Overview of Computer Science: Part II
What is Software Development ?

Software

Wikipedia: Software is a general term used to describe a collection of computer programs, procedures and documentation that perform some tasks on a computer system.

- Computer programs: instructions for the computer/computer system (sequences of instructions, collections of instructions).
- Documentation: hard-copy or electronic documents describing
  - what a program should (must) do;
  - how the program does it;
  - how the program is to be used.
- Procedures: specific methodology for developing, testing and maintaining computer programs.

Software Engineering: a field within Computer Science that studies software develpment procedures.

Software Development Lifecycle

Textbook calls it The Software Development Method.

1. Requirements specification/analysis.
   - *What should the software do?*
2. Software Design.
   - *How do we make software do it?*
3. Software Implementation.
   - *Let's build the programs!*

4. Testing (Verification and Validation)
    - *Where did we make errors?*
    - *Did we build the program right?*
    - *Did we build the right program?*
5. Maintenance and Support
    - *Can we make it work under a new OS?*
    - *How about adding more features?*
    - *Testing did not reveal this fault, let's fix it now!*

Note: The **actual programming** occurs on Step 3 of the Lifecycle, which typically takes about 20-25% of the time/effort within the entire lifecycle.

Requirements Specification

Goals:

- *Understand the problem clearly!*
- Eliminate unimportant stuff.
- Concentrate on key aspects.

Process:

- Meet with clients.
- Ask questions.
- Listen to answers.
- Organize thoughts.
- Create a document.

Requirements engineers: software engineers whose main task in their organizations is formulation of requirements for future software systems.

Artifact: Requirements document.

Analysis

Goals:

- *"translate"* the problem(s) into the language of Computer Science/Software development.
- Discover any missing information/requirements.

Process:

- Study the requirements document.
- Determine if it is complete, if not, revise.
- Determine
    - available input information;
    - desired output information;
    - format for the input information;
    - format in which output needs to be provided;

Artifact: Revised Requirements document.

Design

Goals:

- **Solve the problems!**
- Determine **how** the system and its specific components will be implemented.

Process:

- Study the requirements document.
- Top-down design:
  - Break the overall task/problem into a number of smaller (and, often, independent) tasks.
  - Work on design for each smaller task (this may involve subdividing it as well).
  - Combine the designs of smaller tasks.

Artifact: Design document.

Implementation

Goals:

- **Develop the actual software.**

Process: *Well, this is what we will be studying.*

Artifact: Code.

Testing (Verification & Validation)

Goals:

- Unit testing: ensure that each component of the software system works.
- Verification: ensure that the system was built in the right way.
- Validation: ensure that the right system was built.

Process:

- Prepare test cases. Determine expected software behavior.
- Run test cases.
- Compare expected software behavior to exhibited software behavior.
- Report all failures (bugs/errors).
- Determine faults.

Failure: incorrect behavior of the software system.

Fault: an error in the code that can cause one or more failures.

Testers: software engineers whose main task is to test the software under development.

Quality Assurance (QA).

Maintenance

Goals:

- Ensure that software works with new hardware.
- Ensure that software works with new versions of Operating System.
- Ensure that software conforms to new regulations.

Note: Commonly people who maintain software are not the ones who wrote it.

Software Development in CPE 101

We are not going to be developing major software systems in this course.

Instead, our goal is to develop simple C programs. How does software lifecycle affect this course?

1. Requirements: the course will not involve requirements specificaction tasks. Instructor will be providing requirements for each programming assignment.
2. Design: In some circumstances, design for the program will also be provided. In some other situations, design will be left up to you, or will be provided partially.
3. Implementation: you will be implementing everything in C.
4. Testing: vigorous.
5. Maintenance: none.