LECTURE ONE

COURSE TITLE: SOFTWARE ENGINEERING

COURSE CODE: COMP 306

LECTURER: REGINA CRABBE

PROGRAMME: BSC. COMPUTER SCIENCE

LEVEL: 300

INTRODUCTION TO SOFTWARE ENGINEERING

SOFTWARE ENGINEERING

- Computer software
- Importance of software
- Software engineering
- Software engineers
- Software processes
- Issues that affect different types of software
- Software engineering diversity
- Software engineering ethics

SOFTWARE ENGINEERING

What is it? Computer software (1) instructions (computer programs) that when executed provide desired features, function, and performance; (2) data structures that enable the programs to adequately manipulate information, and (3) descriptive information in both hard copy and virtual forms that describes the operation and use of the programs.

Software engineering encompasses a process, a collection of methods (practice) and an array of tools that allow professionals to build high quality computer software.

SOFTWARE ENGINEERING

- **Who does it?** **Software engineers** build and support software, and virtually everyone in the industrialized world uses it either directly or indirectly.

Why is it important? Software is important because it affects nearly every aspect of our lives and has become pervasive in our commerce, our culture, and our everyday activities. Software engineering is important because it enables us to build complex systems in a timely manner and with high quality.

What is the work product? From the point of view of a software engineer, the work product is the set of programs, content (data), and other work products that are computer software. But from the user's viewpoint, the work product is the resultant information that somehow makes the user's world better.

SOFTWARE ENGINEERING

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:**

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.

SOFTWARE ENGINEERING

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.

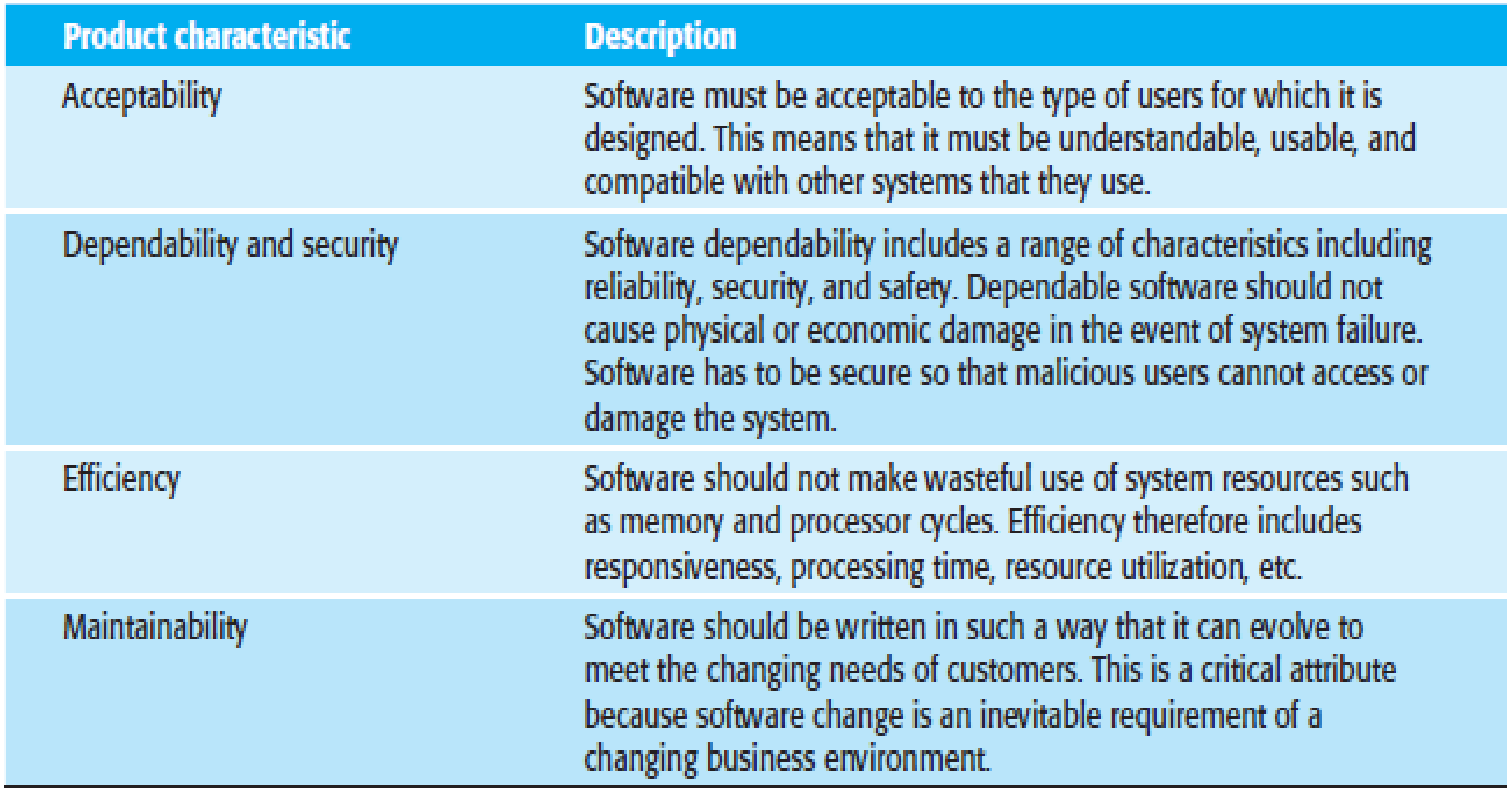
SOFTWARE ENGINEERING

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. 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.

SOFTWARE ENGINEERING

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.



Product characteristic	Description
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

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 a personal programming project. Failure to use software engineering method leads to higher costs for testing, quality assurance, and long-term maintenance.

SOFTWARE PROCESS

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.

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.

SOFTWARE ENGINEERING

There are many different types of software. There are no universal software engineering methods or techniques that may be used. However, there are four related issues that affect many different types of software:

Heterogeneity: Increasingly, systems are required to operate as distributed systems across networks that include different types of computer and mobile devices. As well as running on general-purpose computers, software may also have to execute on mobile phones and tablets. You often have to integrate new software with older legacy systems written in different programming languages. The challenge here is to develop techniques for building dependable software that is flexible enough to cope with this heterogeneity.

SOFTWARE ENGINEERING

Business and social change: Businesses 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. Many traditional software engineering techniques are time consuming, and delivery of new systems often takes longer than planned. They need to evolve so that the time required for software to deliver value to its customers is reduced.

Security and trust : As software is intertwined with all aspects of our lives, it is essential that we can trust that software. This is especially true for remote software systems accessed through a web page or web service interface. We have to make sure that malicious users cannot successfully attack our software and that information security is maintained.

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.

SOFTWARE ENGINEERING

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.

  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:**

TYPES OF APPLICATION

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(Computer-aided design) programs, photo manipulation software, travel apps, productivity apps, and so on.

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. This class of application also includes business systems, where a business provides access to its systems through a web browser or special-purpose client program and cloud-based services, such as mail and photo sharing. Interactive applications often incorporate a large data store that is accessed and updated in each transaction.

TYPES OF APPLICATION

- ***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 in a car, and software in a microwave oven to control the cooking process.

  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.

  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.

TYPES OF APPLICATION

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.

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.

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.

SOFTWARE ENGINEERING ETHICS

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 behaviour are not bound by laws but by the more tenuous notion of professional responsibility.

SOFTWARE ENGINEERING ETHICS

Some of these are:

- ***Confidentiality*** You should normally respect the confidentiality of your employers or clients regardless 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) to extremely serious (dissemination of viruses or other malware).