Introduction to Software Engineering

Definition of software:

- Generally, people equate the term software with computer programs. However, a broader definition is: Software is not just the programs but also all associated documentation and configuration data that is needed to make these programs operate correctly.
- 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.
- Software engineers are concerned with developing software products, i.e., software which can be sold to a customer.

Software and its Types

Types 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 software for PCs such as databases, word processors, drawing packages and project management tools.

Customized (or bespoke) products

- These are systems which are commissioned by a particular customer.
- A software contractor develops 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 and its Types

- 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.
- 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.
- 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 required, and so on.

Fundamental Software Engineering Activities

Software engineering

- Software engineering is an engineering discipline that is concerned with all aspect of software production from the early stages of system specification to maintain in 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, but 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.

Fundamental Software Engineering Activities

- **All aspects of software production**
- Software engineering is not just concerned with the technical processes of software development but also with activities such as software project management and with the development of tools, methods and theories to support software production.
- In general, 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.
- Less formal development is particularly appropriate for the development of web-based systems, which requires a blend of software and graphical design skills.

Fundamental Software Engineering Activities

There are four fundamental process activities that are common to all software processes. These are:

- Software specification: Where customers and engineers define the software 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 adapt it to changing customer and market requirements.

Software Engineering and its Importance

- Software engineering is essential for the functioning of government, society and national and international businesses and institutions.
- 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.
- More than 75% of the world’s population have a software-controlled mobile phone and, and in near future, almost all of these will be Internet-enabled.

Software Engineering and its Importance

- Software systems are abstract and intangible.
- They are not constrained by the properties of materials, governed by physical laws or by manufacturing processes.
- 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.
- There are many different types of software system, from simple embedded systems to complex, worldwide information systems. There are no universal notations, methods or techniques for software engineering because different types of software require different approaches.

Software Engineering and its Importance

- 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.
- All of these applications need software engineering; they do not all need the same software engineering methods and techniques.
- Software engineers can be rightly proud of their achievements. 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.
- All forms of travel would be more dangerous and expensive. Challenges for humanity in the 21st century are climate change, fewer natural resources, changing demographics and an expanding world population.
- We will rely on software engineering to develop the systems that we need to cope with these issues.

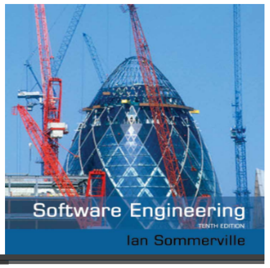
Difference between software engineering and computer science

- Essentially, computer science is concerned with the theories and methods that underlie computers and software systems, whereas software engineering is concerned with the practical problems of producing software.
- Some knowledge of computer science is essential for software engineers in the same way that some knowledge of physics is essential for electrical engineers.
- Ideally, all of software engineering should be underpinned by theories of computer science, but in reality this is not the case.
- Software engineers must often use ad hoc approaches to developing the software.
- Elegant theories of computer science cannot always be applied to real, complex problems that require a software solution.

Difference between software engineering and system engineering

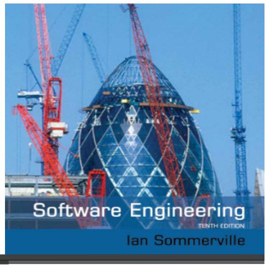
- System engineering is concerned with all aspects of the development and evolution of complex systems where software plays a major role.
- System engineering is therefore concerned with hardware development, policy and process design and system deployment as well as software engineering.
- System engineers are involved in specifying the system, defining its overall architecture and then integrating the different parts to create the finished system.
- They are less concerned with the engineering of the system components (hardware, software, etc.).
- System engineering is an older discipline than software engineering.
- People have been specifying and assembling complex industrial systems such as aircraft and chemical plants for more than a hundred years.
- However, as the percentage of software in systems has increased, software engineering techniques such as use-case modelling and configuration management are being used in the systems engineering process.

Introduction to Software Engineering: summary of key points



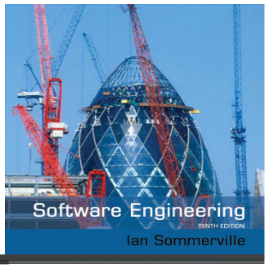
Question	Answer
What is software?	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
What are the attributes of good software?	Good software should deliver the required functionality and performance to 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.

Introduction to Software Engineering: summary of key points



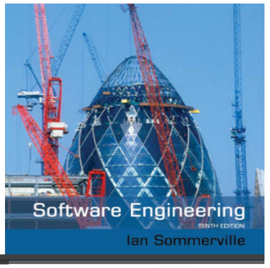
Question	Answer
What are the key challenges facing software engineering?	Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software.
What are the costs of software engineering?	Roughly 60% of software costs are development costs, 40% are testing costs. For 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.

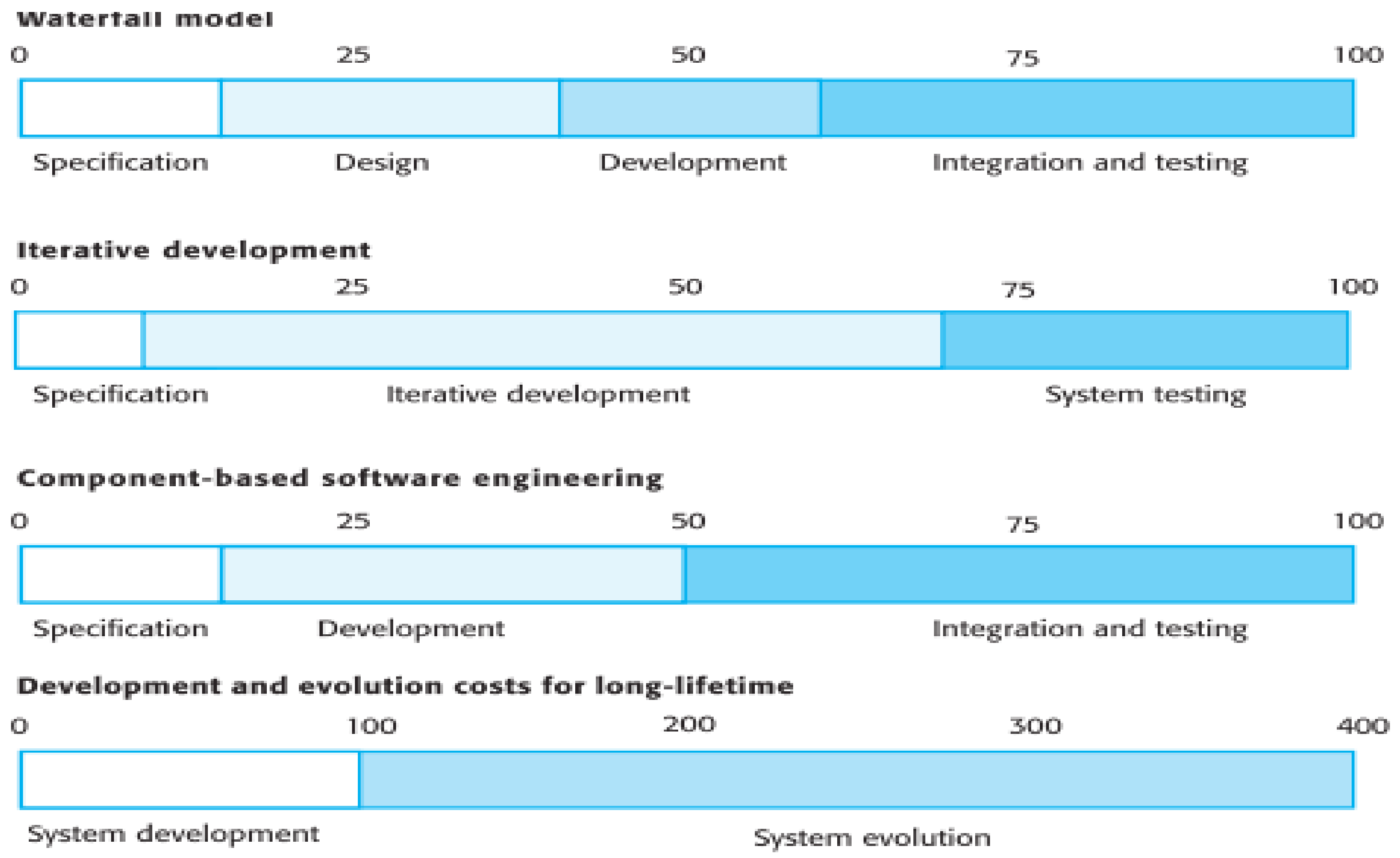
Costs of software engineering



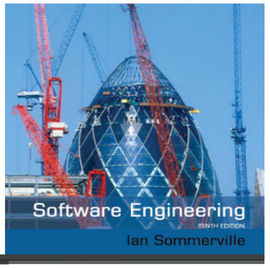
- There is no simple answer to this question as the distribution of costs across the different activities in the software process depends on the process used and the type of software that is being developed.
- For example, real-time software usually requires more extensive validation and testing than web-based systems.
- However, each of the different generic approaches to software development has a different profile of cost distribution across the software process activities.
- If we assume that the total cost of developing a complex software system is 100 cost units then Figure below illustrates how these are spent on different process activities.

Costs of software engineering



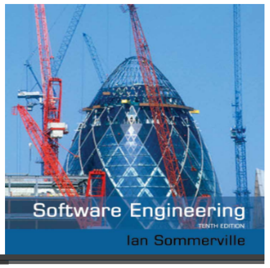


Costs of software engineering



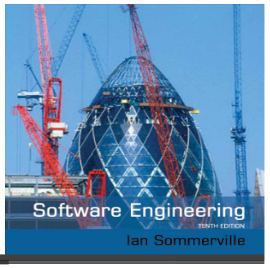
- In the waterfall approach, the costs of specification, design, implementation and integration are measured separately.
- Notice that system integration and testing is the most expensive development activity.
- Normally, this is about 40% of the total development costs but for some critical systems it is likely to be at least 50% of the system development costs.
- If the software is developed using an iterative approach, there is no hard line between specification, design and development.
- Specification costs are reduced because only a high-level specification is produced before development in this approach.
- Specification, design, implementation, integration and testing are carried out in parallel within a development activity.

Costs of software engineering



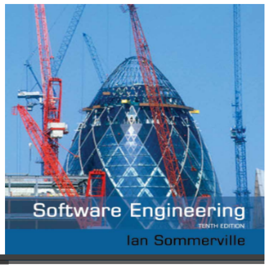
- However, you still need an independent system testing activity once the initial implementation is complete.
- Component-based software engineering has only been widely used for a short time. We don’t have accurate figures for the costs of different software development activities in this approach. However, we know that development costs are reduced relative to integration and testing costs.
- Integration and testing costs are increased because you have to ensure that the components that you use actually meet their specification and work as expected with other components.
- On top of development costs, costs are also incurred in changing the software after it has gone into use.
- The costs of evolution vary dramatically depending on the type of system.

Costs of software engineering



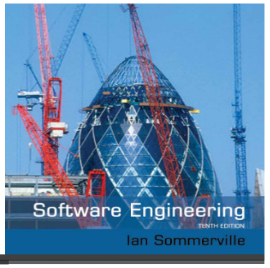
- For long-lifetime software systems, such as command and control systems that may be used for 10 years or more, these costs are likely to exceed the development costs by a factor of 3 or 4.
- However, smaller business systems have a much shorter lifetime and correspondingly reduced evolution costs.
- These cost distributions hold for customized software that is specified by a customer and developed by a contractor.
- For software products that are (mostly) sold for PCs, the cost profile is likely to be different.
- These products are usually developed from an outline specification using an evolutionary development approach.
- Specification costs are relatively low.

Costs of software engineering



- However, because they are intended for use on a range of different configurations, they must be extensively tested.
- The evolution costs for generic software products are particularly hard to estimate.
- Once a version of the product has been released, work starts on the next release and, for marketing reasons, this is likely to be presented as a new (but compatible) product rather than as a modified version of a product that the user has already bought.
- Therefore, the evolution costs are not assessed separately as they are in customized software but are simply the development costs for the next version of the system.

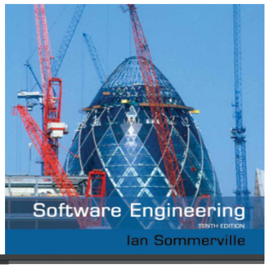
Attributes of good software



- As well as the services that it provides, software products have a number of other

associated attributes that reflect the quality of that software.

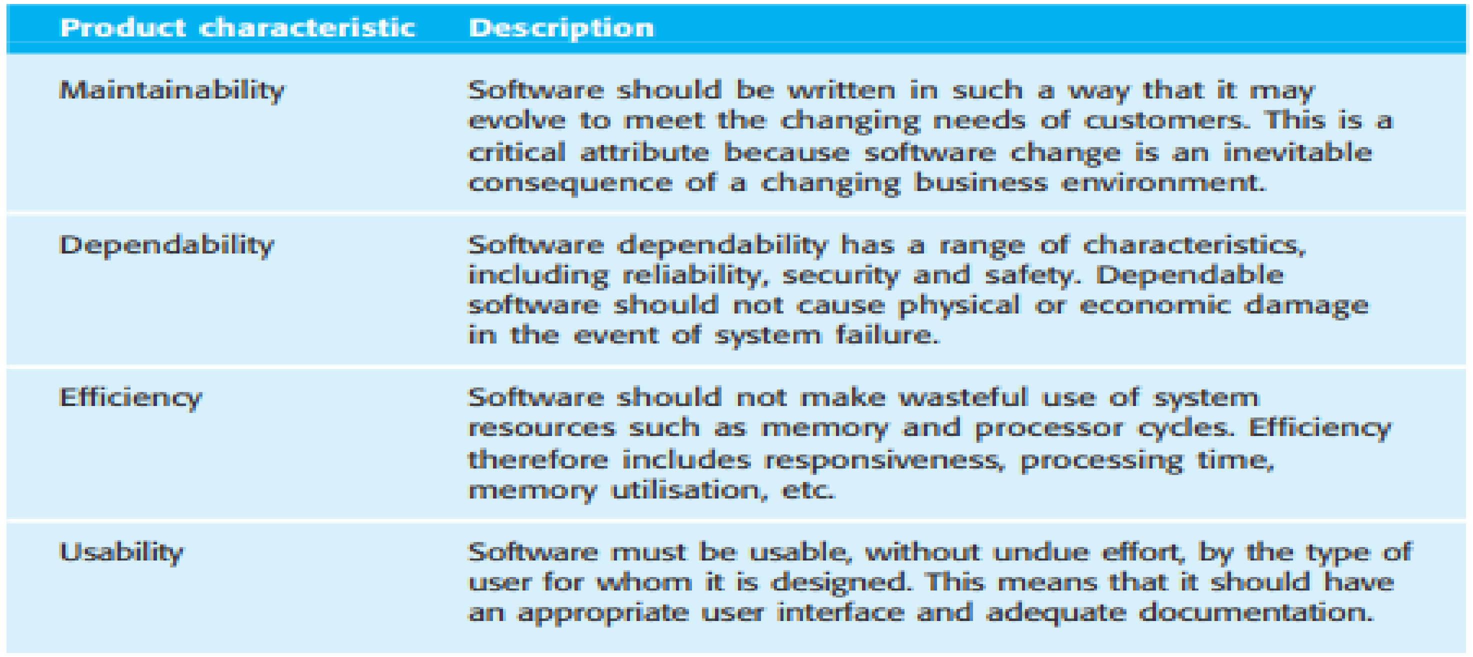
- These attributes are not directly concerned with what the software does.
- Rather, they reflect its behavior while it is executing and the structure and organization of the source program and associated documentation.
- Examples of these attributes (sometimes called nonfunctional attributes) are the software’s response time to a user query and the understandability of the program code.
- The specific set of attributes that you might expect from a software system obviously depends on its application.



Attributes of good software

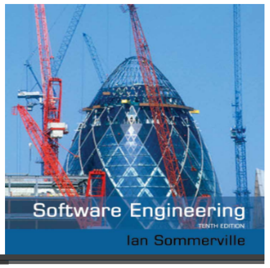
- Therefore, a banking system must be secure, an interactive game must be responsive, a telephone switching system must be reliable, and so on.
- These can be generalized into the set of attributes shown in Figure below, which, are the essential characteristics of a well-designed software system.

Figure 1.5 Essential attributes of good software



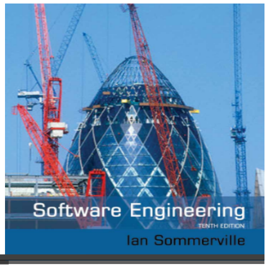
Product characteristic	Description
Maintainability	Software should be written in such a way that it may evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable consequence of a changing business environment.
Dependability	Software dependability has a range of characteristics, including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Usability	Software must be usable, without undue effort, by the type of user for whom it is designed. This means that it should have an appropriate user interface and adequate documentation.

Challenges of software engineering



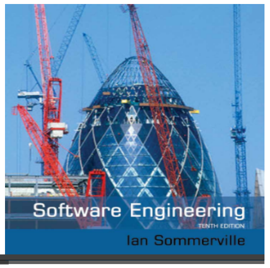
- **The heterogeneity challenge**
- Increasingly, systems are required to operate as distributed systems across networks that include different types of computer and mobile devices.
- As well as running on general-purpose computers, software may also have to execute on mobile phones and tablets.
- It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

Challenges of software engineering



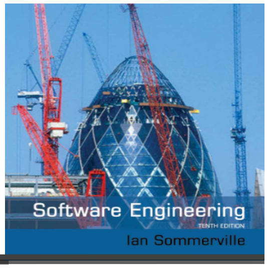
- **Business and social change**
- Businesses and society are changing incredibly quickly as emerging economies develop and new technologies become available.
- They need to be able to change their existing software and to rapidly develop new software.
- Many traditional software engineering techniques 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 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.

Challenges of software engineering



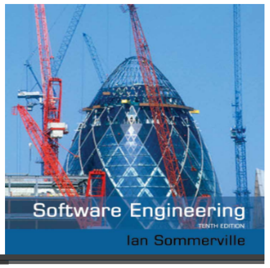
- This is especially true for remote software systems accessed through a web page or web service interface.
- We have to make sure that malicious users cannot successfully attack our software and that information security is maintained.
- **Scale**
- Software has to be developed across a very wide range of scales.
- From very small embedded systems in portable or wearable devices through to Internet-scale, cloud-based systems that serve a global community.
- To address these challenges we will need new tools and techniques as well as innovative ways of combining and using existing software engineering methods.

Professional Software Development



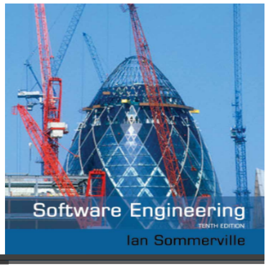
- Lots of people write programs.
- People in business write spreadsheet programs to simplify their jobs; scientists and engineers write programs to process their experimental data; hobbyists write programs for their own interest and enjoyment.
- However, most software development is a professional activity where software is developed for business purposes, for inclusion in other devices or as software products such as information systems, CAD systems, etc.
- The key distinctions are that professional software is intended for use by someone apart from its developer and that teams rather than individuals usually develop the software.
- It is maintained and changed throughout its life.

Professional Software Development

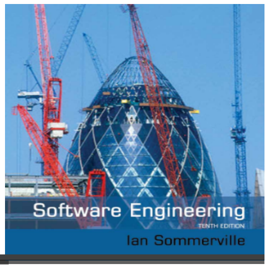


- Software engineering is intended to support professional software development, rather than individual programming.
- It includes techniques that support program specification, design, and evolution, none of which are normally relevant for personal software development.
- A professionally developed software system is often more than a single program, a system may consist of several separate programs and configuration files that are used to set up these programs.
- It may include system documentation, which describes the structure of the system, user documentation, which explains how to use the system and web sites for users to download recent product information.

Professional Software Development



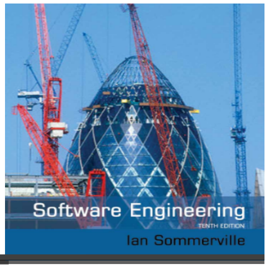
- This is one of the important differences between professional and amateur software development.
- If you are writing a program for yourself, no one else will use it and you don't have to worry about writing program guides, documenting the program design, and so on.
- However, if you are writing software that other people will use and other engineers will change then you usually have to provide additional information as well as the code of the program.



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.
- Rather, a diverse set of software engineering methods and tools has evolved over the past 50 years.
- Perhaps the most significant factor in determining which software engineering methods and techniques are most important is the type of application that is being developed. There are many different types of application including:

Software Engineering Diversity



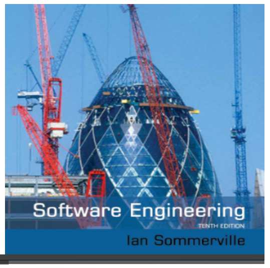
Stand-alone applications:

- 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, photo manipulation software, travel apps, productivity apps, etc.

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.

Software Engineering Diversity

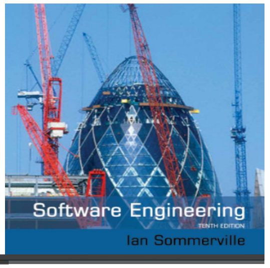


- This class of application also includes business systems, where a business provides access to its systems through a web browser or special-purpose client program and cloud-based services, such as mail and photo sharing.
- Interactive applications often incorporate a large data store that is accessed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

Embedded control systems:

- These are software control systems that control and manage hardware devices.

Software Engineering Diversity



- Examples of embedded systems include the software in a mobile (cell) phone, software that controls anti-lock braking in a car and software in a microwave oven to control the cooking process.

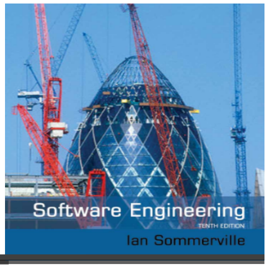
Batch processing systems:

- 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

Software Engineering Diversity

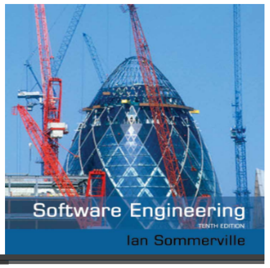


- Most of these systems are games of one kind or another, which may run on special-purpose console hardware.
- The quality of the user interaction offered is the most important distinguishing characteristic of entertainment systems.

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.

Software Engineering Diversity

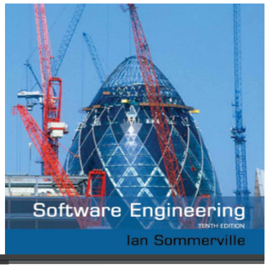


Data collection and analysis systems:

- 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:

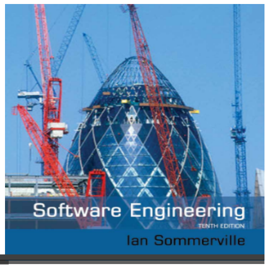
- These are systems, used in enterprises and other large organizations that are composed of a number of other software systems. Some of these may be generic software products, such as an ERP system.
- Other systems in the assembly may be specially written for that environment.



Software Engineering Diversity

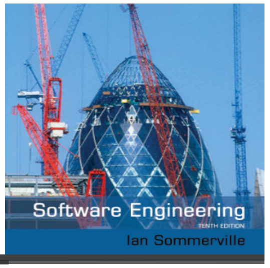
- Of course, the boundaries between these system types are blurred.
- If you develop a game for a phone, you have to take into account the same constraints (power, hardware interaction) as the developers of the phone software.
- Batch processing systems are often used in conjunction with web-based transaction systems.
- For example, in a company, travel expense claims may be submitted through a web application but processed in a batch application for monthly payment.
- Each type of system requires specialized software engineering techniques because the software has different characteristics. For example, an embedded control system in an automobile is safety-critical and is burned into ROM when installed in the vehicle. It is therefore very expensive to change.

Software Engineering Diversity



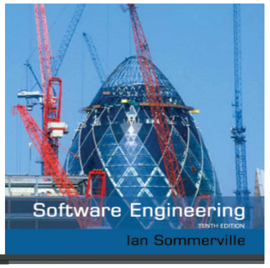
- Such a system needs extensive verification and validation so that the chances of having to recall cars after sale to fix software problems are minimized.
- User interaction is minimal (or perhaps non-existent) so there is no need to use a development process that relies on user interface prototyping.
- For an interactive web-based system or app, iterative development and delivery is the best approach, with the system being composed of reusable components.
- However, such an approach may be impractical for a system of systems, where detailed specifications of the system interactions have to be specified in advance so that each system can be separately developed.
- Nevertheless, there are software engineering fundamentals that apply to all types of software system:

Software Engineering Diversity



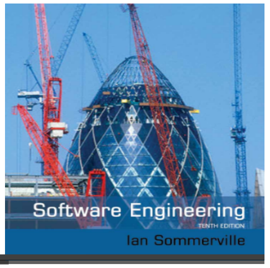
- They should be developed using a managed and understood development process. The organization developing the software should plan the development process and have clear ideas of what will be produced and when it will be completed. Of course, the specific process that you should use depends on the type of software that you are developing.
- Dependability and performance are important for all types of system. Software should behave as expected, without failures and should be available for use when it is required. It should be safe in its operation and, as far as possible, should be secure against external attack. The system should perform efficiently and should not waste resources.
- Understanding and managing the software specification and requirements (what the software should do) are important. You have to know what different customers and users of the system expect from it and you have to manage their expectations so that a useful system can be delivered within budget and to schedule.

Software Engineering Diversity



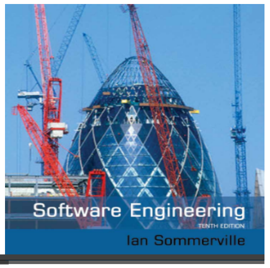
- You should make as effective use as possible of existing resources. This means that, where appropriate, you should reuse software that has already been developed rather than write new software.
- These fundamentals are independent of the program language used for software development.
- For example, a dynamic language, such as Ruby, is the right type of language for interactive system development but is inappropriate for embedded systems engineering.

Internet software engineering



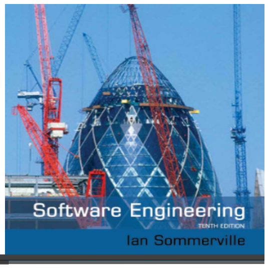
- The development of the Internet and the World Wide Web has had a profound effect on all of our lives.
- Initially, the web was primarily a universally accessible information store and it had little effect on software systems.
- These systems ran on local computers and were only accessible from within an organization.
- Around 2000, the web started to evolve and more and more functionality was added to browsers.
- This meant that web-based systems could be developed where, instead of a special-purpose user interface, these systems could be accessed using a web browser.
- This led to the development of a vast range of new system products that delivered innovative services, accessed over the web. These are often funded by adverts that are displayed on the user's screen and do not involve direct payment from users.

Internet software engineering



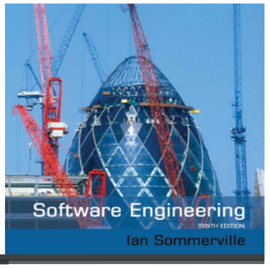
- As well as these system products, the development of web browsers that could run small programs and do some local processing led to an evolution in business and organizational software.
- Instead of writing software and deploying it on users' PCs, the software was deployed on a web server.
- This made it much cheaper to change and upgrade the software, as there was no need to install the software on every PC.
- It also reduced costs, as user interface development is particularly expensive. Wherever it has been possible to do so, businesses have moved to web-based interaction with company software systems.

Internet software engineering



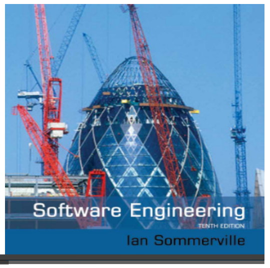
- The notion of ‘software as a service’ was proposed early in the 21st century.
- This has now become the standard approach to the delivery of web based system products such as Google Apps, Microsoft Office 365 and Adobe Creative Suite.
- More and more software runs on remote ‘clouds’ instead of local servers and is accessed over the Internet.
- A computing cloud is a huge number of linked computer systems that is shared by many users.
- Users do not buy software but pay according to how much the software is used or are given free access in return for watching adverts that are displayed on their screen.
- If you use services such as web-based mail, storage or video, you are using a cloud-based system.

Internet software engineering



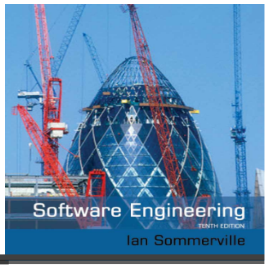
- The advent of the web has led to a dramatic change in the way that business software is organized.
- Before the web, business applications were mostly monolithic, single programs running on single computers or computer clusters.
- Communications were local, within an organization. Now, software is highly distributed, sometimes across the world.
- Business applications are not programmed from scratch but involve extensive reuse of components and programs.
- This change in software organization has had a major effect on software engineering for web-based systems.

Internet software engineering



- Software reuse has become 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, often bundled together in a framework.
- It is now generally recognized that it is impractical to specify all the requirements for such systems in advance. Web-based systems are always developed and delivered incrementally.
- Software may be implemented using service-oriented software engineering, where the software components are stand-alone web services.
- Interface development technology such as AJAX (Holdener 2008) and HTML5 (Freeman 2011) have emerged that support the creation of rich interfaces within a web browser.
- The fundamental ideas of software engineering, apply to web-based software as they do to other types of software. Web based systems are getting larger and larger so the development of software engineering techniques to deal with scale and complexity are relevant here.

Software Engineering Ethics

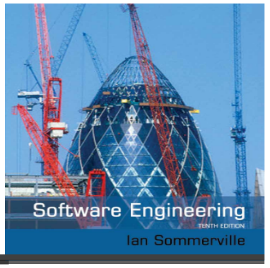


Professional and ethical responsibility

- Like other engineering disciplines, software engineering is carried out within a legal

  and social framework that limits the freedom of engineers.
- Software engineers must accept that their job involves wider responsibilities than simply the application o technical skills.
- They must also behave in an ethical and morally responsible way if they are to be respected as professionals.
- It goes without saying that you should always 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.

Software Engineering Ethics



- However, there are areas where standards of acceptable behavior are not bounded by laws but by the more tenuous notion of professional responsibility. Some of these are:

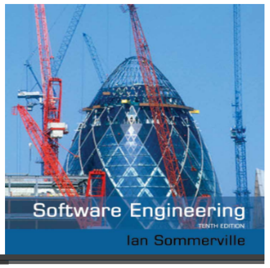
Confidentiality: You should normally respect the confidentiality of your employers or clients irrespective of whether a formal confidentiality agreement has been signed.

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.

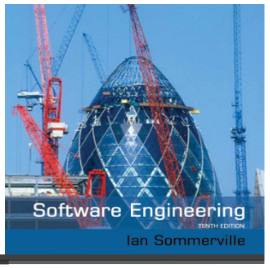
Computer misuse: You should not use your technical skills to misuse other people’s computers. Computer misuse ranges from relatively trivial (game playing on an employer’s machine, say) to extremely serious (dissemination of viruses)

Software Engineering Ethics



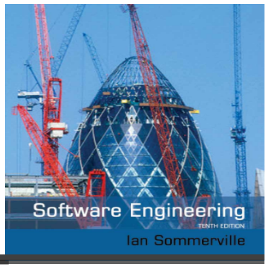
- Professional societies and institutions have an important role to play in setting ethical standards.
- Organizations such as the ACM, the IEEE (Institute of Electrical and Electronic Engineers) and the British Computer Society publish a code of professional conduct or code of ethics.
- Members of these organizations undertake to follow that code when they sign up for membership. These codes of conduct are generally concerned with fundamental ethical behavior.
- The ACM and the IEEE have cooperated to produce a joint code of ethics and professional practice. This code exists in both a short form, shown in below and a longer form that adds detail and substance to the shorter version.

Software Engineering Ethics



Professional Practices

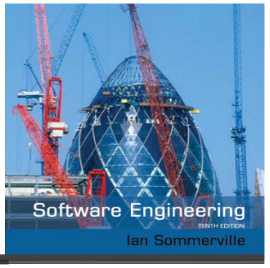
- Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession.
- In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:
- 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.



Software Engineering Ethics

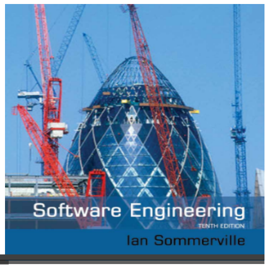
- 4. JUDGMENT – Software engineers shall maintain integrity and independence in their professional judgment.
- 5. MANAGEMENT – Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
- 6. PROFESSION – Software 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.

Software Engineering Ethics



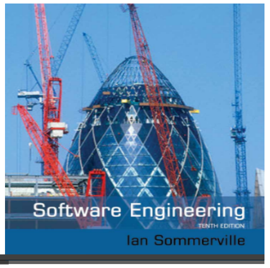
- The rationale behind above code is summarized in the first two paragraphs of the longer form which is described as below:
- Computers have a central and growing role in commerce, industry, government, medicine, education, entertainment and society at large.
- Software engineers are those who contribute 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.

Software Engineering Ethics



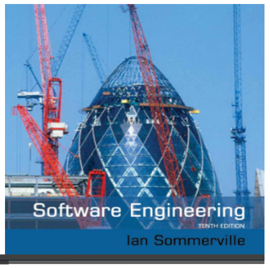
- In accordance with that commitment, software engineers shall adhere to the Code of Ethics and Professional Practice.
- The Code contains eight Principles as described in previous slides, related to the behavior of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession.
- The Principles identify the ethically responsible relationships in which individuals, groups, and organizations participate and the primary obligations within these relationships.
- The Clauses of each Principle are illustrations of some of the obligations included in these relationships.
- These obligations are founded in the software engineer’s humanity, in special care owed to people affected by the work of software engineers, and the unique elements of the practice of software engineering. The Code prescribes these as obligations of anyone claiming to be or aspiring to be a software engineer.

Software Engineering Ethics



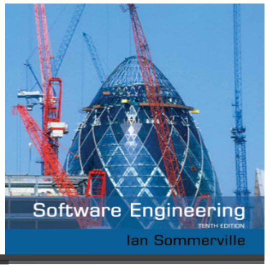
- In any situation where different people have different views and objectives, you are likely to be faced with ethical dilemmas.
- For example, if you disagree, in principle, with the policies of more senior management in the company, how should you react?
- Clearly, this depends on the particular individuals and the nature of the disagreement.
- Is it best to argue a case for your position from within the organization or to resign in principle?
- If you feel that there are problems with a software project, when do you reveal these to management?
- If you discuss these while they are just a suspicion, you may be overreacting to a situation; if you leave it too late, it may be impossible to resolve the difficulties.

Software Engineering Ethics



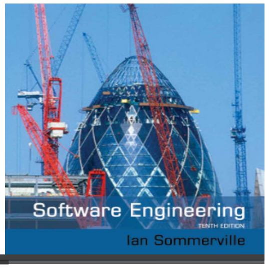
- Such ethical dilemmas face all of us in our professional lives and, fortunately, in most cases they are either relatively minor or can be resolved without too much difficulty.
- Where they cannot be resolved, the engineer is faced with, perhaps, another problem.
- The principled action may be to resign from their job, but this may well affect others such as their partner or their children.
- A particularly difficult situation for professional engineers arises when their employer acts in an unethical way.
- Say a company is responsible for developing a safety-critical system and because of time-pressure, falsifies the safety validation records.
- Is the engineer’s responsibility to maintain confidentiality or to alert the customer or publicize, in some way, that the delivered system may be unsafe?

Software Engineering Ethics



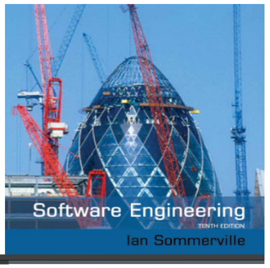
- The problem here is that there are no absolutes when it comes to safety.
- Although the system may not have been validated according to predefined criteria, these criteria may be too strict.
- The system may actually operate safely throughout its lifetime. It is also the case that, even when properly validated, a system may fail and cause an accident.
- Early disclosure of problems may result in damage to the employer and other employees; failure to disclose problems may result in damage to others.
- You must make up your own mind in these matters. The appropriate ethical position here depends entirely on the views of the individuals who are involved.
- In this case, the potential for damage, the extent of the damage and the people affected by the damage should influence the decision.

Software Engineering Ethics



- If the situation is very dangerous, it may be justified to publicize it using the national press (say).
- However, you should always try to resolve the situation while respecting the rights of your employer.
- Another ethical issue is participation in the development of military and nuclear systems.
- Some people feel strongly about these issues and do not wish to participate in any systems development associated with military systems.
- Others will work on military systems but not on weapons systems.
- Yet others feel that national security is an overriding principle and have no ethical objections to working on weapons systems.

Software Engineering Ethics



- In this situation it is important that both employers and employees should make their views known to each other in advance.
- Where an organization is involved in military or nuclear work, it should be able to specify that employees must be willing to accept any work assignment.
- Equally, if an employee is taken on and makes clear that he does not wish to work on such systems, employers should not put pressure on him to do so at some later date.
- The general area of ethics and professional responsibility is one that has received increasing attention over the past few years.
- It can be considered from a philosophical standpoint where the basic principles of ethics are considered, and software engineering ethics are discussed with reference to these basic principles.