Computer Science and Software Engineering...

Alex Dekhtyar

Definitions

Wikipedia[1] defines Computer Science as

> Computer Science ... [is] the scientific and mathematical approach in computing.

This encompasses a wide range of topics of both theoretical and practical importance. Less formally, computer science:

- studies computational problems;
- studies how to solve computational problems; and
- studies how to build computer programs that solve computational problems.

Computational problems. A computational problem is essentially any problem that can be solved using a computer.

One of the core aspects of CHEM 441 is to teach you to translate life sciences problems into computational problems.

Solutions of *computational problems* take the form of *software*.

Software. Software[2] is a set of *programs*, *procedures*, *algorithms* and its *documentation* concerned with the operation of a data processing system.

Software Development: the process of building software.

Software Engineering. Wikipedia defines software engineering as follows [3]:

> Software engineering (SE) is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches.

Informally, software engineering is a sub-discipline of computer science that deals with the craft of software development.

[1] http://en.wikipedia.org/wiki/Computer_Science
[2] http://en.wikipedia.org/wiki/Software
[3] http://en.wikipedia.org/wiki/Software_Engineering

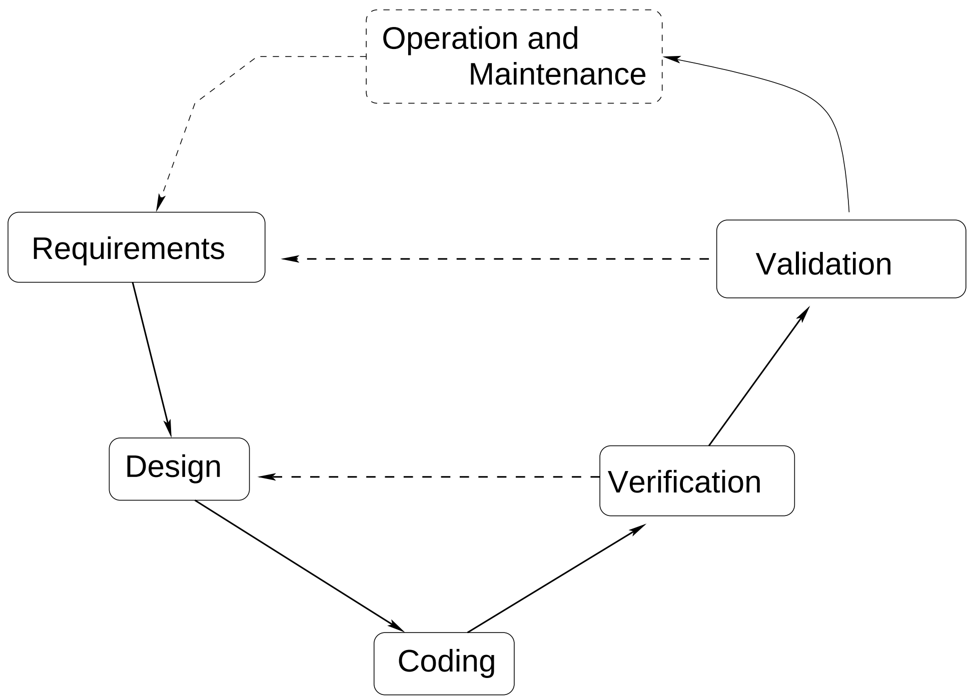

Figure 1: Software Lifecycle diagram: CHEM 441/CSC 448 edition.

Software Lifecycle

Throughout the course, you will develop software. This does not mean that you will write programs yourselves. However, you will participate in software development efforts together with people who will write programs.

Software Lifecycle. A software development project goes through a number of stages, each providing the information that is necessary on the next stage. The entire process of building a software system is called software lifecycle.

There are different ways to model software lifecycles. The software built in CHEM 441/CSC 448 will have lifecycle depicted in Figure 1.

Software Development Process: Roles

There are multiple categories of professionals who participate in the process of software development. The key categories are:

Customers: people who requested that the software be developed.

Domain Experts (aka subject-matter experts): people with special knowlege and skills in the area for which the software is being built. Sometimes, customers play the role of subject-matter experts, and sometimes, customers bring third-party subject matter experts.

Developers: computer scienctists and software engineers who develop the software. On large projects, developers can be responsible for different stages of software development:

- **Analysts:** developers responsible for deciding *how* the software will be built.
- **Programmers:** developers responsible for the actual programming.
- **Testers:** developers responsible for ensuring that the software works properly.

Users: people who will be using the completed and deployed software.

During this quarter:

- **CHEM 441 students** will take on the roles of customers, subject-matter experts and users.
- **CSC 448 students** will take on the roles of analysts and programmers.
- **Both CHEM 441** and **CSC 448 students** will be filling the role of testers.

Stages of Software Lifecycle

Requirements Analysis. The first stage of the software lifecycle.

- Customers and/or subject-matter experts convey to the analysts, what the softwre system is supposed to do.
- The desiderata for the software system are broken into individual components called **requirements**.
- Each **requirement** outlines *one desired property of the software system.*
- Requirements are put together into a document called the **requirements document** or the **requirements specification**.

Software Design. The second stage of the software lifecycle.

- Analysts go over the requirements document and determine **how** the requirements will be met by the software:
  - how the overall archtecture of the software system will look like;
  - how to break the software into individual components;
  - what each individual component of the software shall do;
  - what data structures will be used to store and manipulate the data used in the program;
  - how the output will be produced and conveyed to the user;
  - and so on...
- Analysts produce a **design document** which documents all decisions made.

Coding. The third stage of the software lifecycle. This is the stage when the actual programming is done.

- Programmers use the design document to write computer code implementing all design specifications.
- Programmers assemble all written components into the software.

Verification. Also known as **unit and system testing**. The term **verification** means the *is the process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase*[4]. Simply put, **verification** ensures that the software is written as prescribed in the design specification.

On this stage:

- Testers test individual components (functions, procedures, classes, methods) of the software to determine if these components have been developed as specified.
- All errors (bugs) discovered on this stage are reported to the programmers who modify code to fix them.

Validation. Also known as **acceptance testing**. **Validation** is [5]: *the process of evaluating a system ... at the end of the development process to determine whether it satisfies specified requirements*. Informally, **validation** stage asks the question *"did we build the right software product?"*.

During this stage:

- Testers perform evaluation of the entire software system to determine if its behavior matches the specified requirements.
- Any discovered discrepancies between the behavior of the software and the requirements are referred back to the analysts and programmers.
- Analysts and programmers redesign and re-implement any affected parts of the software to fix observed discrepancies.

Operation and Maintenance. The last stage of the software lifecycle.

- Developers deploy the verified and validated software system.
- Users begin using the software for their needs.
- Customers observe how the system is use, and based on these observations form new requirements for the next development cycle of the system.

Requirements

Requirement. A **requirement** is an expression of desired software behavior.

Requirements elicitation: a process of determining requirements for a software system. Usually, this process, *part of the requirements analysis stage*, involves customers/subject-matter experts and analysts. This is the part of the requirements analysis in which the requirements are actually **conveyed** to the development team.

[4] IEEE Standard Glossary of Software Engineering Terminology
[5] IEEE Standard Glossary of Software Engineering Terminology

Types of requirements. There are a number of different types of requirements, each with a different purpose:

1. **Functional requirement:** a description of desired behavior in terms of activities required to achieve it.
2. **Non-functional requirement:** a description of a quality characteristic of the software system.
3. **Design constraint:** an already-made design decision, that restricts the set of possible solutions.
4. **Process constraint:** a restriction on techniques and resources available to build the system.

Example

As an example, consider the problem of developing software that determines how many codons for Glycine are found in a given DNA sequence.

The customers and the users for this software are **CHEM 441** students. Additionally, because of their background knowledge in life sciences, we assume that **CHEM 441** students also play the role of subject-matter experts, i.e., they posses all the technical knowledge in the field of biology necessary to explain to the development team, i.e., **CSC 448** students, how to complete this task successfully.

The *context* of the problem involves working with a DNA sequence obtained from an on-line genomics resource. The DNA sequence is in the *nucleotide alphabet*, and is stored on disk in a file that has FASTA format[6] A sample input might look as follows:

```
> SEQ1: retrieved from ncbi.gov by Bob
AGTACGTAGTAGCTGCTGCTACGTGCGCTAGCTAGTACGTCA
CGACGTAGATGCTAGGCTGGACCTCCGATGC
```

Dr. Goodman wants to see *all occurrences* of Glycine counted: both on the positive and on the negative strand of the DNA fragment, and on any of the reading frames.

Based on the information supplied above, developers can elicit from the subject-matter experts the following **functional requirements** (note, this is not the full list of requirements, just some examples):

FR1. The program shall take as input a name of a file in FASTA format.

FR2. The program shall read the contents of the input file.

FR3. The following four combinations of three nucleotides represent Glycene: `GGT`, `GGC`, `GGA` and `GGG`.

FR4. The program shall find and count all occurrences of nucleotide triples representing Glycene.

FR5. The program shall also find and count all occurrences of nucleotide triples representing Glycene on the *reverse complement* of the input DNA sequence.

[6] A data format you will learn about from Dr. Goodman shortly in the course.

FR6. The program shall print the total number of Glycene codons found in the input sequence to screen.

Some **non-functional requirements** can also be specified:

NF1. Input Size. The program shall be able to compute the total number of Glycene codons for a DNA sequence of length of up to 2 million base pairs.

NF2. Speed. The program shall compute the total number of Glycene codons for a DNA sequence at the speed of 1 second per 100,000 base pairs of input.

NF3. Output language. The program shall print all its output in English.

Some **design constraints** on the program are:

DC1. Environment. The program has to run under the MS Windows family of operating systems.

DC2. Runtime. The development team shall supply the users with an executable file (.exe) of the program.

Some **process constraints** on the program are:

PC1. Timeline. The users need to start using the program on April 1.

PC2. Prototype. The developers shall build the program in a way that makes an early *usable* prototype of the program available to the users on March 29.

What it means for you

Throughout this quarter you will be playing the roles of

- customers,
- subject-matter experts,
- users, and
- testers (on occasion),

while working on the same team with **CSC 448** students, who will play the role of the development team.

You will participate in the following software lifecycle stages:

- Requirements analysis.
- Validation.
- Operation and Maintenance.

Your job, within the confines of each individual assignment will be:

- to convey (and, in later stages - to deterimine) the proper requirements for the software to the development team;

- to provide test and real data to be used by the software;
- to test (validate) the software built by the development team, to report any bugs found in the software to the development team, and to determine if the software produces the *correct desired output*;
- to use the sotware built by the development team to analyze your data and to find answers to the questions in your assignment.

Early in the quarter, you will be essentially given the requirements for the software to be built in your laboratory assignments. You are expected to simply be able to *convey* these requirements to your teammeates from **CSC 448** (note, the matching assignments for **CSC 448** students will essentially be "obtain requirements for the software you need to write from **CSC 441** members of your team").

Later in the quarter, you will be given assignments, in which it will be your job to determine what program you want written. This will make you (rather than Dr. Goodman) responsible for actually *coming up with appropriate requirements* (with the help of your **CSC 448** teammates).