Software Engineering

Unit 1: Introduction

Compiled by: Sagar Rana Magar

Course Outline

- Software: Definition, Types, Characteristics
- Attributes of Good Software
- Definition of Software Engineering
- Software Engineering Costs
- Key Challenges that Software Engineering Facing
- System Engineering and Software Engineering
- Professional Practice

Program

- A computer program is a sequence of instructions in a programming language that a computer can execute or interpret.
- A program is an executable code, which serves some computational purpose.

Software

- Software is set of Computer programs associated with documentation & configuration data that is needed to make these programs operate correctly.
- Software is more than just a program code.
- Software is considered to be a collection of executable programming code, associated libraries and documentations.
- In general, software is a set of instructions(computer programs) that when executed provide desired features, functions and performance.

Software (contd…)

Some of the constituent items of software are described below:

1. Program
    - The program or code is definitely included in the software.
2. Data
    - The data on which the program operates is also considered as part of the software.
3. Documentation
    - All the documents related to the software are also considered as part of the software.

- So the software is not just the code, it also includes the data and all the documentation related to the program.

Difference between Program & Software

- If we talk about our daily uses, then software and program can be used interchangeably.
- But there is a huge difference in between software and program in technical language.
- Generally,
  - A software is a collection of programs and data files that are designed to perform some operations, and
  - On the other hand, program is a set of instructions that perform only a specific task that it is made for.

Difference between Program & Software

Software

- A software is a set of programs, procedures, data or instructions combined to execute a task.
- Software's are mainly dependent on operating system.
- Software's can be a program that generally runs on computer.
- If software's are not present in computers, then computer is useless.

Program

- A program is a set of instructions that are built to perform a specific task.
- Programs are mainly dependent on compiler.
- Programs cannot be a software.
- If programs are not present in computer, then also computer can function well because of operating system.

Types of Software

- There are two basic types of software related to the operation of a computer.

1. **System Software**
   - Manages the operation of, and the information flow through, all of the computer’s hardware devices (input, processing, storage and output).
2. **Application Software**
   - Includes the programs that do real work for user.
   - Performs a collection of related but specific tasks . In other words, it is designed for a specific tasks.

System Software

- Any computer software which manages and controls the hardware so that application software can perform a task.
- System Software are designed to control the operation and extend the processing functionalities of a computer system.
- Systems software carries out middleman tasks to ensure communication between other software and hardware to allow harmonious coexistence with the user.
- They actually enable functional interaction between hardware, software and the user.

Types of System Software

- System software is of five main types :
  - Operating system:
    - Harnesses communication between hardware, system programs, and other applications.
  - Language processor:
    - Translates high-level languages to low-level machine codes.
  - Utility software:
    - Ensures optimum functionality of devices and applications.
  - Device Driver:
    - Enables device communication with the OS and other programs
  - Firmware:
    - Enables device control and identification.

Operating System

- Operating system is the system software which makes a computer to actually work.
- The operating system is a type of system software kernel that sits between computer hardware and end user.
- It is installed first on a computer to allow devices and applications to be identified and therefore functional.
- Operating software is the first layer of software to be loaded into memory every time a computer is powered up.
- Eg: Microsoft Windows, Mac OS, Linux, Unix, etc

Programming Translators

- These are intermediate programs to translate high-level language source code to machine language code.
- The former is a collection of programming languages that are easy for humans to comprehend and code (i.e., high level language like Java, C++, Python, PHP, BASIC or assembly language).
- The latter is a complex code only understood by the processor (machine code).
- Translator programs may perform a complete translation of program codes or translate every other instruction at a time.
- Eg: Assemblers, Interpreter, Compiler

Utility Software

- Utilities software is system software which is manufactured
    - to help, analyze, configure, optimize or to maintain a computer.
- It also helps in maintenance and problem solving of a computer.
- These apps are usually bundled with an OS. They track performance and alert the system if there's a problem
- Example:
    - Backup Software,
    - Antivirus Software,
    - Scanning and defragmenting disk,
    - file compression software,
    - Troubleshooting software

Device Drivers

- Driver software is a type of system software which brings computer devices and peripherals to life.
- Driver software communicates with hardware and control devices and peripherals attached to a computer.
- It does this by gathering input from the OS (operating system) and giving instructions to the hardware to perform an action or other designated task.
- Without drivers, the OS would not assign any duties.
- If a device is newer than the operating system, the user may have to download drivers from manufacturer websites or alternative sources.

Firmware

- Firmware is the operational software embedded within a flash, ROM, or EPROM memory chip for the OS to identify it.
- It directly manages and controls all activities of any single hardware.
- It provides essential information regarding how the device interacts with other hardware.
- Traditionally, it was installed on non-volatile chips and could be upgraded only by swapping them with new, preprogrammed chips.
- Today, firmware is stored in flash chips, which can be upgraded without swapping semiconductor chips.
- Firmware in computer can be accessed through BIOS(Basic Input Output System) or UEFI(Unified Extended Firmware Interface) platform.

Application Software

- Application Software is also known as end-user programs or application package.
- Application Software is a program that does real work for the user.
- It is mostly created to perform a specific task for a user.
- It can also be referred to as non-essential software
  - as their requirement is highly subjective, and
  - their absence does not affect the functioning of the system.
- Eg: Word Processor, DBMS, Spreadsheet, notepad, etc

Types of Application Software

- General Purpose Application Software
  - A collection of programs designed for general use.
- Specific Purpose Application Software
  - Software that is created or designed by big companies and government departments.

General Purpose Application Software

- It is also referred to as off-the-shelf applications.
- It is a type of application that can be used for a variety of tasks. It is not limited to one particular function.
- General purpose applications are designed as fully-featured packages.
- General purpose applications are available in standalone versions or are bundled together to make up application suites.
- Application suites such as MS Office, Apache OpenOffice, iWork, WPS Office, CorelDRAW Graphics Suite, Adobe Creative Suite are bundles of applications with different functionality

Specific Purpose Application Software

- Special purpose application software is a type of software created to execute one specific task.
- For example, scientific calculator software can carry out calculations, but it cannot be used to perform any other tasks,
  - such as writing an essay or designing a logo.
- Other examples of special purpose application software are
  - web browsers,
  - media players,
  - Web browser,
  - calendar programs etc.

Custom Application Software

- Also called as Bespoke Software or Tailored Software.
- Custom software is tailor-made for a client's specific needs.
- Sometimes, users need a piece of software that can perform a certain task,
  - but the general purpose software that is available does not offer the functionality required.
- Eg: Attendance system, security code system

Characteristics of Software

1. Operational:
   - This tells how good a software works on operations
   - So, budget , usability, efficiency, correctness ,functionality , dependability , security and safety.
2. Transitional:
   - Transitional is important when an application is shifted from one platform to another.
   - So, portability, reusability and adaptability come in this area.
3. Maintenance:
   - This specifies how good a software works in the changing environment.
   - Modularity, maintainability, flexibility and scalability come in maintenance part.

Attributes of Good Software

1. **Functionality:**
   - A good software must be able to do what it was designed to do.
2. **Usability:**
   - Software must be useable without wasting effort of user for whom it is designed.
   - This means that it should have appropriate user interface and adequate documentation.
3. **Efficiency:**
   - Efficiency means that perform it's operations with minimal time, processing power and system resources.
   - A good software uses the least amount of processing power and memory needed to achieve the desired result.
4. **Maintainability:**
   - A good software must evolve with changing requirements.
5. **Dependability**
   - Software dependability includes characteristics like 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.
6. **Portability**
   - A system that operates in different environments from which it's originally designed makes it good.

Engineering

- Engineering is the application of scientific and practical knowledge
  - to invent, design, build, maintain, and improve frameworks, processes, etc
- Engineering is all about selecting the most appropriate method for a set of circumstances so a more creative, less formal approach to development may be effective in some circumstances.

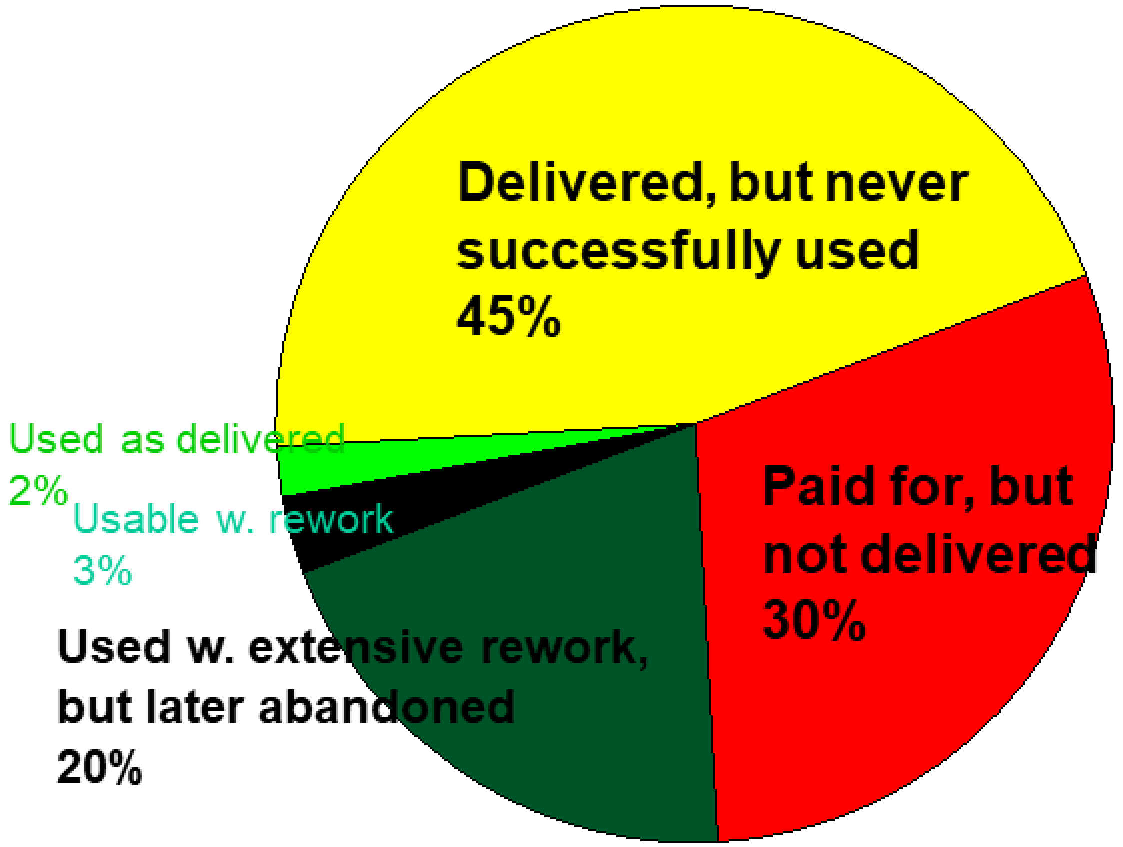
Software Engineering

- Software engineering is a detailed study of engineering to the design, development and maintenance of software.
- 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.
- IEEE defines software engineering as:
  - The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software.
- According to Boehm
  - The practical application of scientific knowledge in the design and construction of computer programs and the associated documentation required to develop, operate, and maintain them.

Software Engineering (contd…)

- Software engineers should
  - adopt a **systematic and organized approach** to their work
  - Use appropriate tools and technique depending upon the problems to be solved, the development constraints and the resource available.
- Software engineering is important because:
  1. Todays technological world relies on the advanced software systems. Thus, 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 software systems.
     - Software engineering is concerned with cost-effective software development

Software Development without SE



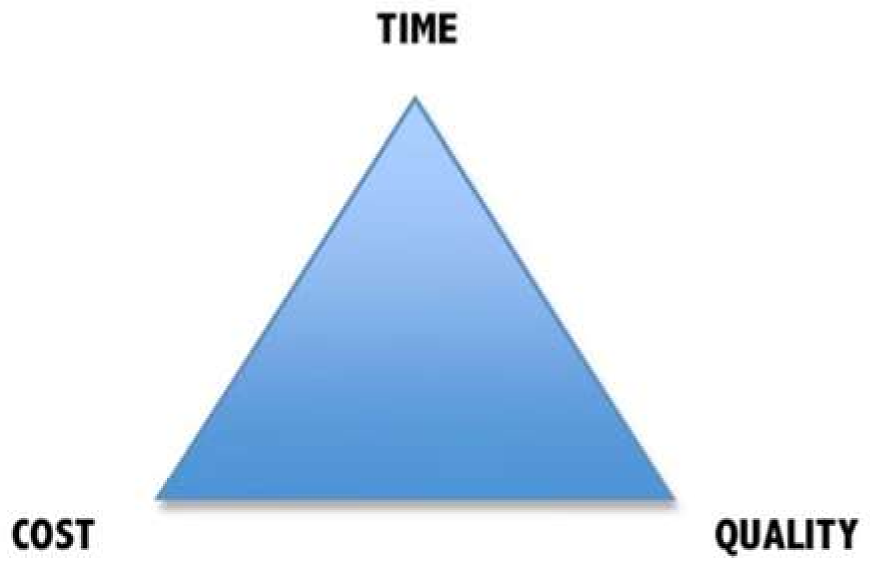
Why SE?

- Complex system need a disciplined approach for designing, developing and managing them.

With out SE in software development…

Projects were:

- Late.
- Over budget.
- Unreliable.
- Difficult to maintain.
- Performed poorly.



Software Engineering Costs

- Software Engineering cost is typically concerned with the financial spend on the effort to develop and test the software
- This can also include requirements review, maintenance, training, managing and buying extra equipment, servers and software.
- Roughly 60% of costs are development costs, 40% are testing costs.
- For custom software, evolution costs often exceed development costs.
- Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and system reliability.
- Distribution of costs depends on the development model that is used

Challenges that Software Engineering face

- **Heterogeneity**
  - Increasingly, software are required to operate across networks that include different types of computer and with different kinds of support systems.
  - Thus, need of developing techniques for building software that can cope with heterogeneous platforms and execution environments.
- **Delivery**
  - Developing techniques that lead to faster delivery of software
  - The delivery challenge is the challenge of shortening delivery times for large and complex systems without compromising system quality.
- **Trust**
  - Developing techniques that demonstrate that software can be trusted by its users.

Other Challenges in Software Engineering

- **Quality assurance**
  - Sometimes, to save time and submit the project before the deadlines, the developers tend to suppress the errors and do not review the codes.
  - This creates problems for the end product which does not meet the requirements of the users.
- **Project infrastructure**
  - A suitable project environment facilitates the creation of the software in the best possible way.
  - Project infrastructure could mean a high performance software development tools, powerful computing platforms, inefficient data storage architectures or improper networks and connectivity.
  - However, if a project environment is unestablished, then it feigns umpteen issues on the project delivery.
  - In the absence of an environment, the company cannot continue with the project in the given budget and time.

Other Challenges in Software Engineering

- **Cross-Platform Functionality**
  - Today, there’s the expectation that companies need to offer a unified—or rather “seamless”—experience across all platforms, channels, and devices.
  - One of the biggest challenges facing software development teams is the pressure to maintain consistency across all touchpoints and
  - be ready to provide on-demand support wherever customers decide to make contact.
- **Rapid technology advancement**
  - Every technology advancement is a blessing for the IT industry.
  - But at the same time, technology evolving at a phenomenal rate leads to an added pressure for software development professionals to leverage these upcoming technology trends in software product development to gain a cutting edge over competitors and stand out in the market.

Other Challenges in Software Engineering

- **Increasing customer demands**
  - Software projects are generally conceptual and are aimed at designing and developing software products that meet varied customer demands.
  - To develop even the simplest application or product, developers must clearly understand the underlying business concept and bring in the required features to satisfy the growing customer demands.
  - Changing Requirements during the development process brings challenges for the software developers. Sometimes they won't be able to deal with changing requirements.
- **Time limitations**
  - Software development is a time-game.
  - Developers work under pressured environments and strive to complete project requirements within strict and scanty timelines.
  - Time constraints often bring down efficiencies of development teams and lead to mediocre quality software products in the end.

Software Engineering Vs Computer Science

- Computer science is concerned with theory and fundamental;
- Software engineering is concerned with the practicalities of developing and delivering useful software.

System Engineering & Software 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 process concerned with developing the software infrastructure, control, applications and databases in the system.

- System engineers are involved in system specification, architectural design, integration and deployment.

Professional And Ethical Responsibility

- Software engineering involves **wider responsibilities** than simply the application of technical skills.
- Their work is carried out within a legal and social framework.
- Software engineers must behave in an ethical and morally responsible way if they are to be respected as professionals.
- Ethical behavior is more than simply upholding the law.
- It goes without saying that you should uphold normal standards of honesty and integrity.

Issue of Professional Responsibility

1. **Confidentiality**
   - Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
2. **Competence**
   - Engineers should not misrepresent their level of competence.
   - They should not knowingly accept work which is out of their competence.
3. **Intellectual property rights**
   - Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc.
   - They should be careful to ensure that intellectual property of employers & clients is protected.
4. **Computer misuse**
   - Software engineers should not use their technical skills to misuse other people's computers.

The Software Engineering Code of Ethics and Professional Practice

- Software Engineering Code of Ethics and Professional Practice (Version 5.2) as recommended by the (Association for Computing Machinery) ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices and jointly approved by the ACM and the IEEE-CS as the standard for teaching and practicing software engineering.

- The Code contains eight Principles related to the behavior 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.

The Software Engineering Code of Ethics and Professional Practice

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 (moral) 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.