

# Introduction to Software Engineering Chapter One - Part Three

**CMPT 276**

# Objectives

We will complete **Chapter One** of the **textbook** (Sommerville, 10<sup>th</sup> Ed.)

Today

Part 1 Introduction to Software Engineering

Chapter 1: Introduction

Chapter 2: Software processes

Chapter 3: Agile software development

Chapter 4: Requirements engineering

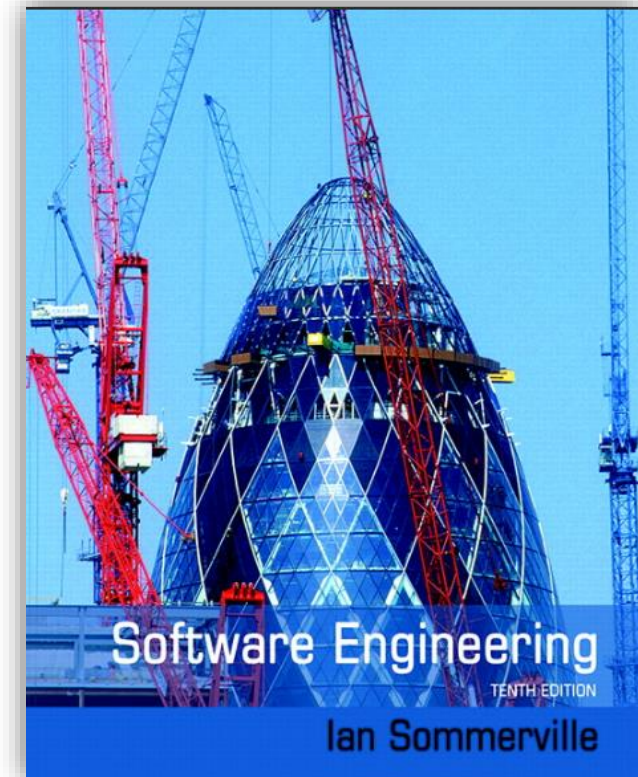
Chapter 5: System modeling

Chapter 6: Architectural design

Chapter 7: Design and Implementation

Chapter 8: Software testing

Chapter 9: Software Evolution





---

# Table of Contents

<b>Part 1</b>	<b>Introduction to Software Engineering</b>	<b>1</b>
<hr/>		
<b>Chapter 1</b>	<b>Introduction</b>	<b>3</b>
1.1	Professional software development	5
1.2	Software engineering ethics	14
1.3	Case studies	17

# Table of Contents

## Part 1 Introduction to Software Engineering

1

### Chapter 1

## Objectives

The objectives 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 different types, that will be used as examples throughout the book.

Today

# Software Engineering Ethics (Section 1.2)

**ETHICS**



 Bill Sourour Follow  
DevMastery.com | Consultant | Teacher  
Nov 13, 2016 · 5 min read

## The code I'm still ashamed of



If you write code for a living, there's a chance that at some point in your career, someone will ask you to code something a little deceitful – if not outright unethical.



Bill Sourout [Follow](#)  
DevMastery.com | Consultant | Teacher  
Nov 23, 2016 · 5 min read

freecodecamp (🔥)

[Follow](#)



[HOME](#) [DEV](#) [DESIGN](#) [DATA](#) | [LEARN TO CODE FOR FREE](#) [Search](#)



Quincy Larson [Follow](#)  
Teacher at <https://FreeCodeCamp.com>  
Mar 3 · 4 min read

## What do Uber, Volkswagen and Zenefits have in common? They all used hidden code to break the law.

If you write code for a living, there's a chance that at some point in your career, someone will ask you to code something a little deceitful – if not outright unethical.



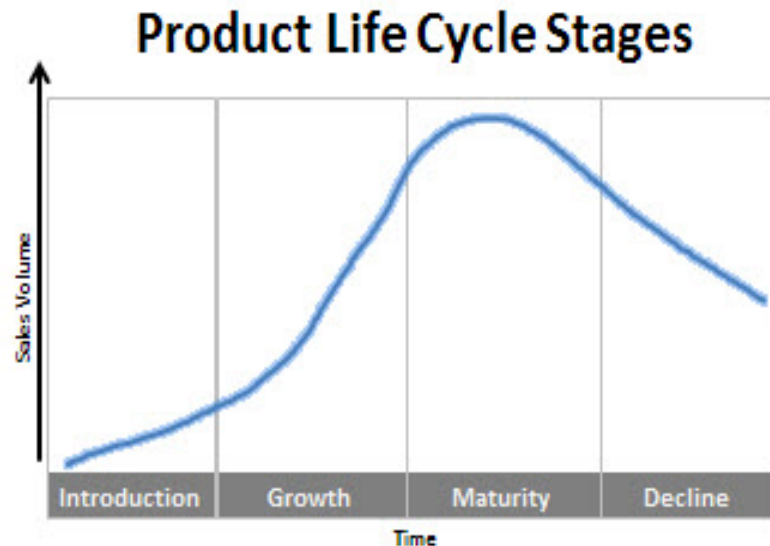
MAY 12, 2017

# Product Life Cycle Stages

*product life cycle stages explained*[Home](#)[Introduction](#)[Growth](#)[Maturity](#)[Decline](#)[Examples](#)[New Product Development](#)

*Is this where SWE is heading?  
It has already happened in the  
discipline of Engineering –  
planned obsolescence.*

## Product Life Cycle Stages



As consumers, we buy millions of products every year. And just like us, these products have a life cycle. Older, long-established products eventually become less popular, while in contrast, the demand for new, more modern goods usually increases quite rapidly after they are launched.

Because most companies understand the different product life cycle stages, and that the products they sell all have a limited lifespan, the majority of them will invest heavily in new product development in order to make sure that their businesses continue to grow.

## Product Life Cycle Stages Explained

See “The Lightbulb Conspiracy”.



# ethic

## Definition of ETHIC

- 1 *plural but sing or plural in constr* : the discipline dealing with what is good and bad and with moral duty and obligation
- 2 **a** : a set of moral principles : a theory or system of moral values <the present-day materialistic *ethic*> <an old-fashioned work *ethic*> —often used in plural but singular or plural in construction <an elaborate *ethics*> <Christian *ethics*>  
**b** *plural but sing or plural in constr* : the principles of conduct governing an individual or a group <professional *ethics*>  
**c** : a guiding philosophy  
**d** : a consciousness of moral importance <forge a conservation *ethic*>
- 3 *plural* : a set of moral issues or aspects (as rightness) <debated the *ethics* of human cloning>

# 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 outwith 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.

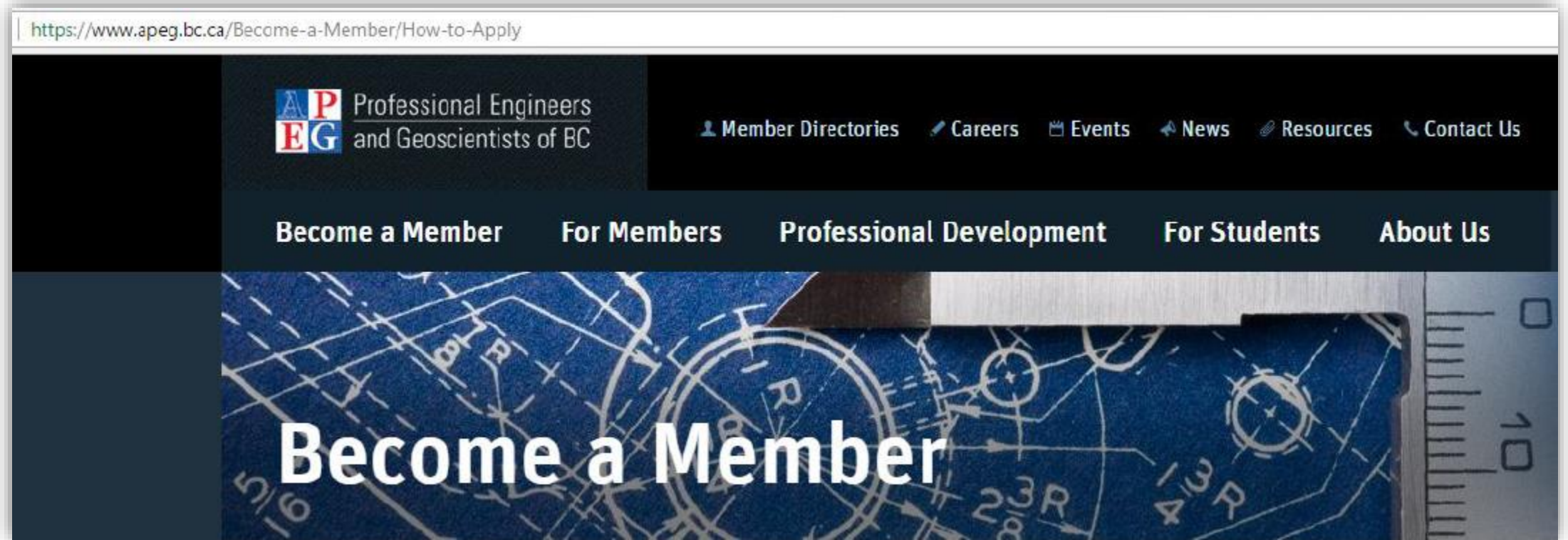
Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the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

# Question:

## Is there a professional designation for SWE?



Professional Designations: P.Eng. and P.Geo.



# Question:

## Is there a professional designation for SWE?

### What is a Professional Engineer?

The practice of professional engineering is defined in Section 1 of the *Professional Engineers Act* and comprises of three tests. Professional engineering is:

1. any act of planning, designing, composing, evaluating, advising, reporting, directing or supervising (or the managing of any such act);
2. that requires the application of engineering principles; and
3. concerns the safeguarding of life, health, property, economic interests, the public welfare or the environment, or the managing of any such act.

If what you do meets all three tests, you are practising professional engineering and must be licensed by the association.

# Question:

## Is there a professional designation for SWE?

A colleague writes:

I think it has been considered in the past. I know that some engineering school have objected to the term "Software Engineer" for people who weren't didn't have engineering degrees. I think a factor is that computing has emerged as more of a vocation program than a professional program. An equivalent argument is that computing is a less mature subject and less in the way of a theoretical foundation is covered in most programs.

Some years ago a Software Engineering Institute (<https://www.sei.cmu.edu>) was established, I believe with the intent that it should be the oversee the Software Engineering field.

In summary, I am not really sure.


# Question:

## Is there a professional designation for SWE?

Refer to *Wikipedia* for a current summary if you are interested.

### Professional certification

From Wikipedia, the free encyclopedia

 [hide] **This article has multiple issues.** Please help [improve it](#) or discuss these issues on the [talk page](#). (*Learn how and when to remove these template messages*)



This article's **use of [external links](#)** may not follow Wikipedia's policies or **guidelines**. (*October 2015*)


This article **contains [embedded lists](#)** that may be better presented **using [prose](#)**. (*November 2015*)

This article **relies too much on [references](#) to [primary sources](#)**. (*November 2016*)


**Professional certification**, **trade certification**, or **professional designation**, often called simply *certification* or *qualification*, is a designation earned by a person to assure qualification to perform a job or task. Not all certifications that use [post-nominal letters](#) are an acknowledgement of educational achievement, or an agency appointed to safeguard the public interest.



Here it is



**ENGINEERS &  
GEOSCIENTISTS**  
BRITISH COLUMBIA

SEARCH... LOGIN

BECOME A MEMBERMEMBER PROGRAMSPRACTICE RESOURCESCOMPLAINTS & DISCIPLINEABOUT

HOME ▶ BECOME A MEMBER ▶ HOW TO APPLY ▶ PROFESSIONAL MEMBERSHIP AND LICENCE ▶ ENGINEER FIRST TIME APPLYING IN CANADA ▶ SOFTWARE ENGINEERING APPLICANTS

▼ How to Apply

- > Professional Membership and Licence
- > Specialist Designation and National Registers
- > Members-in-Training
- > Student Membership
- > Resume Practice Rights and Reinstate Membership
- > Academic Examinations and Syllabi
- > Bridging Pilot Program

## SOFTWARE ENGINEERING APPLICANTS

In British Columbia, software engineering is a discipline of professional engineering and is regulated by Engineers and Geoscientists BC under the *Engineers and Geoscientists Act*, RSBC 1996 c. 116.

The information provided below covers:

- [Regulation of software engineering in BC](#)
- [What is considered software engineering](#)
- [Who needs to be licensed/registered](#)
- [What happens if I don't get registered](#)
- [Where can I find information about registration/licensing](#)

### REGULATION OF SOFTWARE ENGINEERING IN BC

Computer and software engineering have been designated as disciplines of professional engineering since 1988 and 1999, and fall within the definition of professional engineering in the *Engineers and Geoscientists Act*. Currently, there are over 700 professional engineer and engineer-in-training members who are working in software engineering in BC. They work across all industries, including aerospace, manufacturing, mining, transportation, telecommunication, finance, government, and education.

Differences between Canada and the U.S. or foreign jurisdictions can sometimes lead to confusion about registration obligations. In British Columbia, anyone who practices software engineering, or who uses the title "software engineer" (or a similar title that implies that they are a software engineer, like "firmware engineer", "mobile app engineer", etc.), must be registered with Engineers and Geoscientists BC.

# Professional Organizations for Computing Science



## Association for Computing Machinery

The [Association for Computing Machinery](#) services the scientific computing community worldwide through comprehensive newsletters and other publications that promote the field of computing as a profession and field of study. Students enjoy a variety of benefits including a three-month subscription to the association's flagship publication, an email address assigned by the chapter, access to several additional publications on the latest industry news, a subscription to the organization's student magazine, and a quarterly email newsletter showcasing current programs and opportunities for computer science professionals.

## Association for Information Science and Technology



Members of the [Association for Information Science and Technology](#) explore innovation in the field of information science. Established in 1937, the organization now encompasses approximately 4,000 members in a wide range of fields related to computer science and information technology. Students enjoy benefits such as various subscriptions in the organization's publications, discounts on special services for members, voting rights and free online seminars among other benefits.

# Professional Organizations for Computing Science



## Association for Women in Computing

Founded in 1978 as one of the first professional associations specifically for women in this field, the [Association for Women in Computing](#) now hosts chapters nationwide for individuals, groups and universities. Students can form a chapter with only five individual members, and members receive several significant benefits for joining the organization: practical seminars, career assistance, skill enhancement, scholarships, leadership training and various newsletters. The organization's individual membership feature helps professionals who can't attend meetings gain the benefits of regular membership.



## Institute of Electrical and Electronics Engineers

The [Institute of Electrical and Electronics Engineers](#) is one of the largest and most globally recognized organization for professionals and students who plan to pursue careers in computer science and engineering. The organization values innovation and offers a host of benefits to members worldwide. Top benefits include access to downloadable Microsoft software products, career assistance, discounts on other products such as Rosetta Stone, a subscription to the organization's magazine, grants for student travel, and professional networking opportunities. Students can use the organization's interactive website to find benefits that apply to their areas of interest.

# Ethical and Moral Dilemmas

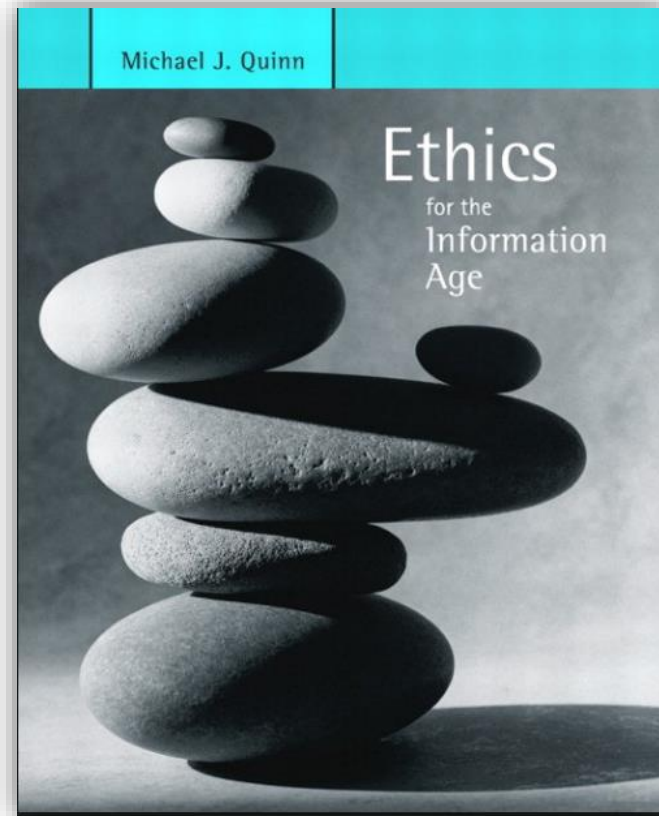
# Ethical and Moral Dilemmas

- 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 (**morals** at the individual level).





# Excellent Reference on Ethics for Computing Scientists



2005 – 1<sup>st</sup> Edition

# Excellent Reference on Ethics for Computing Scientists



2017 – 7<sup>th</sup> Edition



A close-up photograph of a computer keyboard. The central focus is a bright blue key with the words "case studies" printed in white, lowercase, sans-serif font. The key is slightly raised and has rounded edges. Surrounding it are several white keys with standard symbols like hyphens, equals, and apostrophes. The lighting is soft, creating subtle shadows and highlights on the keys' surfaces.

case studies

Section 1.3

# Case studies

SWE practice depends on the type of system under scrutiny. The following examples are quite disparate from one another and require different approaches for development:

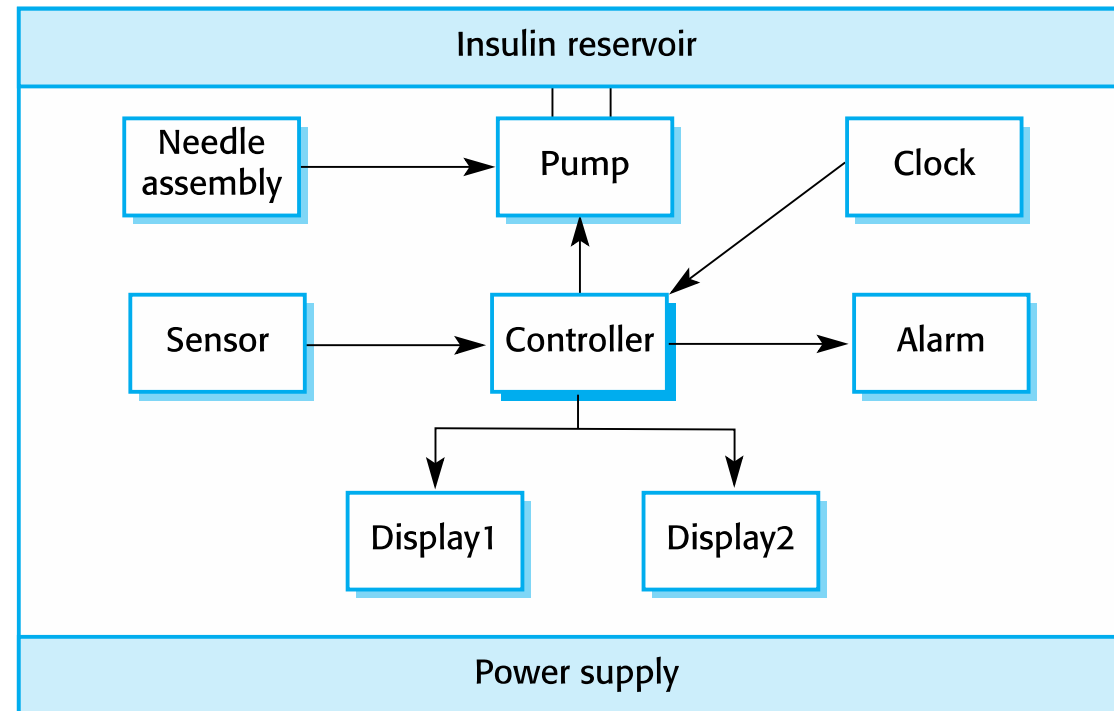
- A personal insulin pump:
  - An *embedded system* in an insulin pump used by diabetics to maintain blood glucose control.
- A mental health case patient management system:
  - Mentcare. An *information system* used to maintain records of people receiving care for mental health problems.
- A wilderness weather station:
  - A *sensor-based data collection system* that collects data about weather conditions in remote areas.
- iLearn: a digital learning environment (10<sup>th</sup> Ed.):
  - A system to support *learning* in schools.

# Insulin pump control system

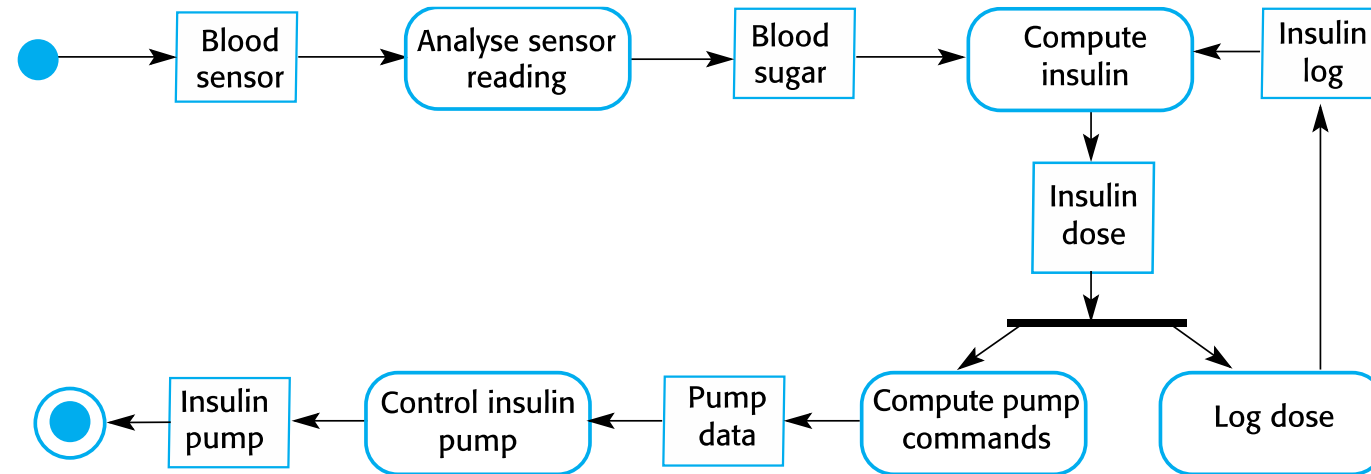
- Collects data from a blood sugar sensor and calculates the amount of insulin required to be injected.
- Calculation based on the rate of change of blood sugar levels.
- Sends signals to a micro-pump to deliver the correct dose of insulin.
- Safety-critical system as low blood sugars can lead to brain malfunctioning, coma and death; high-blood sugar levels have long-term consequences such as eye and kidney damage.



# Insulin pump hardware architecture



# Activity (UML) model of the insulin pump



# Essential high-level requirements

- The system shall be available to deliver insulin when required.
- The system shall perform reliably and deliver the correct amount of insulin to counteract the current level of blood sugar.
- The system must therefore be designed and implemented to ensure that the system always meets these requirements.

# Mentcare: A patient information system for mental health care

- A patient information system to support mental 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 to attend, these clinics are not just run in hospitals. They may also be held in local medical practices or community centres.

# Mentcare

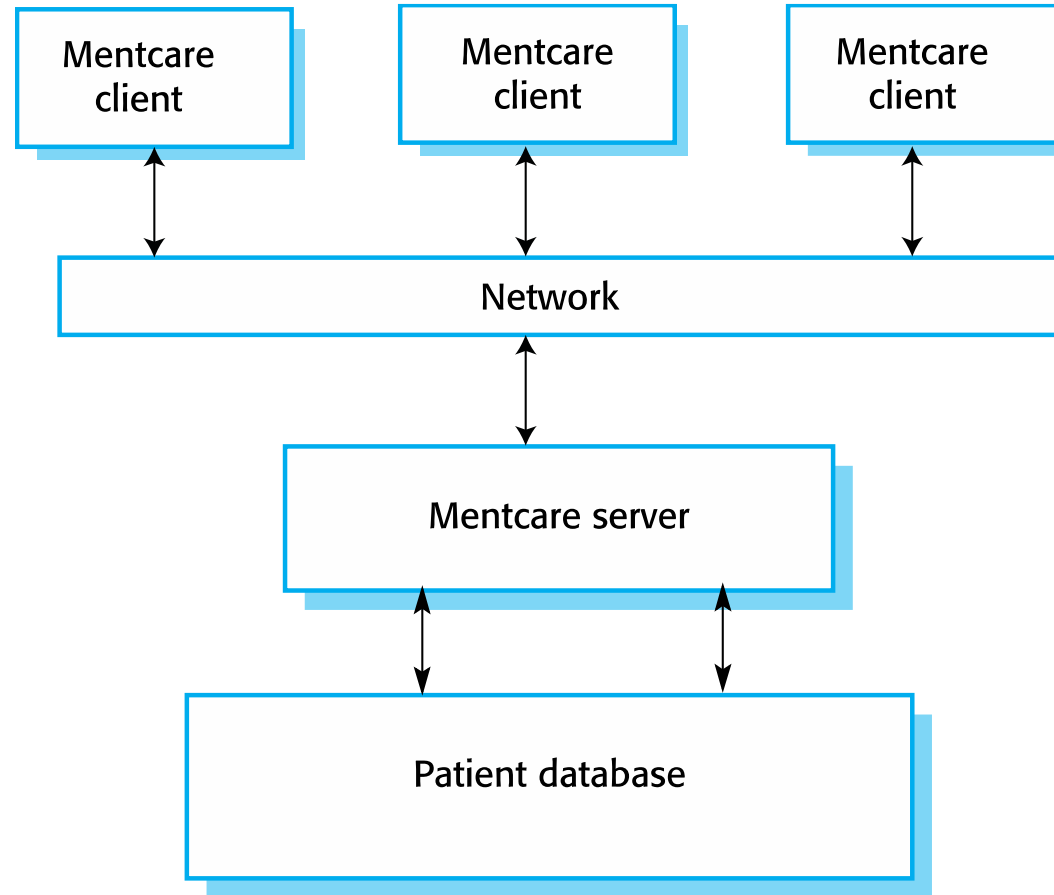
- Mentcare is an information system that is intended for use in clinics.
- It makes use of a centralized database of patient information but has also been designed to run on a PC, so that it may be accessed and used from sites that do not have secure network connectivity.
- When the local systems have secure network access, they use patient information in the database but they can download and use local copies of patient records when they are disconnected.



# Mentcare Goals

- To generate management information that allows health service managers to assess performance against local and government targets.
- To provide medical staff with timely information to support the treatment of patients.

# The organization of the Mentcare system



# Key features of the Mentcare system

- Individual care management
  - Clinicians can create records for patients, edit the information in the system, view patient history, etc. The system supports data summaries so that doctors can quickly learn about the key problems and treatments that have been prescribed.
- Patient monitoring
  - The system monitors the records of patients that are involved in treatment and issues warnings if possible problems are detected.
- 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 drugs prescribed and their costs, etc.

# Mentcare system concerns

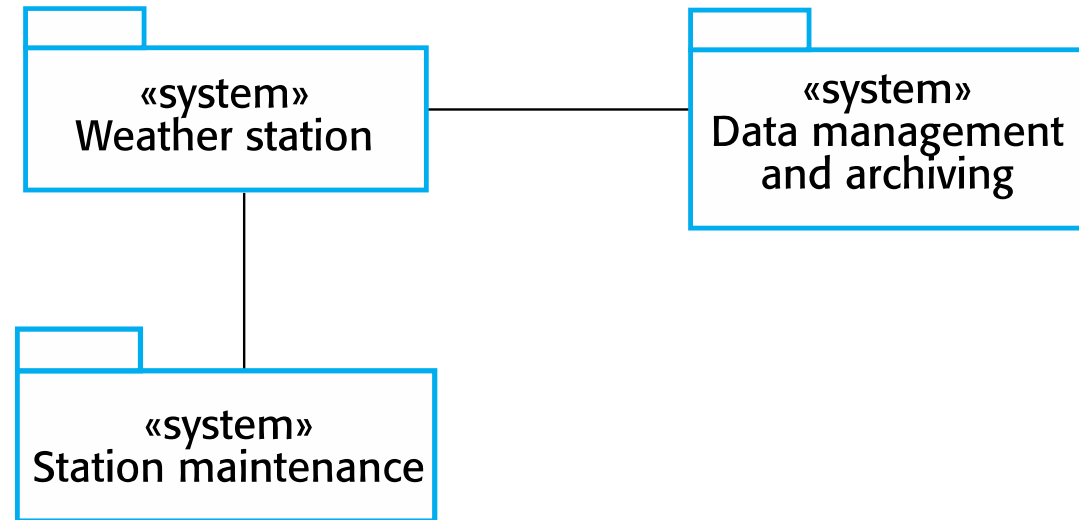
- Privacy
  - It is essential that patient information is confidential and is never disclosed to anyone apart from authorised medical staff and the patient themselves.
- Safety
  - Some mental illnesses cause patients to become suicidal or a danger to other people. Wherever possible, the system should warn medical staff about potentially suicidal or dangerous patients.
  - The system must be available when needed otherwise safety may be compromised and it may be impossible to prescribe the correct medication to patients.

# Wilderness weather station

- The government of a country with large areas of wilderness decides to deploy several hundred weather stations in remote areas.
- Weather stations collect data from a set of instruments that measure temperature and pressure, sunshine, rainfall, wind speed and wind direction.
  - The 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. Each of these instruments is controlled by a software system that takes parameter readings periodically and manages the data collected from the instruments.
-



# The weather station's environment



# Weather information system

- The weather station system
  - This is responsible for collecting weather data, carrying out some initial data processing and transmitting it to the data management system.
- The data management and archiving system
  - This system collects the data from all of the wilderness weather stations, carries out data processing and analysis and archives the data.
- The station maintenance system
  - This system can communicate by satellite with all wilderness weather stations to monitor the health of these systems and provide reports of problems.

# Additional software functionality

- Monitor the instruments, power and communication hardware and report faults to the management system.
- Manage the system power, ensuring that batteries are charged whenever the environmental conditions permit but also that generators are shut down in potentially damaging weather conditions, such as high wind.
- Support dynamic reconfiguration where parts of the software are replaced with new versions and where backup instruments are switched into the system in the event of system failure.

# iLearn: A digital learning environment (10<sup>th</sup> Ed.)

- A digital learning environment is a framework in which a set of general-purpose and specially designed tools for learning may be embedded plus a set of applications that are geared to the needs of the learners using the system.
- The tools included in each version of the environment are chosen by teachers and learners to suit their specific needs.
  - These can be general applications such as spreadsheets, learning management applications such as a Virtual Learning Environment (VLE) to manage homework submission and assessment, games and simulations.

# Service-oriented systems

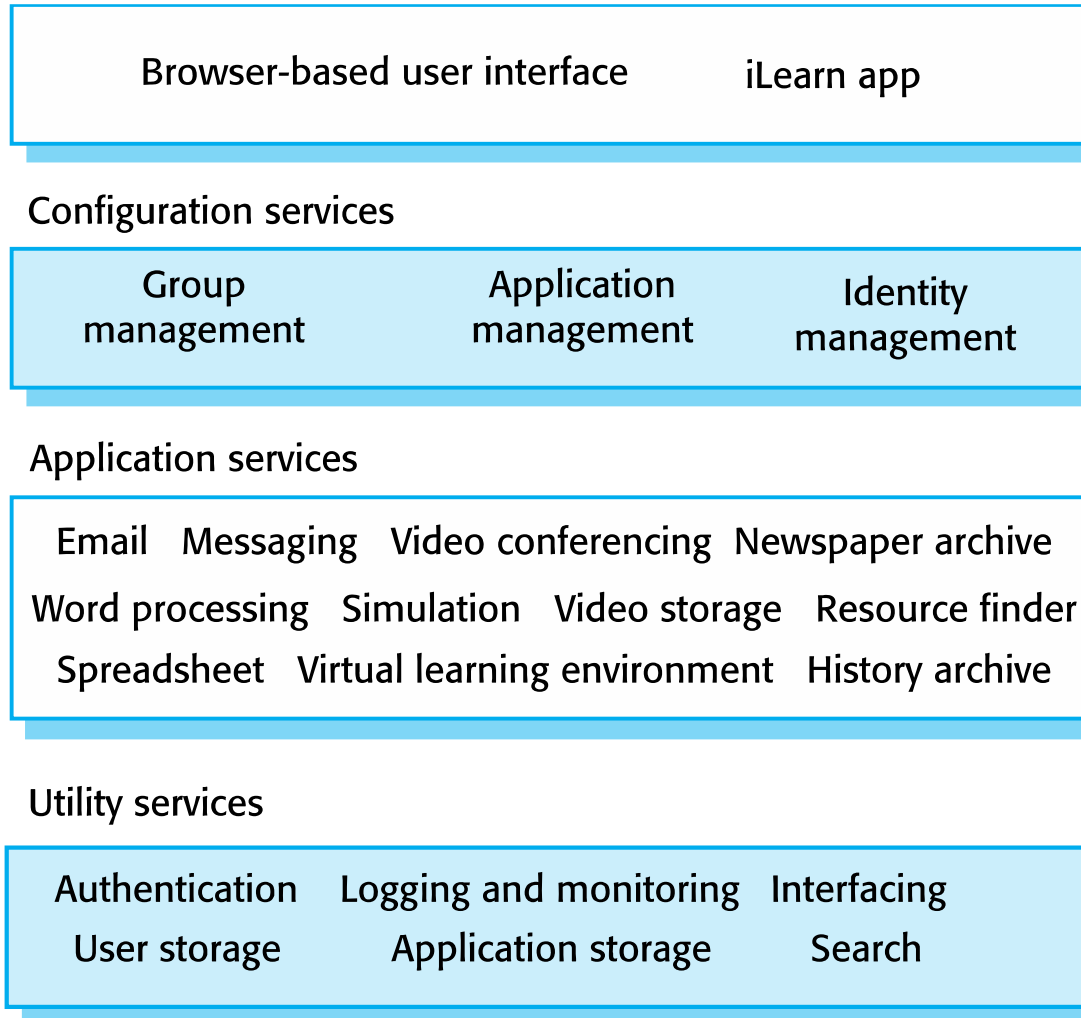
- The system is a service-oriented system with all system components considered to be a replaceable service.
- This allows the system to be updated incrementally as new services become available.
- It also makes it possible to rapidly configure the system to create versions of the environment for different groups such as very young children who cannot read, senior students, etc.

# iLearn services

- *Utility services* that provide basic application-independent functionality and which may be used by other services in the system.
- *Application services* that provide specific applications such as email, conferencing, photo sharing etc. and access to specific educational content such as scientific films or historical resources.
- *Configuration services* that are used to adapt the environment with a specific set of application services and do define how services are shared between students, teachers and their parents.



# iLearn architecture



# iLearn service integration

- *Integrated services* are services which offer an API (application programming interface) and which can be accessed by other services through that API. Direct service-to-service communication is therefore possible.
- *Independent services* are services which are simply accessed through a browser interface and which operate independently of other services. Information can only be shared with other services through explicit user actions such as copy and paste; re-authentication may be required for each independent service.

# Chapter One Summary

- 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.

# Chapter One Summary




- 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.

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 from initial conception to operation and maintenance.
What are the fundamental software engineering activities?	Software specification, software development, software validation and software evolution.
What is the difference between software engineering and computer science?	Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
What is the difference between software engineering and system engineering?	System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process.
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. There are no methods and techniques that are good for everything.
What differences has the Internet made to software engineering?	Not only has the Internet led to the development of massive, highly distributed, service-based systems, it has also supported the creation of an "app" industry for mobile devices which has changed the economics of software.

Copyright ©2016 Pearson Education, All Rights Reserved

**Figure 1.1** Frequently asked questions about software engineering

# Wikipedia Summary

V · T · E		Software engineering	[hide]
Fields	Computer programming · Requirements engineering · Software deployment · Software design · Software maintenance · Software testing · Systems analysis · Formal methods		
Concepts	Data modeling · Enterprise architecture · Functional specification · Modeling language · Orthogonality · Programming paradigm · Software · Software archaeology · Software architecture · Software configuration management · Software development methodology · Software development process · Software quality · Software quality assurance · Software verification and validation · Structured analysis		
Orientations	Agile · Aspect-oriented · Object orientation · Ontology · Service orientation · SDLC		
Models	Developmental	Agile · EUP · Executable UML · Incremental model · Iterative model · Prototype model · RAD · UP · Scrum · Spiral model · V-Model · Waterfall model · XP	
	Other	SPICE · CMMI · Data model · ER model · Function model · Information model · Metamodeling · Object model · Systems model · View model	
	Languages	IDEF · UML · SysML	
Software engineers	<del>Victor Basili · Kent Beck</del> · Grady Booch · Fred Brooks · Barry Boehm · Ward Cunningham · Tom DeMarco · Edsger W. Dijkstra · Martin Fowler · <del>C. A. R. Hoare</del> · Watts Humphrey · Michael A. Jackson · Ivar Jacobson · Stephen J. Mellor · Bertrand Meyer · David Parnas · Winston W. Royce · James Rumbaugh · Niklaus Wirth · Edward Yourdon · Mohamed Fayad		
Related fields	Computer science · Computer engineering · Project management · Risk management · Systems engineering		
<div><div> Category</div><div> Commons</div></div>			
Authority control	GND: 4806620-5 		

Categories: Software project management | Software development philosophies | Agile software development



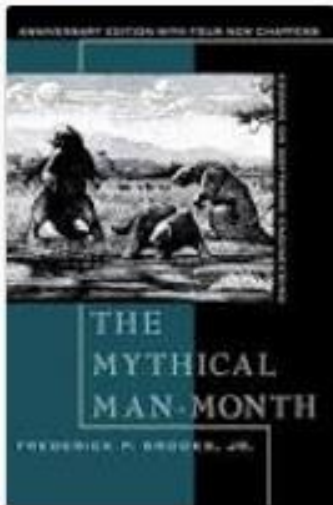
# Mandatory Reading $\forall$ Software Engineers

REPORTS

## **SOFTWARE ASPECTS OF STRATEGIC DEFENSE SYSTEMS**

*A former member of the SDIO Panel on Computing in Support of Battle Management explains why he believes the “star wars” effort will not achieve its stated goals.*

**DAVID LORGE PARNAS**



The Mythical  
Man-Month  
Fred Brooks...



Code  
Complete  
Steve McCo...



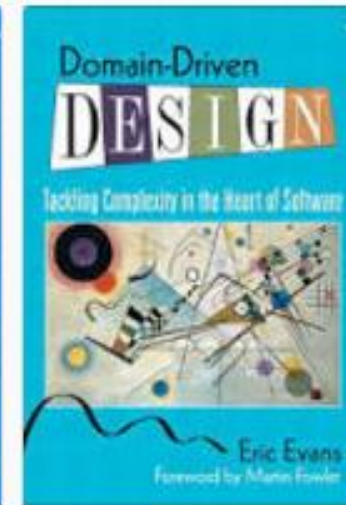
The  
Pragmatic P...  
1999



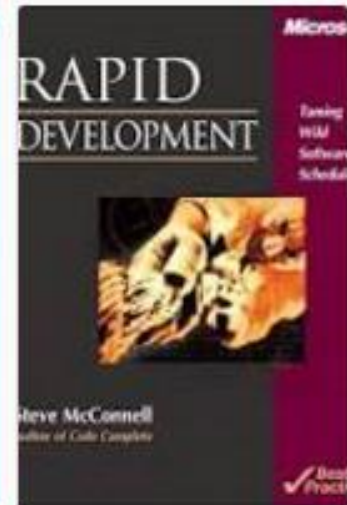
Clean Code  
Robert Cecil...



Design  
Patterns: El...  
1994



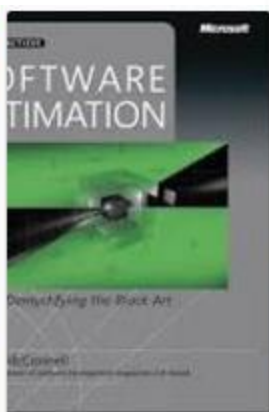
Domain-  
driven design  
Eric J. Evan...



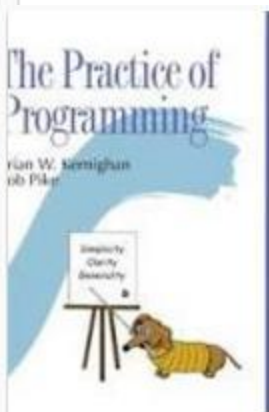
Rapid  
development  
Steve McCo...



Software  
Engineering:...  
Roger S. Pr...



Software  
Estimation  
Steve McCo...



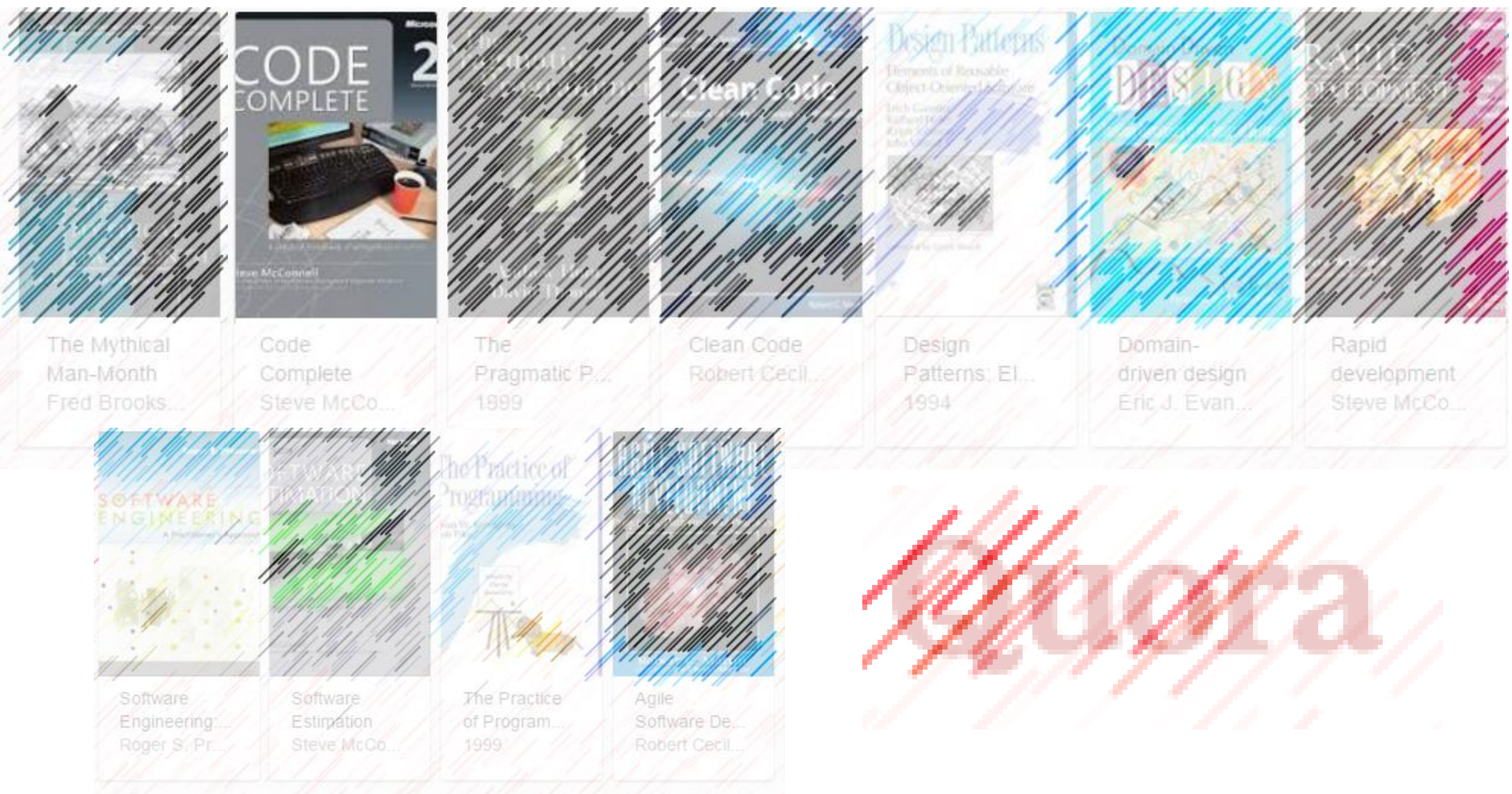
The Practice  
of Program...  
1999



Agile  
Software De...  
Robert Cecil...

# Quora





**THE ALL-TIME CLASSIC**

# Closing Thought from Dijkstra

A number of these phenomena have been bundled under the name "Software Engineering". As economics is known as "The Miserable Science", software engineering should be known as "The Doomed Discipline", doomed because it cannot even approach its goal since its goal is self-contradictory. Software engineering, of course, presents itself as another worthy cause, but that is eyewash: if you carefully read its literature and analyse what its devotees actually do, you will discover that software engineering has accepted as its charter "How to program if you cannot.".

From Edsger W. Dijkstra, "On the cruelty of really teaching computing science“, 1988.

- You MUST read the criticism section of the Wikipedia webpage:  
[https://en.wikipedia.org/wiki/Software\\_engineering](https://en.wikipedia.org/wiki/Software_engineering)
- You Must also read "SOFTWARE ASPECTS OF STRATEGIC  
DEFENSE SYSTEMS", by Parnas at  
<https://pdfs.semanticscholar.org/26ae/89f725ede99590b3a63e0780872c32a6871a.pdf>



# Closing Thought from Your Instructor

- Don't think you're going to learn "how to perform SWE" after completing this course.
- This is NOT science and is extremely situation-dependent.
- You're going to learn about tools, processes and how to apply them.
- But what determines how things will ultimately turn out is on **YOU, YOUR TEAMMATES** and, above all, **THE UNEXPECTED**.

# Closing Thoughts From Experienced S/W Engineers



# Closing Thoughts From Experienced S/W Engineers

Ubiquity

Volume 2014, Number March (2014), Pages 1-6

**Ubiquity symposium: The science in computer science: where's the science in software engineering?**

Walter F. Tichy

DOI: [10.1145/2590528.2590529](https://doi.org/10.1145/2590528.2590529)

*The article is a personal account of the methodological evolution of software engineering research from the '70s to the present. In the '70s, technology development dominated. An attitude of "seeing is believing" prevailed and arguing the pros and cons of a new technology was considered to be enough. With time, this advocacy style of research began to bother me. I thought that something was missing from a scientific approach, and that was the experiment. Controlled experiments and other empirical methods were indeed rare in software research. Fortunately, the situation began to change in the nineties, when journals and conferences began to demand evidence showing that a new tool or method would indeed improve software development. This evolution was gradual, but the situation today is that software research is now by and large following the scientific paradigm. However, the cost of experiments can be staggering. It is difficult to find professional programmers willing to participate in controlled studies and it is time-consuming and expensive to train them in the methods to be studied. I suggest an alternative to experiments with human subjects that can help produce results in certain situations more quickly.*

I feel you man. I just graduated little over a year ago in fact, jumped on the first job offer that came my way and got the biggest shock of my life.



Things I didn't expect:

**School stress and Work stress aren't the same** - The stress of working on a school project with friends, or working solo, even with that looming thesis deadline or special project defense does not compare to the stress of seemingly unreasonable work deadlines, communication problems, (a little of office politics) and crunch times.

**Lack of Best Practices** - Same as your experience on professionalism. Before taking my first job and during my training period, I rushed off reviewing and reading about best practices in both programming and software engineering. These aren't followed as well as they should for impractical and, to be fair, practical reasons. And sometimes, your knowledge counts very little against others who are merely afraid of the unknown and treat these practices with disdain.

**What they taught in school was just the tip of the iceberg** - Thinking that what I learned self-studying and from classes was enough to get me through, I was shocked to say the least as I stared dumbfounded at the first piece of code I was supposed to maintain. A lot of the skills I use now were learned on the job or during my job that I keep on wondering if I could've made it without a college degree at all. XD

**The Importance of Communication** - Made me realize what all those English classes were for. Before the real world, I could not see the relevance of having three to four different English classes in college when it's been taught since we were in the first grade. You're no use in your job when you can talk to a computer but fail to talk to people.

In a traditional undergraduate computer science program you learn *just* programming. But the real world doesn't want people who are *just* programmers. The real world wants real software engineers. I know many job descriptions don't seem to express this distinction, which only confuses the matter. In the real world you need to be able to:



- Gather and analyze requirements when they aren't directly given to you.
- Design and analyze architecture with near endless possibilities.
- Create test plans and act on them to evaluate and improve the quality of a system.
- Work collaboratively on a team of people with different backgrounds and experience levels.
- Estimate and plan work even if you don't know exactly what to build.
- Communicate effectively with stakeholders who have different needs that don't necessarily align.
- Negotiate schedule, budget, quality, and features without disappointing stakeholders.



At university, your teacher gives you:

- A well defined, isolated problem, the solution of which can be provided within a short and well-defined time span (and it will be discarded afterward).
- A well-defined set of tools that you were introduced to prior to assignment
- A well-defined measure for the quality of your solution, with which you can easily determine whether your solution is good enough or not.



In the "Real World":

- The problem is blurry, complex and embedded in context. It's a set of contradictory requirements that change over time and your solution must be flexible and robust enough for you to react to those changes in an acceptable time.
- The tools must be picked by you. Maybe there's already something usable in your team's 10-year-old codebase, maybe there's some open source project, or maybe a commercial library will have it. Or, maybe you will have to write it on your own.
- To determine whether the current iteration of your software is an improvement (because you're almost never actually done with a software project), you need to do regression testing and usability testing, the latter of which usually means that the blurry, complex, contradictory, context-embedded requirements shift once again.

# Closing Thoughts From Experienced S/W Engineers

## Conclusion

Programming in school and programming in the real world are so inherently different to the point where there's actually very little overlap. CS will prepare you for "real world" software development like athletics training would prepare an army for battle.

**Welcome,  
to the real  
world**





# 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 of the 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.



---

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?

---

---

## 1.2 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?

---

The essential difference is that in generic software product development, the specification is owned by the product developer. For custom product development, the specification is owned and controlled by the customer. The implications of this are significant – the developer can quickly decide to change the specification in response to some external change (e.g. a competing product) but, when the customer owns the specification, changes have to be negotiated between the customer and the developer and may have contractual implications.

For users of generic products, this means they have no control over the software specification so cannot control the evolution of the product. The developer may decide to include/exclude features and change the user interface. This could have implications for the user's business processes and add extra training costs when new versions of the system are installed. It also may limit the customer's flexibility to change their own business processes.

---

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

---

---

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

---

Four important attributes are maintainability, dependability, performance and usability. Other attributes that may be significant could be reusability (can it be reused in other applications), distributability (can it be distributed over a network of processors), portability (can it operate on multiple platforms e.g laptop and mobile platforms) and inter-operability (can it work with a wide range of other software systems).

*Decompositions of the 4 key attributes e.g. dependability decomposes to security, safety, availability, etc. is also a valid answer to this question.*



Presentation Terminated