Software Engineering

Lecture 1

Introduction to Software Engineering

Dr. Obuhuma James

Description

This topic aims at introducing you to Software Engineering and its importance by exploring the categories of software products, attributes of a good software, software process activities and the issues affecting most software. It also explores the desired critical ethical and professional behaviour of software engineers.

Learning Outcomes

By the end of this topic, you will be able to:

- Understand the term Software Engineering as used in the world of Computing
- Describe the attributes of a good software and issue affecting most software
- Appreciate the ethical and professional behaviour requirement of Software Engineers

Overview of Software Engineering

The current digital world revolves around existence and evolution of new hardware systems with software that controls their functioning. In this case, hardware refers to the physical, tangible elements of a computing system. On the other hand, software refers to a set of instructions written using a computer programing language that control operations of hardware or performs specific tasks. To be more precise, software is categorized as: System Software and Application Software where []:

- System Software

  This are software used by the computer to perform computer-specific tasks. Examples include the operating system, utilities and device drivers.
- Application Software

  This are software that allows users to complete tasks on a computer like document production, communications, among others.

With all these, it is clear that economies of all developed nations are dependent on software that runs the massive emerging hardware and performs the various tasks for users. Thus, the need for proper knowledge on Software Engineering. Software Engineering is hence concerned with the theories, methods and tools that facilitate professional software development [1]. On a more general scale, the cost of developing and maintaining software is often greater than that of hardware acquisition. More so, the cost of software maintenance

is always rated higher than its development cost, particularly for systems with a long-life span. Thus, as part of Software Engineering, the element of cost-effectiveness in software development is a crucial consideration.

Categories of Software Products

There are two broad categories of software products, namely, generic and customized software products. Generic software products are stand-alone systems marketed and sold to customers wishing to buy them. They are mostly distributed as commercial off the shelves (COTS) products. Specifications of the functionalities of the software are determined by the developer, and so are the decisions on changes to be made on such products. On the other hand, customized software products are designed to meet requirements and needs for specific customers. Unlike for the case of generic products, specifications of the functionalities of customized software are determined by the customer, and so are the decisions on changes to be made on such products.

Attributes of a Good Software

Despite the existence of many different software designed for specific functions, usability always stands out as one of the desired attributes of a good software. The customer, who is the end user of the software will always want the software to be easy to learn and use. It is however worth noting the following essential attributes of a good software as outline by Sommerville [1] that could be used as yard sticks in choice of software:

a) Maintainability

   This attribute allows software to be designed in a manner that permits evolution to accommodate the ever-changing user needs. It is worth pointing out that changing business environments result to the inevitable nature of software.

b) Dependability and Security

   The dependability attribute encompasses a wide range of characteristics that guarantees reliability, security and safety for software users. Reliability creates a sense of trustworthiness for both usage, data that gets into the software and information retrieved from the software. On the other hand, security and safety guarantees protection of the data and information circulating within the software. In line with this, secure and safe software shield against physical and economic damage in cases

of system failure. It also protects against access or damage of the system by malicious users.

c) Efficiency

Efficiency in terms of resource utilisation is a key positive desire of many software users. Resources in this case refer to memory, time and money. A good software should not result to wasteful use of such resources. Thus, software responsiveness, processing time, memory utilization among others builds up to the efficiency attribute of a good software.

d) Acceptability

Software is always designed for a target user or user groups. Thus, it must be acceptable to the desired target. A software is deemed to be acceptable if it is understandable and usable. Furthermore, the acceptability attribute also includes an element of software compatibility with other systems in place.

Software Engineering and its Importance

According to Sommerville [1], Software Engineering is concerned with all aspects of software production spanning from the early stages of system specification through to maintaining the system under use.

- As a discipline of engineering, Software Engineering emphasizes on the use of appropriate theories and methods in problem solving in fully consideration of organizational and financial constraints.
- Apart from the technical processes of development, Software Engineering also includes project management and the development of tools and methods in support of software production.

Through appropriate application of Software Engineering, reliable and trustworthy systems can be produced in a more economical and faster manner. It is worth noting that the highest cost of a software is always experienced in its lifetime, during maintenance. This is normally as a result of the need to change the system. Such unanticipated costs can however be lowered if proper Software Engineering isn involved in the production of software. For instance, if the maintainability attribute of a given software is well taken care of during the production period, then the cost of maintenance will be lowered to some extent.

Software Process Activities

The lifecycle of any given software involves varied sets of activities ranging from software specification all the way to software maintenance. Different Software Engineering book authors tend to express the activities in a different fashion using different names. However, in general, the following process activities end up being part of the process in that order:

- Specification

  This process activity entails definition of the requirements for the software to be developed and the constraints under which the software will operate. Software specification is normally a joint task done by customers and developers.
- Design and Implementation

  This process activity involved actual software design and programming. It is the point during which the specifications are transformed into an actual system.
- Validation

  This process activity gives room for confirmation of whether the developed software addresses customer requirements. The software is tested against the requirements specified during the software specification process activity.
- Maintenance

  This process activity applies to the period when the software is under use by the customer. It entails modification of the software to reflect any changes in customer and market needs.

Considering the existence of many varied types of software and the lack of a universal set of techniques applicable to all of these, Software Engineering methods and tools used end up being aligned to the type of application being developed, customer requirements and the development team's background. Examples of types of applications include: stand-alone applications, interactive transaction-based applications, embedded control systems, batch processing systems, entertainment systems, simulation and modeling systems, data collection systems, among others.

Despite the different types of application systems and development techniques, some fundamental principles Software Engineering principles should be applicable. Some of which include:

- There should be a manageable and well understood development process guiding system development.
- A full consideration of dependability and performance is crucial for all types of systems.
- A thorough understanding and management of the software specification and requirements is vital.
- A consideration on software reuse rather than complete development of new software should be considered.

Issues Affecting Most Software

Any given software is susceptible to issues in its lifetime, during usage. Sommerville [1] notes the following three general issues that affect most software:

- Heterogeneity

  The current world of technology has led to a major growth in distributed systems, where distributed systems refer to a computing environment in which various components are spread across multiple computers (or other computing devices) on a network. These devices split up the work, coordinating their efforts to complete the job more efficiently than if a single device had been responsible for the task. Having a software that survives well in such a heterogeneous environment is an issue of major concern.
- Business and social changes

  The fast growth in technology coupled with the ever-changing user needs render software change demands inevitable. The frequency of such change requests is not predictable and may sometimes end up being overwhelming to software developers.
- Security and trust

  As part of the key requirements of a good software, security and trust is essential. It is essential for any software to be secure and trustworthy as a way of enhancing on dependability by the customer.

The three issue end up being crucial for software developers desiring to develop and have in place software that meet the essential attributes of a good software.

Ethics and Professionalism in Software Engineering

Having processes and technical skills alone in place is not a final solution to successful Software Engineering. The ethical conduct of software engineers is paramount. According to Sommerville [1], ethical behaviour should be viewed in a manner that involves following a set of principles that are morally correct. This is beyond just upholding the law. For better professionalism in Software Engineering, the following are some of the professional responsibilities that Software Engineers should adhere to:

- Confidentiality

  Software Engineers should always respect and uphold to the confidentiality of their employers or clients with or without the existence of a signed confidentiality agreement.
- Competence

  Software Engineers should never misrepresent their competence levels that may result to acceptance of work that is beyond their competence.
- Intellectual property rights

  Software Engineers should be familiar of local laws governing the use of intellectual property rights and should protect the intellectual property rights of employers and clients. Such rights include patents, copyrights, etc.
- Computer misuse

  Software Engineers should not misuse other people's computers through the use of their technical skills.

Professional societies in the US cooperated to produce the ACM/IEEE code of ethical practice that requires members of these organisations to sign up to the code of practice upon employment [1]. The Code contains the following eight principles [2] related to the behaviour of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession.

1. PUBLIC - Software engineers shall act consistently with the public interest.
2. CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.

3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.
5. MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.
8. SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Summary

The topic has introduced the course by describing Software Engineering and its importance. A clear view of the two broad categories of software products, essential attributes of a good software and the process activities used in the development of software has been covered. Finally, the main issues affecting most software have also been brought out with a link on how the ethics and professional conduct of software engineers is vital.

Check Points

1. Describe the term Software Engineering and its importance.
2. Differentiate between the two categories of software products.
3. Describe the crucial attributes of a good software.
4. Outline the main software process activities.
5. State and explain the main issues affecting most software.
6. State and explain some of the professional responsibility requirements for Software Engineers.

Learning Resources

Core Textbooks

1. Sommerville, I., Software Engineering, 10th Edition, Addison Wesley, 2016.

Other Resources

2. Gotterbarn, D., Miller, K., & Rogerson, S. (1997). Software engineering code of ethics. Communications of the ACM, 40(11), 110-118.

References

[1] Sommerville, I., Software Engineering, 10th Edition, Addison Wesley, 2016.

[2] Gotterbarn, D., Miller, K., & Rogerson, S. (1997). Software engineering code of ethics. Communications of the ACM, 40(11), 110-118.

[3] Pressman, R. S. (2005). Software engineering: a practitioner's approach. Palgrave macmillan.

[4] Kendall, K. E. and Kendall, J. E., Systems Analysis and Design, 8th Edition, Pearson, 2011.