**CHAPTER 1**

**Q1. Explain why professional software is not just the program that are developed for the customer?**

Ans. A professionally developed software system is often more than a single program. A system may consist of several separate programs and configuration files that are used to set up these programs. It may include system documentation, which describes the structure of the system, user documentation, which explains how to use the system, and websites for users to download recent product information. At the end, software is not just the programs themselves but also all associated documentation, libraries, support websites, and configuration data that are needed to make these programs useful.

**Q2. What is the most important difference between generic software product development and custom software development? What might this mean in practice for users of generic software product?**

Ans. *Generic Product:* These are stand-alone system that produced by a development organization and sold on the open market to any customer who is able to buy them. Examples of this type of product include apps for mobile devices, software for PCs such as Databases, word processor, drawing packages, and project management tool. This kind of software also includes “vertical” applications designed for a specific market such as library information systems, accounting systems, or systems for maintaining dental records.

*Customized Software:* These are systems that are commissioned by and developed for a particular customer. A software contractor design and implements the software especially for that customer. Example of this type of software include control system for electronic devices, system written to support a particular business process, and air traffic control systems.

The critical distinction between these types of software is that, in generic products, the organization that develops the software controls the software specification. This means that if they run into development problems, they can rethink what is to be developed. For custom products, the specification is developed and controlled by the organization that is buying the software. The software developers must work to that specification.

**Q3. Briefly discuss why it is usually cheaper in the long run to use software engineering methods and techniques for software systems.**

Ans. The longevity of software largely depends on the use of software engineering methods. Typically, most software projects usually follow four (4) patterns;

* Software Specification: This entails the definition of the software requirements as well as the identified constraints.
* Software Development: This is the stage where the software product is developed based on the identified specification.
* Software Validation: The software validation stage is where the developed software is checked against the initial requirements. The goal of this is to ensure that the software meets the needs of the end user.
* Software Evolution: Software evolution is the concept that a given software must be developed in a manner that is flexible and easy to modify to reflect changing business rules, or market requirements.

Having highlighted the software engineering methods above, it is easy to see why it is cheaper to use software engineering methods and techniques when developing software systems. A good software must follow a software engineering method or technique.

**Q4: Software engineering is not only concerned with issues like system heterogeneity, business and social change, trust, and security, but also with ethical issues affecting the domain. Give some examples of ethical issues that have an impact on the software engineering domain**

Ans. The examples of the ethical issued that have an impact on the domain is as follows:

* Need more developers as rapid changes occurs
* Lack of resources
* High cost
* Need more and more scaling to the software
* Cyber security
* Data privacy
* Customer experience

**Q5. Based on your own knowledge of some of the application types discussed in Section 1.1.2, explain, with examples, why different application types require specialized software engineering techniques to support their design and development.**

Ans.Different application types require the use of different development techniques fora number of reasons:

Systems which are very expensive to change need extensive upfront analysis to ensure that the requirements are consistent and extensive validation to ensure that the system meets its specification. Different systems have different priorities for non-functional requirements. The techniques used to achieve safety are not required for interactive gaming; the extensive UI design required for games is not needed in safety-critical control systems. Some software systems have a relatively short lifetime (many web-based systems), others have a lifetime of tens of years (large command and control systems).

The techniques used to develop short-lifetime, rapid delivery systems (e. g. use of scripting languages, prototyping, etc.) are inappropriate for long-lifetime systems which require techniques that allow for long-term support such as design modelling

**Q6. Explain why the fundamental software engineering principles of process, dependability, requirements management, and reuse are relevant to all types of software system.**

Ans.The points to show the reasons are as given below:

* The process of requirement gathering is required in both generic and customized software.
* A document to define the development process is required in all type of software application.
* Updating is required in each type of software. For doing so, new version of each type of software is released.
* Maintenance is an important part of software development. It is required in each type of application development.
* Software has some minimum requirements to execute. So, platform dependability is considered in all software development.

**Q7. Explain how electronic connectivity between various development teams can support software engineering activities.**

Ans.The key enabler of team collaboration is communication. Without an effective communication medium, collaborating with others on a project is almost impossible.

In software development, it is crucial for team members to keep track of their individual development process. Electronic connectivity makes this process relatively stress-free. Availability of tools like GitHub make it easy for team members to collaborate and contribute code to a single remote repository. Use of electronic connectivity tools not only makes the development process a lot faster, but also improves the quality of the end-product or software.

**Q8. Noncertified individuals are still allowed to practice software engineering. Discuss some of the possible drawbacks of this.**

Ans.Some possible drawbacks of this situation are given below:

Non-certified individuals can misapply software engineering processes, they might make the software ineffective for what its designed for this could lead to a waste of resources.

Certified individuals may have to work along with non-certified individuals. This could give rise to disagreeing ideas about security and development processes.

At worst, these non-certified individuals can be absolutely counterproductive, create pointless confusion, or make else simple code awfully complex as the confused programmer, they try to shoehorn in activities where they do not belong.

**Q9. For each of the clauses in the ACM/IEEE Code of Ethics shown in Figure 1.4, propose an appropriate example that illustrates that clause.**

Ans.The main thing which actually defines the clause in 1.4 is some advantages and disadvantages of certifications:

Public > An example of acting in the public's interest is not to share any private information that the software engineer is privy to while working on a project.

Client and Employer > A good software engineer will not let the needs of the client or the wants of the employer harm the public.

Product > A good software engineer would not allow software to be released that they know is faulty and will crash.

Judgement > An example of using good judgement is, if a software engineer is approached by a client that wants them to build software that causes airplanes to crash, the software engineer knows that they must decline since although that may be in the client's interest, it is not in the publics.

Management > For example, Software engineering managers and leaders will not encourage employees to slack off when they are aware there is a deadline coming up.

Profession > For example, software engineers will not accept payment for inadequate software that they have admitted will not work as intended. This would reflect badly on other software engineers.

Colleagues > An example of following this clause would be a software engineer not stealing their coworkers code without permission.

Self > For example, a good software engineer will keep up with changes in technology and encourage others to do the same.

**Q10. The “Drone Revolution” is currently being debated and discussed all over the world. Drones are unmanned flying machines that are built and equipped with various kinds of software systems that allow them to see, hear, and act. Discuss some of the societal challenges of building such kinds of systems.**

Ans.Drones (Dynamic Remotely Operated Navigation Equipment) are our one of the greatest achievements in technology. It can be used in many fields like traffic, farming, geologically etc.

Drones are capable of doing many things for example:

Ingenuity is a small robotic solar helicopter and lightest spacecraft operating on Mars as part of NASA's Mars 2020 mission. On April 19, 2021, it successfully completed the first powered controlled flight by an aircraft on a planet besides Earth, taking off vertically, hovering and landing.