Chapter 1

Introduction
Topics covered

- Professional software development
  - What is meant by “software engineering”.
  - Addendum to Sommerville’s FAQs

- Software engineering ethics
  - A brief introduction to ethical issues that affect software engineering.

- Case studies
  - An introduction to three examples that are used in later chapters in the book.
Software engineering

- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled. (E.g., gas centrifuges, power grids, nuclear power plants, ballpoint pens.)
- Software engineering (SE) is concerned with **theories**, **methods** and **tools** for professional software development.
- SE expenditures represents a significant fraction of GNP in all developed countries.
Electronic Pen?
The electronic pen itself looks and even feels like a regular ballpoint pen, the company says, but contains a digital sensor and image microprocessor. As you write, or follow the lines of a map with the pen, the nib automatically scans your movements in conjunction with a digital watermark at a rate of 75 snapshots per second, effectively giving the impression of scanning while you write.
Software costs

- Often dominate computer system costs*. (The costs of software on a PC are often greater than the hardware cost.)

- Software costs more to maintain than it does to develop*. (For systems with a long life, maintenance costs may be several times development costs.)

- SE is concerned with cost-effective software development.

FAQs about software engineering

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

(cont’d)
FAQs about software engineering (cont’d)

- What are (some of) the *key challenges* facing software engineering?
- What are the *costs* of software engineering?
- What are the *best* software engineering *techniques* and *methods*?
- What differences has the *web* made to software engineering?
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.
  - **Generic products** - developed to be sold to any customer wishing to buy them.
    - Specification owned by developer and change decisions made by developer
  - **Customized products** – commissioned by specific customers to meet their needs.
    - Specification owned by customer and change requirements identified by customer
What are the (essential) attributes (product characteristics) of good (professional) software?

- Must deliver the required functionality and performance.
- **Maintainability**: must be able to evolve to meet changing needs.
- **Dependability**: must be trustworthy.
- **Efficiency**: should not make wasteful use of system resources.
- **Usability**: must be usable by the users for which it was designed.
What is *software engineering*?

- SE is an *engineering discipline* which is concerned with **all aspects of software production and maintenance**...
  - Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.
  - Not just the technical process of *development*. Also *project management* and the *development of tools, methods, etc.*, to support development and maintenance.

How *old* is the term “software engineering”?
What are the fundamental software engineering \textit{process activities}?

- **Software specification**: customers and engineers identify the \textit{functionality} of the software that is to be produced and the \textit{constraints} on its operation.
- **Software development**: the software is \textit{designed} and \textit{programmed}.
- **Software validation**: the software is checked to ensure that it is what the customer \textit{requires/needs}.
- **Software evolution**: the software is modified to reflect changing customer and market requirements.

So where does software \textit{verification} fit-in?
What is the difference between software engineering and computer science?

- Computer science is concerned with theory and computing fundamentals.
- Software engineering is concerned with the practicalities of developing, delivering, and maintaining useful software.
What is the difference between software engineering and software engineering?

- **(Computer-based)** system engineering is concerned with all aspects of computer-based systems development, including hardware, software and process engineering. Software engineering is just part of this process...

- **System engineers** are involved in (overall) specification, architectural design, integration, and deployment of computer-based systems.
What are (some of) the key challenges facing software engineering?

- Coping with increasing diversity, demands for reduced delivery times, and developing software that is demonstrably trustworthy. Related issues:
  - Heterogeneity – increasingly, systems are distributed and include a mix of hardware and software.
  - Business and social change – there is increasing pressure for faster delivery of software.
  - Security and trust – as software is intertwined with all aspects of our lives, it is essential that we can trust software.

Evolving corporate goals:

1980’s: improve quality
1990’s: reduce cycle time
What are the *costs* of software engineering?

- Roughly 60% are “development” costs, 40% are “testing” costs.
- For most types of systems, the majority of costs are associated with *changing the software after it has gone into use*. 60-90% of overall life-cycle costs is not unusual
- Costs vary depending on the type of system being developed and the requirements of system attributes such as performance and reliability.
- *Distribution* of costs depends on the development model that is used. Waterfall, XP, Cleanroom, etc.
Activity cost distribution

Waterfall model

Iterative development

Component-based software engineering

Development and evolution costs for long-lifetime systems
Activity cost distribution (cont’d)

A more specific example: Generic software developed using an incremental approach intended for several different platforms.

0 25 50 75 100

Specification Development System testing
What are the best SE *techniques and methods*?

- There is no universal set of software techniques that is applicable to all of the many different types of software systems.
- The SE methods and tools used depend (among other things) on the *type of application*, the *requirements of the customer*, and the *nature/background of the development team*.
(8 Examples of) Application types

- **Stand-alone applications** that run on a local computer, such as a PC. (They include all necessary functionality and do not need to be connected to a network.)

- **Interactive transaction-based applications** that execute on a remote computer and are accessed by users from their own PCs or terminals. (These include web applications such as e-commerce solutions.)

- **Embedded control systems** that control and manage hardware devices. (There are probably more embedded systems than any other type.)

(cont'd)
Application types (cont’d)

- **Batch processing (business) systems** designed to process data in large batches. (They process large numbers of individual inputs to create corresponding outputs.)

- **Entertainment systems.** (Most are personal systems primarily intended to entertain the user.)

- **Systems for modeling and simulation.** (They are developed to model physical processes that may include many separate interacting objects.)
Application types (cont’d)

- **Data collection systems.** (They collect data from their environment using a set of sensors and send it to other systems for processing.)

- **Systems of systems.** (Systems that are composed of a number of other software systems.)
Some fundamental SE principles that apply universally to all types of software

- Systems should be developed using a **managed and understood development process**. (Of course, different processes are used for different types of software.)
- **Dependability and performance** are important for all types of systems.
- Understanding and managing **software requirements** (functional and non-functional) is important.
- **Where appropriate**, one should **reuse software** that has already been developed rather than develop new software.
What differences has the *web* made to SE?

- **Organizations are increasingly developing web-based systems** rather than local systems.

- **Web services** allow application functionality to be accessed over the web.

- **Cloud computing** is an approach to the provision of computer services whereby applications run remotely “in the cloud.” (Users do not buy software but pay according to use.)

(cont'd)
Web-based software engineering

- *Software reuse* is the dominant approach for constructing web-based systems.
- Web-based systems are normally developed and delivered *incrementally*.
- User interfaces are *constrained by the capabilities of web browsers*.
- Web-based systems are complex distributed systems but *the fundamental principles of software engineering still apply*.
Addendum to Sommerville’s FAQs…

Keynote Address

8th Conference on Software Engineering Education

Presented by

Richard E. Fairley, Ph.D.
A Quote from Winston Royce

The construction of new software which is pleasing to both user and buyer, and that does not contain errors, is an unexpectedly hard problem. **It is perhaps the most difficult problem in engineering today.** It is often referred to as the “software crisis.” **It has become the longest continuing crisis in the engineering world, and it continues unabated.**

-1987
What is Software?

- At the most fundamental level, software is the state of magnetization, voltage level, or current flow in electronic devices.
- Software is therefore intangible; it has no mass, no volume, taste, odor, or other physical properties.*
- Requirements specifications, design documents, source code, and other work products of software engineering are *representations* of the inherently invisible software product.

* Boeing aeronautical engineer: How much does the software weigh?
Foundations of Software Engineering

- Logic
- Computer science
- Economics
- Management science
Engineering involves the application of scientific principles to the production of *material* artifacts within a framework of economic, social, and ethical considerations.
NSPE Guidelines

National Society of Professional Engineer’s Guidelines for an engineering discipline...

1. Define a body of knowledge.
2. Adopt a set of recommended practices.
3. Establish a code of ethics.
4. Develop accreditation and certification guidelines.
5. Define a curriculum.
Guide to the ("generally accepted") Software Engineering Body of Knowledge (SWEBOK)

- '04 IEEE Computer Society effort to:
  - characterize the contents of the SE discipline
  - promote a consistent view of SE worldwide
  - clarify SE's place with respect to other disciplines
  - provide a foundation for training materials and curriculum development, and
  - provide a basis for certification and licensing of software engineers.

- (‘14 revision -- Version 3.0 -- now available...)

- See: [http://www.computer.org/portal/web/swebok](http://www.computer.org/portal/web/swebok)
Fairley’s final observation…

The tragedy of software engineering is not that we don’t know how to engineer software, but that we know how and don’t do it.
Professional and ethical responsibility
Professional and ethical responsibility

- SE involves wider responsibilities than simply the application of technical skills.
- Software engineers must behave honestly and ethically if they are to be respected as professionals.
- This goes beyond simply upholding the law.
- Consider four important issues of professional responsibility...
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 never misrepresent their level of competence.
  - They should never knowingly accept work which is outside their competence.

(cont’d)
Issues of professional responsibility

- **Intellectual property rights**
  - Engineers should be aware of applicable 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.
ACM/IEEE Code of Ethics ~’99

- The ACM and IEEE have cooperated to produce a code of ethical practice.
- Members must adopt the code when they join.
- It is based on eight principles related to the behaviour and decisions made by professional software engineers (+ educators, managers, supervisors, policy makers, trainees, and students of the profession).
Stated rationale for a 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.
Code of ethics - short and full versions

- Short version summarizes aspirations at a high level.
- Full version gives examples and details of how aspirations change the way software engineering professionals should act.
- The aspirations + examples/details form a cohesive code.
Code of ethics – preamble

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**:
Code of ethics – 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.

(cont'd)
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.

(cont'd)
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.
Examples of potential ethical dilemmas

- Disagreement in principle with the policies of senior management.
- Your employer acts in an unethical way by releasing a safety-critical system before testing is complete.
- Participation in the development of military weapons systems (e.g., nuclear arms).
Case Studies

- **A PERSONAL INSULIN PUMP**: embedded software system in an insulin pump used by diabetics to maintain blood glucose control.

- **A MENTAL HEALTH PATIENT INFORMATION MANAGEMENT SYSTEM**: used to maintain records of people receiving care for mental health problems.

- **A WILDERNESS WEATHER STATION**: a system that collects and transmits weather condition data in remote areas.
Key points

- Software engineering is an engineering discipline concerned with all aspects of software production.
- “Essential product attributes” are maintainability, dependability, efficiency, usability, and delivering the required functionality and performance.
- The high-level activities of specification, development, V&V, and evolution are part of all software engineering processes.

(cont’d)
Key points (cont’d)

- There are many different types of systems and each requires appropriate software engineering tools and techniques for their development.

- However, the fundamental principles of software engineering are applicable to all types of software systems.
Key points (and finally…)

- 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.
Chapter 1

Introduction