Introduction to Software Engineering

Chapter : Topic Covered

- Evolving Role of Software
- Hardware vs. Software
- Software characteristics
- Changing nature of software
- Evolution of Software
- Software Myths

Evolving Role of Software

Software is a product

- Transforms information - produces, manages, acquires, modifies, displays, or transmits information
- Delivers computing potential of hardware and networks

Software is a vehicle for delivering a product

- Controls other programs (operating system)
- Effects communications (networking software)
- Helps build other software (software tools & environments)

What is Software ?

Software can defined as:

- Instruction – executed provide desire features, function & performance.
- Data structure – to adequately manipulate operation.
- Documents – operation and use of the program.

Software products may be developed for a particular customer or may be developed for a general market.

- Software products may be
  - **Generic** - developed to be sold to a range of different customers e.g. PC software such as Excel or Word.
  - **Bespoke** (**custom**) - developed for a single customer according to their specification.

Hardware vs. Software

Hardware

- Manufactured
- wear out
- Built using components
- Relatively simple

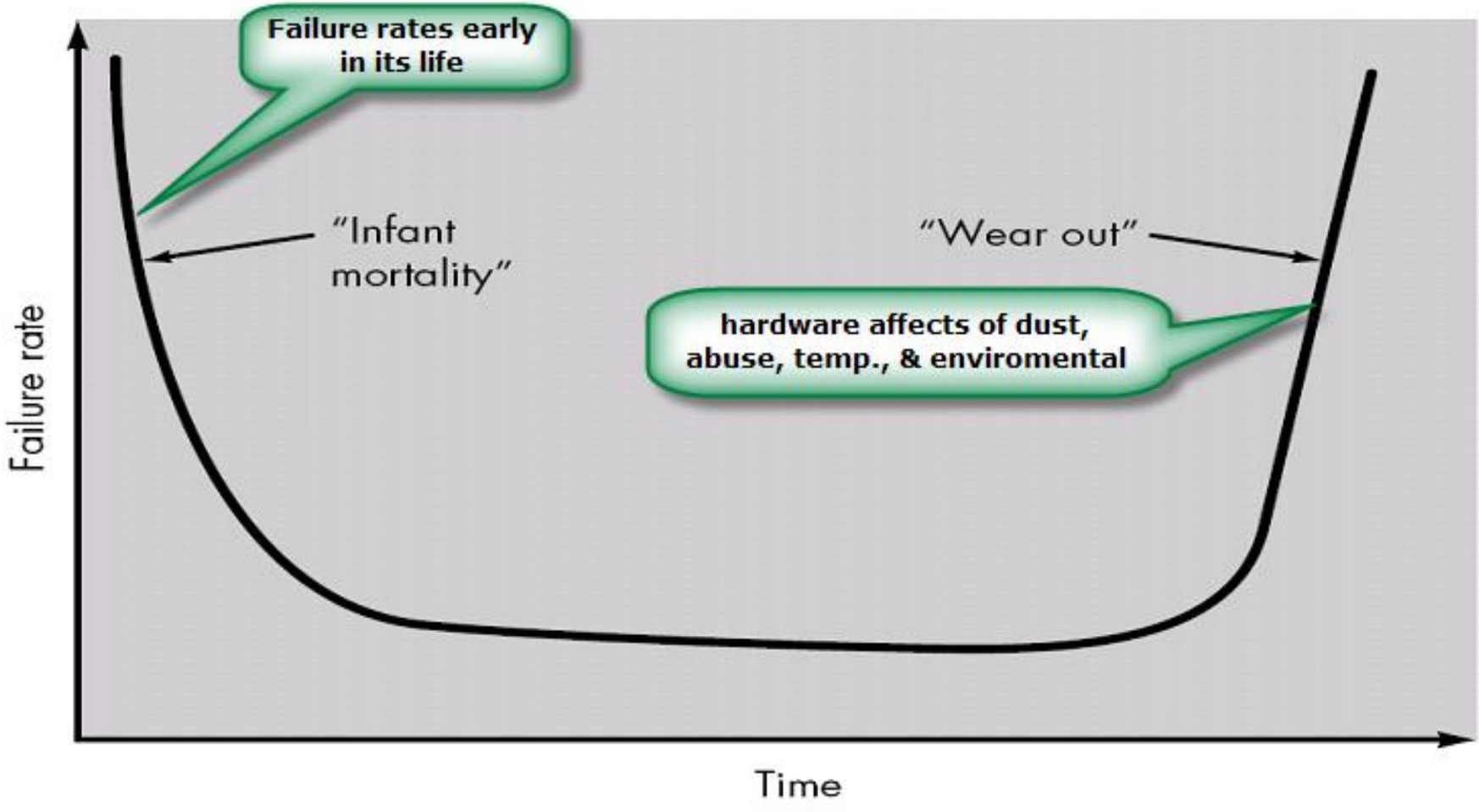
Software

- Developed/ engineered
- deteriorate/ weaken
- Custom built
- Complex

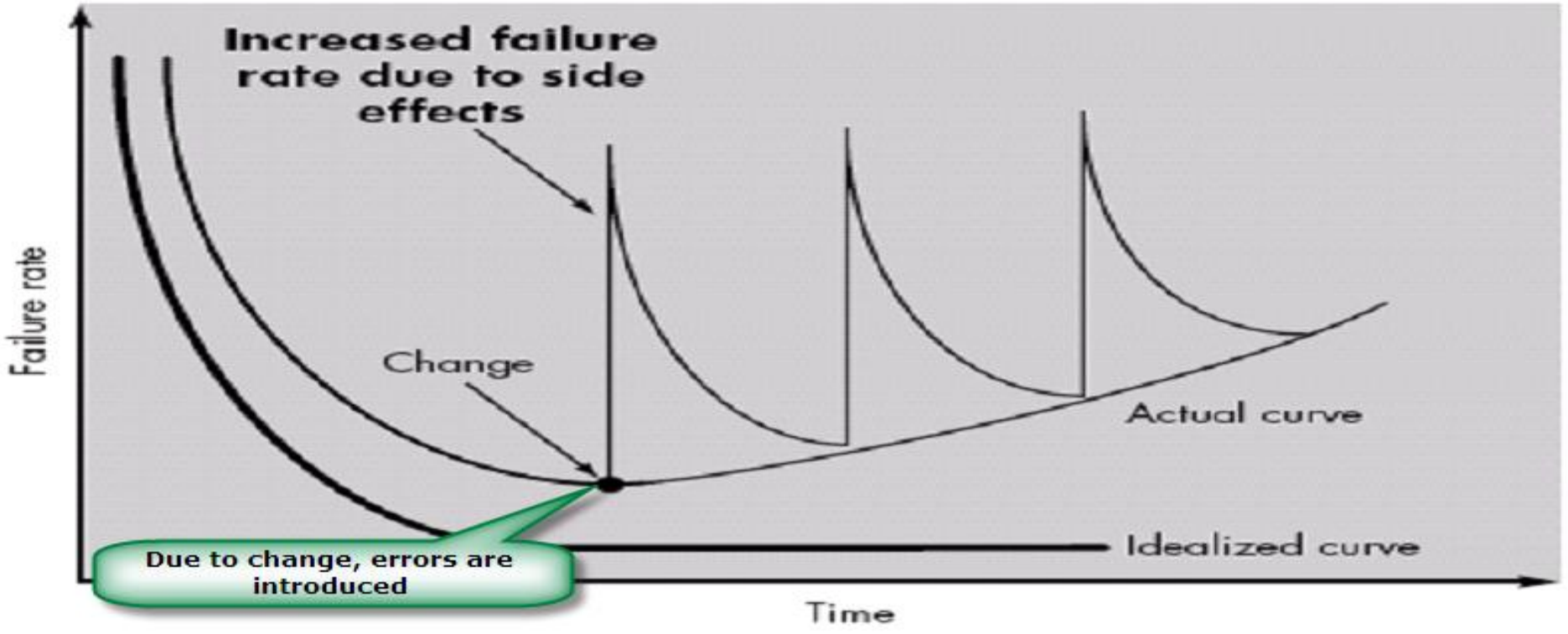
Manufacturing vs. Development

- Once a hardware product has been manufactured, it is difficult or impossible to modify. In contrast, software products are routinely modified and upgraded.
- In hardware, hiring more people allows you to accomplish more work, but the same does not necessarily hold true in software engineering.
- Unlike hardware, software costs are concentrated in design rather than production.

Failure curve for Hardware



Failure curve for Software



When a hardware component wears out, it is replaced by a spare part.

There are no software spare parts. Every software failure indicates an error in design or in the process through which design was translated into machine executable code. Therefore, software maintenance involves considerably more complexity

Component Based vs. Custom Built

- Hardware products typically employ many standardized design components.
- Most software continues to be custom built.
- The software industry does seem to be moving (slowly) toward component-based construction.

Software characteristics

- Software is developed or engineered; it is not manufactured.
- Software does not “wear out” but it does deteriorate.
- Software continues to be custom built, as industry is moving toward component based construction.

CAUSES FOR CHANGE IN ROLE OF SOFTWARE

- Dramatic improvements in hardware performance.
- Profound changes in computing architectures.
- Vast increases in memory and storage capacity.
- Programmers has been replaced by teams of software specialists, each focusing on one part of the technology required to deliver a complex application.

Changing nature of software/ Types/Applications of Software

- System software
- Application software
- Engineering/scientific software
- Embedded software
- Product line software
- Web applications
- Artificial intelligence software

System Software:

- System software is a collection of programs written to service other programs.
- It is characterized by heavy interaction with computer hardware; heavy usage by multiple users; concurrent operation that requires scheduling, resource sharing, and sophisticated process management; complex data structures; and multiple external interfaces.

Ex. Compilers, operating system, drivers etc.

Application Software :

- Application software consists of standalone programs that solve a specific business need.
- Application software is used to control the business function in real-time.

Engineering /Scientific software:

- Characterized by "number crunching" algorithms.
- Applications range from astronomy to volcano logy, from automotive stress analysis to space shuttle orbital dynamics, and from molecular biology to automated manufacturing.

Ex. Computer Aided Design (CAD), system stimulation etc.

Embedded Software:

- It resides in read-only memory and is used to control products and systems
- Embedded software can perform limited and esoteric functions.

Ex. keypad control for a microwave oven.

Product line software:

- Designed to provide a specific capability for use by many different customers, product line software can focus on a limited and esoteric marketplace.

Ex. Word processing, spreadsheet, CG, multimedia, etc.

Web Applications:

- Web apps can be little more than a set of linked hypertext files.
- It evolving into sophisticated computing environments that not only provide standalone features, functions but also integrated with corporate database and business applications.

Artificial Intelligence software

- AI software makes use of non-numerical algorithms to solve complex problems that are not amenable to computation or straightforward analysis

Ex. Robotics, expert system, game playing, etc.

Software Myths

Definition: Beliefs about software and the process used to build it. Myths have number of attributes that have made them insidious (i.e. dangerous).

Misleading Attitudes - caused serious problem for managers and technical people.

Management myths

Managers in most disciplines, are often under pressure to maintain budgets, keep schedules on time, and improve quality.

Myth1: 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 :

- Are software practitioners aware of existence standards?
- Does it reflect modern software engineering practice?
- Is it complete? Is it streamlined to improve time to delivery while still maintaining a focus on quality?

Myth2*:* If we get behind schedule, we can add more programmers and catch up

Reality*:* Software development is not a mechanistic process like manufacturing. Adding people to a late software project makes it later.

- People can be added but only in a planned and well-coordinated manner

Myth3*:* If I decide to outsource the software project to a third party, I can just relax and let that firm build it.

Reality*:* If an organization does not understand how to manage and control software projects internally, it will invariably struggle when it outsource software projects

Customer Myths

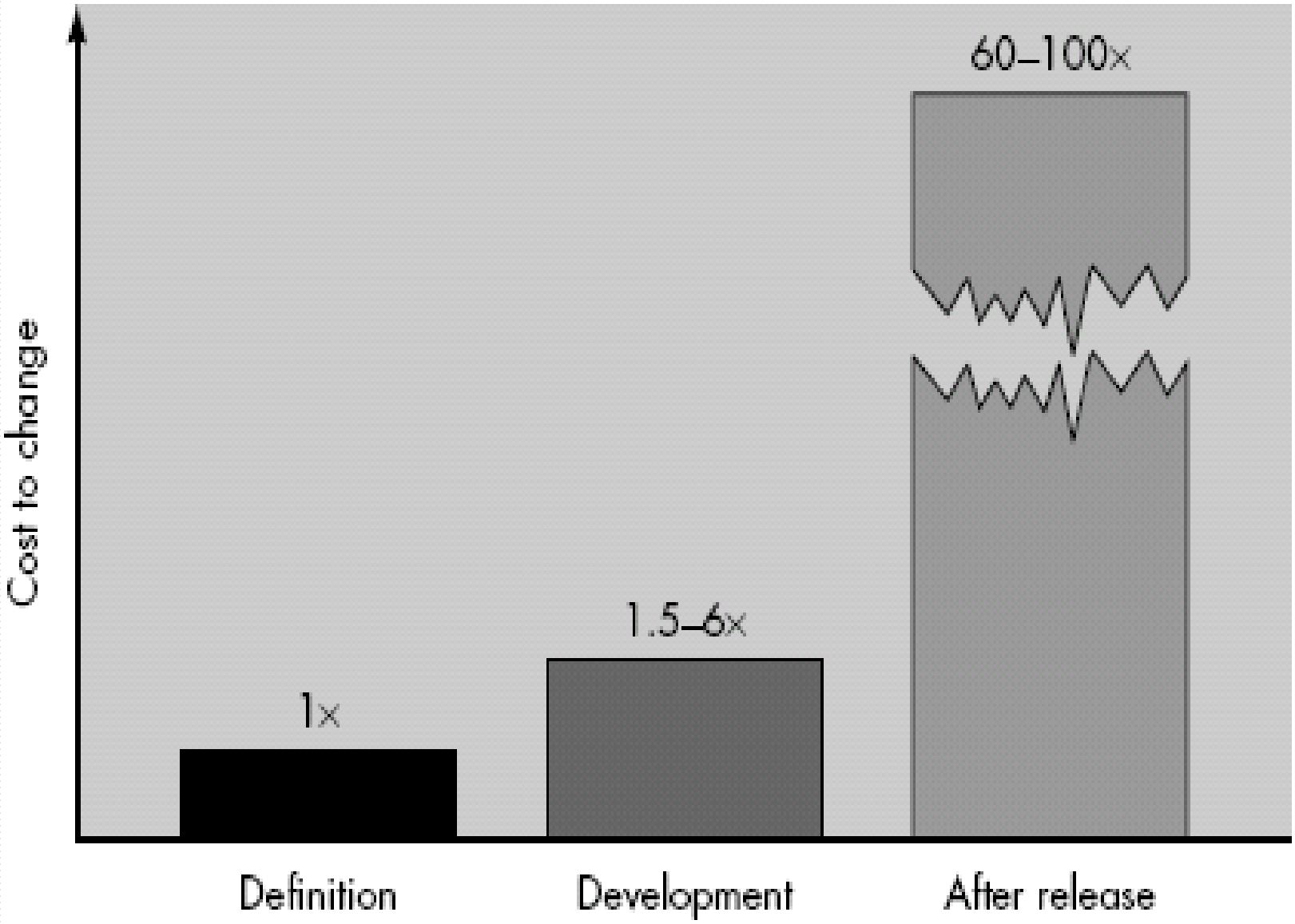
Customer may be a person from inside or outside the company that has requested software under contract.

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

Reality*:* A poor up-front definition is the major cause of failed software efforts. A formal and detailed description of the information domain, function, behavior, performance, interfaces, design constraints, and validation criteria is essential. These characteristics can be determined only after thorough communication between customer and developer.

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

Reality: Customer can review requirements and recommend modifications with relatively little impact on cost. When changes are requested during software design, the cost impact grows rapidly. Below mentioned *figure* for reference.


Cost to change
60–100×
1.5–6×
1×
Definition
Development
After release

Practitioner's myths

Myth1*:* Once we write the program and get it to work, our job is done.

Reality*:* Someone once said that "the sooner you begin 'writing code', the longer it'll take you to get done." Industry data indicate that between 60 and 80 percent of all effort expended on software will be expended after it is delivered to the customer for the first time.

Myth2*:* Until I get the program "running" I have no way of assessing its quality.

Reality: One of the most effective software quality assurance mechanisms can be applied from the inception of a project—the *formal technical review.*

Myth3: The only deliverable work product for a successful project is the working program.

Reality*:* A working program is only one part of a *software configuration* that includes many elements. Documentation provides a foundation for successful engineering and, more important, guidance for software support.

Myth4 *:* Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.

Reality*:* Software engineering is not about creating documents. It is about creating quality. Better quality leads to reduced rework. And reduced rework results in faster delivery times.