An Introduction to Software Engineering

WHAT'S SOFTWARE ENGINEERING?

“Software Systems are perhaps the most intricate and complex … of the things humanity makes.”

-Fred Brooks

What is Software?

- Software is developed or **engineered, it is not manufactured** in the classical sense.
- Software **doesn't wear out**, but deterioration.
- Although the industry is moving toward component-based construction, most **software continues to be custom-built.**

What is Software?

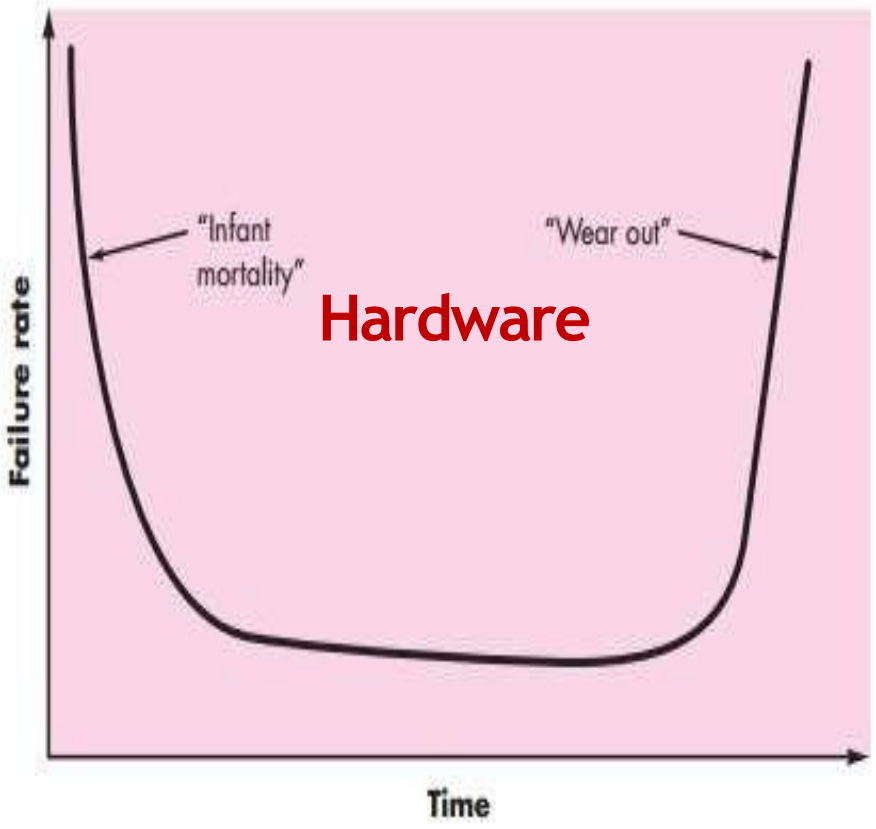
Software is:

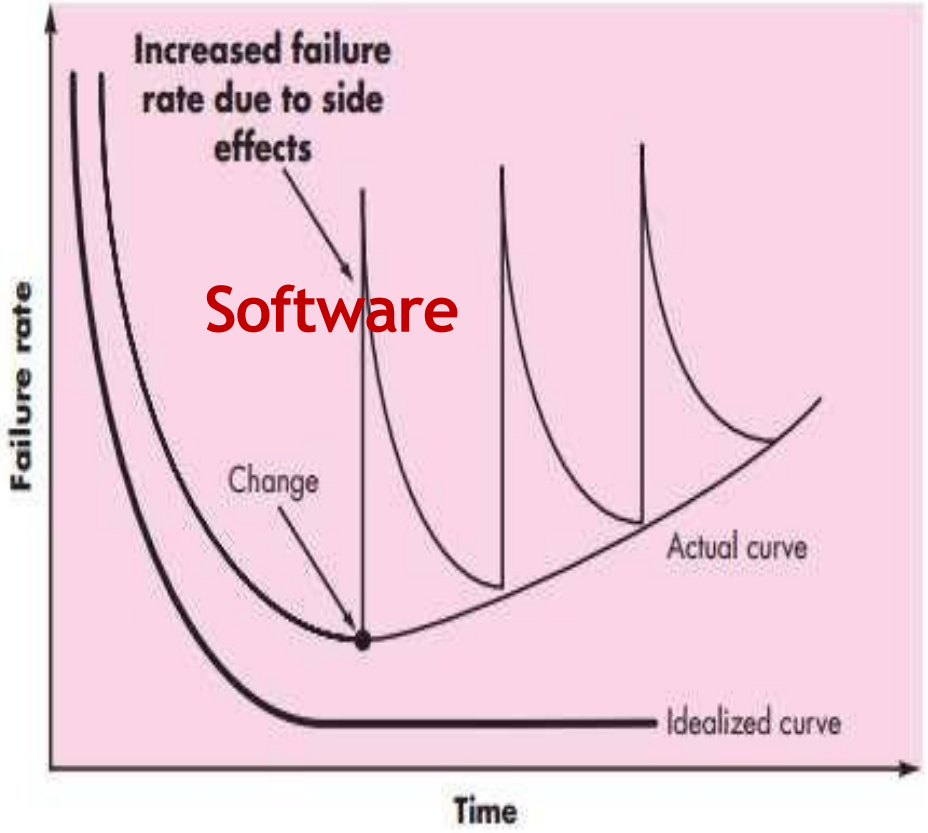
(1) instructions (computer programs) that when executed provide desired features, function, and performance;

(2) data structures that enable the programs to adequately manipulate information and

(3) documentation that describes the operation and use of the programs.

Wear-out vs. Deterioration





Engineering: A definition

"The systematic and regular application of **scientific and mathematical knowledge** to the design, construction, and operation of machines, systems, and so on of **practical use** and, hence, of **economic value**. Particular characteristic of engineers is that they take seriously their **responsibility for correctness**, **suitability, and safety of the results of their efforts**. In this regard they consider themselves to be responsible to their customer (including their employers where relevant), to the users of their machines and systems, and to the public at large."

-***Robert Baber***

Software Engineering

- Has progressed very far in a short time:
  - 50 years ago, most programming were done by Scientists trying to solve mathematical problems.
  - Today, we build monstrous systems used everywhere.
- But, software industry is in crisis:
  - A software project overshoots its schedule by a half.
  - Three quarters of all large systems are operating failures.

Software Engineering as Engineering

- Practical use, economic value:
  - We need to determine the content and build the best product value.
- Responsibility for correctness, suitability and safety:
  - The work you do could impact the safety, business and well being of the customer.
- Regular application of scientific and mathematical knowledge:
  - Computer Science, psychology, economics and management

Software Engineering: A Definition

"Software engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software"

-IEEE

What is Software Engineering?

- The term **software engineering** is the product of two words, **software**, and **engineering**.
- The **software** is a collection of integrated programs.
- Software subsists of carefully-organized instructions and code written by developers on any of various particular computer languages.
- Computer programs and related documentation such as requirements, design models and user manuals.
- **Engineering** is the application of **scientific** and **practical** knowledge to **invent, design, build, maintain**, and **improve frameworks, processes, etc**.


Software Product
Deployment
Testing
Code Design
Requirements
System Analysis
System Design
Maintenance
Updates

Software Engineering is an engineering branch related to the **evolution** of software product using well-defined **scientific principles, techniques, and procedures**. The result of software engineering is an effective and reliable software product.

Why is Software Engineering required?

- Software Engineering is required due to the following reasons:
- To manage Large software
- For more Scalability
- Cost Management
- To manage the dynamic nature of software
- For better quality Management

Need of Software Engineering

The necessity of software engineering appears because of a higher rate of progress in user requirements and the environment on which the program is working.

Huge Programming: It is simpler to manufacture a wall than to a house or building, similarly, as the measure of programming become extensive engineering has to step to give it a scientific process.

Adaptability: If the software procedure were not based on scientific and engineering ideas, it would be simpler to re-create new software than to scale an existing one.

Cost: As the hardware industry has demonstrated its skills and huge manufacturing has let down the cost of computer and electronic hardware. But the cost of programming remains high if the proper process is not adapted.

Dynamic Nature: The continually growing and adapting nature of programming hugely depends upon the environment in which the client works. If the quality of the software is continually changing, new upgrades need to be done in the existing one.

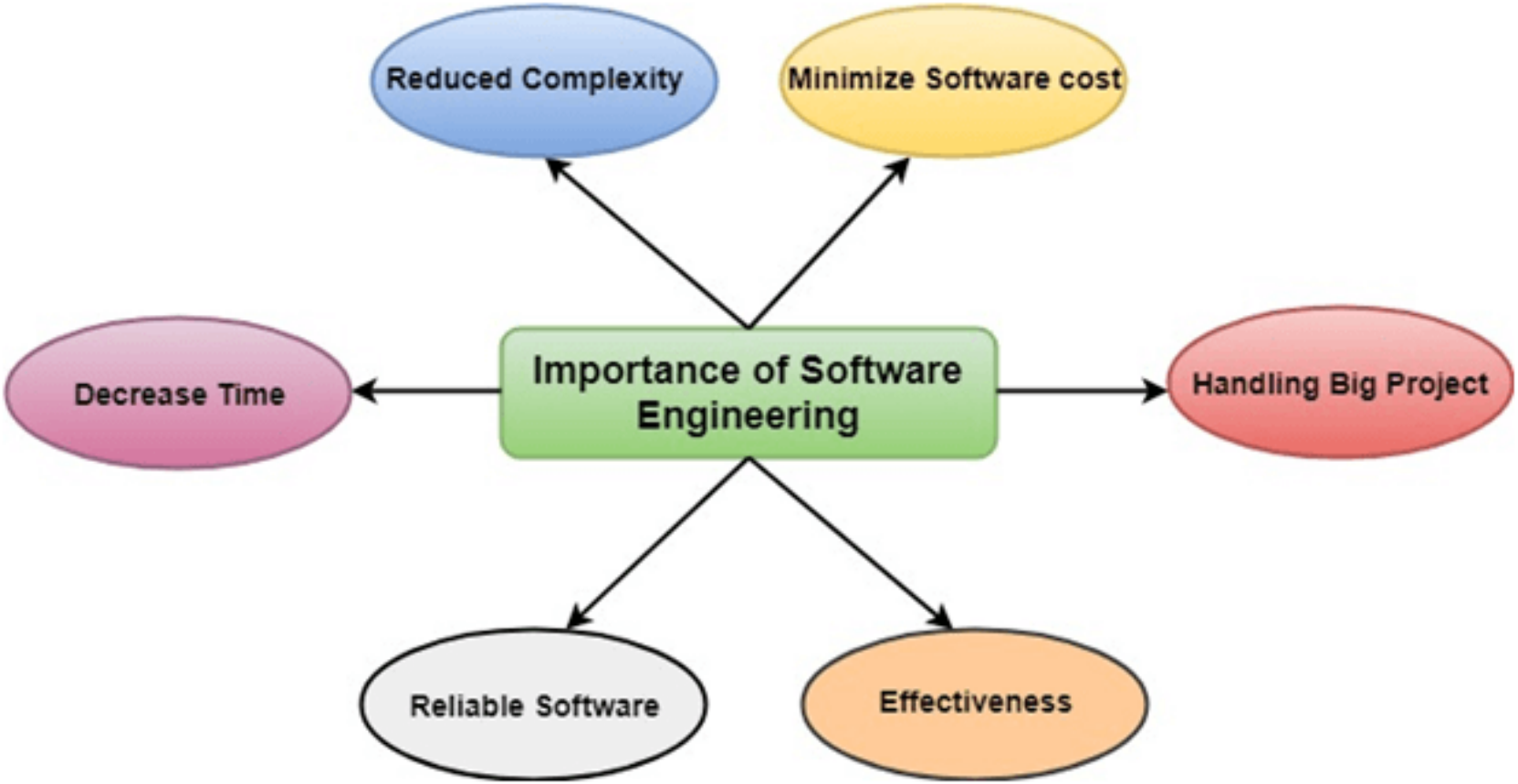
Quality Management: Better procedure of software development provides a better and quality software product.

Characteristics of a good software engineer

The features that good software engineers should possess are as follows:

- Exposure to systematic methods, i.e., familiarity with software engineering principles.
- Good technical knowledge of the project range (Domain knowledge).
- Good programming abilities.
- Good communication skills. These skills comprise of oral, written, and interpersonal skills.
- High motivation.
- Sound knowledge of fundamentals of computer science.
- Intelligence.
- Ability to work in a team
- Discipline, etc.

Importance of Software Engineering



The importance of Software engineering is as follows:

Reduces complexity: Big software is always complicated and challenging to progress. Software engineering has a great solution to reduce the complication of any project. Software engineering divides big problems into various small issues. And then start solving each small issue one by one. All these small problems are solved independently to each other.

To minimize software cost: Software needs a lot of hardwork and software engineers are highly paid experts. A lot of manpower is required to develop software with a large number of codes. But in software engineering, programmers project everything and decrease all those things that are not needed. In turn, the cost for software productions becomes less as compared to any software that does not use software engineering method.

To decrease time: Anything that is not made according to the project always wastes time. And if you are making great software, then you may need to run many codes to get the definitive running code. This is a very time-consuming procedure, and if it is not well handled, then this can take a lot of time. So if you are making your software according to the software engineering method, then it will decrease a lot of time.

Handling big projects: Big projects are not done in a couple of days, and they need lots of patience, planning, and management. And to invest six and seven months of any company, it requires heaps of planning, direction, testing, and maintenance. No one can say that he has given four months of a company to the task, and the project is still in its first stage. Because the company has provided many resources to the plan and it should be completed. So to handle a big project without any problem, the company has to go for a software

Reliable software: Software should be secure, means if you have delivered the software, then it should work for at least its given time or subscription. And if any bugs come in the software, the company is responsible for solving all these bugs. Because in software engineering, testing and maintenance are given, so there is no worry of its reliability.

Effectiveness: Effectiveness comes if anything has made according to the standards. Software standards are the big target of companies to make it more effective. So Software becomes more effective in the act with the help of software engineering.

FAQ about Software Engineering

Question	Answer
What is software?	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed or 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.

FAQ about Software Engineering

Question	Answer
What are the fundamental software engineering activities?	Software specification, software development, software validation and software evolution.
What is the difference between software engineering nd 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 rocess engineering. Software engineering is part of this more general process.

FAQ about Software Engineering

Question	Answer
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. You can't, therefore, say that one method is better than another.

What differences has the web made to software engineering?	The web has led to the availability of software services and the possibility of developing highly distributed service-based systems. Web-based systems development has led to important advances in programming languages and software reuse.

Essensial Atributes 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 Myths (Management)

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

The book of standards may very well exist, but is it used? Are software practitioners aware of its existence? Does it reflect modern software engineering practice? Is it complete? Is it adaptable? Is it streamlined to improve time-to-delivery while still maintaining a focus on quality? In many cases, the answer to all of these questions is "no."

Software Myths (Management)

Myth

If we get behind schedule, we can add more programmers and catch up (sometimes called the "Mongolian horde" concept).

Reality

Software development is not a mechanistic process like manufacturing. In the words of Brooks: "adding people to a late software project makes it later." At first, this statement may seem counterintuitive.

However, as new people are added, people who were working must spend time educating the newcomers, thereby reducing the amount of time spent on productive development effort. People can be added but only in a planned and well coordinated manner.

Software Myths (Management)

Myth

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 outsources software projects.

Software Myths (Customer)

Myth

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

Reality

Although a comprehensive and stable statement of requirements is not always possible, an ambiguous "statement of objectives" is a recipe for disaster.

Unambiguous requirements (usually derived iteratively) are developed only through effective and continuous communication between customer and developer

Software Myths (Customer)

Myth

Software requirements continually change, but change can be easily accommodated because software is flexible.

Reality

It is true that software requirements change, but the impact of change varies with the time at which it is introduced. When requirements changes are requested early (before design or code has been started), the cost impact is relatively small.16 However, as time passes, the cost impact grows rapidly — resources have been committed, a design framework has been established, and change can cause upheaval that requires additional resources and major design modification.

Software Myths (Practitioners)

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

Software Myths (Practitioners)

Myth

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 technical review.
Software reviews are a “quality filter” that have been found to be more effective than testing for finding certain classes of software defects.

Software Myths (Practitioners)

Myth

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. A variety of work products (e.g., models, documents, plans) provide a foundation for successful engineering and, more important, guidance for software support

Software Myths (Practitioners)

Myth	Reality
Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.	Software engineering is not about creating documents. It is about creating a quality product. Better quality leads to reduced rework. And reduced rework results in faster delivery times.

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).