CSE 4103: Software Engineering and Information System

Software & Software Engineering

Software Engineering -

Textbook: “Software Engineering -- A Practitioner’s Approach,” 6th Edition by Roger S. Pressman

Now software, of course is different from other engineering domains

we should attempt to estimate the cost and effort
We should plan and schedule the work
We should involve users in defining requirements

14

Industries concern about software and the manner it is developed

- Why does it take so long to get software finished?
- Why are development costs so high?
- Why can’t we find all errors before we give the software to our customers?
- Why do we spend so much time and effort maintaining existing programs?
- Why do we continue to have difficulty in measuring progress as software is being developed and maintained?

15

WHAT IS SOFTWARE ?

Software is

(1) Instructions (computer programs) that when executed provide desired features, function and performance;

(2) Data structures that enable the programs to adequately manipulate information;

and

(3) documents that describe the operation and use of the programs.

16

Some Software Characteristics

- Software does not “ wear out”.
- Software is developed or engineered ***not*** manufactured in the classical sense.

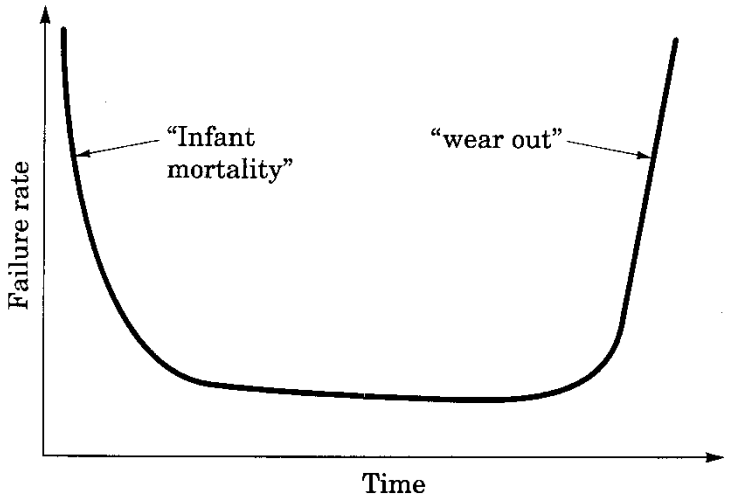
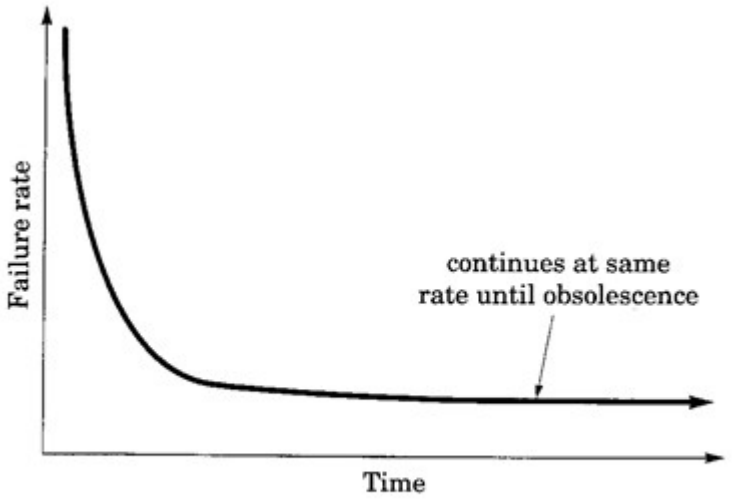


FIGURE 1.2.
Failure curve for hardware.

17

Some Software Characteristics

- In theory, software does not wear out at all.



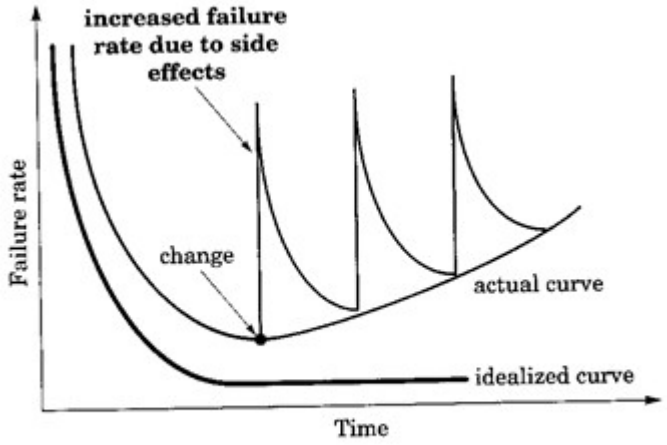
Failure curve for software (idealized).

- **BUT,**
  - Hardware upgrades.
  - Software upgrades.

18

Some Software Characteristics

- Thus, reality is more like this.



Actual failure curve for software.

  - Most serious corporations control and constrain changes

- Although the industry is moving toward component based construction, most software continues to be custom built.

Types of Software Applications

- Systems software
- Application software
- Engineering/Scientific software
- Embedded Software
- Product-line software
- Web-applications
- Artificial Intelligence Software

- *Ubiquitous computing*
- *Netsourcing*
- *Open source*
- *The new economy*

Legacy Software

- Older programs
- The quality of Legacy software is poor
- Sometimes has inextensible design, convoluted code, poor or non-existent documentation, test cases and results that were never achieved, a poorly managed change history…

---The software must be adapted

---must be enhanced

---must be extended

--must be re-architected

21

Theory of Software evolution

- The law of continuing change (1974)
- The law of increasing complexity (1974)
- The law of self-regulation (1974)
- The law of conservation of organizational stability (1980)
- The law of conservation of familiarity (1980)
- The law of continuing growth 1980)
- The law of declining quality (1996)
- The feedback system law (1996)

[Manny Lehman etel., 1997]

E-type system: software that has been implemented in a real world computing context and will therefore evolve over time.

22

Software Myths (Managerial myths)

- **Myth**: We already have a book that's full of standards and procedures for building software. Won't that provide my people with everything they need to know.

  Reality: is it complete or modern or adaptable?

- **Myth**: If we get behind schedule, we can add more programmers and catch up

  Reality: Maybe. It increases coordination efforts and ***may slow things down***.

23

Software Myths (Managerial myths)

- **Myth**: If I decide to outsource the software project to a third party, I can just relax and let that firm build it.

Reality: Lack of proper experience in managing and control software project internally, will cause problem.

24

Customer myths

- **Myth:** A general statement of objectives is sufficient to begin writing programs- we can fill in the details later.

  Reality: Incomplete/ambiguous statement of requirements create huge problem

- **Myth:** Project requirements continually change, but change can easily be accommodated because software is flexible.

  Reality: Impact of change varies with the time

Software Myths (practitioner's myths)

- **Myth**: Once we write the program and get it to work, our job is done.

  Reality: Coding may be as little as 10% of the effort, and 50 - 70% may occur after delivery.

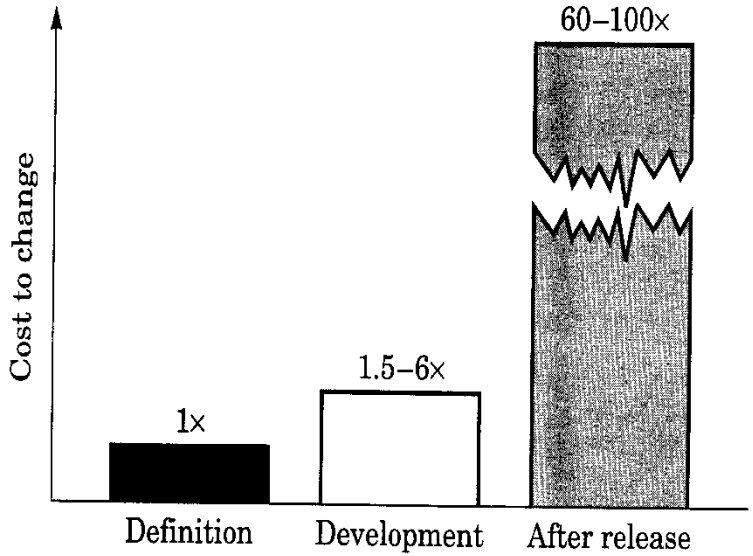


FIGURE 1.5. The impact of change.

- **Myth:** Until I get the program running, I have no way of assessing its quality.

  Reality: Software quality assurance mechanisms from the inception of the project.

- **Myth:** The only deliverable work product for a successful project is the working programs.

  Reality: Working program is only one part of software configuration.

- **Myth:** Software engineering will make us create voluminous and unnecessary documentation and invariably slow us down.

  Reality: Software engineering is about quality.

What is Software?

The product that software professionals build and then support over the long term.

Software encompasses: (1) instructions (computer programs) that when executed provide desired features, function, and performance; (2) data structures that enable the programs to adequately store and manipulate information and (3) documentation that describes the operation and use of the programs.

Software products

Generic products

- Stand-alone systems that are marketed and sold to **any customer** who wishes to buy them.
- Examples – PC software such as editing, graphics programs, project management tools; CAD software; software for specific markets such as appointments systems for dentists.

Customized products

- Software that is commissioned by **a specific customer** to meet their own needs.
- Examples – embedded control systems, air traffic control software, traffic monitoring systems.

Why Software is Important?

- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled (transportation, medical, telecommunications, military, industrial, entertainment,)
- Software engineering is concerned with theories, methods and tools for professional software development.
- Expenditure on software represents a significant fraction of GNP in all developed countries.

Software costs

- Software costs often dominate computer system costs. The costs of software 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.
- Software engineering is concerned with cost-effective software development.

Features of Software?

- Its characteristics that make it different from other things human being build.

Features of such logical system:

- Software is developed or engineered, it is not manufactured in the classical sense which has quality problem.
- Software doesn't "wear out." but it deteriorates (due to change). Hardware has bathtub curve of failure rate (high failure rate in the beginning, then drop to steady state, then cumulative effects of dust, vibration, abuse occurs).
- Although the industry is moving toward component-based construction (e.g. standard screws and off-the-shelf integrated circuits), most software continues to be custom-built. Modern reusable components encapsulate data and processing into software parts to be reused by different programs. E.g. graphical user interface, window, pull-down menus in library etc.

Software Applications

1. **System software**: such as compilers, editors, file management utilities
2. **Application software:** stand-alone programs for specific needs.
3. **Engineering/scientific software**: Characterized by "number crunching"algorithms. such as automotive stress analysis, molecular biology, orbital dynamics etc
4. **Embedded software** resides within a product or system. (key pad control of a microwave oven, digital function of dashboard display in a car)
5. **Product-line software** focus on a limited marketplace to address mass consumer market. (word processing, graphics, database management)
6. **WebApps** (Web applications) network centric software. As web 2.0 emerges, more sophisticated computing environments is supported integrated with remote database and business applications.
7. **AI** software uses non-numerical algorithm to solve complex problem. Robotics, expert system, pattern recognition game playing

Software—New Categories

- **Open world computing**—pervasive, ubiquitous, distributed computing due to wireless networking. How to allow mobile devices, personal computer, enterprise system to **communicate across vast network**.
- **Netsourcing**—the Web as a computing engine. How to architect simple and sophisticated applications to target end-users worldwide.
- **Open source**—"free" source code open to the computing community (a blessing, but also a potential curse!)
- Also …
  - Data mining
  - Grid computing
  - Cognitive machines
  - Software for nanotechnologies

Software Engineering Definition

The seminal definition:

[Software engineering is] the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

The IEEE definition:

Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).

Importance of Software Engineering

- More and more, individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.
- It is usually cheaper, in the long run, to use 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 system, the majority of costs are the costs of changing the software after it has gone into use.

FAQ about software engineering

Question	Answer
What is software?	Computer programs, data structures and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
What are the attributes of good software?	Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.
What is software engineering?	Software engineering is an engineering discipline that is concerned with all aspects of software production.
What 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.

Essential attributes of good software

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

Software Engineering: The Good

- Software engineering has helped to produce systems that improve our lives in numerous ways
  - helping us to perform tasks more quickly and effectively
  - supporting advances in medicine, agriculture, communication, transportation, and other industries
- Indeed, software-based systems are now ubiquitous

Software Engineering: The Bad

- Software is not without its problems
  - Systems function, but not in the way we expect
  - Or systems crash, generate the wrong output, etc.
  - Or the process for producing a system is riddled with problems leading to
    - a failure to produce the entire system
- many projects get cancelled without ever producing a system

Software Engineering: The Bad

- One study in the late 80s found that in a survey of 600 firms, more than 35% reported having a runaway development project. A runway project is one in which the budget and schedule are completely out of control.

Software Engineering: The Bad

CHAOS Report from Standish Group

- Has studied over 40,000 industry software development projects over the course of 1994 to 2004.
- Success rates (projects completed on-time, within budget) in 2004 was 34%, up from 16.2% in 1994
- Failure rates (projects cancelled before completion) in 2004 was 15%, down from 31% in 1994.
- In 2004, “challenged” projects made up 51% of the projects included in the survey.
  - A **challenged project** is one that was over time, over budget and/or missing critical functionality

Software Engineering: The Bad

- Most challenged projects in 2004 had a cost overrun of under 20% of the budget, compared to 60% in 1994
- The average cost overrun in 2004 was 43% versus an average cost overrun of 180% in 1994.
- In 2004, total U.S. project waste was 55 billion dollars with 17 billion of that in cost overruns; Total project spending in 2004 was 255 billion
- In 1994, total U.S. project waste was 140 billion (80 billion from failed projects) out of a total of 250 billion in project spending

Software Engineering: The Bad

- So, things are getting better (attributed to better project management skills industry wide), but we still have a long way to go.
- 66% of the surveyed projects in 2004 did not succeed!

Software Engineering: The Ugly

- Loss of NASA's Mars Climate Observer
  - due to mismatch of English and Metric units!
  - even worse: problem was known but politics between JPL and Houston prevented fix from being deployed
- Denver International Airport
  - Luggage system: 16 months late, 3.2 billion dollars over budget!

Software Engineering: The Ugly

- Therac-25 (safety critical system: failure poses threat to life or health)
  - Machine had two modes:
    - “electron beam” and “megavolt x-ray”
    - “megavolt” mode delivered x-rays to a patient by colliding high energy electrons into a “target”
  - Patients died when a “race condition” in the software allowed themegavolt mode to engage when the target was not in position
- Related to a race between a “type ahead” feature in the user interface and the process for rotating the target into position

What is Good Software?

- **"Good"** is often associated with some definition of quality. The higher the quality, the better the software.
- The problem? Many different definitions of quality!
  - **Transcendental**: where quality is something we can recognize but not define ("I know it when I see it")
  - **User**: where quality is determined by evaluating the fitness of a system for a particular purpose or task (or set of tasks)
  - **Manufacturing**: quality is conformance to a specification

What is Good Software?

- **Product**: quality is determined by internal characteristics (e.g. number of bugs, complexity of modules, etc.)
- **Value**: quality depends on the amount customers are willing to pay

- customers adopt “user view”; developers adopt “manufacturing view”, researchers adopt “product view”; “value view” can help to tie these together

What is Good Software?

- Good software engineering must always include a **strategy for producing high quality software**
- Three common ways that SE considers quality:
  - The quality of the product (product view)
  - The quality of the process (manufacturing view)
  - The quality of the product in the context of a business environment (user view)
- The results of the first two are termed the “technical value of a system”; The latter is the “business value of a system”

The Quality of the Process (I)

- Quality of the development and maintenance process is as important as the product quality
- The development process needs to be modeled

The Quality of the Process (II)

- Modeling will address questions such as
  - What steps are needed and in what order?
  - Where steps in the process are effective for finding faults?
  - How can you shape the process to find faults earlier?
  - How can you shape the process to build fault tolerance into a system?

The Quality of the Process (II)

- Models for Process Improvement
  - SEI's Capability Maturity Model (CMM)
  - ISO 9000
  - Software Process Improvement and Capability dEtermination (SPICE)

Software Engineering: **More than just Programming**

- It should now be clear that software engineering is more than just
  - programming, data structures, algorithms, etc.
- It takes advantage of these very useful computer science techniques but adds
  - **quality concerns**
    - testing, code reviews, validation and verification of requirements
  - **process concerns**
    - Are we using the right software life cycle? Are we monitoring our ability to execute the process? Are we consistent? Are we getting better?
  - **reliance on tools, people, and support processes**
    - debugging, profiling, configuration management, deployment, issue tracking