TASK

Thinking Like a Software Engineer I — Introduction to Software Engineering

Visit our website

Introduction

WELCOME TO THE INTRODUCTION TO SOFTWARE ENGINEERING TASK!

Software affects almost every aspect of our daily lives. You cannot go a day without encountering some software; it is everywhere from microwaves to large military systems. This task aims to define software engineering and the attributes of a good Software Engineer. It also introduces an essential tool used by Software Engineers: the Integrated Development Environment.

Get in touch

Connect for support

Remember that with our courses, you're not alone! You can contact your mentor to get support on any aspect of your course.

The best way to get help is to login to **www.hyperiondev.com/portal** to start a chat with your mentor. You can also schedule a call or get support via email.

Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

WHAT IS SOFTWARE ENGINEERING?

To understand the concept of software engineering, you first need to understand what software is. You might think that a Software Engineer is simply concerned with computer programs that provide desired features, function, and performance when executed. However, this is not entirely the case. A Software Engineer is not just responsible for computer programs, but also all associated documentation and configuration data which ensures the correct operation of the program. Software that can be sold to a customer is known as a **software product**.

Now, what is software engineering? The IEEE (1993) defines software engineering as follows:

> *software engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).*

In simpler terms, software engineering is an engineering discipline, which is concerned with the development of a software product using well-defined scientific principles, methods and procedures.

SOFTWARE ENGINEERING VS PROGRAMMING

You might be tempted to use the terms *programmer* and *Software Engineer* interchangeably since they both develop software applications. However, this would be inaccurate. There are some distinct differences between the responsibilities and approaches to the job for these two roles.

Much like other engineering disciplines, software engineering approaches developing software as a formal process of understanding requirements, designing software, deploying, testing and maintaining it. Computer programmers, on the other hand, write code to the specifications given to them by Software Engineers. Software engineering is also a team activity, while programming is a solitary one.

WHAT MAKES A GOOD SOFTWARE ENGINEER?

You might assume that the most important quality that a Software Engineer can possess is technical experience and knowledge. However, this is not entirely true. A good Software Engineer is more than the sum of the technologies with which they have worked. Below is a list of some important traits that set successful Software Engineers apart:

- **Passion for Code** — This will come as no surprise, but a good Software Engineer should have a passion for programming. A Software Engineer should have an inherent interest in programming and enjoy working with code, as it is something that you will be working with almost every day.

- **Team Player** — Very few projects are small enough to be undertaken by a single person. Therefore, Software Engineers often find themselves working in a team. Software Engineers must have excellent teamwork skills.

- **Excellent Communication Skills** — Software Engineers should be able to communicate successfully with team members as well as clients. They should have both great written and verbal communication skills and be able to write reports as well as give instructions.

- **Curious** — Good Software Engineers are curious people. They want to know why and how things happen, like how the code and conditions produce certain software behaviour. This curiosity enables these engineers to think outside the box and come up with creative solutions to problems.

- **Wide-Ranging Technical Experience** — You can't escape the fact that a successful Software Engineer should have broad technical experience. Knowing a single programming language is not sufficient. A good Software Engineer should be well-versed in best practices like agile, task management software, version control and working in different development environments. A good Software Engineer should also always be willing to learn new technologies.

This is by no means an exhaustive list of the qualities of a good Software Engineer. However, it gives you a sense of what it takes to become successful in this field.

THE ECLIPSE IDE

An IDE, or Integrated Development Environment, is an essential tool used by Software Engineers to develop code. It consists of a source code editor, a compiler and a debugger. It increases a Software Engineer's productivity by condensing the entire process of code creation, compilation and testing. The Eclipse IDE is the most popular IDE for Java, with a market share of 65% in 2014. Eclipse is primarily used for developing Java applications. However, it can also be used for developing applications in other programming languages via plug-ins. These languages include JavaScript, PHP, Python and Ruby to name a few.

A note from the

HyperionDev Team

The History of Software Engineering

In 1968 a conference was organised by NATO to discuss the "software crisis". The software crisis was the name given to the difficulties encountered in the 1960s with developing large, complex systems. This was when the term "software engineering" was first introduced. It was suggested that the use of the engineering approach to develop software would reduce development costs and lead to more reliable software.

In the 1970s, the concepts of structured programming were developed. The concepts of object-oriented development were also introduced with the development of Smalltalk languages. During the late 1970s, the first programming environments were created.

During the 1980s, the Ada programming language was developed. This language included concepts of structured programming and information hiding. CASE (Computer-Aided Software Engineering) tools were also introduced during this time to support design methods. The 1980s also saw increased use of object-oriented programming with the use of languages such as C++ and Objective-C as well as object-oriented design methods.

In the 1990s, object-oriented programming became mainstream and Java was developed and released. During this time, the concept of software architecture received a great deal of attention. There was also increased use of client-server distributed architectures. The concept of component-based software engineering as well as the UML was also proposed.

During the early 2000s, the use of IDEs (Integrated development environments) became more common. The use of UML also became widespread. Use of scripting languages, like Python, for software development also increased. C# was also developed in the early 2000s and was seen as a competitor to Java.

Compulsory Task

- This task will walk you through how to install Eclipse.
- To use Eclipse for Java, you will first need to install the Java Development Kit (JDK). If you do not have it already installed, you can find a tutorial on how to install it **here**.
- Once the JDK is installed, you can download Eclipse from **https://www.eclipse.org/downloads/**
    - Under “Get Eclipse Neon” select “Download Packages.”
    - Then choose “Eclipse IDE for Java Developers.”
    - Under “Download Links” select your operating system.
    - You then click on the “Download” button to begin your download.
    - Once the file has downloaded, unzip it into a directory of your choice and click on the eclipse.exe file (Windows) to start the IDE.
- When you have successfully installed and downloaded Eclipse, you will be required to select a workspace directory. Once you have chosen a directory, select OK.
- You are then greeted by the welcome screen. Take a screenshot of this screen and send it to us to show that you have successfully installed Eclipse.

If you are having any difficulties installing or downloading Eclipse or the JDK, please feel free to contact your mentor.

Rate us

Share your thoughts

HyperionDev strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved or think we've done a good job?

Click here to share your thoughts anonymously.

References:

IEEE. (1993). *IEEE Standards Collection: Software Engineering*. IEEE Standard 610.12-1990.