EECS 22L: Software Engineering Project in C Language

Lecture 1

Rainer Dömer

doemer@uci.edu

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

Lecture 1: Overview

- Introduction
  - Programming Courses in EECS
  - EECS 22L course outline and overview
- Course Administration
  - Projects and deliverables
  - Grading policy and exams
  - Team work!
  - Web page and programming setup
- Introduction to Software Engineering
  - General software engineering
  - Software development process in EECS 22L

Programming Courses in EECS

- Introductory Programming
  - EECS 10: uses C programming language (for EE)
  - EECS 12: uses Python programming language (for CpE)
- Programming from the Ground Up
  - EECS 20: starts with Assembly language (on bare CPU), then introduces C programming language
- Advanced Programming Courses
  - EECS 22: *"Advanced C Programming"* (in ANSI C)
  - ➢ EECS 22L: *"Software Engineering Project in C"* (ANSI C/C++)
- Object-Oriented Programming
  - EECS 40: introduces objects and classes, hierarchy, and higher object-oriented programming concepts using Java

EECS 22L: Software Eng. Project in C

- *"Developing real C Programs in a Team"*
  - Hands-on experience with larger software projects
  - Introduction to software engineering
    - Specification, documentation, implementation, testing
  - Team work
- Features
  - Design efficient data structures, APIs
  - Utilize programming modules, build libraries, GUIs
  - Develop and optimize contemporary software applications
- Tools
  - Software development, version control: **`ssh`**, **`gcc`**, **`cvs`**, **`chmod`**
  - Compilation, scripting, packaging: **`make`**, **`bash`**, **`groff`**, **`gtar`**
  - Testing and debugging with **`gdb`**, **`ddd`**, **`gprof`**, **`gcov`**, …

EECS 22L: Software Eng. Project in C

- Catalogue Data
  - **EECS 22L Software Engineering Project in C Language (Credit Units: 3) W.**
  - Hands-on experience with the ANSI-C programming language.
  - Medium-sized programming projects, team work.
  - Software specification, documentation, implementation, testing.
  - Definition of data structures and application programming interface.
  - Creation of program modules, linking with external libraries.
  - Rule-based compilation, version control.
  - Prerequisites: EECS 22
  - (Design Units: 3)

EECS 22L: Software Eng. Project in C

- Course Outline
  - Software engineering topics, including specification, documentation, implementation, testing, debugging, project planning, organization, maintenance, version control, organization of source files, header files, modules
  - Compilation flow, Makefile, shell scripting
  - Definition of data structures and application programming interface
  - External libraries, system programming, POSIX API, interrupts
  - Introduction to C++ language, syntax and semantics, references, inline functions, default arguments, classes, members, and methods, object creation and deletion (constructors, destructors)

Course Overview

Week	Lecture topics	Project tasks	
1	Introduction to software engineering	Project 1	Application specification
2	Software architecture, design flow, documentation		Software architecture specification
3	Introduction to version control, GUI programming		Documentation, implementation
4	Software development, testing, documentation		Implementation, testing, debugging
5	Software packaging, installation, deployment		Delivery, installation, deployment
6	Project planning, organization, maintenance	Project 2	Application specification
7	Data structure and API design		Software architecture specification
8	System programming, shell scripting, Linux tools		Documentation, implementation
9	Intro to object-oriented programming in C++		Implementation, testing, debugging
10	Course wrap up		Delivery, installation, deployment

Course Overview

- Class Schedule
  - Quote from EECS 22L course outline:
    EECS 22L *"Meets for 1 hour of lecture, 1 hour of discussion and 3 hours of laboratory each week for 10 weeks"*
  - However, current schedule of classes lists 3 hours of lecture, 1 hour of discussion and 3 hours of laboratory
  - ➢ Use lecture slots for actual lectures, as needed
  - ➢ Use remaining lecture slots for team meetings and team presentations
- Detailed Class Schedule
  - Online at course web site:
    `https://eee.uci.edu/14w/18020/schedule.html`

Course Administration

- Projects and Deliverables

Project	Task	Points	Deliverable	Due
Project 1: Chess Game	Application specification	100	Chess_UserManual.pdf	Jan 13, 12pm (noon)
	Software specification	100	Chess_SoftwareSpec.pdf	Jan 20, 12pm (noon)
	Software alpha version	100	Chess_Alpha.tar.gz Chess_Alpha_src.tar.gz	Jan 27, 12pm (noon)
	Software release	100 (+X)	Chess_V1.0.tar.gz Chess_V1.0_src.tar.gz	Feb 3, 12pm (noon)
Project 2: TBD	Application specification	100	TBD	Feb 17, 12pm (noon)
	…	…	…	…
	Software release	100 (+X)		Mar 17, 12pm (noon)

Course Administration

- Effort Assessment
  - Team: Project deliverables
  - Individual student: Exams, plus feedback from peers, TAs
- Grading Policy
  - Programming projects 50% (team effort)
  - Participation 5% (individual effort)
  - Midterm examination 15% (individual effort)
  - Final examination 30% (individual effort)
- Exams
  - Midterm exam Project 1 contribution (week 5)
  - Final exam Project 2 contribution (final week)
    - Short oral exams by individual students at the computer
    - Explain original contribution to the team, and answer few ad-hoc questions

Course Administration

- Team Work
  - Projects will be performed by student teams
    - Project 1: 10 teams of 7-8 students
    - Project 2: TBD
    - ➢ EEE Survey on team preferences open until 2pm today!
  - *Team work* is an essential aspect of this class!
    - ➢ Every student needs to contribute to the team effort!
    - ➢ Tasks may be assigned to individual team members, but all members share the responsibility for deliverables
  - Collaboration
    - Team meeting at least once a week
    - Dedicated team account on the server
    - Share code, data, and documents (within your team only!)
  - Competition
    - Teams compete for extra credit!

Course Administration

- Course web pages online at **`http://eee.uci.edu/14w/18020/`**
  - Instructor information
  - Course description and contents
  - Course policies and resources
  - Course and project schedule
  - Course communication
    - Message board (announcements, class discussion)
    - Email (administrative issues)
    - Office hours (instructor and TAs)
- Linux system environment
  - Same as for EECS 22
  - EECS Linux servers **`crystalcove`** and **`zuma`**
  - Additionally shared team accounts: **`team1`**, **`team2`**, …

Introduction to Software Engineering

- What is Software Engineering?
  - Software engineering is the application of *engineering* to *software*
  - Software engineering can be defined as:
    - *The application of, or*
    - *the study of*
      a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software.
- EECS 22L …
  - … is *not* a complete course on software engineering!
  - … consists of projects that demonstrate the essential tasks and tools of software development in ANSI C

Introduction to Software Engineering

- General Software Engineering Process
  - Project feasibility and planning
  - Requirements analysis, definition, and specification
  - Design and documentation of the system and software
    - E.g. using UML (Unified Modeling Language)
  - Implementation
    - Programming (modules, system)
    - Testing against the specification (units, system)
  - Delivery, operation, maintenance
- EECS 22L Software Development Process
  1. Application specification and documentation
  2. Software architecture design and specification
  3. Implementation, testing, and debugging
  4. Software release

Introduction to Software Engineering

- EECS 22L *does not* cover
  General Software Engineering Topics
  - General processes of software engineering
  - General feasibility study and requirements engineering
  - General design strategy and documention
    - E.g. UML
  - Usability and reliability studies
  - Legacy systems and evolution of software
  - General project or personnel management
  - Consideration of economic, legal, social and other factors
  - Verification of software
  - …

Software Development Process

- EECS 22L *does* cover
  the essential tasks and tools of software development
  - Using ANSI-C programming language
    - With an outlook into object-oriented design, i.e. C++
  - In Linux environment
    - With typical Linux tools chain,
      e.g. **`gcc`**, **`make`**, **`gdb`**, **`ssh`**, **`cvs`**, **`gtar`**, **`bash`**, **`gprof`**, …
  - With focus on practical aspects
    - Medium-size projects
    - Programming practice
    - Communication
    - Team work!

Software Development Process

- EECS 22L Software Development Process
  1. Application specification
     - User's perspective (aka. client, customer, consumer)
     - Documentation
  2. Software architecture design and specification
     - Developer's perspective (aka. producer)
     - Software layers and modules
     - 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 (customer, client, consumer) want?
     - What does he need to provide? What does he get?
     - What does the software do? What features does it have?
   - Deliverable: Software User Manual (as anticipated)
     - 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 "undefined"
   - ➢ Specification document is typically an early version of the final documentation: *User Manual*!

Software User Manual

- Contents of a User Manual for a Software Product (1/2)
  - Title page
    - Software title, version
    - Author/producer, affiliation
  - Front matter
    - Table of contents
    - Glossary
  - Overview (or Tutorial)
    - Introduction, goals, usage scenario
    - Typical screenshot
    - Main features
  - Installation
    - System requirements
    - Setup and configuration
    - Uninstalling

Software User Manual

- Contents of a User Manual for a Software Product (2/2)
  - Documentation of functionality
    - Detailed description of functions, menu options
    - User input, program output
    - Screen shots
  - Back matter
    - Trouble shooting, error messages
    - Copyright, contact information
    - Legal, license, disclaimer of warranty
    - Index
    - References
    - Appendix

Software Development Process

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

Software Architecture Document

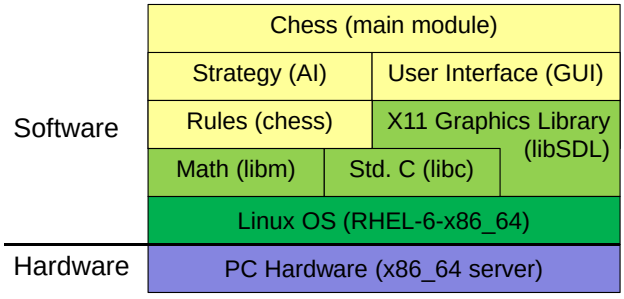
- Contents of a Software Architecture Document (1/2)
  - Title page
    - Software title, version
    - Author/producer, affiliation
  - Front matter
    - Table of contents
    - Glossary
  - Software Architecture Overview
    - Introduction, goals, features
    - Major software components (e.g. module hierarchy), diagrams
    - Major interfaces (e.g. application procedural interfaces), diagrams
  - Installation
    - System requirements, compatibility
    - Setup and configuration
    - Building, compilation, and installation

Software Architecture Document

- Contents of a Software Architecture Document (2/2)
  - Various views on the software architecture
    - Use-case view, logical view, process view, deployment view (typically described in Unified Modeling Language, UML)
  - Documentation of packages, modules, interfaces
    - Detailed description of data structures
    - Detailed description of functions and parameters
    - Detailed description of data input and output (incl. format)
  - Development plan and timeline
    - Partitioning of tasks
    - Timeline of development, testing, releases
  - Back matter
    - Copyright, contact information
    - Legal, license, disclaimer of warranty
    - Index, References, Appendix

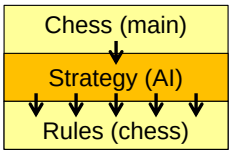
Software Architecture Document

- Example: Diagram of Software Layers and Modules
  - Stack of major components in the HW/SW architecture
    - Application modules
    - OS and third-party libraries
    - Operating system (OS) infrastructure
    - Hardware platform



Software Architecture Document

- Example: Documentation of Chess Strategy Module
  - Module dependencies
    - Provides: Evaluation of potential moves
    - Requires: **`libChessRules.a`**, **`libc.a`**



  - Exported functions

```
t_Move *SelectBestMove(
                t_MoveList *LegalMoves,
                t_Board    *Board,
                t_Player   Color)
```

    - Arguments:
      - **`LegalMoves`** list of potential moves (which must be legal)
      - **`Board`** current board position
      - **`Color`** player to make the next move
    - Result:
      - **`BestMove`** pointer to the "smartest" move in the **`LegalMoves`** list
    - Notes:
      - Returns **`NULL`** if list of **`LegalMoves`** is empty

Software Development Process

3. Implementation, Testing, and Debugging
   - Goal: Develop and build the software!
     - Implement the modules and integrate them
     - Perform unit testing
     - Perform system testing
   - Deliverables: Early version of the software packages
     - ➢ *Alpha version: Demonstrate feasibility to the user*
     - ➢ *Beta version: Preview software to the user*

   1. Software program package (for users)
      - Executable program
      - User manual (with known limitations)
   2. Source code package (for developers)
      - Source code file hierarchy
      - Software architecture document

Software Development Process

4. Software Release
   - Goal: Release, install, operate and maintain the software!
     - Complete the implementation and testing
     - Complete the documentation
   - Deliverables: Final version of the software packages
     - ➢ *Everything needed for users (client, customer, consumer) to install, learn and operate the software!*
     - ➢ *Everything needed for developers to install, maintain and upgrade the software!*

     1. Software program package (for users)
        - Executable program
        - User manual
     2. Source code package (for developers)
        - Source code file hierarchy
        - Software architecture document