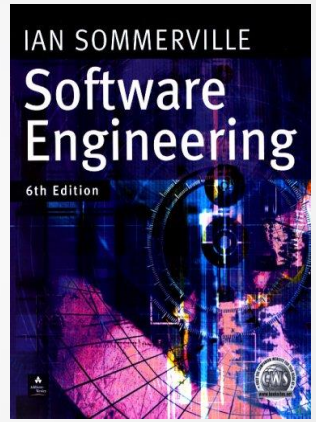
Software Engineering
Comp 433
Chapter 1: Introduction

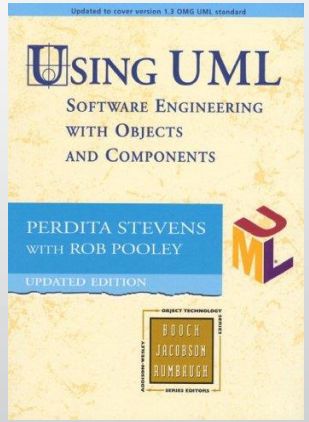
Computer Science Dept, Birzeit University, Samer Zein (Ph.D). Updated by Saad Mansour, 2024

- No phones and No Eating in Class
- Raise your hand before answering or asking

Recommended Course Textbooks

- Sommerville I. (2010)
  Software Engineering 9th Edition, Addison-Wesley, Harlow, Essex,UK (6th ,7th,or 8th would suffice)
- Bruegge and Dutoit, ***Object-Oriented Software Engineering Using UML, Patterns, and Java***, Prentice Hall **3rd** Edition
- Stevens P. with Pooley, R. (2005)
  Using UML: Software Engineering with Objects and Components,
  2nd Ed., Addison-Wesley, Harlow, Essex, UK
- Jeffrey A. Hoffer, Joey F. George, Joseph S. Valacich. (2005)
  Modern System Analysis and Design 4th - 6th Edition, Prentice Hall.
- *Roger Pressman (2014),* ***Software Engineering: A Practitioner's Approach*** *6-8th Edition, McGraw-Hill.*





Objectives

When you have read this chapter, you will:

- 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;

Value of software

- We can’t run the modern world without software.
  - National infrastructure
  - Electric products.
  - Industrial manufacturing
  - Goods distribution.
  - Education, Banks
  - Organizations
  - Mobile apps, the new player!

- However, **building and maintaining <u>software is hard and getting harder</u>**

SE history

- SE introduced first in 1968 – conference about “software crisis” when the introduction of third generation computer hardware led to more complex software systems than before
- Early approaches based on informal methodologies led to:
  - Delays in software delivery
  - Higher costs than initially estimated
  - Unreliable, difficult to maintain software
- Need for new methods and techniques to manage the production of complex software.

Industry and Software Engineering

- **Increasing demands:** systems have to be built and delivered more quickly; larger, even more complex systems are required

- **Failure to use software engineering methods:** it is relatively easy to write computer programs without using software engineering methods and techniques.
  - Many companies do not use software engineering methods in their everyday work.
  - Consequently, their software is often more expensive and less reliable than it should be.

Software failures

- **Therac-25 (1985-1987)**: six people overexposed during treatments for cancer
- **Taurus (1993)**: the planned automatic transaction settlement system for London Stock Exchange cancelled after five years of development
- **Ariane 5 (1996)**: rocket exploded soon after its launch due error conversion (16 floating point into 16-bit integer)
- **The Mars Climate Orbiter** assumed to be lost by NASA officials (1999): different measurement systems (Imperial and metric)

More Software failures

- Passport System delays cause backlog (1999, UK)
- Ferry Company left thousands of lorries stranded for 12 hours (back up also failed, 1999, UK)
- Inland Revenue (IR) 'losing tax records' (2000, UK)

  => IR spokesman said 'All major IT initiatives have some kind of teething problems ....'

  => Guardian (20 July 2000) 'At the centre of the crisis are two computer systems .... Files appear to have gone missing somewhere between the two'
- General Motors Ford Cars (2016, USA + Worldwide): A "software bug" that may cause human safety, 4.5M cars recalled.

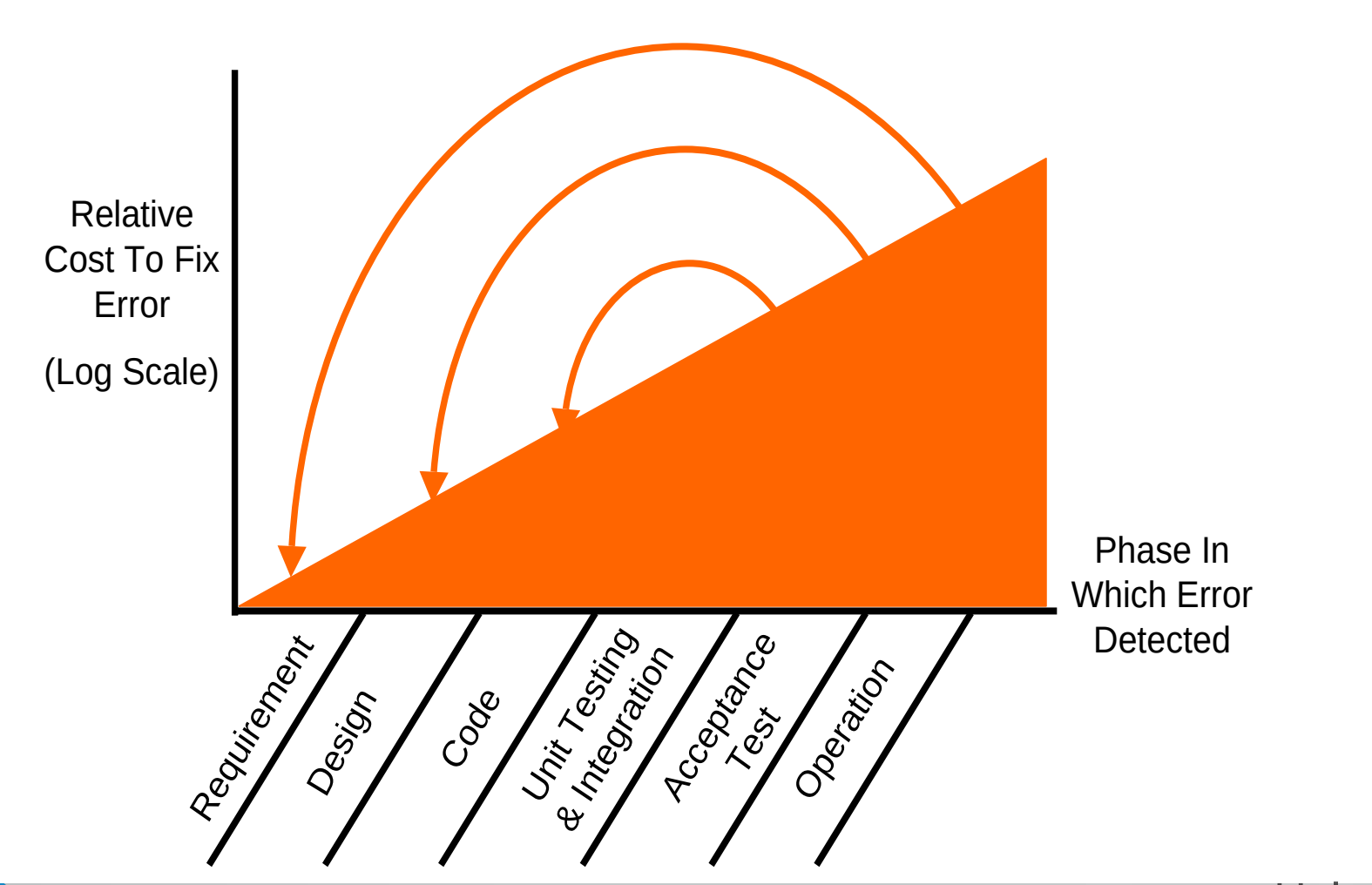
Even More Software failures

- In 1995 annual US spending on software projects reached 250 billion dollars
- This involved some 175,000 projects
- Of this spend:
  Overspend cost 59 billion dollars
  Cancelled projects cost 81 billion dollars

Causes of Software Failures?

Causes:

- Cost of delayed error detection



Why software projects fail?

- Inaccurate understanding of customer needs
- Inability to deal with changing requirements
- Modules that do not fit together
- Software that are hard to maintain/extend
- Poor Quality
  - Testing: normally should cost 40%
- Unacceptable performance
- Technology change and team-member change over time in long period projects
- Two main factors:
  - Increasing system complexity and demand
  - Failure to use software engineering methods

Different than regular engineers?

- Yes, normal engineers for buildings and bridges for example, have faced similar examples, risks, solutions.
- But we are not so lucky
- So, what is the problem, really?
  - **Complexity**
  - **Change**

Complexity:

- **Software development is Complex**!
- Abstract and intangible.
- **Important to distinguish "small" systems** (*one developer, one user, experimental use only*) **from "Complex" systems** (*multiple developers, multiple users, products*)
- **Experience with "small" systems is misleading**
  - *One person techniques do not scale up*
- **Analogy with bridge building:**
  - Over a stream = easy, one person job
  - Over a River … ? (*the techniques do not scale)*

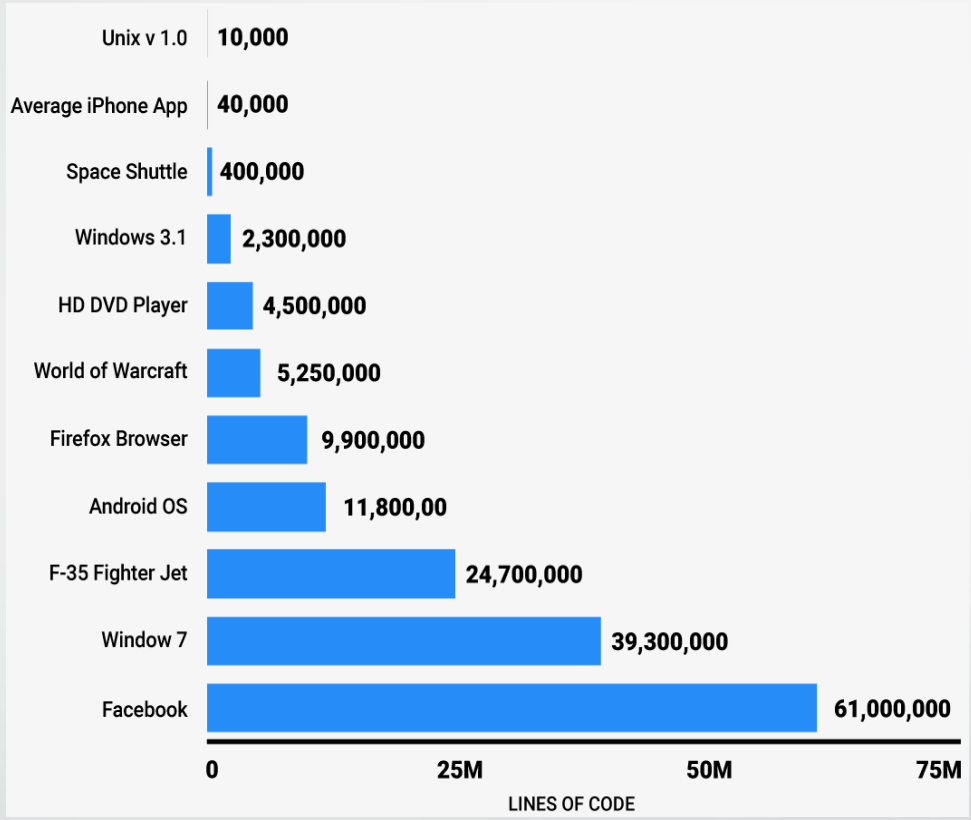
Why Software Engineering ?

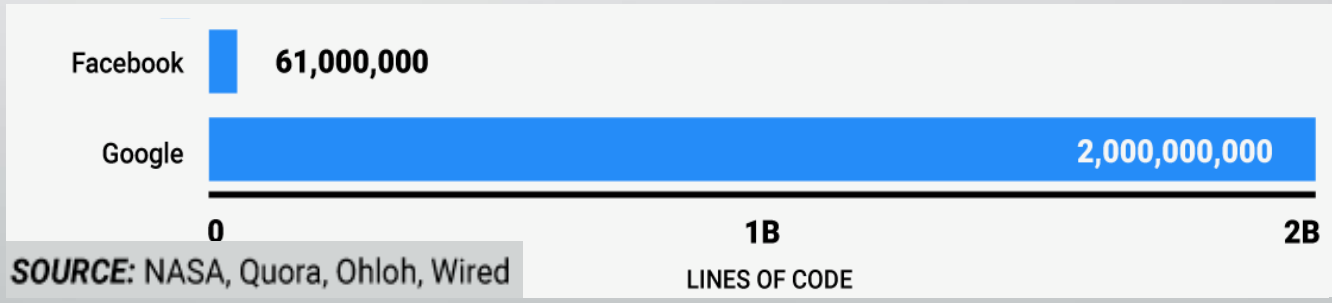
Complexity

increases *as*

Size

increases!





1.1 Professional software development

Figure 1.1: Frequently asked questions about software engineering 1

Question	Answer
What is software?	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
What are the attributes of good software?	Good software should deliver the required functionality and performance to the user and should be maintainable, dependable, and usable.
What is software engineering?	Software engineering is an engineering discipline that is concerned with all aspects of software production.
What are the fundamental software engineering activities?	Software specification, software development, software validation, and software evolution.
What is the difference between software engineering and computer science?	Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
What is the difference between software engineering and 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 more general process.

1.1 Professional software development Cont.

Figure 1.1: Frequently asked questions about software engineering 2

What are the key challenges facing software engineering?	Coping with increasing diversity, demands for reduced delivery times, and developing trustworthy software.
What are the costs of software engineering?	Roughly 60% of software costs are development costs; 40% are testing costs. For custom software, evolution costs often exceed development costs.
What are the best software engineering techniques and methods?	While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. You can't, therefore, say that one method is better than another.
What differences has the Web made to software engineering?	The Web has led to the availability of software services and the possibility of developing highly distributed service-based systems. Web-based systems development has led to important advances in programming languages and software reuse.

What is the difference between software engineering and computer science?

Computer Science	Software Engineering
is concerned with	
➢ theory ➢ fundamentals Algorithms, data structures, complexity theory, numerical methods	➢ the practicalities of developing and ➢ delivering useful software SE deals with practical problems in complex software products

Computer science theories are currently insufficient to act as a complete underpinning for software engineering, BUT it is a foundation for practical aspects of software engineering

1.1 Professional software development Cont.

- Software is not just the programs themselves but also all associated documentation, libraries, support websites, and configuration data that are needed to make these programs useful.
- The vast majority of software development is a professional activity:
  - Special team of developers.
  - For specific business purpose.
  - Formal and systematic (Process).
  - Software will be maintained and changed throughout its life
- Software Engineering supports professional SW development, rather than individual programming.
- Two types of Software product:
  - Generic products: MS Office
  - Customized Products: developed for a particular customer, examples: Ramallah Library System, Jawwal حسابي

Attributes of a Good Software

- Good software:
  - deliver the required functionality
  - Acceptable performance to the user
  - and should be maintainable,
  - dependable,
  - and usable
- Therefore, **a banking system** must be **secure**,
- An interactive **game** must be **responsive**,
- A **medical software** system must be **reliable**, and so on.

Figure 1.2 Essential attributes of good software

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.

Tip: Software Reliability is *the probability of failure-free software operation for a specified period of time in a specified environment*.

Software myths

- **Management myths**
  - *Standards and procedures for building software exist*
  - *Add more programmers if behind schedule*
- **Customer myths**
  - *A general description of objectives enough to start coding*
  - *Requirements may change as software is flexible*
- **Practitioner myths**
  - *Task accomplished when the program works*
  - *"Working program" the only project deliverable*

1.1.1 Software engineering

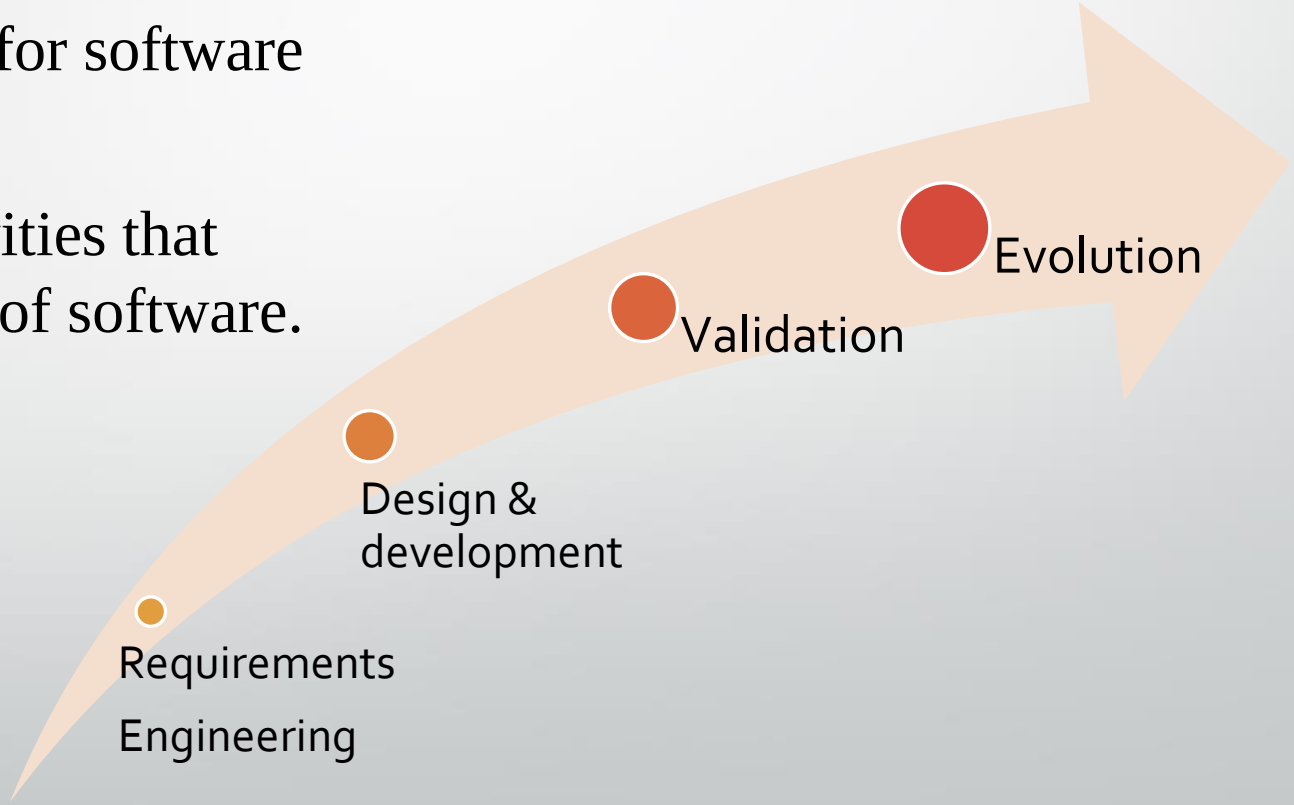
- Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.
- Engineering is about getting results of the required quality within schedule and budget.
- Software engineering is important for two reasons:
  - 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. For most types of systems, the majority of costs are the costs of changing the software after it has gone into use. Failure to use software engineering method leads to higher costs for testing, quality assurance, and long-term maintenance.

1.1.1 Software engineering Cont.

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

Software Development Process

- A systematic approach for software development.
- It is a sequence of activities that leads to the production of software.



Software Development Process Cont.

- **Different** types of systems need **different** development processes.
- Each type of system requires specialized software engineering techniques because the software has different characteristics. Examples:
  - E-commerce and MIS systems → **Incremental**
  - While real time (**critical**) system for an aircraft must be **completely specified** in details before design & implementation
  - An embedded control system in an automobile is safety-critical. It is very expensive to change. Such a system needs extensive verification and validation so that the chances of having to recall cars after sale to fix software problems are minimized.
  - For an interactive web-based system or app, iterative development and delivery is the best approach.

Different Approaches!

- It is pointless to look for universal notations, methods, or techniques for software engineering because different types of software require different approaches.
- All of these applications need software engineering; they do not all need the same software engineering techniques.

Different Types of Software → Different Types of Approaches

- There are many different types of application including:
  - **Stand-alone applications: Office products**
  - **Interactive transaction-based applications: execute on a remote computer such as: Ritaj, cloud-based services, such as mail and photo sharing.**
  - **Embedded control systems: car software systems**
  - **Entertainment systems: games**
  - **Data collection and analysis systems: Mars rovers**
  - **Systems for modeling and simulation: driving simulator.**

So, what is a successful software project?

- Good software should:
  - Deliver the required functionality
  - Efficient: does not waste voluble resources, response time
  - Usable
  - Dependable: reliable, secure, and safe.
  - Maintainable
  - **Within budget and time**

Software Engineering fundamentals that Apply to all Types of Software Systems

- Clear Development Process
- Dependability and performance
- Clear Requirements
- Reusability

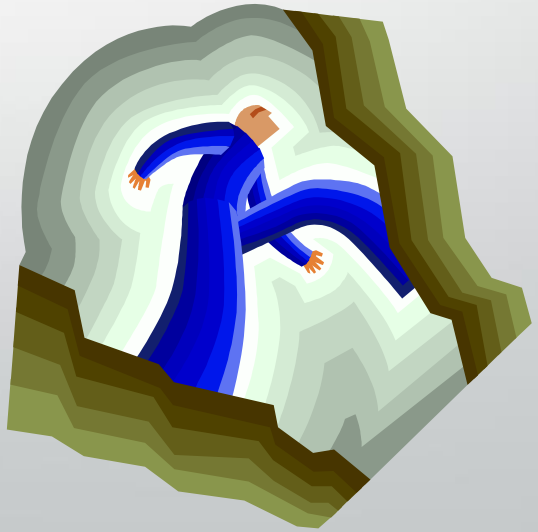
1.1.3 Software engineering and the Web

- Instead of writing software and deploying it on users' PCs, the software is deployed on a web server. This made it much cheaper to change and upgrade the software, as there was no need to install the software on every PC
- Software as a service has now become the standard approach to the delivery of web-based system products such as Google Apps, Microsoft Office 365, and Adobe Creative Suite.
- If you use services such as web-based mail, storage, or video, you are using a cloud-based system.

What are the key challenges facing software engineering?

Software engineering in the 22st century faces three key challenges:

- **Legacy systems**
  - Old, valuable systems must be maintained and updated
- **Increasing Diversity and Heterogeneity**
  - Systems are distributed and include a mix of different hardware and software
- **Dependability and Delivery**
  - Having trustworthy software with faster delivery of software (time-to-market)



1.2 Software engineering ethics

As a professional engineer, you should behave in an ethical and morally responsible way:

- *Confidentiality*
- *Competence*
- *Intellectual property:* such as patents and copyright.
- *Computer misuse*

KEY POINTS

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Software is not just a program or programs but also includes documentation.
- Essential software product attributes are maintainability, dependability, security, efficiency, and acceptability.
- The software process includes all of the activities involved in software development. 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. These fundamentals include software processes, dependability, security, requirements, and reuse.
- There are many different types of systems, and each requires appropriate software engineering tools and techniques for their development. There are few, if any, specific design and implementation techniques that are applicable to all kinds of systems.
- The fundamental ideas of software engineering are applicable to all types of software systems. These fundamentals include managed software processes, software dependability and security, requirements engineering, and software reuse.
- Software engineers have responsibilities to the engineering profession and society. They should not simply be concerned with technical issues.