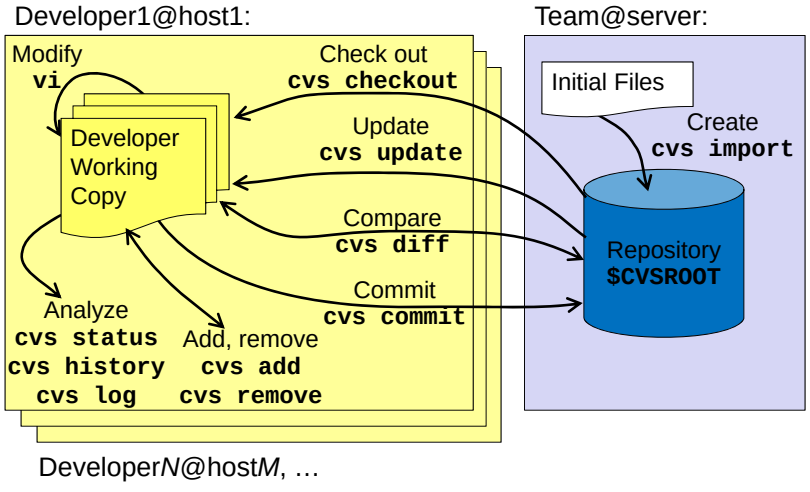
- Version Trees
  - Software products consist of versions
    - Release *versions*
    - Development *revisions* (internal)
  - Concurrent feature development requires multiple parallel branches
    - Separate common and feature files
      - Only a few of the files actually differ
  - Version trees consist of
    - *Root* (e.g. revision 1.0) and main *trunk*
    - *Branches* for features (1.0.1, 1.0.2, …)
      - May be active or dead
      - May be merged into other branches
    - Minor development revisions (e.g. 1.1, 1.2, …)
    - Major / release versions (e.g. 2.0, 3.0)



EECS22L: Software Engineering Project in C, Lecture 2 (c) 2013 R. Doemer 13

Source Code Management

- Example: Concurrent Versions System (CVS)



EECS22L: Software Engineering Project in C, Lecture 2 (c) 2013 R. Doemer 14