# 1. Introduction to Software Engineering

IT2206 - Fundamentals of Software Engineering

Level I - Semester 2

# Learning Outcomes

After successfully completing this course you should be able to;

- explain the software engineering principles and techniques that are used in developing quality software products
- apply software engineering principles and techniques appropriately to develop a moderately complex software system

# What is Software?

- Software is any set of machine-readable instructions to the computer's processor to perform specific operations.
- As a product, a Software system is a collection of intercommunicating components, programs, configuration files, system documentation and user documentation
- Software products may be developed for a particular customer or may be developed for a general market.
  - **Generic** - developed to be sold to a range of different customers
  - **Bespoke (custom)** - developed for a single customer according to their specification

# Software Projects vs. Other Projects

The main difference in software engineering compared to other engineering disciplines are as follows;

- It is difficult  for a customer to specify requirements completely.
- It  is difficult for the developer to understand fully the customer needs.
- Software requirements change regularly.
- The environments to which software is made change regularly.
- Software is primarily intangible; much of the process of creating software is also intangible, involving experience, thought and imagination (a brain product).
- Because of intangibility it is difficult to specify software.
- Customers and developers have communication gaps.
- It is difficult to test software exhaustively.

# Software Projects vs. Other Projects

- Not like other engineering disciplines that have narrow scope, software can be developed in diverse contexts ranging from embedded systems to giant manufacturing systems.
- The replication of software is trivial and as a consequence software is elastic and gradually becomes more complex as changes are made over time.
- A small change to a few lines of code could do a big change in the overall behavior of the system.

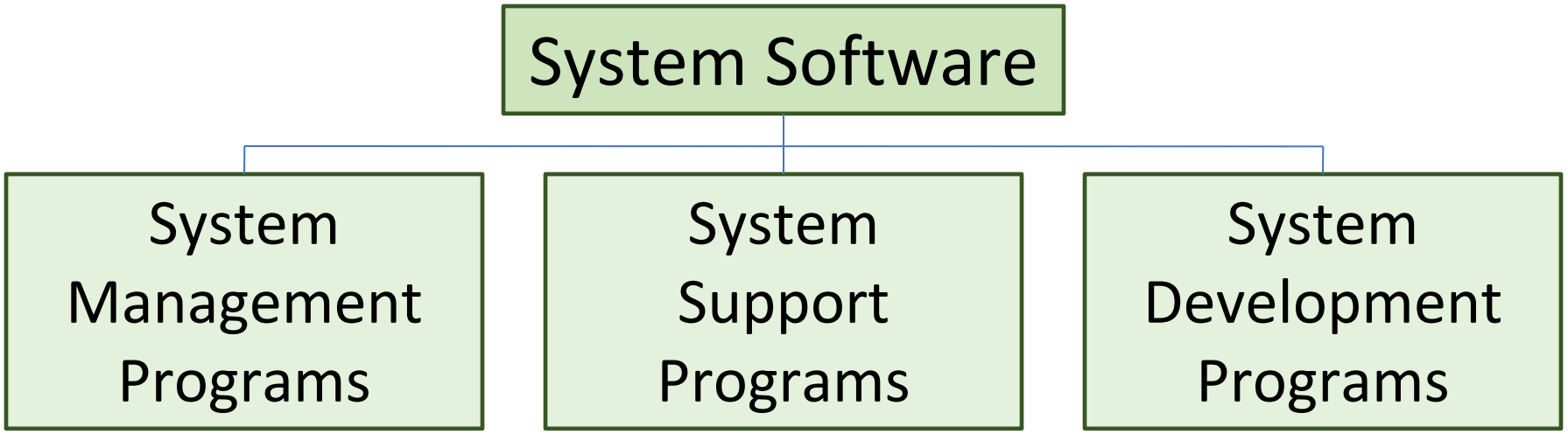
# Main Types of Software

- There are two main types of software;

1. System Software
   - computer software designed to operate and control the computer hardware and to provide a platform for running application software

2. Application Software
   - set of one or more programs designed to carry out operations for a specific application

# System Software Types

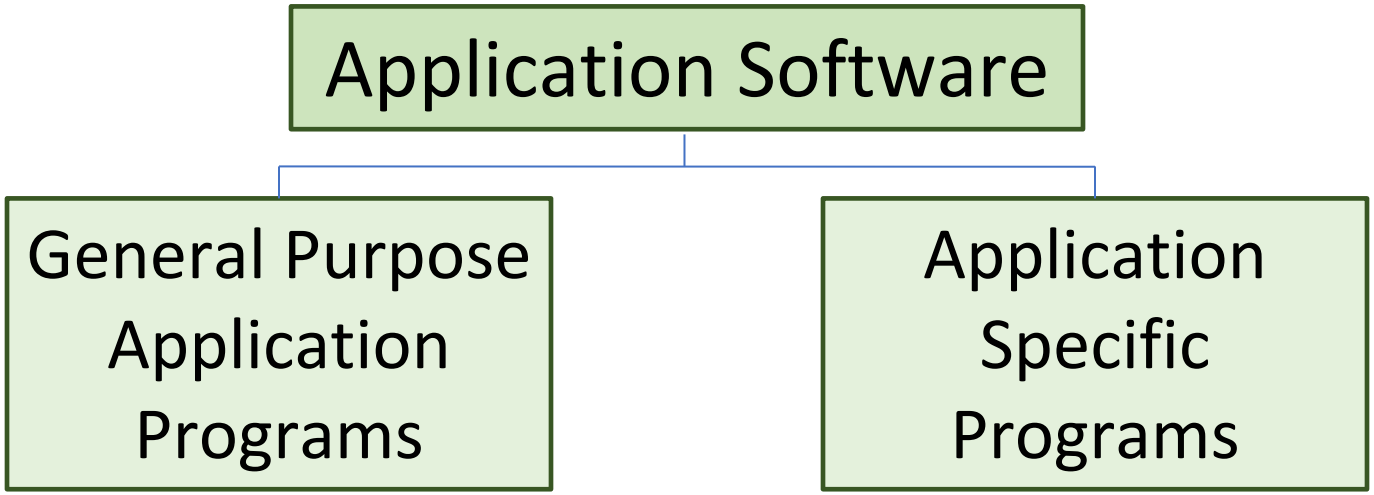



- Operating Systems
- Operating Environments
- Database Management Systems

- System Utilities
- Performance Monitors
- Security Monitors

- Programming Language Translators
- Programming Environments
- Computer Aided Software Engineering (CASE) Packages

# Application Software Types




- Word Processing
- Electronic Spread Sheets
- Database Managers
- Graphics Software
- Integrated Packages

- Accounting, General Legers etc.
- Marketing-Sales Analysis etc.
- Manufacturing-Production Control etc.
- Finance-Capital Budgeting etc.

# Application Software Types Cont...

- **Stand-alone applications**
  - These are application systems that run on a local computer, such as a PC. They include all necessary functionality and do not need to be connected to a network.
- **Interactive transaction-based applications**
  - Applications that execute on a remote computer and are accessed by users from their own PCs or terminals. These include web applications such as e-commerce applications.
- **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.
- **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.

# Application Software Types Cont...

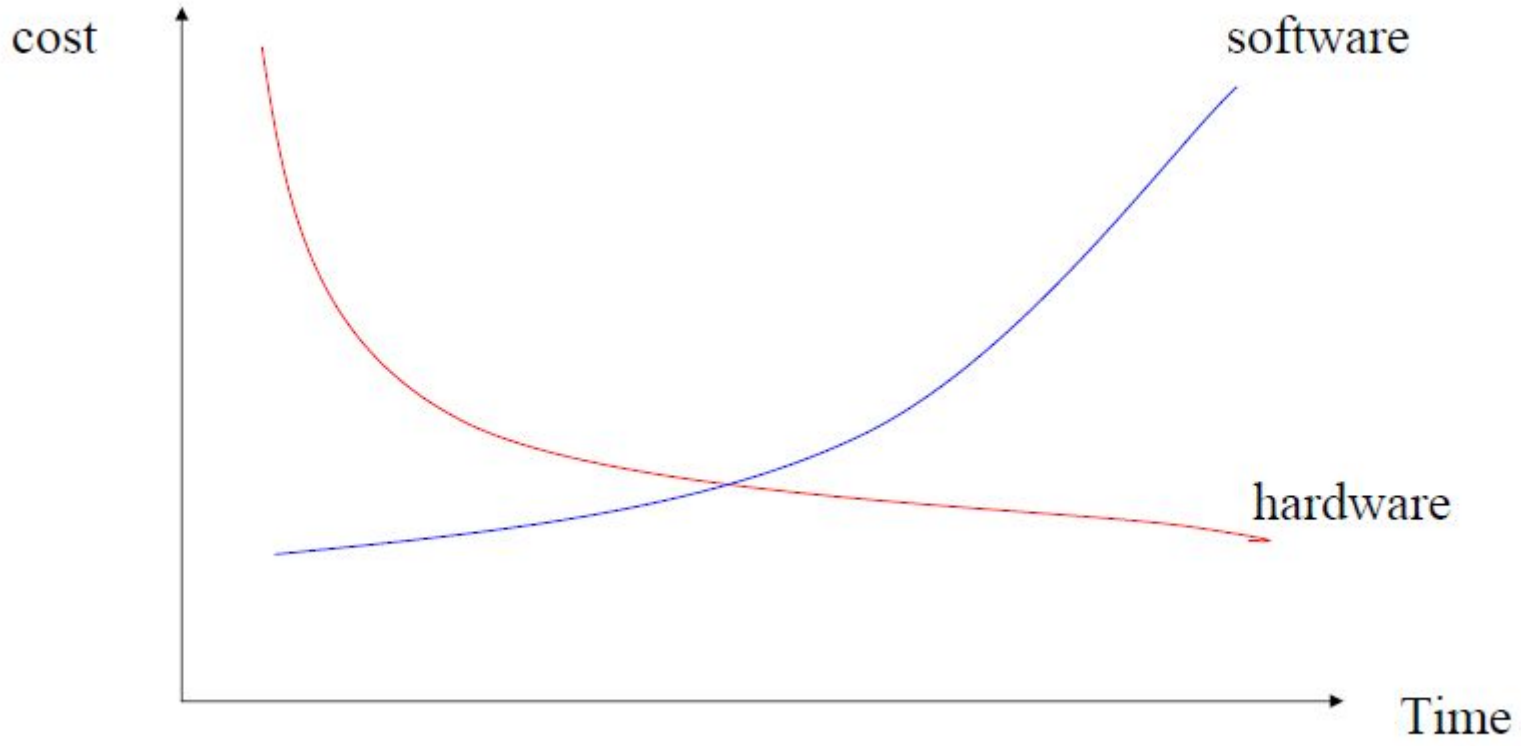
- **Entertainment systems**
  - These are systems that are primarily for personal use and which are intended to entertain the user.
- **Systems for modelling and simulation**
  - These are systems that are developed by scientists and engineers to model physical processes or situations, which include many, separate, interacting objects.
- **Data collection systems**
  - These are systems that collect data from their environment using a set of sensors and send that data to other systems for processing.
- **Systems of systems**
  - These are systems that are composed of a number of other software systems.

# Software Costs

- Software costs often dominate computer system costs. The costs of software on a PC are often greater than the hardware cost.
- Software costs more to maintain than it does to develop. For systems with a long life, maintenance costs may be several times development costs.
- Software engineering is concerned with cost-effective software development.

# Cost of Hardware vs. Software

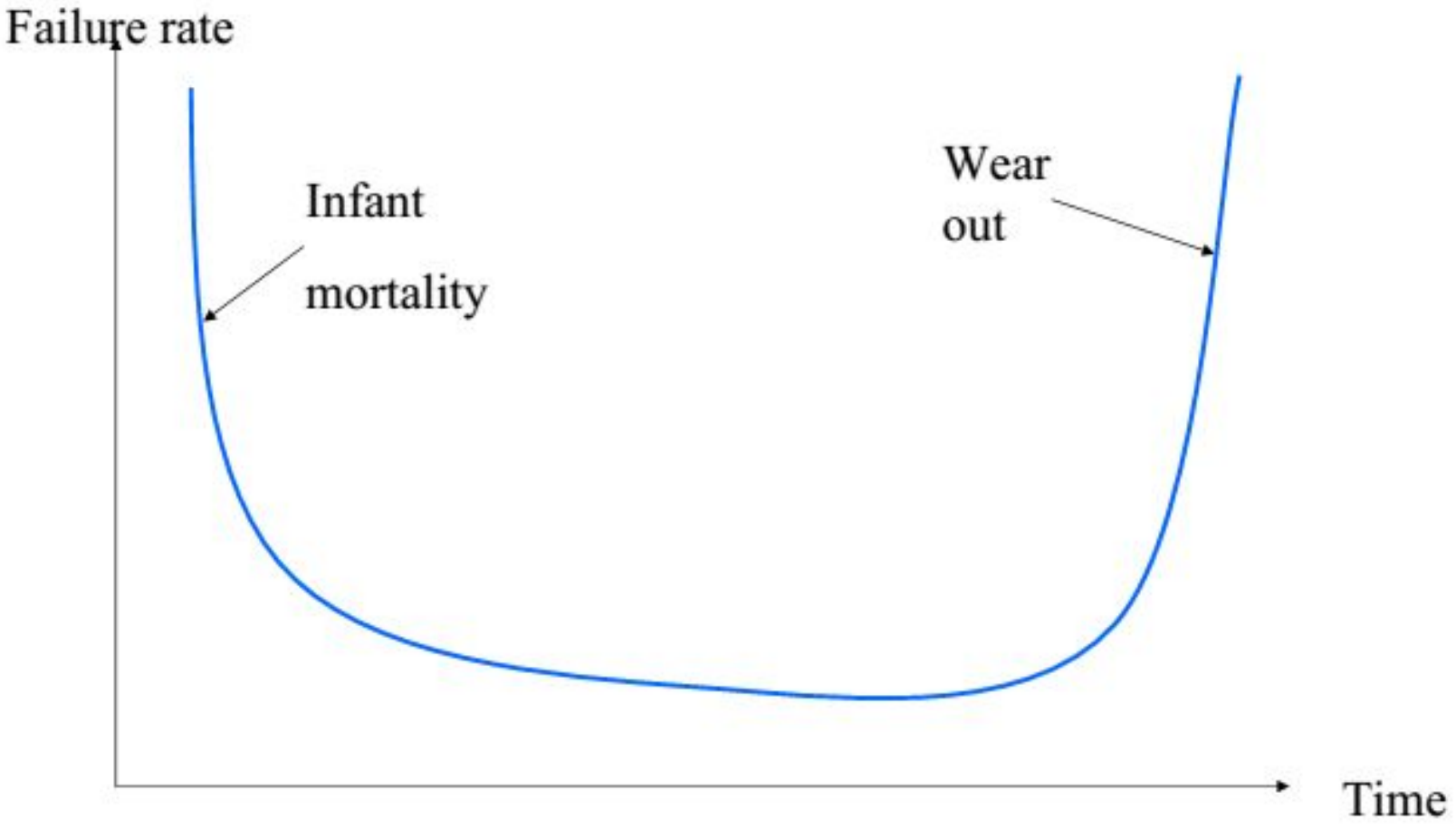
- As per the graph shown below, the software development cost increased rapidly over the time while the hardware becomes cheaper and cheaper with the development of the technology.



# Failure Curves for Software & Hardware

- Software is engineered or developed, not manufactured in the traditional sense.
- Software does not wear out in the same sense as hardware.
- So, the failing rate of a hardware is high at the initial stage.
- The failure rate decreases with the time and the hardware become more stable since the quality of the hardware is improved.
- However, failure rate again goes up in a particular hardware, when it comes to the Wear out stage.
- So, the failure curve for the Hardware looks like a “bathtub” as shown in the next slide.

# Failure “Bathtub” Curve for Hardware



# Failure Curve for Software

- Software projects can be failed due to two main reasons.

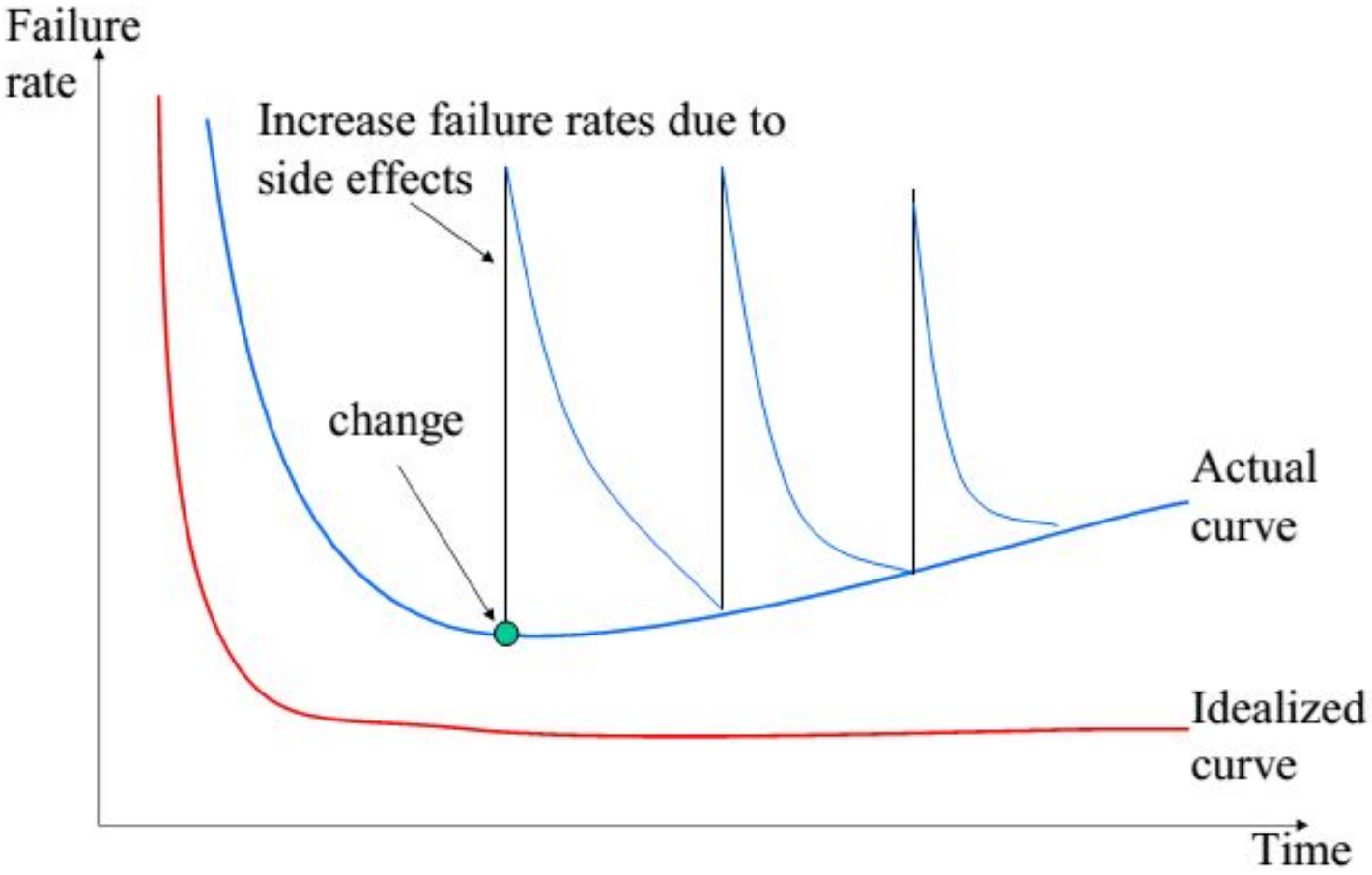
1. Increasing system complexity
   - As new software engineering techniques help us to build larger, more complex systems, the demands change. Systems have to be built and delivered more quickly; larger, even more complex systems are required; systems have to have new capabilities that were previously thought to be impossible.

2. Failure to use software engineering methods
   - It is fairly easy to write computer programs without using software engineering methods and techniques. Many companies have drifted into software development as their products and services have evolved. They do not use software engineering methods in their everyday work. Consequently, their software is often more expensive and less reliable than it should be.

# Failure Curve for Software Cont...

- Failure rate of a software also is high, like in hardware, because there can be un-detected bugs/errors in the system.
- However, software becomes stable once it improves with the time.
- Unlike hardware, software do NOT wear out with the time, so it does NOT increase the failure rate of a particular software, unless it become unstable by adding new features continuously.
- Failure rate of  software can go high, when a new change introduce to the software.
- Next slide shows the idealized (expected) curve and the actual curve of a software.

# Failure Curve for Software Cont...



# 1.1: Professional Software Development

IT2206 - Fundamentals of Software Engineering

Level I - Semester 2

# Software Development

- Software development is the computer programming, documenting, testing, and bug fixing resulting in a software product.
- **A software development methodology** (process, model, or life cycle) is a framework that is used to structure, plan, and control the process of developing software systems.
- Software development process is considered to be a life cycle of a software product.

# Problems in Software Development

- Large software is usually designed to solve 'wicked' problems
- Software engineering requires a great deal of coordination across disciplines
  - Almost infinite possibilities for design trade-offs across components
  - Mutual distrust and lack of understanding across engineering disciplines
- Systems must be designed to last many years in a changing environment.
- The process of efficiently and effectively developing requirements.
- Tooling required to create the solutions, may change as quick as the clients mind.

# Problems in Software Development Cont...

- User expectations:
  - User expectations increase as the technology becomes more and more sophisticated
- The mythical man-month factor:
  - Adding personnel to a project may not increase productivity
  - Adding personnel to a late project will just make it later
- Communications:
  - Communications among the various constituencies is a difficult problem. Sometimes different constituencies speak completely different languages. For example, developers may not have the domain knowledge of clients and / or users. The larger the project, the more difficult the communications problems become.

# Why do we need Professional Software Development?

- Software development is different from other types of development products due to following reasons;
  1. It is difficult for a customer to specify requirements completely
  2. It is difficult for the developer to understand fully the customer needs
  3. Software requirements change regularly
  4. Software is primarily intangible; much of the process of creating software is also intangible, involving experience, thought and imagination
  5. It is difficult to test software exhaustively
- So, we need a proper approach to develop a good quality software.

# Key Features 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.

# Key Features of a Good Software Cont...

- 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.

# So, how to make a good quality software?

The main concern is, with the nature a software, how do we develop a quality software including all the key features mentioned above?

You can imagine that we can NOT develop a software simply by knowing some programming languages.

You should follow some process and proper approach in order to develop a good quality software.

# The Solution: Software 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.

**Engineering discipline**

- Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.

**All aspects of software production**

- Not just technical process of development. Also project management and the development of tools, methods etc. to support software production.

# The Solution: Software Engineering Cont...

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.

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 system, the majority of costs are the costs of changing the software after it has gone into use.

# Software Engineering: Definitions

Simple Definition:

*Designing, building and maintaining large software systems*

A generic definition:

*Use of systematic, engineering approach in all stages of software development and project management to develop high quality and economical software using appropriate software tools*

# Software Engineering: Definitions

Software engineering is concerned with the theories, methods and tools for developing, managing and evolving software products'

- I Sommerville

'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'

- B.W.Boehm

# Software Engineering: Definitions

'The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines'

- F.L. Bauer

'The application of systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software'

- IEEE Standard 610.12

# 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. 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.

# Key Activities of Software Process

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

# Key Challenges facing Software Engineering

**Heterogeneity**

- Increasingly, systems are required to operate as distributed systems across networks that include different types of computer and mobile devices.

**Business and social change**

- Business and society are changing incredibly quickly as emerging economies develop and new technologies become available. They need to be able to change their existing software and to rapidly develop new software.

**Scale**

- Software has to be developed across a very wide range of scales, from very small embedded systems in portable or wearable devices through to Internet-scale, cloud-based systems that serve a global community.

**Security and trust**

- As software is intertwined with all aspects of our lives, it is essential that we can trust that software.

# Software Engineering vs. 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 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.

# Software Engineering Diversity

There are many different types of software system and there is no universal set of software techniques that is applicable to all of these.

The software engineering methods and tools used depend on the type of application being developed, the requirements of the customer and the background of the development team.

# Web-based Software Engineering

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.

# Web-based Software Engineering Cont...

Though the web-based systems are complex distributed systems, the fundamental principles of software engineering discussed previously are as applicable to them as they are to any other types of system.

The fundamental ideas of software engineering apply to web-based software in the same way that they apply to other types of software system.

# Features of Web-based Software Engineering

Software reuse

- Software reuse is the dominant approach for constructing web-based systems. When building these systems, you think about how you can assemble them from pre-existing software components and systems.

Incremental and agile development

- Web-based systems should be developed and delivered incrementally. It is now generally recognized that it is impractical to specify all the requirements for such systems in advance.

# Features of Web-based Software Engineering Cont...

Service-oriented systems

- Software may be implemented using service-oriented software engineering, where the software components are stand-alone web services.

Rich interfaces

- Interface development technologies such as AJAX and HTML5 have emerged that support the creation of rich interfaces within a web browser.

# 1.2: Software engineering ethics

IT2206 - Fundamentals of Software Engineering

Level I - Semester 2

# Why Ethics in Software Engineering?

- 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 behaviour is more than simply upholding the law but involves following a set of principles that are morally correct.

# Issues of Professional Responsibility

- **Confidentiality**
  - Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
- **Competence**
  - Engineers should not misrepresent their level of competence. They should not knowingly accept work which is out with their competence.
- **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, say) to extremely serious (dissemination of viruses).

# ACM/IEEE Code of Ethics

- The professional societies in the US have cooperated to produce a code of ethical practice.
- Members of these organisations sign up to the code of practice when they join.
- The Code contains eight Principles related to the behaviour 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.

# Rationale for the Code of Ethics

- Computers have a central and growing role in commerce, industry, government, medicine, education, entertainment and society at large. Software engineers are those who contribute by direct participation or by teaching, to the analysis, specification, design, development, certification, maintenance and testing of software systems.
- Because of their roles in developing software systems, software engineers have significant opportunities to do good or cause harm, to enable others to do good or cause harm, or to influence others to do good or cause harm. To ensure, as much as possible, that their efforts will be used for good, software engineers must commit themselves to making software engineering a beneficial and respected profession.

# The ACM/IEEE Code of Ethics

**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:

# Ethical 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

- Disagreement in principle with the policies of senior management.
- 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.

# Activity

- Identify the differences between Software development projects and other type of development projects.
- Identify the differences between generic software product development and custom software development?
- Briefly discuss why it is usually cheaper in the long run to use software engineering methods and techniques for software systems.

# Summary

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Essential software product attributes are maintainability, dependability and security, efficiency and acceptability.
- The high-level activities of specification, development, validation and evolution are part of all software processes.
- The fundamental notions of software engineering are universally applicable to all types of system development.
- There are many different types of system and each requires appropriate software engineering tools and techniques for their development.

# Summary

- The fundamental ideas of software engineering are applicable to all types of software system.
- Software engineers have responsibilities to the engineering profession and society. They should not simply be concerned with technical issues.
- Professional societies publish codes of conduct which set out the standards of behaviour expected of their members.