Software Engineering Section 1

Difference between Software and Computer programs

Software engineering is intended to support professional software development, rather than individual programming. It includes techniques that support program specification, design, and evolution, none of which are normally relevant for personal software development.

Many people think that software is simply another word for **computer programs.** However, when we are talking about software engineering, software is not just the programs themselves but also **all associated documentation and configuration data that is required to make these programs operate correctly.** A professionally developed software system is often **more than a single program.** The system usually consists of a number of 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.

This is one of the important differences between professional and amateur software development. If you are writing a program for yourself, no one else will use it and you don't have to worry about writing program guides, documenting the program design, etc. However, if you are writing software that other people will use and other engineers will change then you usually have to provide additional information as well as the code of the program.

Many applications need software engineering; they do not all need the same software engineering techniques. There are still many reports of software projects going wrong and **'software failures'.** Software engineering is criticized as inadequate for modern software development. However, in my view, many of these so-called software failures are a consequence of two factors:

- **Increasing demands**
- **Low expectations**

When we talk about the quality of professional software, we have to take into account that the software is used and changed by people apart from its developers. Quality is therefore not just concerned with what the software does. Rather, it has to include the software's behavior while it is executing. This is reflected in so-called **quality or non-functional software attributes.**
Examples of these attributes are the software's response time to a user query and the understandability of the program code. The specific set of attributes that you might expect from a software system obviously depends on its application. Therefore, a banking system must be **secure**, an interactive game must be **responsive,** a telephone switching system must be reliable, and so on.

Essential attributes of a good software

Maintainability Software should be written in such a way so that it can **evolve** to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security Software dependability includes a range of characteristics including reliability, security, and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilization, etc.
Acceptability Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable, and compatible with other systems that they use.

SFW Engineering

Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use. In this definition, there are two key phrases:

- Engineering discipline
- All aspects of software production

Key challenges facing software engineering

- Coping with increasing diversity, demands for reduced delivery times.
- Developing trustworthy software.

Software engineering is important for two reasons:

- Individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.

- It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of systems, the majority of costs are the costs of changing the software **after it has gone into use.**

Fundamental SFW engineering activities

The systematic approach that is used in software engineering is sometimes called a *software process.* ***A software process*** is a sequence of activities that leads to the production of a software product. There are four fundamental activities that are common to all software processes. These activities are:

- Software specification.
- Software development.
- Software validation.
- Software evolution.

There are many different types of software. Hence, there is no universal software engineering method or technique that is applicable for all of these. However, **there are three general issues** that affect many different types of software:

- Heterogeneity (different types of computer and mobile devices)
- Business and social change
- Security and trust (instapay)
- Scale

Software engineering diversity

Software engineering is a systematic approach to the production of software that takes into account:

- practical cost
- Schedule
- dependability issues
- The needs of software customers and producers.

How this systematic approach is actually implemented varies dramatically depending on the organization developing the software, the type of software, and the people involved in the development process.

There are no universal software engineering methods and techniques that are suitable for all systems and all companies. The most significant factor in determining which software engineering methods and techniques are most important is **the ty**
developed.

You use different software engineering techniques for each type of system because the software has quite different characteristics. For example, an embedded control system in an automobile is safety-critical and is burned into ROM when installed in the vehicle. It is therefore very expensive to change.

Software engineering fundamentals that apply to all types of software system:

1. **Developed using a managed and understood development process.** The organization developing the software should plan the development process and have clear ideas of what will be produced and when it will be completed. Of course, different processes are used for different types of software.
2. **Dependability and performance are important for all types of systems.** Software should behave as expected, without failures and should be available for use when it is required.
3. **Understanding and managing the software specification and requirements** (what the software should do) are important.
4. **Effective use as much as possible of existing resources.** This means that, where appropriate, you should reuse software that has already been developed rather than write new software.

Software engineering ethics

Like other engineering disciplines, software engineering is carried out within a social and legal framework that limits the freedom of people working in that area. As a software engineer, you must accept that your job involves wider responsibilities than simply the application of technical skills. You must also behave in an ethical and morally responsible way if you are to be respected as a professional engineer.

Issues of professional responsibility

- **Confidentiality** You should normally respect the confidentiality of your employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
- **Competence** You should not misrepresent your level of competence. You should not knowingly accept work that is outside your competence.
- **Intellectual property rights** You should be aware of local laws governing the use of intellectual property such as patents and copyright. You should be careful to ensure that the intellectual property of employers and clients is protected.

- **Computer misuse** You should not use your technical skills to misuse other people’s computers. Computer misuse ranges from relatively trivial (game playing on an employer’s machine, say) to extremely serious (dissemination of viruses or other malware).