Software Engineering

Dr. Fatma ElSayed

Computer Science Department
fatma.elsayed@fci.bu.edu.eg

Course Information

Contents

- Introduction to Software Engineering
- Software Processes
- Requirements Engineering
- System Design
- System Modelling
- System Architecture
- Software Testing Strategies
- Software Testing Techniques
- Technical Metrics for Software

Course Information

Contents

- Introduction to Software Engineering
- Software Processes
- Requirements Engineering
- System Design
- System Modelling
- System Architecture
- Software Testing Strategies
- Software Testing Techniques
- Technical Metrics for Software

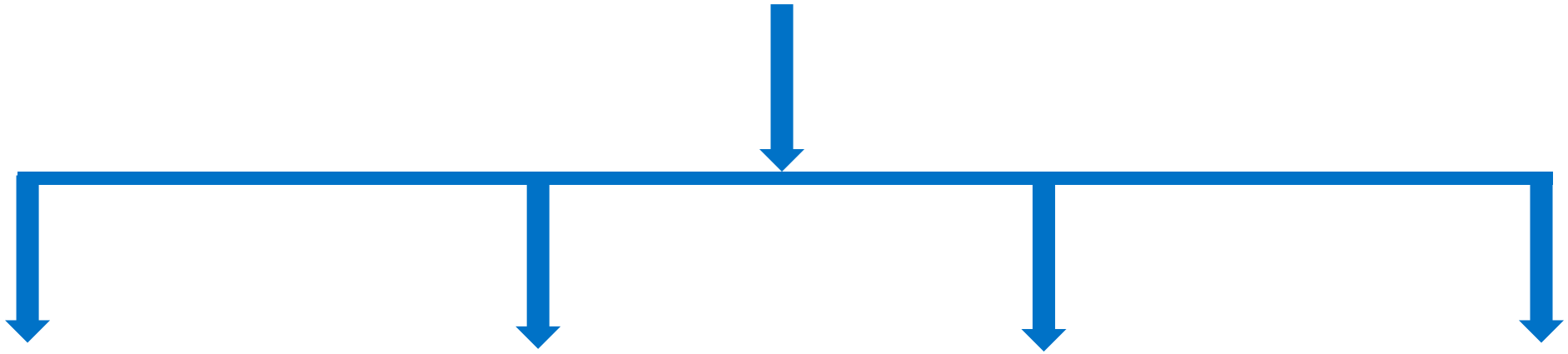
Chapter 1: Introduction to Software Engineering

What is Software?

- Computer programs and associated documentation.
- Software is not just the programs themselves but also all **associated documentation** that is required to set up these programs and make them operate correctly, explains how to use the system, and **websites for users to download recent product information**.
- Software engineering is intended to support **professional** software development, rather than **individual** programming.
    - Professional software, intended for use by someone apart from its developer, is usually **developed by teams rather than individuals.** It is **maintained and changed** throughout its life.

Attributes of Good Software

- The software should deliver the required functionality and performance to the user. **Four** important attributes that professional software must have:



Maintainability

Software must meet changing needs and future upgrades can be made easily, quickly, and cheaply

Dependability

Software must be trustworthy, shouldn't cause physical or economic damage in case of system failure

Efficiency

Software should not make wasteful use of system resources, memory and processor usage should be minimized

Acceptability

Software must accepted by the users for which it was designed. This means it must be understandable, usable and compatible with other systems.

What is Software Engineering?

- Software engineering is an engineering discipline that is concerned with all aspects of software production using systematic well-defined scientific techniques.
- The result of software engineering is an effective and reliable software product.
- Software engineers should adopt a systematic approach, using appropriate tools and techniques based on the problem, constraints, and resources.
- Software engineers design, develop, and test software programs that apply computer technology to everyday processes.

Software Engineer vs Software Developer

Software Developer	Software Engineer
A professional who builds software which runs across various types of computer.	A professional who applies the principles of software engineering for designing, development, maintenance, and testing computer software
Primarily a solitary activity.	Team activity.
Write a complete program.	Works with other components of the hardware system whereas software.
Use readymade tools to build apps.	Creates the tools to develop software.
Tend to do everything engineer do, but on a limited scale.	Tend to solve issues on a much larger scale

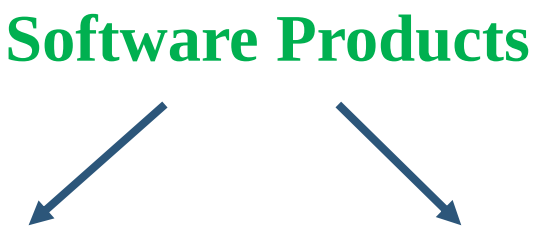
Software Engineering vs. System Engineering & Computer Science

- System engineering is concerned with all aspects of computer-based systems development including **hardware, software, human work, and procedures** needed to fulfill this mission. Software engineering is part of this process concerned with developing the software.

- Computer science focuses on **theory** and fundamentals *(programming languages, graphics, artificial intelligence and machine learning,….)*; software engineering is concerned with the practicalities of developing and delivering useful software.
- Computer science gives the scientific foundation for software as electrical engineering depends on physics

Software Products

- Software products are software systems delivered to a customer with the documentation which describes how to install and use the system.



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.

Customized Products

These are systems that are specially built for a particular customer. The software is developed especially for that customer by some developer.

Software Products

Generic Products

Universal software produced for the open market

Examples:

Software for PCs such as databases, word processors, drawing packages, and project-management tools.

Customized Products

Unique solution for specific customers

Examples:

Systems written to support a particular business process, admission and registration system for some universities, air traffic control systems, and control systems for electronic devices

Difference between Generic and Customized

Comparison parameter	Generic	Custom
Functionality	has functionality designed to resolve a specific problem for several entities.	has functionality made to resolve a problem for a specific entity.
Specifications	are produced internally by the marketing department of the product company	according to a contract between customer and developer
Quality	quality isn't the most parameter a development company follows	quality is the main criterion in custom software product
Number of users	large	limited users
Marketing	marketing is required	marketing is not required
Needs and updates	defined by market demand generic software is done based on future updates	custom software is done according to the time, budget and needs are defined by the customer
Development cost	usually not high	high(single customer)

Software Process

- A software process is a sequence of activities that leads to the production of a software product.
- Different types of systems need different development processes.
- There are **four fundamental** activities that are common to all software processes.
  - **Specification**
  - **Development *[Design, Implementation]***
  - **Validation**
  - **Evolution**

Software Process: Activities

Specification	Development	Validation	Evolution
where customers and engineers define the software that is to be produced and the constraints on its operation. what the system should do and its development constraints	where the software is designed and programmed. production of the software	where the software is evaluated to confirm that it meets the customer's requirements.	where the software is updated to reflect changing demands.

Key Challenges facing Software Engineering

- Any company is going to develop new software it should deal with these key challenges

1. Coping with increasing diversity **(Heterogeneity)**

   The diversity of software and hardware platforms and fast changing in technology should not affect the developing software.

2. Demands for reduced delivery times **(Delivery)**

   Developing techniques that lead to faster delivery of software.

3. Developing trustworthy software. **(Trust)**

   Developing techniques that demonstrate that software can be trusted by its users.

Software Costs

- 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.
- Software engineering is concerned with **cost-effective** software development.

Best Software Engineering Techniques and Methods

- While all software projects have to be professionally managed and developed, **different types of software systems require different development techniques.**
- You can't, therefore, say that one method is better than another.
- There are **no universal** software engineering methods and techniques that are suitable for all systems and all companies.
- The **key factor** in choosing software engineering methods and techniques is the **type of application** being developed.

Different Types of Application

1. **Stand-alone applications**

   They are designed to operate independently on a local computer without requiring a network connection. They include all the necessary components and functionalities within the application itself.

   Examples: office applications on a PC, photo manipulation software, etc.

2. **Embedded control systems**

   They are specialized software systems designed to control and manage hardware devices. They are tailored for performance and integration with the hardware components.

   Examples: software that controls anti-lock braking in a car, and software in a microwave oven to control the cooking process.

Different Types of Application

3. Interactive transaction-based applications

They are designed to allow users to interact with a remote system through their own PCs or terminals. These applications enable real-time user interaction with a remote system. Users can perform actions, make queries, and receive responses from the system.

Examples: e-commerce applications (e.g. Amazon), online banking systems.

4. Batch processing systems

These are business systems that are designed to process data in large batches rather than in real time. This approach is often used for tasks that don't require immediate processing. They process large numbers of individual inputs to create corresponding outputs.

Examples: include periodic billing systems (phone billing systems).

Different Types of Application

5. Data collection systems

These systems are designed to gather data from various sources through sensors and transmit this data to other systems for further processing or analysis. They often operate in challenging environments, such as within engines or remote locations, and the software must be robust enough to handle these conditions while interacting with the sensors effectively.

Examples: agricultural systems (use sensors to collect data on soil moisture, crop health, and weather conditions) and healthcare monitoring systems (wearable devices and medical sensors collect data on vital signs such as heart rate, blood glucose levels, and oxygen saturation)

Different Types of Application

6. **Entertainment systems**

   These are systems that are primarily for personal use and which are intended to entertain the user.

7. **Systems for modelling and simulation**

   These are systems that are developed by scientists and engineers to model physical processes or situations, which include many, separate, interacting objects.

8. **Systems of systems**

   These are systems that are composed of a number of other software systems.

Software Engineering Fundamentals

- Some fundamental principles apply to all types of software system, irrespective of the development techniques used:
  - Systems should be developed using a managed and understood development process. Of course, different processes are used for different types of software.
  - Dependability and performance are important for all types of system.
  - Understanding and managing the software specification and requirements (what the software should do) are important.
  - Where appropriate, you should reuse software that has already been developed rather than write new software.

Software Engineering and the web

- The Web is now a platform for running application and organizations are increasingly developing web-based systems rather than local systems.
- Web **services** allow application functionality to be accessed over the web.
- **Cloud computing** is an approach to the provision of computer services where applications run remotely on the ‘cloud’.
  - Users do not buy software buy pay according to use.

Software Engineering Ethics

Professional and Ethical Responsibility

- Software engineering involves wider responsibilities than simply the application of technical skills.
- Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals.
- Ethical behavior is more than simply upholding the law.

Issues of Professional Responsibility

- **Confidentiality**
  - Engineers should normally respect the confidentiality of their employers or clients of whether or not a formal confidentiality agreement has been signed. (Keep sensitive information private and protected from unauthorized individuals or entities.)

- **Competence**

  Engineers should not misrepresent your level of competence. They should be honest about their abilities, skills, and qualifications. (If a task or job is outside their current level of expertise, they should avoid accepting it, as doing so could lead to poor performance, errors, or harm to the employer, client, or project)

Issues of Professional Responsibility

- **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 the intellectual property of employers and clients is protected.

- **Computer misuse**

  Software engineers should not use their technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine) to extremely serious (spreading of viruses).

  e.g. Test software for the company or client. This could involve accessing the computer to install programs, run tests, or troubleshoot issues.

ACM/IEEE Code of Ethics

- The professional societies in the US have cooperated to produce a code of ethical practice.

- 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.

  (Software engineers should follow in their professional work. By following this code, software engineers ensure they behave ethically, contribute positively to society, and maintain trust in their profession.)

Software Engineering Code of Ethics and Professional Practice

ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices

PREAMBLE

The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code.

Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

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.

Ethical Dilemmas

- In any situation where different people have different views and objectives you are likely to be faced with **ethical dilemmas**.
  - Disagreement in principle with the policies of senior management. (conflict between your own moral principles or professional values)
  - Your employer acts in an unethical way and releases a safety-critical system without finishing the testing of the system.
  - Participation in the development of military weapons systems or nuclear systems.

Case studies

1. **A personal insulin pump**

   An embedded system in an insulin pump used by diabetics to maintain blood glucose control.

2. **A mental health case patient management system**

   Mentcare. A system used to maintain records of people receiving care for mental health problems.

3. **A wilderness weather station**

   A data collection system that collects data about weather conditions in remote areas.

NEXT: Software Processes

Thanks! .. Questions?