

## Unit-1: Introduction

### **Professional Software Development:**

Lots of people write programs. People in business write spreadsheet programs to simplify their jobs; scientists and engineers write programs to process their experimental data; hobbyists write programs for their own interest and enjoyment.

However, most software development is a professional activity in which software is developed for business purposes, for inclusion in other devices, or as software products such as information systems and computer-aided design systems.

The key distinctions are that professional software is intended for use by someone apart from its developer. The teams rather than individuals usually develop the professional software. It is maintained and changed throughout its life.

**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, libraries, support websites, and configuration data that are needed to make these programs useful. 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.

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, and so on. 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. This is the main difference between program and professional software.

Software engineers are concerned with developing software products, that is, software that can be sold to a customer. There are two kinds of software product:

1. **Generic products:** These are stand-alone systems that are 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 processors, drawing packages, and project management tools. 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.
2. **Customized (or bespoke) software:** These are systems that are commissioned by and developed for a particular customer. A software contractor designs and implements the software especially for that customer. Examples of this type of software include control systems for electronic devices, systems written to support a particular business process, and air traffic control systems.

### **Essential attributes of good software:**

**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.

**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. Software has to be secure so that malicious users cannot 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, resource utilization, etc.

**Maintainability:** Software should be written in such a way 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.

## **Software Engineering:**

Engineering is about getting results of the required quality within schedule and budget. 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:** Engineers make things work. They apply theories, methods, and tools where these are appropriate. However, they use them selectively and always try to discover solutions to problems even when there are no applicable theories and methods. Engineers also recognize that they must work within organizational and financial constraints, and they must look for solutions within these constraints.
- **All aspects of software production:** Software engineering is not just concerned with the technical processes of software development. It also includes activities such as software project management and the development of tools, methods, and theories to support software development.

Software engineering is important for two reasons:

1. More and more, individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.
2. It is usually cheaper, in the long run, to use software engineering methods and techniques for professional software systems rather than just write programs as personal programming project. Failure to use software engineering method leads to higher costs for testing, quality assurance, and long-term maintenance.

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. Four fundamental activities are common to all software processes.

1. Software specification, where customers and engineers define the software that is to be produced and the constraints on its operation.
2. Software development, where the software is designed and programmed.
3. Software validation, where the software is checked to ensure that it is what the customer requires.
4. Software evolution, where the software is modified to reflect changing customer and market requirements.

Software engineering is related to both computer science and systems engineering.

1. **Computer science** is concerned with the theories and methods that underlie computers and software systems, whereas **software engineering** is concerned with the practical problems of producing software. Some knowledge of computer science is essential for software engineers in the same way that some knowledge of physics is essential for electrical engineers. Computer science theory, however, is often most applicable to relatively small programs. Elegant theories of computer science are rarely relevant to large, complex problems that require a software solution.
2. **System engineering** is concerned with all aspects of the development and evolution of complex systems where software plays a major role. System engineering is therefore concerned with hardware development, policy and process design, and system deployment, as well as software engineering. System engineers are involved in specifying the system, defining its overall architecture, and then integrating the different parts to create the finished system.

### **Software Engineering Diversity:**

Software engineering is a systematic approach to the production of software that takes into account practical cost, schedule, and dependability issues, as well as the needs of software customers and producers. The specific methods, tools, and techniques used depend 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 that are suitable for all systems and all companies.

Rather, a diverse set of software engineering methods and tools has evolved over the past 50 years. However, the SEMAT initiative (Jacobson et al. 2013) proposes that there can be a fundamental meta-process that can be instantiated to create different kinds of process. This is at an early stage of development and may be a basis for improving our current software engineering methods.

Perhaps the most significant factor in determining which software engineering methods and techniques are most important is the type of application being developed.

There are many different types of application, including:

**1. *Stand-alone applications:*** These are application systems that run on a personal computer or apps that run on a mobile device. They include all necessary functionality and may not need to be connected to a network. Examples of such applications are office applications on a PC, CAD programs, photo manipulation software, travel apps, productivity apps, and so on.

**2. *Interactive transaction-based applications:*** These are applications that execute on a remote computer and that are accessed by users from their own computers, phones, or tablets. Obviously, these include web applications such as e-commerce applications where you interact with a remote system to buy goods and services.

**3. *Embedded control systems:*** These are software control systems that control and manage hardware devices. Numerically, there are probably more embedded systems than any other type of system. Examples of embedded systems include the software in a mobile (cell) phone, software that controls antilock braking (ABS) in a car, and software in a microwave oven to control the cooking process.

**4. *Batch processing systems:*** These are business systems that are designed to process data in large batches. They process large numbers of individual inputs to create corresponding outputs. Examples of batch systems are periodic billing systems, such as phone billing systems, and salary payment systems.

**5. *Entertainment systems:*** These are systems for personal use that are intended to entertain the user. Most of these systems are games of one kind or another, which may run on special-purpose console hardware. The quality of the user interaction

offered is the most important distinguishing characteristic of entertainment systems.

**6. Systems for modeling and simulation:** These are systems that are developed by scientists and engineers to model physical processes or situations, which include many separate, interacting objects. These are often computationally intensive and require high-performance parallel systems for execution.

**7. Data collection and analysis systems:** Data collection systems are systems that collect data from their environment and send that data to other systems for processing. The software may have to interact with sensors and often is installed in a hostile environment such as inside an engine or in a remote location. “Big data” analysis may involve cloud-based systems carrying out statistical analysis and looking for relationships in the collected data.

**8. Systems of systems:** These are systems, used in enterprises and other large organizations that are composed of a number of other software systems. Some of these may be generic software products, such as an ERP system. Other systems in the assembly may be specially written for that environment.

Each type of system requires specialized software engineering techniques because the software has different characteristics. For example, an ***embedded control system*** in an automobile is safety-critical and is burned into ROM (read-only memory) when installed in the vehicle. It is therefore very expensive to change. Such a system needs extensive verification and validation so that the chances of having to recall cars after sale to fix software problems are minimized. User interaction is minimal (or perhaps nonexistent), so there is no need to use a development process that relies on user interface prototyping.

For an ***interactive web-based system or app, iterative development and delivery*** is the best approach, with the system being composed of reusable components. However, such an approach may be impractical for a system of systems, where detailed specifications of the system interactions have to be specified in advance so that each system can be separately developed.

Nevertheless, there are software engineering fundamentals that apply to all types of software systems:



1. They should be 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, the specific process that you should use depends on the type of software that you are developing.
2. Dependability and performance are important for all types of system. Software should behave as expected, without failures, and should be available for use when it is required. It should be safe in its operation and, as far as possible, should be secure against external attack. The system should perform efficiently and should not waste resources.
3. Understanding and managing the software specification and requirements (what the software should do) are important. You have to know what different customers and users of the system expect from it, and you have to manage their expectations so that a useful system can be delivered within budget and to schedule.
4. You should make effective use of existing resources. This means that, where appropriate, you should reuse software that has already been developed rather than write new software.

### **Internet Software Engineering:**

The development of the Internet and the World Wide Web has had a profound effect on all of our lives. Initially, the web was primarily a universally accessible information store, and it had little effect on software systems. These systems ran on local computers and were only accessible from within an organization. Around 2000, the web started to evolve, and more and more functionality was added to browsers. This meant that web-based systems could be developed where, instead of a special-purpose user interface, these systems could be accessed using a web browser. This led to the development of a vast range of new system products that delivered innovative services, accessed over the web. These are often funded by adverts that are displayed on the user's screen and do not involve direct payment from users.

As well as these system products, the development of web browsers that could run small programs and do some local processing led to an evolution in business and organizational software. Instead of writing software and deploying it on users'

PCs, the software was deployed on a web server. This made it much cheaper to change and upgrade the software, as there was no need to install the software on every PC. It also reduced costs, as user interface development is particularly expensive. Wherever it has been possible to do so, businesses have moved to web-based interaction with company software systems.

The notion of software as a service (SaS) was proposed early in the 21<sup>st</sup> century. This has now become the standard approach to the delivery of web-based system products such as Google Apps, Microsoft Office 365, and Adobe Creative Suite. More and more software runs on remote “clouds” instead of local servers and is accessed over the Internet. A computing cloud is a huge number of linked computer systems that is shared by many users. Users do not buy software but pay according to how much the software is used or are given free access in return for watching adverts that are displayed on their screen. If you use services such as web based mail, storage, or video, you are using a cloud-based system.

The advent (arrival) of the web has led to a dramatic change in the way that business software is organized. Before the web, business applications were mostly monolithic, single programs running on single computers or computer clusters. Communications were local, within an organization. Now, software is highly distributed, sometimes across the world. Business applications are not programmed from scratch but involve extensive reuse of components and programs.

This change in software organization has had a major effect on software engineering for web-based systems. For example:

1. Software reuse has become the dominant approach for constructing web-based systems. When building these systems, you think about how you can assemble them from preexisting software components and systems, often bundled together in a framework.
2. It is now generally recognized that it is impractical to specify all the requirements for such systems in advance. Web-based systems are always developed and delivered incrementally.
3. Software may be implemented using service-oriented software engineering, where the software components are stand-alone web services.



4. Interface development technology such as AJAX and HTML5 have emerged that supports the creation of rich interfaces within a web browser.

### **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.

It goes without saying that you should uphold normal standards of honesty and integrity. You should not use your skills and abilities to behave in a dishonest way or in a way that will bring disrepute to the software engineering profession. However, there are areas where standards of acceptable behavior are not bound by laws but by the more tenuous notion of professional responsibility. Some of these are:

1. **Confidentiality:** You should normally respect the confidentiality of your employers or clients regardless of whether or not a formal confidentiality agreement has been signed.
2. **Competence:** You should not misrepresent your level of competence. You should not knowingly accept work that is outside your competence.
3. **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.
4. **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) to extremely serious (dissemination of viruses or other malware).

A difficult situation for professional engineers arises when their employer acts in an unethical way. Say a company is responsible for developing a safety-critical system and, because of time pressure, falsifies the safety validation records. Is the

engineer's responsibility to maintain confidentiality or to alert the customer or publicize, in some way, that the delivered system may be unsafe?

The problem here is that there are no absolutes when it comes to safety. Although the system may not have been validated according to predefined criteria, these criteria may be too strict. The system may actually operate safely throughout its lifetime. It is also the case that, even when properly validated, the system may fail and cause an accident.

Another ethical issue is participation in the development of military and nuclear systems. Some people feel strongly about these issues and do not wish to participate in any systems development associated with defense systems. Others will work on military systems but not on weapons systems. Yet others feel that national security is an overriding principle and have no ethical objections to working on weapons systems.

In this situation, it is important that both employers and employees should make their views known to each other in advance. Where an organization is involved in military or nuclear work, it should be able to specify that employees must be willing to accept any work assignment. Equally, if an employee is taken on and makes clear that he or she does not wish to work on such systems, employers should not exert pressure to do so at some later date.

## **FAQs**

### ***What is software?***

It is a collection of computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.

### ***What are the attributes of good software?***

Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.

### ***What is software engineering?***

Software engineering is an engineering discipline that is concerned with all aspects of software production from initial conception to operation and maintenance.

***What are the fundamental software engineering activities?***

- Software specification
- Software development
- Software validation and
- Software evolution

***What is the difference between software engineering and computer science?***

Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.

***What is the difference between software engineering and system engineering?***

System engineering is concerned with all aspects of computer based systems development including hardware, software and process engineering. Software engineering is part of this more general process.

***What are the key challenges facing software engineering?***

Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software.

***What are the costs of software engineering?***

Roughly 60% of software costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.

***What are the best software engineering techniques and methods?***

While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system.

For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. There are no methods and techniques that are good for everything.

***What differences has the Internet made to software engineering?***

Not only has the Internet led to the development of massive, highly distributed, service-based systems, it has also supported the creation of an “app” industry for mobile devices which has changed the economics of software.

**Assignment:**

1. To be a professional software developer, what are the requirements you need to fulfill.
2. Discuss the ethical consideration a software engineer should be aware.
3. Justify, why there are no universal software engineering methods that are suitable for all systems and all companies.
4. If you need to develop a software for a company. Then what are the minimum attributes your software should hold to be accepted by the company.
5. Write short notes on
  - a. Software engineering
  - b. System engineering
  - c. Computer science

**End of Unit-1**