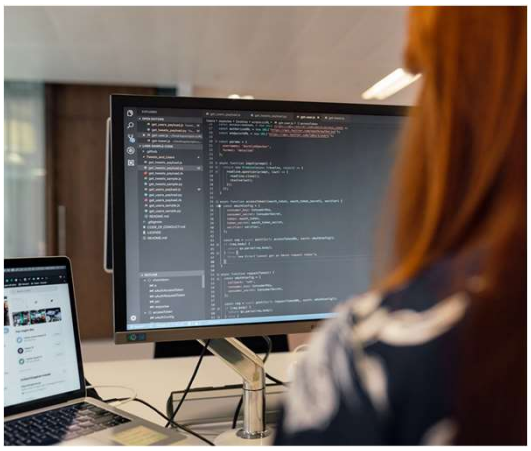
Software Engineering Introduction



Welcome to module 1.

This is an introduction to the study of software engineering. We have five video lectures for you.

Before we get into that, we need to point out that there is a discussion forum for this module. It's where you get to introduce yourself to the rest of the class and you get to see who else is here. We need you to finish your introduction by the first due date, as you can see in the course outlined. This is so that there will be enough time for the teams to form by the end of the module. The teams for the project are self selected. Based on the information in the introductions you need to find several other people in the class that you would like to work with on the team project this semester. We would prefer that the team size be at least three people, but not more than five. When you have found the members for your team, please post a note in the Discussion Forum telling everyone who's on your team. We've set up a forum named Project Teams for this purpose. Put the working name of your team in the subject line. You can change the official name of your team at any time up until you turn in your team charter document.

Finally, there's a quiz that will test your grasp of the information provided in the videos. Remember, that you'll have two chances to take each of the quizzes this course. If you do take a quiz twice, your two scores will be average together. Once you start a quiz you must finish it in one sitting. Each quiz is due at midnight on the due date. If you turn in your quiz late, it must be manually graded. We may not do this grading right away, so don’t be

alarmed if your late quiz hasn't been graded. We will normally not deduct points if your quiz is turned in late for a good reason, such as time zone differences, internet difficulties, etc.

So, let's get started with our introduction to software engineering.

Overview

- Definitions
- History
- Goals and principles of software engineering
- Is software engineering a profession?

2

- We start by defining terms, including the obvious one, “Software Engineering.”
- We will use a little history lesson to see why we need an engineering approach to the development of software.
- The main goals of software engineering are introduced, followed by a discussion of some of the important software engineering principles. We will re-visit many of these same topics later in the course, so it is important to get a foundational understanding of them now.
- Lastly, since many people consider engineering a profession, we might ask, “Is Software Engineering a Profession?” We will examine this issue from both sides.

Objectives

- Here is what you should be able to do upon completion of this module:
  - Define Software Engineering
  - Explain why an engineering discipline is needed for software development today
  - Discuss some of the problems that result if you don't apply engineering to the development of software
  - Recognize the goals and principles of Software Engineering
  - Argue why Software Engineering is, and is not, a Profession

3

- Here is what you should be able to do upon completion of this module:
  - Define Software Engineering.
  - Explain why an engineering discipline is needed for software development today.
  - Discuss some of the problems that result if you don't apply engineering to the development of software.
  - Recognize the goals and principles of Software Engineering.
  - Argue why Software Engineering is, and is not, a Profession.

Outline

- Definitions
- Origins of Software Engineering
- Goals and Principles of Software Engineering
- Software Engineering as a Profession

Here is the order of business.

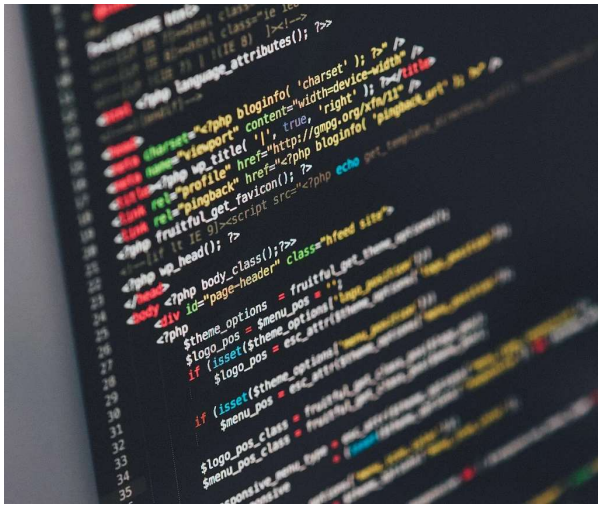
Software Engineering

Definitions

5

Before we get too far into our discussion of software engineering, we should step back and define our terms. Specifically, what do the terms “software” and “engineering” mean?

What is Software?



- Not Hardware
  - Program -- instructions that direct computer hardware to perform a task
  - Procedures and documentation
  - Data
- Types of software:
  - Application
  - System

The term, software, first appeared in a publication by John Tukey in 1958. (https://en.wikipedia.org/wiki/John_Tukey).

We use the term soft to distinguish it from hardware. It is soft in the sense that it can more easily be modified than hardware.

We usually think of programs when we think of software. A program is a collection of instructions, stored in the computer's memory, that can be executed by the computer hardware to perform some given task.

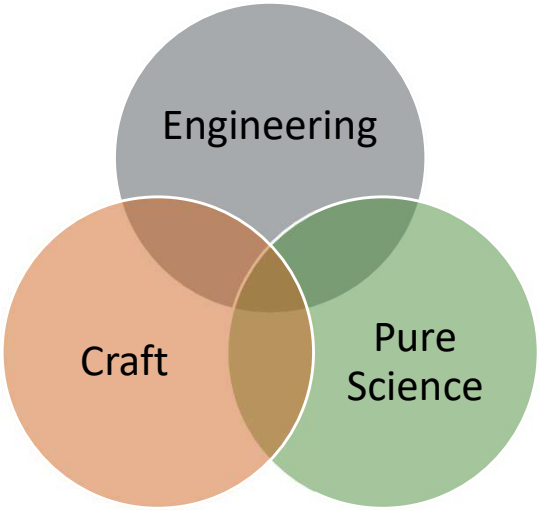
We can also include procedures and documentation in our definition of software. A program by itself requires documentation for people to be able to use it.

We also include data in our definition. Think about it. A program without data as input and output is not very useful.

We can partition software into two main categories

- Application software that performs a task for an end user. Application software is special purpose. Most of the software that is downloadable from the app store is application software.
- System software that makes it easier to use the computer. System software is general purpose. Examples include operating systems (such as IOS or Windows), and tools such as compilers or editors.

What is engineering?



For our definition of engineering, let's contrast it with endeavors such as Crafts or Pure Science.

Craft

- Produce real products for real people
- Artisans and craftspeople may obtain skills through apprenticeship
- No scientific background or formal training required

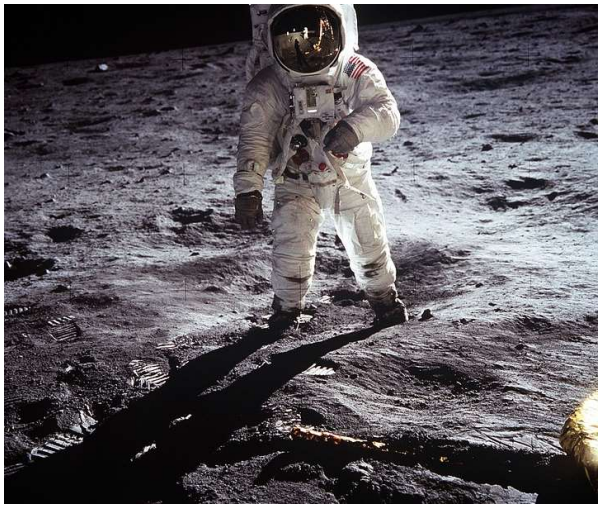
A craft is an activity that is performed by a craftsperson or artisan to produce real things for real people.

This would include arts and crafts such as weaving or painting as well as practical crafts such as carpentry or cooking.

The skills necessary to perform a craft are usually learned on the job through formal or informal apprenticeship.

Even if the craft is based on a scientific foundation, formal training in the basics is not usually necessary. For example, building codes are based on physics, but most carpenters don't really need to know the theory in order to apply the rules.

Science



- Discovery of facts and truth about the universe
- Application of the *scientific method* (running tests to prove or disprove a hypothesis)
- Based on scientific principles
- Academic training necessary

The purpose of pure science is to acquire knowledge and doesn't necessarily need to result in any practical application or result.

Scientists apply what is known as the scientific method. We start with a hypothesis or theory. Then we devise a set of tests to either prove or disprove the hypothesis.

Think Hubble telescope or the Large Hadron Collider. They don't produce any useful products, but they help us expand our knowledge of the universe.

In order to devise and analyze the proper experiments and tests to prove the hypotheses, we need to rely on sound scientific principles. This means that scientists must have an academic background.

Engineering

- Goal: to produce real products for real people
- Well defined, repeatable processes and procedures
  - Based on science
- Academic training necessary

Engineering shares some of the attributes of both science and craft.

Like a craft, the goal of engineers is to make real products for real people, unlike pure science that is mainly concerned with learning about the universe.

Like with pure science, engineering is based on scientific principles and requires academic training, unlike crafts that people can perform without necessarily knowing the underlying science.

One important aspect of engineering is that engineers devise and follow well defined, repeatable processes and procedures. These processes and procedures are in turn based on the science.

Examples

Craft	Science	Engineering
Plumber	Fluid dynamics	Water Resources Engineer
Carpenter	Static physics	Structural engineer
Electronics hobbyist	Solid state physics	Computer chip designer
Chemistry hobbyist	Organic Chemistry	Chemical engineer

Here are some examples of the differences among craft, science and engineering.

Consider the electronics hobbyist. You can build a complete computer by combining parts such as a chassis, motherboard, hard drive, memory chips, graphics controller, power supply, and so forth. You can buy all the necessary parts off the shelf. You don't really need to know exactly how each component works, just how to wire them up properly.

The computer components are designed by engineers. For example, a computer chip has a specification of what it does and how it interconnects with other components. The engineer refines this specification through analysis and design processes and procedures to devise a properly working chip.

The chip design, in turn, relies on principles from scientific fields of study such as solid state physics.

Software Engineering

- Real products for real people
- Based on scientific principles (computer science)
- Requires academic background
- Repeatable, well-defined procedures and processes

Craft	Science	Engineering
Hacking	Computer Science	Software Engineering

To sum up, then:

Software Engineering, like other engineering disciplines, is focused on building real software applications for real people.

Software engineering has an underlying theoretical basis: computer science. It is necessary to understand computer science to be a successful software engineer, but it is not sufficient. Software engineering processes go beyond what we know from computer science.

Because software engineering is based on computer science, an academic background is usually necessary.

So, to add another line to our chart from the previous page, we can think of a hobbyist that learns a programming language and puts together an application. The application might work for a specific purpose, but it may be difficult to modify to fix bugs or extend the capability. The application may take quite a bit of hacking to get it to work properly.

The proper way to build an application is to apply the processes and procedures of software engineering. These involve more than just getting the thing to work. There is a big difference between getting a program to work and getting it right. Software engineering is not the same as programming. Programming is only one part of the software engineering process.

As we will see throughout the semester, some of the processes and procedures of software engineering are based on the theory of computer science. Many are borrowed from other types of engineering.

Next

- Origins of software engineering

In our next video, we will explore the origins of software engineering.