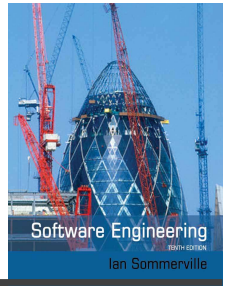
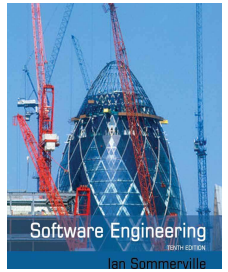


MSP Context: Software Engineering and Systems Engineering

Some software news…



Impostos

Portal das Finanças ainda apresenta problemas

Paula Cravina de Sousa
01/06/11 17:05

1 Leitores Online

Docentes excluídos das ofertas de escola por erro informático da plataforma online da DGRHE

Dados de dinheiro rumo a offshores perdidos por "erro informático"

Economia

Segurança Social envia cartas a 34 mil mortos

Entidade usou base de dados desactualizada

Por Redacção VC | 2012-01-13 08:43

Fenprof pede que listas sejam retiradas

Ministra nega erros na colocação de professores

14.10.2004 - 21:39 Por PUBLICO.PT

Segurança Social cobra dívidas pagas devido a erro informático

18 | 07 | 2007 08.30H

Galp diz que não reflectiu a descida do IVA numa botija de gás butano devido a um erro informático

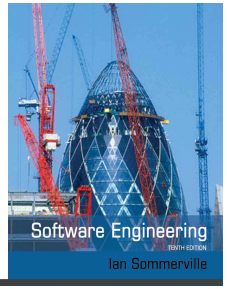
09 Julho 2008 | 18:55
Jornal de Negócios Online - negocios@negocios.pt

Partilhar Tweet 0

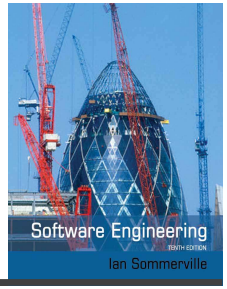
Contribuintes de IRS penalizados por erro informático

JOÃO RAMOS DE ALMEIDA
28/05/2005 - 00:00

Standish Group



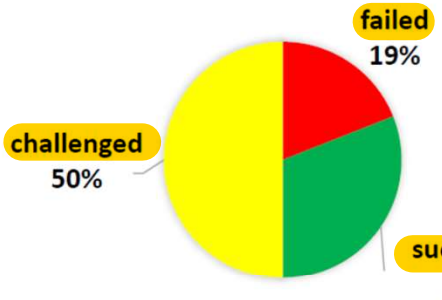
- The **Standish Group International, Inc.** or **Standish Group** is an independent international IT research advisory firm founded in 1985, known from their reports about information systems implementation projects in the public and private sector
- The firm focuses on mission-critical software applications, especially focusing on failures and possible improvements in IT projects



Project Success

Quick Reference Card

Based on CHAOS 2020: Beyond Infinity Overview. January'2021, QRC by Henny Portman



Modern measurement (software projects)

Good Sponsor, Good Team, and Good Place are the only things we need to improve and build on to improve project performance.

The Good Place is where the sponsor and team work to create the product. It's made up of the people who support both sponsor and team. These people can be helpful or destructive. It's imperative that the organization work to improve their skills if a project is to succeed. This area is the hardest to mitigate, since each project is touched by so many people. Principles for a Good Place are:

- The Decision Latency Principle
- The Emotional Maturity Principle
- The Communication Principle
- The User Involvement Principle
- The Five Deadly Sins Principle
- The Negotiation Principle
- The Competency Principle
- The Optimization Principle
- The Rapid Execution Principle
- The Enterprise Architecture Principle

Successful project Resolution by Good Place Maturity Level:

highly mature	50%
mature	34%
moderately mature	23%
not mature	23%

The Good Team is the project's workhorse. They do the heavy lifting. The sponsor breathes life into the project, but the team takes that breath and uses it to create a viable product that the organization can use and from which it derives value. Since we recommend small teams, this is the second easiest area to improve. Principles for a Good Team are:

- The Influential Principle
- The Mindfulness Principle
- The Five Deadly Sins Principle
- The Problem-Solver Principle
- The Communication Principle
- The Acceptance Principle
- The Respectfulness Principle
- The Confrontationist Principle
- The Civility Principle
- The Driven Principle

Successful project Resolution by Good Team Maturity Level:

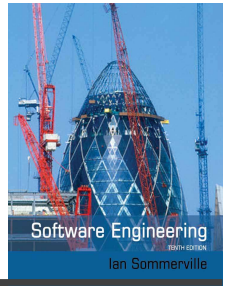
highly mature	66%
mature	46%
moderately mature	21%
not mature	1%

The Good Sponsor is the soul of the project. The sponsor breathes life into a project, and without the sponsor there is no project. Improving the skills of the project sponsor is the number-one factor of success – and also the easiest to improve upon, since each project has only one.
Principles for a Good Sponsor are:

- The Decision Latency principle
- The Vision Principle
- The Work Smart Principle
- The Daydream Principle
- The Influence Principle
- The Passionate Principle
- The People Principle
- The Tension Principle
- The Torque Principle
- The Progress Principle

Successful project Resolution by Good Sponsor Maturity Level:

highly mature	67%
mature	33%
moderately mature	21%
not mature	18%

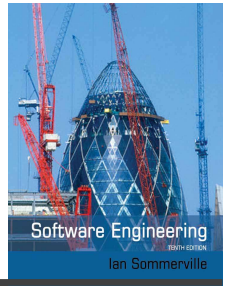


The Good Team

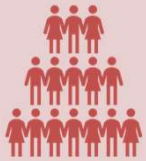
The Good Team discusses the skills involved in being a good team. The good team is the project's workhorse. They do the heavy lifting.

The sponsor breathes life into the project, but the team takes that breath and uses it to create a viable product that the organization can use and from which it derives value.

The Good Place



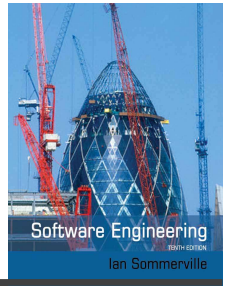
The Good Place covers what's needed to provide a good place for projects to thrive. The good place is where the sponsor and team work to create the product.

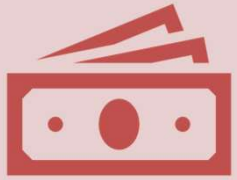


It's made up of the people who support both sponsor and team. These people can be helpful or destructive. It's imperative that the organization work to improve their skills if a project is to succeed.

This area is the hardest to mitigate, since each project is touched by so many people.

The Good Sponsor





The Good Sponsor discusses the skills needed to be a good sponsor. The good sponsor is the soul of the project.

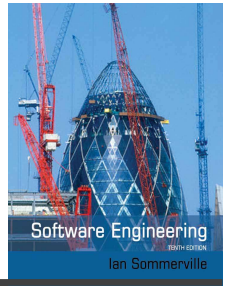
The sponsor breathes life into a project, and without the sponsor there is no project.

Improving the skills of the project sponsor is the number-one factor of success – and also the easiest to improve upon, since each project has only one.

Project Management's Five Deadly Sins (Standish Group's CHAOS Report)

1. **Over Ambition:** Project manager taking on too much at once.
2. **Prestige:** Project manager having an over confident attitude towards team members.
3. **Ignorance:** Poor understanding of project goals and overall objective.
4. **Absence:** Key member or decision-maker are not dedicating enough time.
5. **Dishonesty:** Concealing facts.

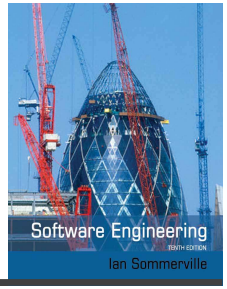
CHAOS Report



Failure Factors
(as identified by the surveyed projects owners)

Incomplete requirements	13,1%
Lack of user involvement	12,4%
Lack of resources	10,6%
Unrealistic expectations	9,9%
Lack of executive support	9,3%
Changing requirements and specifications	8,7%
Lack of planning	8,1%
Didn't need it any longer	7,5%
Lack of IT management	6,2%
Technology illiteracy	4,3%

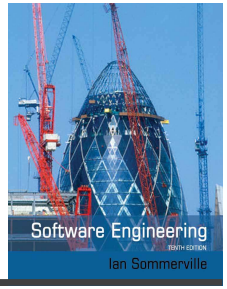
CHAOS Report



Success Factors
(as identified by the surveyed projects owners)

User Involvement	15.9%
Executive Management Support	13.9%
Clear Statement of Requirements	13.0%
Proper Planning	9.6%
Realistic Expectations	8.2%
Smaller Project Milestones	7.7%
Competent Staff	7.2%
Clear Vision & Objectives	2.9%
Hard-Working, Focused Staff	2.4%
Other	13.9%

Initial concepts



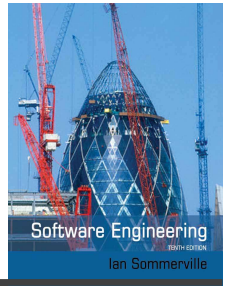
Professional software development

What is meant by software engineering.

Software engineering ethics

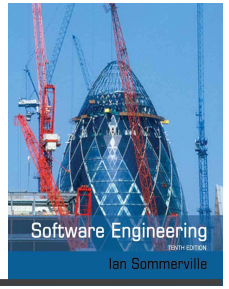
A brief introduction to ethical issues that affect software engineering.

Software engineering



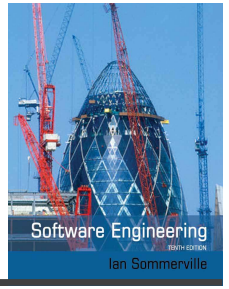
- The economies of ALL developed nations are dependent on software.
- Most of systems are software controlled
- Software engineering is concerned with theories, methods and tools for professional software development.
- Expenditure on software represents a significant fraction of GNP in all developed countries.

Software costs

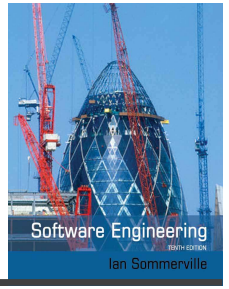


- Software costs often dominate computer system costs. The costs of software are often greater than the hardware cost.
- Software **costs more to maintain** than it does to develop. For systems with a long life, maintenance costs may be several times development costs.
- Software engineering is concerned with cost-effective software development.

Software project failure

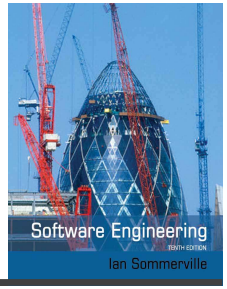


- *Increasing system complexity*
  - As new SE techniques help us to build larger, more complex systems, the demands change
  - Systems have to be built and delivered more quickly
  - larger, even more complex systems are required
  - systems have to have new capabilities that were previously thought to be impossible
- *Failure to use software engineering methods*
  - It is easy to write computer programs without using SE methods
  - Many companies have drifted into software development as their products and services have evolved
  - They do not use SE methods in their everyday work
  - Consequently, their SW is often more expensive and less reliable



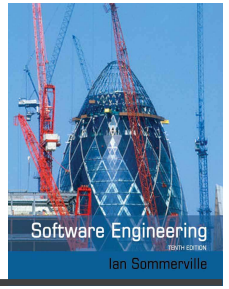
Professional software development

Frequently asked questions about software engineering



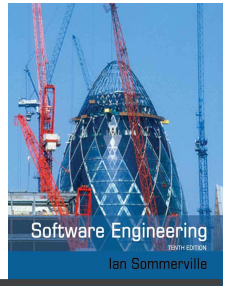
Question	Answer
What is software?	
What are the attributes of good software?	
What is software engineering?	
What are the fundamental software engineering activities?	
What is the difference between software engineering and computer science?	
What is the difference between software engineering and system engineering?	

Frequently asked questions about software engineering



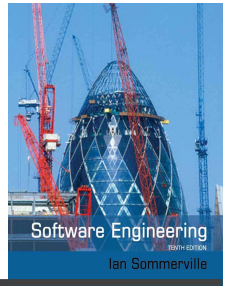
Question	Answer
What is software?	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
What are the attributes of good software?	Good software should deliver the required functionality and performance to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

Frequently asked questions about software engineering



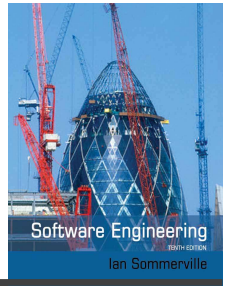
Question	Answer
What are the key challenges facing software engineering?	
What are the costs of software engineering?	
What are the best software engineering techniques and methods?	
What differences has the web made to software engineering?	

Frequently asked questions about software engineering



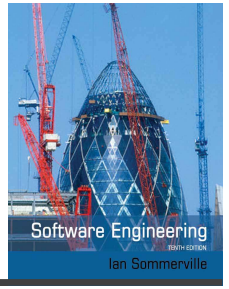
Question	Answer
What are the key challenges facing software engineering?	Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software.
What are the costs of software engineering?	Roughly 60% of software costs are development costs, 40% are testing costs. For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

Software engineering



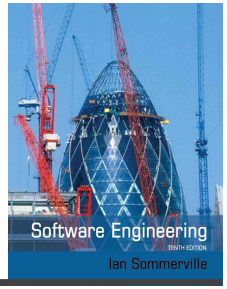
- Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.
- Engineering discipline
  - Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.
- All aspects of software production
  - Not just technical process of development. Also project management and the development of tools, methods etc. to support software production.

Importance of software engineering



- We need to be able to produce reliable and trustworthy systems economically and quickly.
- It is usually cheaper, in the long run, to use SE methods and techniques for software systems
- Costs
  - For most types of system, **the majority of costs are the costs of changing the software** after it has gone into use.

Software products



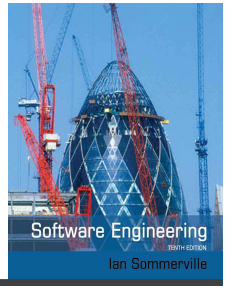
✧ Generic products

- The specification of what the software should do is owned by the software developer and decisions on software change are made by the developer.
- Stand-alone systems that are marketed and sold to any customer who wishes to buy them.
- Examples – Apps, graphics programs, project management tools; CAD software; software for specific markets such as appointments systems

✧ Customized products

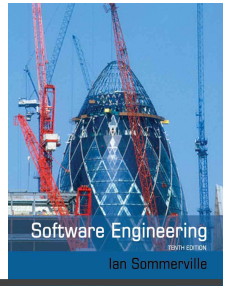
- The specification of what the software should do is owned by the customer for the software and they make decisions on software changes that are required.
- Software that is commissioned by a specific customer to meet their own needs.
- embedded control systems, air traffic control software, traffic monitoring systems.

Essential attributes of good software



Product characteristic	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 utilisation, 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.

Other issues that affect software



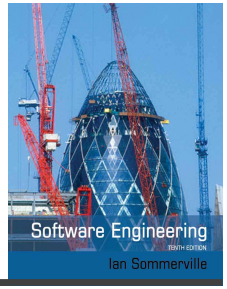
✧ Heterogeneity

- Increasingly, systems are required to operate as distributed systems across networks that include different types of computer and mobile devices.

✧ Business and social change

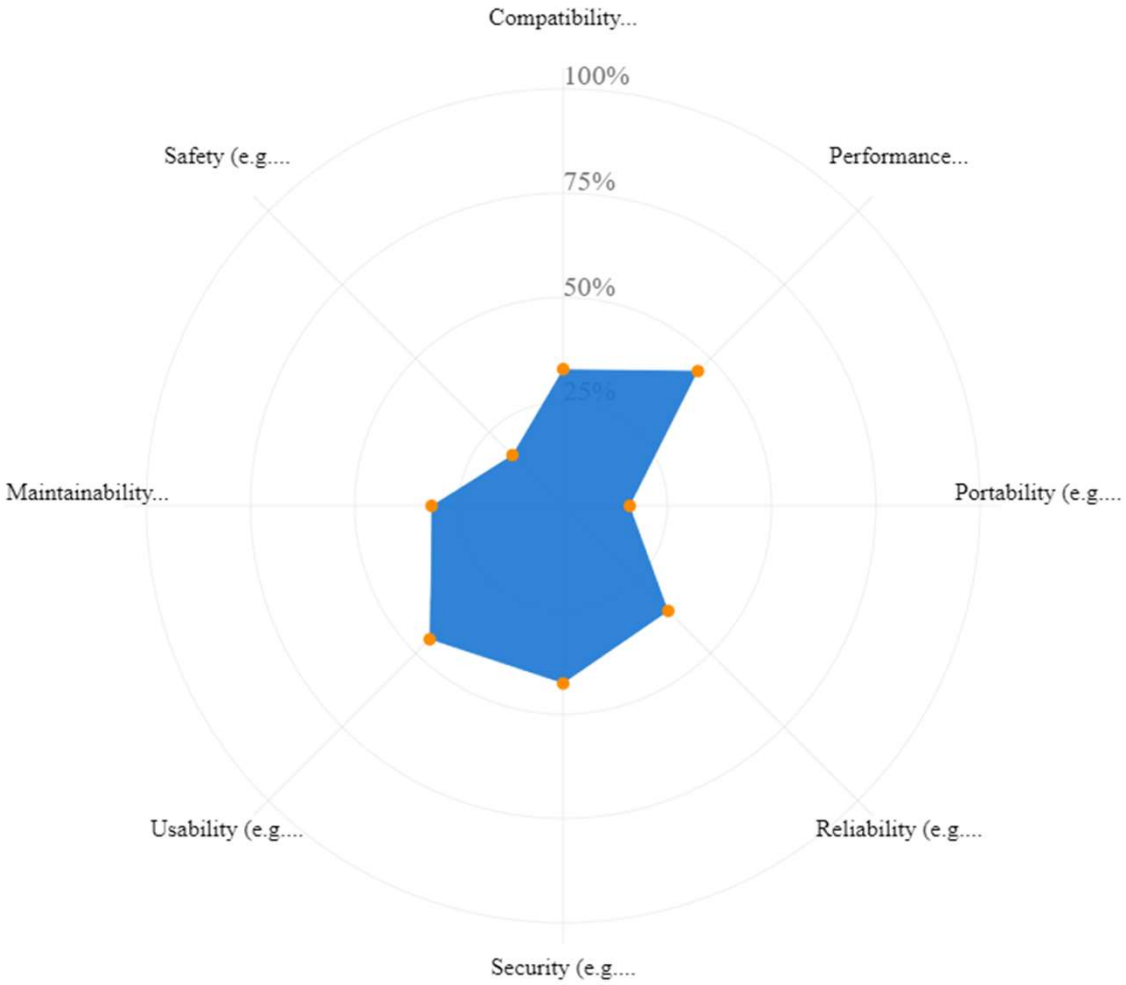
- Business and society are changing incredibly quickly as emerging economies develop and new technologies become available. They need to be able to change their existing software and to rapidly develop new software.

Other issues that affect software

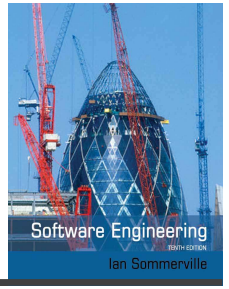


- Security and trust
  - As software is intertwined with all aspects of our lives, it is essential that we can trust that software.
- Scale
  - Software has to be developed across a very wide range of scales, from very small embedded systems in portable or wearable devices through to Internet-scale, cloud-based systems that serve a global community.

Napire: Considered NFRs

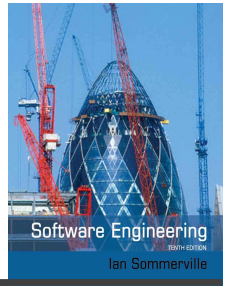


Software process activities



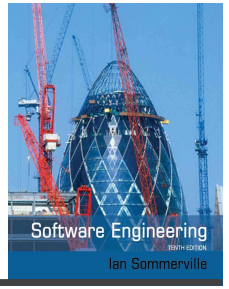
- Software specification, where customers and engineers define the software that is to be produced and the constraints on its operation.
- Software development, where the software is designed and programmed.
- Software validation, where the software is checked to ensure that it is what the customer requires.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

Software engineering diversity



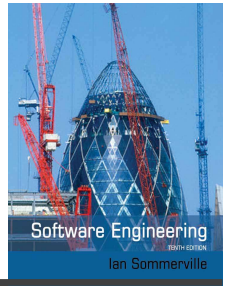
- There are many different types of software system and there is no universal set of software techniques that is applicable to all of these.
- The software engineering methods and tools used depend on the type of application being developed, the requirements of the customer and the background of the development team.

Application types



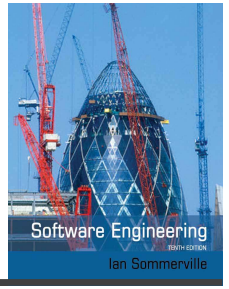
- Stand-alone applications
  - These are application systems that run on a local computer or device, such as a PC, mobile. They include all necessary functionality and do not need to be connected to a network.
- Interactive transaction-based applications
  - Applications that execute on a remote computer and are accessed by users from their own devices. These include web applications such as e-commerce applications, social networks.
- Embedded (cyber-physical) 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



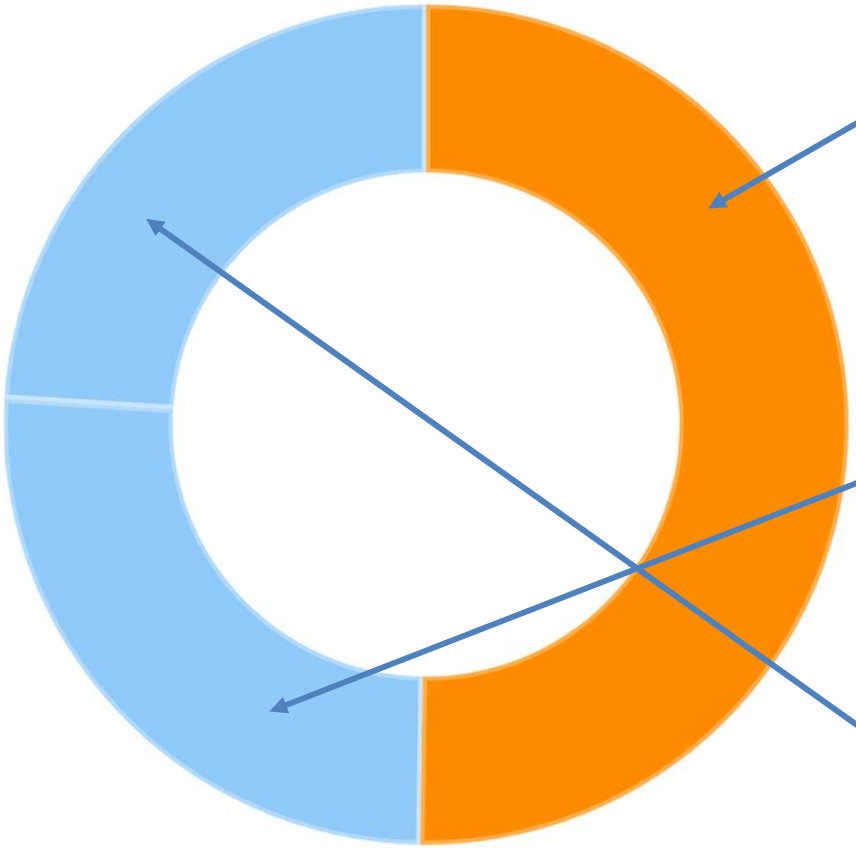
- Batch processing systems
  - These are business systems that are designed to process data in large batches. They process large numbers of individual inputs to create corresponding outputs.
- Entertainment systems
  - These are systems that are primarily for personal use and which are intended to entertain 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.
  - Systems of systems, while still being investigated predominantly in the defense sector, is also seeing application in such fields as national air and auto transportation and space exploration.
  - Other applications where it can be applied include health care, and energy management

System classes (Napire)



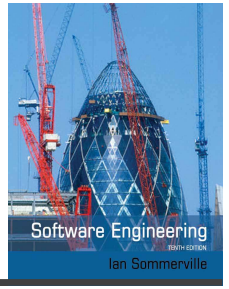
01. Business information systems (50%)

02. Hybrid of both software-intensive en

embedded systems and business information systems (26%)

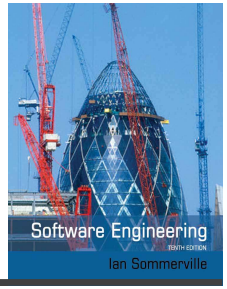
03. Software-intensive embedded system (24%)

Software engineering fundamentals



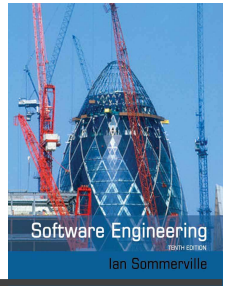
- Some fundamental principles apply to all types of software system, irrespective of the development techniques used:
  - Systems should be developed using a managed and understood development process. Of course, different processes are used for different types of software.
  - Dependability and performance are important for all types of system.
  - Understanding and managing the software specification and requirements (what the software should do) are crucial
  - Where appropriate, you should reuse software that has already been developed rather than write new software.

Cloud/Internet software engineering



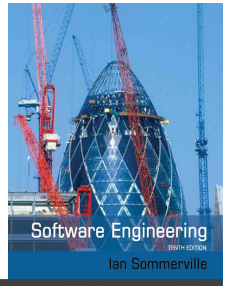
- The Cloud/Web is now a platform for running application and organizations are increasingly developing cloud/web-based systems rather than local systems.
- Cloud/Web services allow application functionality to be accessed over the web.
- Cloud computing is an approach to the provision of computer services where applications run remotely on the ‘cloud’.
  - Users do not buy software but pay according to use.

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 apply to web-based software in the same way that they apply to other types of software system.

Cloud/Web software engineering



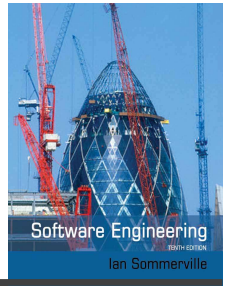
✧ Software reuse

- Software reuse is the dominant approach for constructing cloud/web-based systems. When building these systems, you think about how you can assemble them from pre-existing software components and systems.

✧ Incremental and agile development

- Cloud/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.

Web software engineering

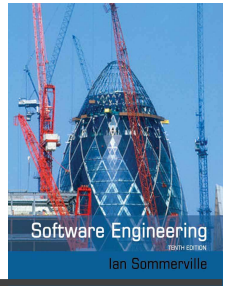


✧ Service-oriented systems

- Software may be implemented using service-oriented software engineering, where the software components are stand-alone web services.

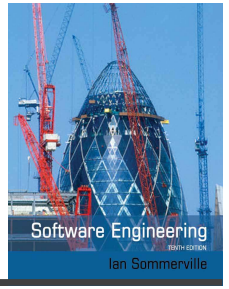
✧ Rich interfaces

- Interface development technologies such as AJAX and HTML5 have emerged that support the creation of rich interfaces within a web browser.



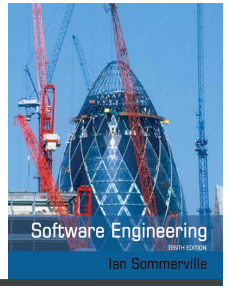
Software engineering ethics

Software engineering ethics



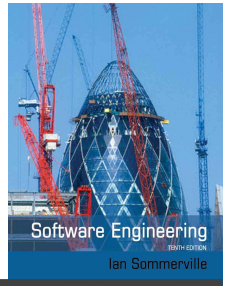
- Software engineering involves wider responsibilities than simply the application of technical skills.
- Software engineers must behave in an honest and ethically responsible way **if** they are to be respected as professionals.
- Ethical behaviour is more than simply upholding the law but involves following a set of principles that are morally correct.

Issues of professional responsibility



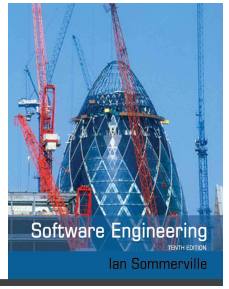
- Confidentiality
  - Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
- Competence
  - Engineers should not misrepresent their level of competence. They should not knowingly accept work which is out with their competence.

Issues of professional responsibility



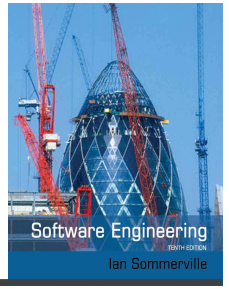
- Intellectual property rights
  - Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.
- Computer misuse
  - Software engineers should not use their 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).

ACM/IEEE Code of Ethics



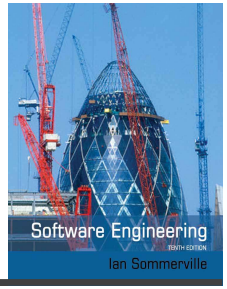
- The professional societies in the US have cooperated to produce a code of ethical practice.
- Members of these organisations sign up to the code of practice when they join.
- The Code contains eight Principles related to the behaviour of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession.

Rationale for the code of ethics



- *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.*

The ACM/IEEE Code of Ethics



Software Engineering Code of Ethics and Professional Practice

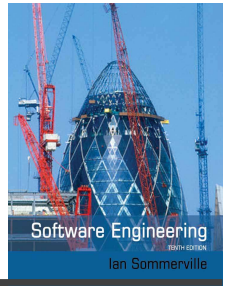
ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices

PREAMBLE

The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code.

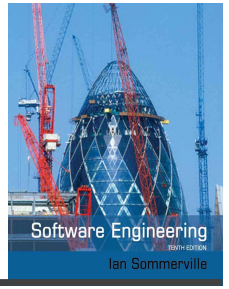
Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the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

Ethical principles



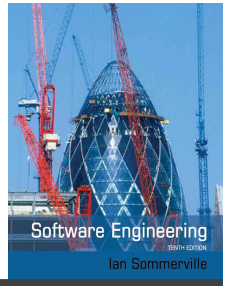
1. PUBLIC - Software engineers shall act consistently with the public interest.
2. CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
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.

Ethical dilemmas



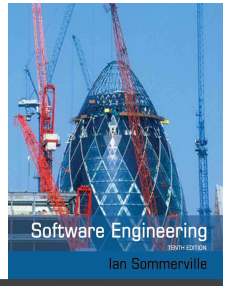
- Disagreement in principle with the policies of senior management.
- Your employer acts in an unethical way and releases a safety-critical system without finishing the testing of the system.
- Participation in the development of military weapons systems or nuclear systems.

Key points



- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Essential software product attributes are maintainability, dependability and security, efficiency and acceptability.
- 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.

Key points



- There are many different types of system and each requires appropriate software engineering tools and techniques for their development.
- The fundamental ideas of software engineering are applicable to all types of software system.
- Software engineers have responsibilities to the engineering profession and society. They should not simply be concerned with technical issues.
- Professional societies publish codes of conduct which set out the standards of behaviour expected of their members.