INTRODUCTION

SOFTWARE ENGINEERING 1

SOON PHEI TIN

OBJECTIVES

- understand what software engineering is and why it is important;
- understand that the development of different types of software systems may require different software engineering techniques;
- understand some ethical and professional issues that are important for software engineers;

INTRODUCTION

- What is a computer software?
  - Computer software is the product that software professionals build and then support over the long term. It encompasses computer programs and associated documentation that execute within a computer of any size and architecture . Software products may be developed for a particular customer or may be developed for a general market.

INTRODUCTION

- 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.
- What is software engineering?
  - Software engineering is an engineering discipline that is concerned with all aspects of software production.

INTRODUCTION

- We can't run the modern world without software.
  - National infrastructures and utilities are controlled by computer-based systems and most electrical products include a computer and controlling software.
  - Industrial manufacturing and distribution is completely computerized, as is the financial system.
  - Entertainment, including the music industry, computer games, and film and television, is software intensive.

INTRODUCTION

- Software systems are abstract and intangible.
  - They are not constrained by the properties of materials, governed by physical laws, or by manufacturing processes.
- Implication?
  - This simplifies software engineering, as there are no natural limits to the potential of software.
  - However, because of the lack of physical constraints, software systems can quickly become extremely complex, difficult to understand, and expensive to change.

SOFTWARE FAILURE & SUCCESS

- Software engineering is criticized as inadequate for modern software development.
- Software failures are a consequence of two factors:
  - Increasing demands
  - Low expectations

SOFTWARE FAILURE & SUCCESS

- Of course, we still have problems developing complex software but, without software engineering, we would not have explored space, would not have the Internet or modern telecommunications.
- Software engineering has contributed a great deal and its contributions in the 21st century will be even greater.

PROFESSIONAL SOFTWARE DEVELOPMENT

- Amateur
  - People in business write spreadsheet programs to simplify their jobs
  - scientists and engineers write programs to process their experimental data
  - hobbyists write programs for their own interest and enjoyment
- Professional
  - developed for specific business purposes, for inclusion in other devices, or as software products such as
  - 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.
- It includes techniques that support program specification, design, validation, and evolution.
- Professional software usually has the following properties: -
  - Strict user requirements
  - Required accuracy and data integrity
  - Higher security standard
  - Stable performance for heavy load
  - Required technical support, etc.

PROFESSIONAL SOFTWARE DEVELOPMENT

Product characteristics	Description
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.

SOFTWARE DEVELOPMENT CHALLENGES

- Why does it take so long to get software finished?
- Why are development costs so high?
- Why can't we find all errors before we give the software to our customers?
- Why do we spend so much time and effort maintaining existing programs?
- Why do we continue to have difficulty in measuring progress as software is being developed and maintained?

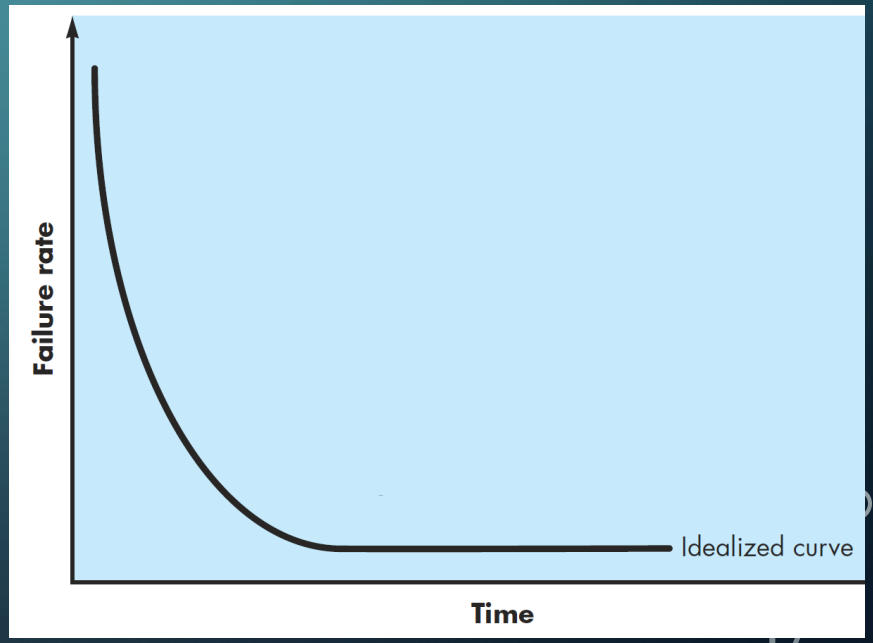
TWO KINDS OF SOFTWARE PRODUCTS

- Generic software products
  - These are systems that are produced by a development organization and sold on the open market to any customer who is able to buy them.
- Customized software products
  - These are systems that are commissioned by a particular customer. A software contractor develops the software especially for that customer.
- The distinction between these system product types is becoming increasingly blurred

Discuss the differences between a generic software product and a customized software product

SOFTWARE DETERIORATION

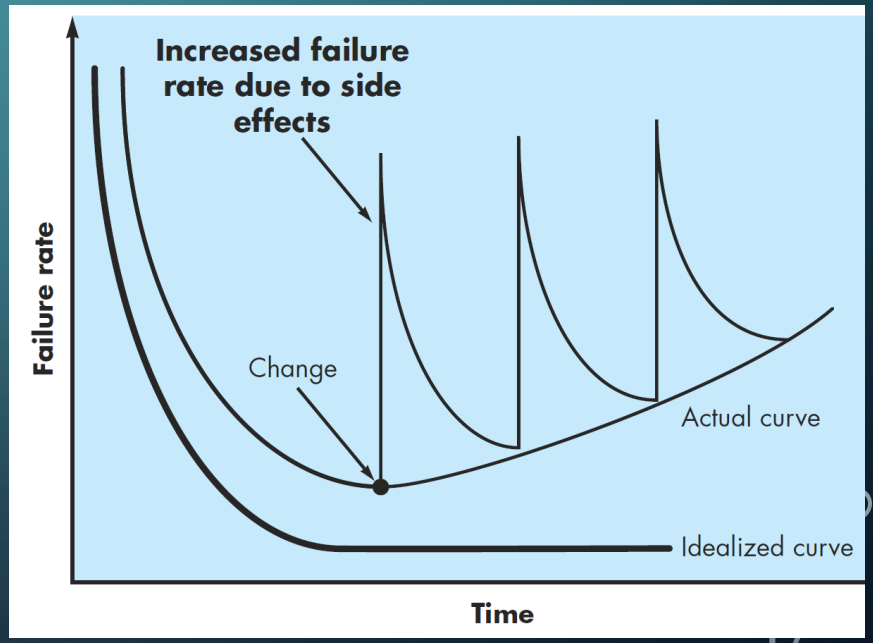
- Software is a logical rather than a physical system element. Therefore, software has one fundamental characteristic that makes it considerably different from hardware: *Software doesn't* ***"wear out."***
- In theory, therefore, the failure rate curve for software should take the form of the "idealized curve" shown in Figure



SOFTWARE DETERIORATION

- Software doesn't wear out. But it does ***deteriorate***
- During its life, software will undergo change. As changes are made, it is likely that errors will be introduced, causing the failure rate curve to spike as shown in the "actual curve" (Figure 1.2).
- Before the curve can return to the original steady-state failure rate, another change is requested, causing the curve to spike again.

Software engineering methods strive to reduce the magnitude of the spikes and the slope of the actual curve



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.

SOFTWARE ENGINEERING

- 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 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

- Why software engineering 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.
  - 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 systems, the majority of costs are the costs of changing the software after it has gone into use.

SOFTWARE ENGINEERING APPROACHES

- 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.
- There are four fundamental activities that are common to all software processes: -
  - Software specification
  - Software development
  - Software validation
  - Software evolution

GENERAL ISSUES THAT AFFECT MOST SOFTWARE

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

SOFTWARE ENGINEERING DIVERSITY

- Different types of systems need different development processes.
- For example, real-time software in an aircraft has to be completely specified before development begins. In e-commerce systems, the specification and the program are usually developed together.

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.
- How this systematic approach is actually implemented varies dramatically depending on the organization developing the software, the type of software, and the people involved in the development process
- 2 major approaches: -
  - Sequential
  - Iterative

APPLICATION TYPES

- 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.

APPLICATION TYPES

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

APPLICATION TYPES

- 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 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.

WEB SOFTWARE ENGINEERING

- 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.
- 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.
- User interfaces are constrained by the capabilities of web browsers.
  - Technologies such as AJAX allow rich interfaces to be created within a web browser but are still difficult to use. Web forms with local scripting are more commonly used.

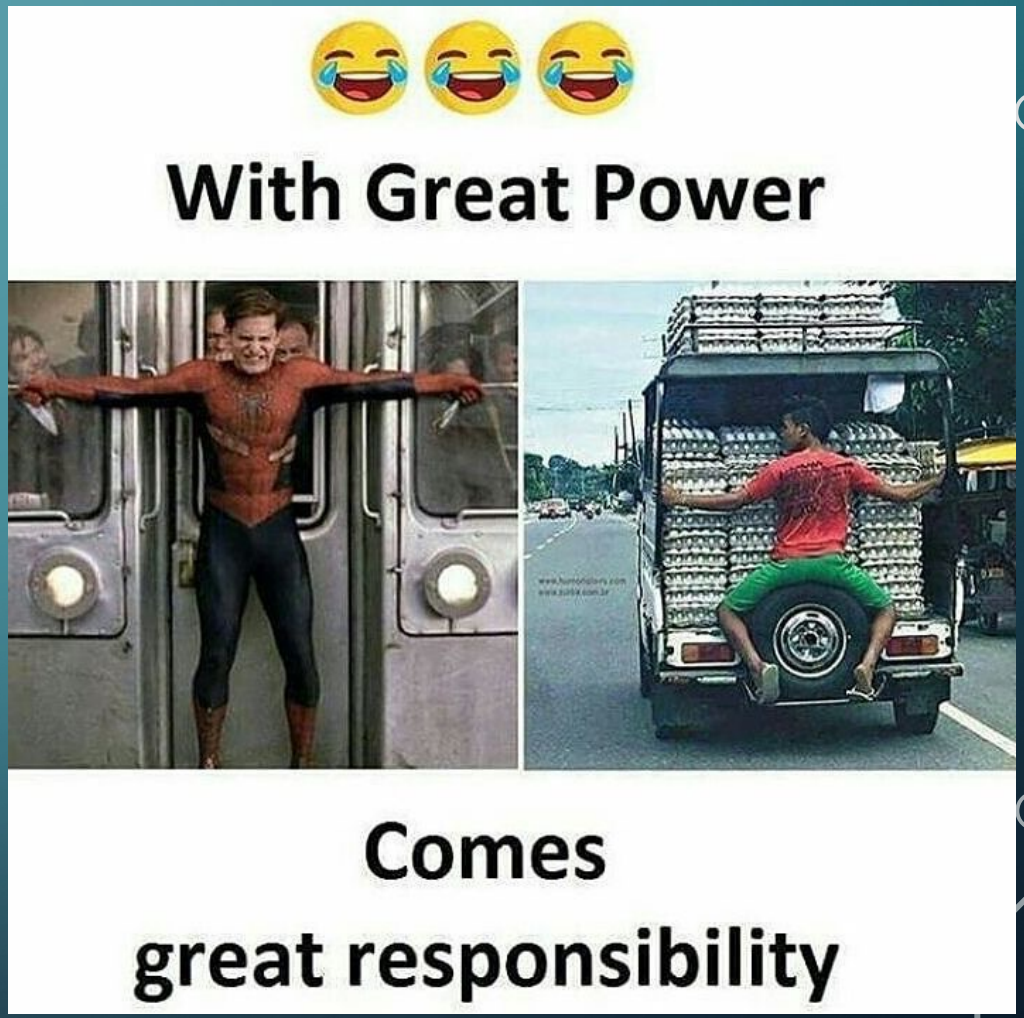
WEB-BASED SOFTWARE ENGINEERING

- Web-based systems are complex distributed systems but 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, discussed in the previous section, apply to web-based software in the same way that they apply to other types of software system.

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.


"With great power comes great responsibility"
~Voltaire
IGN.COM
Compiled by NetSysCon find more inspiring quotes at
http://netsyscon4hr.wordpress.com/category/quotes-by-the-greats/
NSC
NetSysCon


With Great Power
Comes
great responsibility

SOFTWARE ENGINEERING ETHICS

- You must also behave in an ethical and morally responsible way if you are to be respected as a professional engineer.
- 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.

SOFTWARE ENGINEERING ETHICS

- There are areas where standards of acceptable behavior are not bound by laws but by the more tenuous notion of professional responsibility. Some of these are:
  - Confidentiality
  - Competence
  - Intellectual property rights
  - Computer misuse

SOFTWARE DEVELOPMENT RISK

- Many software development projects run into difficulties
  - Does not work as expected
  - Over budget
  - Late delivery
- Much of the functionalities in the software is wasted
  - Wrong requirement
  - User dislike it
  - There are no customer
  - etc

SOFTWARE DEVELOPMENT RISK

- Many software projects fail because the software developers build the wrong software. The software development team must: -
  - Fully understand requirement
  - Validate requirement
- The developer will often add technical insights and suggestions, but remember the client satisfaction is the primary measurement of success in software project.

SOFTWARE DEVELOPMENT RISK

- Failures of software development projects can bankrupt companies.
- What are the consequences if the development of a software: -
  - Late?
  - Over budget?
  - Does not work or full of bugs?

HOW TO MINIMIZE RISK?

- Communication with the client
  - Feasibility studies
  - Separation of requirement from design
  - Milestones and releases
  - Acceptance and user testing