1

Introduction

ObJectives

The objectilles of this chapter are to introduce software engineering and to provide a framework for understanding the rest of the book. 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;

~ have been introduced to three systems, of dlff'erent types, that will be used as examples throughout the book.

Contents

\Vc can't TUn the modem world without software. National infrastructures and utilitics arc controlled by computer-based systems and most electrical product, 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. Therefore, software engineering is essential for the functioning of national and international societies.

Software systems arc abstract and intangible. TIley are not constrained by the properties of materials, governed by physical laws, Or by manufacturing processes. This simplifies software engineering, as ihere are no natural limits (o the potential of software. However, because of the lack of physical constraints, software systems can quickly become extremely complex, difficult LO understand, and expensive to change.

There arc many different types of software systems, from simple embedded systerns to complex, worldwide information systems. It is pointless to look for universal notations. methods, Or techniques for software engineering because different types of software require different approaches. Developing an organizational information system is completely different from developing a controller for a scientific inSITUmcnt, Neither of these systems has much in common with a graphics-intensive computer game. All ofthese applications need software engineering; they do not all need the same software engineering techniques.

There arc still many n..epons of software projects going wrong and 'software failures'. SOf1\VarC engineering is criticized as inadequate for modern software development. However, in my view, many of these so-called software failures arc a consequence of (\\'o factors:

I. *Increasing demands* As new software engineering techniques help us to build larger, more complex systems, the demands change. Systems have to be buill and delivered more quickly; larger, even more complex systems arc required; systems have to have new capabilities that were previously thought to be impossible. Existing software engineering methods cannot cope and new software engineering techniques have to be developed to meet new these new demands.

2. *UJu"expectations* It is relatively easy to write computer programs without using software engineering methods and techniques. Many companies have drifted into software development as their products and services have evolved. They do not usc software engineering methods in their everyday work. Consequently, their software is often mote expensive and 'less reliable than it should be. \\le need better software engineering education and training 10 address this problem.

Software engineers can be rightly proud of their achievements. Of COurse we still have problems developing complex software but. without software engineering, \\'C would not have explored space, would not have thc Imcrnet Or modern tclccommunications. All forms of travel would be JOOte dangerous and expensive, Software enginccring has conmbuted a great deal and I am convinced that ils contributions in the 21 S1 century wiII be even greater,

() History of software engineering

The notion of 'software engineering' was first proposed in 1968 at a conference held to discuss what was then called the 'software crisis' (Naur and Randell, 1969). It became dear that individual approaches to program development did not scale up to large and complex software systems. These were unreliable, cost more than expected} and were delivered late.

Throughout the 19705 and 19805, a variety of new software- engineering techniques and methods were developed, such as structured programming.. information hiding and object-oriented development. Tools and standard notations were developed and are now extensively used.

http:/{www.SoftwareEngineering-9.comjWebjliistory/

Professional software development

LoL' of people write programs. People in business write spreadsheet programs to simplify their jobs. scientists and engineers write programs to process their cxperimental data) and hobbyists write programs for their own interest and enjoyment. However, the vast majority of software development is a professional activity .vhcre software is developed for specific business purposes, for inclusion in other devices, Or as software products such as information systems, CAD systems, etc. Professional software, intended for usc by someone apart from its developer, is usually developed by teams rather than individuals. It is maintained and changed throughout its life.

Software engineering is intended to support professional software development, rather than individual programming. II includes techniques that support program specification, design, and evolution, none of which are normall y relevant for personal software development. To help you to get a broad view of what software engineering is about, J have summarized some frequently asked questions in Piguro 1.1.

Many people think that software is simply another word for computer programs. However, when we arc talking about software engineering, software is not just the programs themselves but also all associated documentation and configuration data that is required to make these programs operate correctly. A professionally developed software system is often more than a single program, The system usually consisrs of a number of separate programs and configuration files that are used to set up these programs, 11may include system documentation, which describes the structure of the system; user documentation, which explains how to use the system, and websites for users LOdownload recent product information.

This is One of the important differences between professional and amateur software development, If you arc writing a program for yourself, no One else \\';11usc it and you don't have co worry about writing program guides, documenting the program design, etc. However, if you arc 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.

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 coneemed 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 softvvare 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 concemed with all aspects of computer-based systems development including hardwa,re,software, and process engineering. Software engineering is part of this more general process.
What are the key challenges facing software engineering?	Coping with increasing diversity,demands for reduced deliverytimes, and developing trustworthy software.
What are the costs of software engineering?	Roughly 60% of software costs are development costs; 40% are testing costs. For rustom 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 ate 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 anatyzable specification to be developed. Youcan't, therefore, say that one method is better than another.
What differences has the Web made to software engineering?	lIle Web has led to the availabilityof software servioes and the possibility of developing highly distributed service-based ~ems. Web·based systems development has led to important advances in programming languages and software reuse.

Figur. 1.1 Frequently asked questions about software

Software engineers arc concerned with developing software products (i.c., software which can he sold to a customer). There arc two kinds of software products:

I. *Generic products* These arc stand-alone systems that are produced by a development organization and sold On the open market to any CU:::lOmCwho is able co buy them. Examples of this type of product include software for PCs such as databases. word processors, drawing packages, and project-management tools. It also includes so..ca11edvertical applications designed for some specific pur.. pose such as library infonnation systems, accounting systems, or systems for maintaining dental records.

2. *Customized (or bespoke) products* These are systems that 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 .

An important difference between these types of software is that) in generic products, the organization that develops the software controls the software specification. For CLL~ torn products, the specification is usually developed and controlled by tbe organization that is buying the software. The software developers must work to that specification.

However, the distinction between these system product types is becoming increasingly blurred. More and more systems are now being built with a generic product as a base, which is then adapted to suit the requirements of a 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 COIn. pan)' by incorporating information about business rules and processes, reports required, and so on.

When we talk about the quality of professional software, we have to take into account that the software is used and changed by people apart from its developers. Quality is therefore not just concerned with what the software does. Rather, it has to include the software's behavior while it is executing and the structure and organization of the system programs and associated documentation. This is reflected in so-called quality or non..functional software attributes. Examples of these attributes axe the soft.. ware'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 obvi.. ously depends on its application. 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 1.2, which J believe are the essential characteristics of a professional software system.

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 maintain.. ing the system after it has gone into use. In this definition. there are two key phrases:

I. *Engineering discipline* Engineers make things work. They apply theories, meth.. ods, and tools where these are appropriate. However, they use them selectively

Maintainability	Softvvare should be WTitten 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
Dependabilityand security	Software dependability indudes a range of characteristics includingreliability,security,and safety.Dependablesoftware should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damagethe system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cydes. Efficiency therefore includes responsiveness, processing time, memory utilization, etc.
Acceptability	Softwaremust be acceptableto the type of users forwhichit is designed. This means that it must be understandable, usable, and compatiblewith other systemsthat they use.

FigureT.2 Essential attributesof good softvvare

and always try to discover solutions to problems even when there are no applicable theories and methods. Engineers also recognize [hat they must work to organizational and financial constraints so they look fOT solutions within these constraints.

2.. *All aspects ofsoftware production* Software engineering is not just concerned with the technical processes of software development, It also includes activities such as software project management and the development of tools, methods, and theories to support software production.

Engineering is about getting results of the required quality within the schedule and budget This often involves making compromises=-cnginecrs cannot be perfcctionists. People ",'filing programs for themselves, hO\A'CVCT,can spend as much time as they wish On the program development,

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 fot a set of circumstances So 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.

Software engineering is important for two reasons:

1. More and more, individuals and society rely on advanced software systems. We need LO be able to produce reliable and trustworthy systems economically and quickly.

2. It is usually cheaper, in [he long run, to Usc software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project, For most types of systems, the majority of costs arc the costs of changing the software after it has gone into usc.

'The systematic approach that is used in software engineering is somcumcs called a software process. A software process is a sequence of activities that leads to the production of a software produce. 'There are four fundamental activities that arc corn-mon to all software processes. 111CSC activities arc:

I. Software specification, where customers and engineers define the software that is (0 be produced and the constraints On its operation.
2. SOfL\VarC 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.

Different types of systems need di lfcrent development processes. For example. real-time software in an aircraft bas to be completely specified before development begins. In c-commerce systems, the specification and the program are usually devel-oped together. Consequently, these generic activities may be organized in different ways and described at different levels of detail depending on the type of software being developed. I describe software processes in more detail in Chapter 2.

Software engineering is related to both computer science and systems engineering:

I. Computer science is concerned with the theories and methods that underlie com-putcrs 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. Computer science theory, however, is often most applicable to relatively small programs. Elegant theories of computer science cannot always be applied to large, complex problems that require a soft-ware solution.
2. System engineering is concerned with all aspects of the development and evo-lution of complex systems where software plays a major role. System engineer-ing is therefore concerned with hardware development, policy and process design and system deployment, as well as software engineering, System cngi-nccrs are involved in specifying the system, defining irs 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, ctc.).

As I discus s.in the next section, there arc many different types of software ..There is no universal software engineering method Or technique that is applicable for all of these. However, there art three general issues that affect many different types of software:

1. *Heterogeneity* Increasingly, systems are required to operate as distributed systems aCrOSS networks that include different types of computer and mobile devices. A~ well as running On general-purpose computers, software may also have to execute On mobile phones. You often have to integrate new software with older legacy sys- terns written in different programming languages. The challenge here is to develop techniques for building dependable software that is flexible enough to cope with this heterogeneity.

2. *Businessand social chang«* Business and society are changing incredibly quickly as emerging economics develop and new technologies become availablc. They need to be able cochange their existing software and to rapidly develop new soft- ware. Many traditional software engineering techniques arc 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 value to irs customers is reduced.

3. *Security and trust* As software is intertwined with aU aspects of Our lives, it is essential Chat we can trust that software. This is especially true for remote soft.. ware systems accessed through a web page or web service interface. ~\/c have to make SLUt: that malicious users cannot attack our software and that information security is maintained.

Or course, these are not independent issues, For example, it may be necessary to make rapid changes to a legacy system to provide it with a web service interface. To address these challenges we will need new tools and techniques as well as innovative ways of combining and USingexisting software engineering methods.

1.1.2 Software engineering diversity

Software engineering is a systematic approach to the production of software that takes into account practical cost, schedule, and dependability issues, as well as the needs of software customers and producers. How this systematic approach is acru- ally implemented varies dramatically depending 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 and techniques that arc suit- able for all systems and all companies. Rather, a diverse set of software engineering mctheds and tools has evolved over [he past 5"0years.

Perhaps the most significant factor in determining which software engineering methods and techniques arc most important is the type of application that is being developed. There are many different types of application including:

1. *Stand-atone applications* These arc application systems (hat run On a local corn- purer, sucn as a PC. They include all necessary functionality and do not need 10

be connected to a network, Examples of such applications are office applica-[ions on a PC, CAD programs, photo manipulation software, ere,

2. *Interactive transaction-based applicasions* These are applications thai execute On a remote computer and that arc accessed by users from their O'A'JIPCsOr terminals. Obviously. these include web applications such as c-cornmcrce appli-cations where you can interact with a remote system LO buy goods and services. This class of application also includes business systems, where a business provides access to irs 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.

3. *Embedded control systems* These arc software control systems that control and manage hardware devices. Numerically, there arc probably more embedded sys-tems than any other type of system. 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 lOcontrol the cooking process.

4. *Batch processing systems* These are business systems that are designed to process data in large batches. They process large numbers of individual inputs co create corresponding outpuus. Examples of batch systems include periodic billing systems, such as phone billing systems, and salary payment systems.

5. *Entertainment systems* These are systems that arc primarily for personal usc and. which arc intended to entertain the user. Most of these systems arc games of one kind Oranother. The quality of the IL~CTinteraction offered is the most important distinguishing characteristic of entertainment systems.

6. *Systems for modeling and. simulation* These arc systems Charare developed by scientists and engineers to model physical processes Or situations, which include many, separate, interacting objects, These arc often computationally intensive and require high-performance parallel systems for execution.

7. *Data collection systems* These arc systems that collect data from their environ-mcnt using a set of sensors and send that data to other systems for processing. The software has to interact with sensors and often is installed in a hostile cnvi-ronmcnt such as inside an engine or in a remote location.

8. *Systems of systems* These arc systems that are composed of a number of other software systems. Some of these may be generic software products> such as a spreadsheet program. Other systems in the assembly may be specially wriucn for that environment.

Of course, the boundaries between these system types are blurred. If you develop a game for a mobile (cell) phone, you have to take into account the same constraints (power, hardware interaction) as the developers of the phone software. Batch pro-cessing systems arc often used in conjunction with web-based systems. FOiexample, in a company, travel expense claims may be submitted through a web application but processed in a batch application for monthly payment.

You use different software engineering techniques for each t}'PC of system because tbc software has quite different characteristics. For example, an embedded control system in an automobile is safety-critical and is burned into ROM when installed in the vehicle. 1 lis therefore very expensive LOchange. Such a system needs very extensive verification and validation so that the chances of having to recall cars after sale to fix software problems arc minimized. User interaction is minimal (or perhaps nonexistent) so there is no need to usc a development process chat relies on user interface prototyping.

POTa web-based system, an approach based on iterative development and delivery may be appropriate. 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 chat each system can be separately developed

Ncvcnhclcss, there are software engineering fundamentals that apply to all types of software system:

1. They should be developed using a managed and understood development process. TI,e organization developing the software should plan the development process and have clear ideas of what will be produced and when iLwill be com-plctcd. Of course, different processes are used for different types of software.

2. Dependability and performance are important for all types of systems. Software should behave as expected, without failures and should be available for usc 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 reSOurCCS.

3. Understanding and managing the software specification and requirements (what uic software should do) are important. You have to know what different customers and users of the system expect front it and you have to manage their expectations so that a useful system can be delivered within budget and to schedule.

4. 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 fundamental notions of process, dependability, requirements, management, and reuse arc important themes of this book. Different methods reflect them in differcnt ways but they underlie all professional software development.

You should notice that these fundamentals do nOLcover implementation and programming.I don't cover specific programming techniques in this book because these vary dramatically from One type of system to another. For example, a scripting language such as Rub)' is used for web-based system programming but would be completely inappropriate for embedded systems engineering.

1.1.3 Software engineering and the Web

The development of 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 Irorn within an organization. Around 2000, the Web started to evolve and mOTC and more functionality was added to browsers. This meant Chat 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 [he development of a vast range 0ľ new system products that delivered innovative serv-ices, accessed over the Web. These are often funded by adverts !hat arc displayed on the user's SCreen and do not involve direct payment from users.

As well as these system products, (he development of web browsers that could Tun SInal) programs and do some local processing led to an evolution in business and organizational software. Instead of writing software and deploying iLOn users' PCs, the software was deployed On a web S(,'T\'Cr. 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 IISCr interface development is particularly expensive. Consequently, wherever it has been possible to do so, Irian), businesses have moved to web-based interaction with company software systems.

The next stage in the development of web-based systems was the notion of web services. Web services are software components Chat deliver specific, useful function-alit)' and which are accessed over the web. Applications arc constructed by integrating these web services, which may be provided by different companies. In principle, this linking can be dynamic so that an application may use different web services each time thai it is executed, I cover this approach to software development in Chapter 19.

In the last ICw years, the nouon of 'software as a service' has been developed, It has been proposed that software will not normally run On local computers but will Tun On 'computing clouds' thai arc accessed over the Internet If you lL~a service such M web-based mail, you arc using a cloud-based system. A computing cloud is a huge number of Iinkcd computer systems that is shared by many users. Users do not buy software but pay according to how much (he software is used or 3rC given tree acccss in return for watching adverts char arc displayed On their SCreen.

The advent of the web, therefore, has led to a signi (icant change in the way that business software is organized. Before the web, business applications were mostly monolithic, single programs funning On Single computet's or computer clusters. Communications were local, within an organization. Now, software is highly distrib.. utcd, sometimes aCrOSS the world. Business applications arc not programmed from scratch but involve extensive reuse of components and programs.

TIlls radical chango in software organization has, obviously, ted to changes in the ways that web-based systems 31'e engineered. Pol' example:

1. Software rCUW has become the dominant approach fOT constructing web-based systems. When building these systems. you think about how you can assemble them from pre.existing software components and systems.

2. II is now generally recognized that it is impractical to specify all the requirements for such systems in advance. Web-based systems should be developed and delivered incrementally.

3. User interfaces are constrained by thc capabilities of web browsers. Although rcchnologics such as AJAX (Holdener, 20(8) mean thai rich interfaces can be created within a web browser, these technologies are still difficult to usc. Web forms with local scripting arc more commonly used Application interfaces on web-based systems are often poorer than the specially designed user interfaces OnPC system prOdUCL';.

TL1c fundamental ideas of software engineering, discussed in the previous section, apply to web-based software in the same way that they apply to other types of soft-'A/arcsystem. Experience gained with large system development in the 20th century is still relevant to web-based software.

Software engineering ethics

Like other engineering disciplines. softy/arc engineering is carried our within a social and legal framework that limits the freedom of people working in that area. A.~ a software engineer, you must accept that your job involves wider responsibilities than simply [he application of technical skills. You must also behave in an ethical and morally responsible way if you arc to be respected a, a professional engineer,

II goes without saying that you should uphold normal standards of honesty and integrity. You should not Usc your skills and abilities to behave in a dishonest way or in a way that will bring disrepute to the software engineering profession. However, there arc area'; where standards of acceptable behavior are not bound by laws but by the more tenuous notion of professional responsibility. Some of these are:

I. *Confidentiality* You should normally respect the confidentiality of' your employers Or clients irrespective of whether Or not a formal confidcntialiry agreement bas been signed.

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

3. *Intellectual property rights* You should be aware of local laws governing the LL';C (if intellectual property such a, patents and copyright. You should be careful to ensure that the intellectual property of employers and clients is protected,

4. *Computer misuse* You should not usc your technical skills LOmisuse other people's computers. Computer misuse ranges from relatively trivial (game playing On an employer's machine, say) to extremely serious (dissemination of viruses Or oilier malware).

Software Engineering Cod. of EtI1icsand Promsional Practice

.ACM/IEEEMCS 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 induded in the fun version give examples and details of how these. aspirations change the way we act as software engineering professionals. VVithout 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, softwine engineers shall adhere to the following Eight Principles:

1. PUBLIC - Software engineers shall act consistently with the public interest.
2. CLIENT AND E~IPLOYER - 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 mod; ficat; ons meet the. highest profes st ona 1 standa rds poss ; 'ble.
4. JUDG~IENT - Software engineers shall maintain integrity and independence in their professional judgment.
5. r~AIiAGEME!lT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. PROFESSION - Soft\>lareengi neers shall advance the integrity and reputati on o-f the profession consistent with the public interest.
7. COLLEAGUES - Soft\>lareengineers shall be fair to and supportive of their co 11eagues.
8. SELF - Soft,'Ine en-gineers sha 11 part ictpate in 1ifelong 1ea rnin9 regardi n9 the practice of their profession and shall promote. an ethical approach to the. practice of the profession.

Figure 1.3 The ACM/lEEE Code of Ethics (-~ IEEE/ACM t999)

Professional societies and institutions have an important role to play in setting ethical standards. Organizations such as the ACM, lite lBEE (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 arc gcn-crally concerned with fundamental ethical behavior.

Professional associations, notably thc ACM and thc lEEE, have cooperated to produce a joint code of ethics and professional practice. This code exists in both a short form, shown in Figure 1.3, and a longer form (Gottcrbam et al., 1999) that adds detail and substance to the ShOtLC version. The rationale behind this code is summa-rizcd in the first two paragraphs of the longer form:

> *Computers have a central and growing role in. (.'()lnl11erce- industry, government, medicine, education, ensertainnumt and society at large. Softwar« engineers are those who contribute hy direct participation. or by teaching, to the analysis, spec-ification, design; d.evelopnlcnt, certification, maintenance and testing ofsoftwar«*

systems. Because of their roles in developing software systems, software engineers have significemt 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, tha: their ejJfJrlS will be used for good) software engi.. neers must commit themselves to making software engineering it beneficial. and respected profession: In accordance witt: that commitment, software engineers shall adhere fa the following Code 01 Ethics and Professiona; Practice.

The Code contains eight Principles related to the behaviour of atUll/.ecisiolls made by professional software engineers, including practitioners, educators, 11lanagers, supervisors and policy makers, as well as trainees and students of the profession. The Principles identify the ethically responsible relationships in which individuals, groups, lind organizations participate lind the primary obligations within these relationships. The Clauses of each Principle are illustrations of sonte oj' the obligations included in these relationships. These obli.. gutions are founded ill. thi! software engineer's humanity, in special ca,re owed to people affected by the work ofsoftware engineers, and the unique elements of the practice oj' software engineering. Tire COlle prescribes the..:e as ohliga.. tions of lUt)'OfU! claiming to be or aspiring to be a software engineer.

In any situation where different people have different views and objectives you are likely to bc 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 01'the disagreement. Lit best to argue a case for your position from within the organization or to resign in principle? II' you fed chat there are problems with a software project, when do you reveal these to management? If YOU discuss these while they arc just a suspicion. you may be overreacting to a situation; if you leave it lOO late, it may be impossible to rcsolve the difficulties.

Such ethical dilemmas fare all of us in our professional lives and, "fortunately,in most cases the)' arc either relatively minor Or can be resolved without too much difficulry. 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 M 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 safcry-cruical system and, because of time pressure, falsities the safely validation records. Is the engineer's responsibility to maintain confidentiality Or LO alert the customer or publicize, in some way, that the delivered system may be unsafe?

The problem here is that there arc 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. TI,e system rna)' actually operate safely throughout it~ lifetime. It is also the case that, even when properly validated, the system rna)' fail and cause an accident. Early disclosure of problems may result in damage COthe 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 hero depends entirely On the v1c'ivs of the individuals who arc mvolved, In this case, the potential for damage, the extent of the damage, and the people affected by the damage should influence the decision. *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 Ute 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 10participate 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.

In this Situation, it is important that both employers and employees should make their viC\\IS known to each other in advance. Where an organization is involved in military or nuclear work, they should be able 10specify that employees must be willing to accept any work asslgnmcnt. Equally, if an employee is taken on and makes clear that they do not wish 10 work on such systems, employers should not put pressure On them to do so at some later dale.

The general area of ethics and professional responsibility is becoming more important as software-intensive systems pervade every aspect of work and everyday life. 1 lcan be considered from a philosophical standpoint where the basic principles of ethics arc considered and software engineering ethics arc discussed with reference lOthese basic principles. This is the approach taken by Laudon (1995) and to a lesser extent by Huff and Martin (1995). Johnson's text On computer ethics (200 I) also approaches the topic from a philosophical perspective.

However, I find that this philosophical approach is too abstract and difficult 10 relate to everyday experience. I prefer the more COncrete approach embodied in codes of conduct and practice. I think that ethics arc best discussed in a software engineering context and not as a subject in their own right. In this book, therefore, I do not include abstract ethical discussions but, where appropriate, include examples in the exercises that can be the starting point for a group discussion On ethical issues.

Case studies

To illustrate software engineering concepts, I use examples from three different types of systems throughout the book. The reason why I have nor used a Single case study is lhal one of the key messages in this book is thai software engineering praclice depends On the type of systems being produced. 1 therefore choose an appropriate example when discussing concepts such as safety and dependability, system modeling, reuse, etc.

The three types of systems that I usc M case studies arc:

1. *An embedded system.* This is a system where the software controls a hardware device and is embedded in that device. Issues in embedded systems typically

include physical size, responsiveness, power management, etc. The example 0f an embedded system that 1 CL<icis a software system to control a medical dc'lice.

2. *An information. systCIJI* This is a system whose primary purpose is to manage and provide access to a database of information. Issues in information systems include security, usability; privacy, and maintaining data integrity. The example of an information system that 1use is a medical records system.

3. A *sensor-based dati' collection system.* This is a system whose primary purpose is to COnCCI data from a set of sensors and process that data in some way. The key requirements of such systems arc reliability, even in hostile environmental conditions, and maintainability, The example of a data collection system that 1 use is a wildernesss weather station.

I introduce each of these systems in this chapter, with more information about each of them available On the Web.

1.3.1 An insulin pump control system

An insulin pump is a medical system that simulates rhc operation of the pancreas (an internal organ). The software controlling this system is an embedded system, which collects information from a sensor and controls a pump that delivers a controlled dose of insulin to a user,

People who suffer from diabetes usc the system. Diabetes is a relatively common condition where the human pancreas is unable to produce sufficient quantities of a hormone called insulin. Insulin mcraboliscs glucose (sugar) in the blood. The COn-ventional trcatmcnt of diabetes involves regular injections of genetically engineered insulin. Diabcrics measure their blood sugar levels using an external meter and then calculate the dose of insulin that they should inject.

The problem with this treatment is that the level of insulin required docs not just depend on the blood glucose level but also on the lime of the last insulin injection. This can lead to very low levels of blood glucose (if there is loo much insulin) or very high levels of blood sugar (if there is lOOlittle insulin). Low blood glucose is, in the short term, a more serious condition as it can result in temporary brain malfunctioning and. ultimately, unconsciousness and death. In the long term. however, continual high levels of blood glucose can lead to eye damage, kidney damage, and heart problems.

Current advances in developing miniaturized sensors have meant that it is nO\'lpos-sible to develop automated insulin delivery systems. These systems monitor blood sugar levels and deliver an appropriate dose of insulin when required.Insulin delivery systems like this already exist for the treatment of hospital patients. In the future, it may be pos-sible for many diabetics ro have such systems permanently auachcd lO their bodies.

A soltwarc-controllcd insulin delivery system might work by using a micro-sensor embedded in the patient to measure some blood parameter that is proportional to the sugar level. This is then sent to the pump controller. This controller computes the sugar level and the amount of insulin that is needed. Jt then sends signals to a miniaturized pump to deliver the insulin via a permanently attached needle.

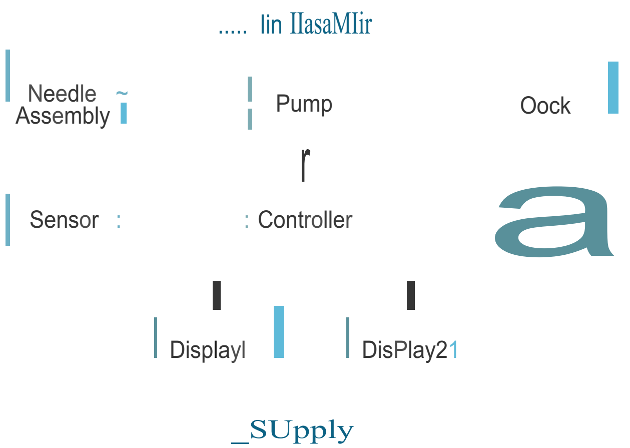

Figure 1.4 Insulin pump hardware

Figure 1.5 Activity model of the insulin pump

Figure 1.4 shows the hardware components and organization of the insulin pump. To understand the examples in this book, all you need CO Know is that the blood sensor measures the electrical conductivity of the blood under different conditions and thaI these values can be related to the blood sugar level. The insulin pump delivers One unit of insulin in response to a single pulse from a controller, Therefore, to deliver I() units of insulin, Ihe controller sends 10 pulses to the pump. Pigure 1.5 is a UML activity model that illustrates how the software transforms an input blood sugar level to a sequence of commands that drive the insulin pump.

Clearly, this is a safety-critical system. If the pump fails 10 operate Or docs not operate correctly, then the user's health may be damaged Or they may fall into a coma because their blood sugar levels arc roo high Or too low, There arc, therefore, two essential high-level requirements that this system must meet:

I. The system shall be available to deliver insulin when required.

2. The system shall perform reliably and deliver the correct amount of insulin to counteract the CLUTCnC level of blood sugar.

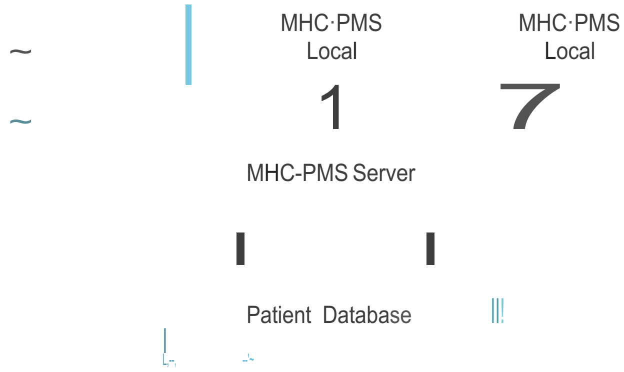

Figure 1.6 The organization of theMHC-PMS

11,e system must therefore he designed and implemented 10ensure thai the system always meers these requirements. More derailed requirements and discussions of how to ensure that the system is safe arc discussed in later chapters.

1.3.2 A patient information system for mental health care

A patient information system 10support menial health care is a medical information system that maintains information about patients suffering from mental health problems and the treatments that they have received. Most mental health patients do not require dedicated hospital treatment but need to attend specialist clinics regularly where they can meet a doctor who has detailed knowledge of their problems. To make it easier for patients 10attend, these clinics arc not just Tun in hospitals. They may also be held in local medical practices Or community centers.

The MHC-PMS (Menial Health Care-Patient Management System) is an information system that is intended for usc in clinics. It makes usc of a centralized database of patient information but has also been designed 10run Ona PC, so thai it may be accessed and used from sites that do nOI have secure network connectivity. Wben the local systerns have secure network access, they IISC patient information in the database but they can download and usc local copies of patient records when they arc disconnected, The system is not a complete medical records system so docs not maintain information about other medical conditions. However, it may interact and exchange data with other clinical information systems, Figure 1.6illustrates the organization of the MHC·PMS.

The lVlHC·PMShas two overall goals:

J. "It) generate management information that allows health service managers to assess performance against local and government targets.

2. To provide medical staff with timely information to support the treatment of patients.

'The nature of mental health problems is such that patients arc oflen disorganized so may miss appointments, deliberately Or accidentally lose prescriptions and medication, forget instructions, and make unreasonable demands On medical staff, Tiley may drop in On clinics unexpectedly. In a minority of cases, they may be a danger to themselves Or to other people. They may regularly change address Or may be homeless on a long-term Or short-tern. basis. Where patients arc dangerous, they may need lO be 'sectioned'<-confined ro a secure hospital for treatment and observation.

Users of the system include clinical staff such as doctors, nurses, and hcauh visi-10rS(nurses who visit people at home to check On their treatment). Nonmedical U.SCrS include receptionists who make appointments, medical records staff who maintain the records system, and administrative staff who generate reports.

The system is used 10 record information about patients (name. address, age. next of kin, ctc.), consultations (date, doctor seen, subjective impressions of the patient, ctc.), conditions, and treatments. Reports arc generated at regular intervals for medical seaff and health authority managers. Typically, reports for medical stall' focus On information about individual patients whereas management reports are anonyrnized and arc concerned with conditions. costs of treatment, CLeo

'The key features of the system arc:

I. *Individual care ltJlnla.genrent* Clinicians can create records for patients, edit the information in the system, view patient history, etc. The system supports data summaries so that doctors who have not previously mer a patient can quickly learn about the key problems and trcarmcnts that have been prescribed.

2. *Patient monitoring* TIle system regularly monitors the records of patients. that arc involved in treatment and issues warnings ifpossible problems arc detected. Therefore, if a patient has not seen a doctor for some lime, a warning may be issued. (me of the most important clements of the moniroring system is to keep track of patients who have been sectioned and lO ensure that the legally required checks arc carried out at the right time.

3. *Administrative reporting* The system generates monthly management reports showing the number of patients treated at each clinic, the number of patients who have entered and left the care system, number of patients sectioned, the drags prescribed and their costs, etc.

Two different laws affect the system. These arc laws On data protection that govern the confidentiality of personal information and mental health laws that govern the toto-pulsory detention of patients deemed to be a danger to themselves Or others. Mental health is unique in this respect as it is the only medical speciality that can recommend the detention of patients against their will. This is subject to vcry strict legislative safeguards. One of the aims of the MHC-PMS is lO ensure that stall' always act in aCCOrdance with the law and lhat their decisions are recorded for judicial review if necessary,

As in all medical systems, privacy is a critical system requirement. It is essential that patient information is confidential and is never disclosed [0 anyone apart from authorized medical stail' and the patient themselves. The MHC-PMS is also a safety-critical

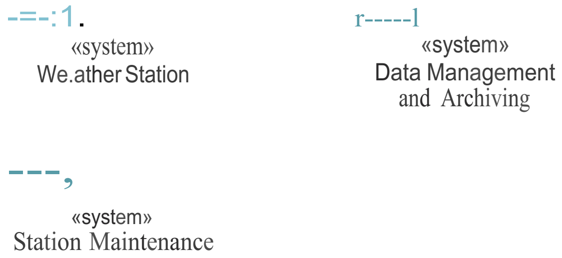

Figure 1.7 The weather station's environment

system. Some mcneal illnesses cause patients to become suicidal or a danger to other people, Wherever possible, the system should warn medical staff about potentially suicidal Ordangerous patients.

The overall design of the system has to take into account privacy and safely requirements. The sysrcm m(L~t be available when needed otherwise safety may be compromised and it rna)' be impossible LO prescribe the correct medication LO patients, There is a potential conflict here-privacy is easiest to maintain when there is only a single copy of the system data. HO\VCVCTt to ensure availability in the event of server failure Or when disconnected from a network, multiple copies of the data should be maintained. I discuss the tradc-offs between these requirements in later chapters,

1.3.3 Awilderness weather station

To help monitor climate change and 10 improve the accuracy of weather forecasts ill remote areas, the government of a country with large areas of wilderness decides to deploy several hundred weather stations in remote areas. These weather stations collcct data from a sec of instruments that measure temperature and pressure, sunshine. rainfall, wind speed, and wind direction.

Wildernc ss weather stations arc part of a larger system (Figure 1.7). which is a weather information system that collects data from weather stations and makes it available to other systems for processing. The SYStCTni iS Figure 1.7 arc:

I. *The weather station system* This is responsible for collecting weather data) carrying Out some initial data processing, and transmiuing it to the daLa management system.

2. *The data management and archiving system* TlJis system COtfCCL<; the data from all of the wilderness weather stations. carries (Jut data processing and analysis, and archives the data in a form that cart be retrieved by other systems, such as weather forecasting systems.

3. *The station. maintenance system.* This system can communicate by satellite with all wilderness weather stations LO monitor the health of these systems and provide reports of problems. 1£ can update the embedded software in these SystCTIIS. ln the event of system problems, this SYStCTII can also be used to remotely control a wilderness weather system.

In Figure 1.7, I have used the UML package symbol to indicate that each system is a collection of components and have identified the separate systems. using the UML stereotype «system». The associations between [he packages indicate there is an exchange of information but, at this stage, then! is no need to define them in any more detail.

Each weather station includes a number of instruments that measure weather parameters such as the wind speed and direction) the ground and air temperatures, the barometric pressure, and the rainfall over a 24-hour period. Bach of these instruments is controlled by a software system that takes parameter readings periodically and manages the data collected from tho instruments.

The weather station system operates by collecting weather observations at frcquent intervals-for example, temperatures arc measured every minute. However, because the bandwidth to the satellite is relatively narrow, the weather SUtLlOn carries out some local processing and aggregation of the data. It then transmits ibis aggrcgated data when requested by the data collection system. If, for whatever reason, it is impossible to make a connection, then the weather station maintains the. data locally until communication can be resumed

Each weather station is battery-powered and must be entirely self-contained=there arc no cxn..nnapower Or network cables available. All communications arc through a rclativcly slow-speed satellite link and the weather station must include some mechanism (solar or wind power) lO charge its batteries. As they arc deployed in wilderness areas, they arc exposed to severe environmental conditions and may be damaged by animals. The station SOft",,3rGis therefore not just concerned with data collectionIt must also:

). Monitor the instruments. power, and communication hardware and report faults to the management system.

2. Manage the. system power, ensuring that batteries arc charged whenever the environmental conditions permit but also that generators arc shut down in potentially damaging weather conditions, such as high wind.

3. Allow for dynamic reconfigurauon where parts of the software arc replaced with new versions and where backup instruments arc switched into the system in the event of system failure.

Because weather stations have to be self-containcd and unattended, this means that the software installed is complex, even though the data collection functionality is fairly simple.

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.

- Professional societies publish codes of conduct that set out the standards of behavior expected of their members.

FURTHER READING

'No silver bullet: Essence and accidents of software engineering'. In spite of its age, this paper is a good general introduction to the problems of software engineering. The essential message of the paper still hasn't changed. (F. P. Brooks, *IEEE Computer,* 20 (4), April 1987.) http://doi.ieeecomputersociety.org/10.1109/MC.1987.1663S32.

'Software engineering code of ethics is approved'. An article that discusses the background to the development of the ACM/IEEE Code ofEthics and that includes both the short and long form ofthe code. *(Comm. ACM,* D. Gotterbarn, K. Miller, and S. Rogerson, October 1999.) http://portal.acm.org/citation.cfm?doidQ31766S.317682.

Professional Issues in Software Engineering. This is an excellent book discussing legal and professional issues as well as ethics. I prefer its practical approach to more theoretical texts on ethics. (F. Bott, A. Coleman, J. Eaton and D. Rowland, 3rd edition, 2000, Taylor and Francis.)

IEEESoftware, March/Apri/2oo2. This is a special issue of the magazlne devoted to the development of Web·based software. This area has changed very quickly so some articles *are* a little *dated* but most are still relevant. *(IEEESoftware,* 19 (2), 2002.)
http://wWW2 .computer.org/ portal/web/software.

'A View of zoth and 21st Century Software Engineering'. A backward and forward look at software engineering from one of the first and most distinguished software engineers. Barry Boehm identifies timeless software engineering principles but also suggests that some commonly used practices are obsolete. (B. Boehm, *Proc. 28th Software Engineering coni,* Shanghai. 2006.)
http://doi.ieeecomputersociety.org/10.114S/113428S·1134288.

'Software Engineering Ethics'. Special issue of IEEEComputer, with a number of papers on the topic. *(IEEE Computer,* 42 (6), June 2009.)

EXERCISES

1.1. Explain why professional software is not just the programs that are developed for a customer.

1.2. What is the most important difference between generic software product development and custom software development? What might this mean in practice for users of generic software products?

1.3. What are the four important attributes that all professional software should have? Suggest four other attributes that may sometimes be significant.

1.4. Apart from the challenges of heterogeneity, business and social change, and trust and security, identify other problems and challenges that software engineering is likely to face in the 21st century (Hint: think about the environment).

1.5. Based on your own knowledge of some of the application types discussed in section 1.1.2, explain. with examples, why different application types require specialized software engineering techniques to support their design and development.

1.6. Explain why there are fundamental ideas of software engineering that apply to all types of software systems.

1.7. Explain how the universal use of the Web has changed software systems.

1.8. Discuss whether professional engineers should be certified in the same way as doctors or lawyers.

1.9. For each ofthe clauses in the ACM/IEEE Code of Ethics shown in Figure 1.3, suggest an appropriate example that illustrates that clause.

1.10. To help counter terrorism, many countries are planning or have developed computer systems that track large numbers of their citizens and their actions. Clearly this has privacy implications. Discuss the ethics of working on the development of this type of system.

REFERENCES

Gotterbam, D., Miller, K. and Rogerson, S. (1999). Software Engineering Code of Ethics is Approved. Comm. ACM, 42 (.0), 102-7.

Holdener, A. T. (2008). *Aiax: The Definitive Guide.* Sebastopol, Ca.: O'Reilly and Associates.

Huff, C. and Martin, C. D. (1995). Computing Consequences: A Framework for Teaching Ethical Computing. Comm. ACM, 38 (.2), 75-84.

Johnson, D. G. (2001). *Computer Ethics.* Englewood Cliffs, NJ: Prentice Hall.

Laudon, K. (1995). Ethical Concepts and Information Technology. Comm. ACM, 38 (12), 33-9.

Naur, P. and Randell, B. (1969). Software Engineering: Report on a Conference sponsored by the NATO Science Committee, Garmisch, Germany. 7th to nth October 1968.