ICS 3105
OBJECT ORIENTED SOFTWARE ENGINEERING

CHAPTER 1

Introduction to Software Engineering

Chapter Objectives

- Understand what software engineering is and why it is important.
- Know the answers to key questions that provide an introduction to software engineering.
- Understand some ethical and professional issues that are important to software engineers.

Introduction to Software

- Virtually all countries now depend on complex computer-based systems.
- National infrastructures and utilities rely on computer-based systems and most electrical products include a computer and controlling software.

Introduction to software

- Industrial manufacturing and distribution is completely computerised, as is the financial system.
- Therefore, producing and maintaining software cost-effectively is essential for the functioning of national and international economies.

Software Process

- A software process is a framework for the tasks that are required to build high-quality software.
- A software process defines the approach that is taken as software is engineered.

Software Process

- But software engineering also encompasses technologies that populate the process—technical methods and automated tools.

Software Process

- There are many different types of software systems, from simple embedded systems to complex, worldwide information systems.
- It is pointless to look for universal notations, methods, or techniques for software engineering because different types of software require different approaches (Sommerville, 2011).

Software Process

- Developing an organizational information system is completely different from developing a controller for a scientific instrument.
- Neither of these systems has much in common with a graphics-intensive computer game.

Software Process

- All of these applications need software engineering; they do not all need the same software engineering techniques

Software Engineering

- There are still many reports of software projects going wrong and ‘software failures’.
- Software engineering is criticized as inadequate for modern software development.

Software Engineering

- Many of these software failures are a consequence of two factors (Sommerville, 2011):
  - Increasing demands.
  - Low expectations.

Increasing Demands

- 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.

Increasing Demands

- Existing software engineering methods cannot cope and new software engineering techniques have to be developed to meet new these new demands.

Low Expectations

- It is relatively 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.

Low Expectations

- 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.
- We need better software engineering education and training to address this problem.

Professional Software Development

- Tools and standard notations were developed and are now extensively used.
  - People in business write spreadsheet programs to simplify their jobs,
  - Scientists and engineers write programs to process their experimental data, and
  - Hobbyists write programs for their own interest and enjoyment.

Professional Software Development

- However, the vast majority of software development is a professional activity where software is developed for specific business purposes, for inclusion in other devices, or as software products such as information systems, CAD systems, etc.

Professional Software Development

- 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.

Professional Software Development

- Software engineering is intended to support professional software development, rather than individual programming.
- Software engineering includes techniques that support program specification, design, and evolution, none of which are normally relevant for personal software development.

Professional 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 and configuration data that is required to make these programs operate correctly.

Professional Software Development

- Associated documentation 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.

What is Software Engineering?

1. Software engineering is an engineering discipline whose focus is the cost effective development of high-quality software systems.
   - It is not constrained by materials, or governed by physical laws or by manufacturing processes.

Software Engineering

- In some ways, this simplifies software engineering as there are no physical limitations on the potential of software.
- However, this lack of natural constraints means that software can easily become extremely complex and hence very difficult to understand.

What is Software Engineering?

2. 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 (Sommerville, 2011).

What is Software Engineering?

3. The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines (Pressman, 2001).

What is Software Engineering?

4. The profession, practiced by developers, concerned with creating and maintaining software applications by applying technologies and practices from computer science, project management, and other fields.

Engineering Discipline

- Engineers make things work.
- They apply theories, methods and tools where these are appropriate, but they use them selectively and always try to discover solutions to problems even when there are no applicable theories and methods.

Engineering discipline

- Engineers also recognise that they must work to organizational and financial constraints, so they 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 production.

Software Engineering

- Generally, software engineers adopt a systematic and organized approach to their work, as this is often the most effective way to produce high-quality software.
- However, engineering is all about selecting the most appropriate method for a set of circumstances and a more creative, less formal approach to development may be effective in some circumstances.

Software engineering

- Less formal development is particularly appropriate for the development of web-based systems, which requires a blend of software and graphical design skills.

History of Software Engineering

- The notion of software engineering was first proposed in 1968 at a conference held to discuss what was then called the "software crisis".
- This software crisis resulted directly from the introduction of new computer hardware based on integrated circuits.

History of Software Engineering

- The notion of ‘software engineering’ was first proposed in 1968 at a conference held to discuss what was then called the ‘software crisis’ (Naur and Randell, 1969).
- It became clear that individual approaches to program development did not scale up to large and complex software systems.

History of Software Engineering

- These were unreliable, cost more than expected, and were delivered late.
- Throughout the 1970s and 1980s, a variety of new software engineering techniques and methods were developed, such as structured programming, information hiding and object-oriented development.

History of Software Engineering

- Their power made hitherto unrealisable computer applications a feasible proposition.
- The resulting software was orders of magnitude larger and more complex than previous software systems.

History of Software Engineering

- Early experience in building these systems showed that informal software development was not good enough.

History of Software Engineering

- Major projects were:
  - Sometimes years late.
  - Software cost much more than predicted.
  - Software was unreliable.
  - Software was difficult to maintain.
  - Software poorly performed.

History of Software Engineering

- Software development was in crisis.
- Hardware costs were tumbling whilst software costs were rising rapidly.
- New techniques and methods were needed to control the complexity inherent in large software systems.

History of Software Engineering

- These techniques have become part of software engineering and are now widely used.
- However, as our ability to produce software has increased, so too has the complexity of the software systems that we need.

History of Software Engineering

- New technologies resulting from the interfaces place new demands on software engineers.
- As many companies still do not apply software engineering techniques effectively, too many projects still produce software that is unreliable, delivered late and over budget.

Software System

- A software system usually consists of:
  - A number of separate programs.
  - Configuration files, which are used to set up these programs.
  - System documentation, which describes the structure of the system, and
  - User documentation, which explains how to use the system and web sites for users to download recent product information.

Types of Software Product

- There are 2 fundamental types:
  - Generic products.
  - Customised (or bespoke) products.

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.
- E.g. databases, word processors, drawing packages and project management tools.

Customised (or Bespoke) Products

- These are products which are commissioned by a particular customer.
- A software contractor develops the software especially for that customer.
- E.g. control systems for electronic devices, systems written to support a particular business process and air traffic control systems.

Customised Products Vs Generic

- An important difference between these types of software is that, in generic products, the organization that develops the software controls the software specification.
- For custom products, the specification is usually developed and controlled by the organization that is buying the software.

Customised Products Vs Generic

- The software developers must work to that specification.
- However, the line between these types of products is becoming increasingly blurred.
- More and more software companies are starting with a generic system and customizing it to the needs of a particular customer.

Customized Products Vs Generic

- Enterprise Resource Planning (ERP) systems, such as the SAP system, are the best examples of this approach.
- Here, a large and complex system is adapted for a company by incorporating information about business rules and processes, reports, e.t.c

Two reasons why Software Engineering is important

- 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.

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.

Software engineering ethics

- 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.

Software engineering ethics

- 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.

Software engineering ethics

- Confidentiality
- Competence
- Intellectual property rights
- Computer misuse.

Software engineering ethics: Confidentiality

- You should normally respect the confidentiality of your employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.

Software engineering ethics: Competence

- Competence You should not misrepresent your level of competence.
- You should not knowingly accept work that is outside your competence.

Software engineering ethics: 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.

Software engineering ethics: Computer misuse

- 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, say) to extremely serious(dissemination of viruses or other malware).

Essential attributes of good software

- Maintainability
- Dependability and security.
- Efficiency.
- Acceptability.

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.

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.

End of Chapter 1