Introduction to Software Engineering

Ashima Tyagi
Assistant Professor
School of Computer Science & Engineering

Outline

- Software & Engineering
- Software Engineering
- Software components
- Software characteristics
- Software crisis
- Software Attributes

Software & Engineering???

- Software is a program or set of programs containing instructions that provide desired functionality.
- Engineering is the process of designing and building something that serves a particular purpose and finds a cost-effective solution to problems.

Software Engineering

Software Engineering is the process of designing, developing, testing, and maintaining software.
It is a systematic and disciplined approach to software development that aims to create high-quality, reliable, and maintainable software.

Software Components

1. Program –

- A computer program is a list of instructions that tell a computer what to do.
- In software engineering, a program is a set of instructions written in a programming language to perform a specific task or solve a particular problem.
- It consists of algorithms, data structures, and control flow mechanisms, organized systematically to achieve the desired functionality within a computer system.

2. Documentation –

- Source information about the product contained in design documents, detailed code comments, etc.
- Software documentation in software engineering encompasses written descriptions of a software system's architecture, design, requirements, and functionality.
- It includes manuals, user guides, technical specifications, API documentation, and code comments.
- Documentation aids in understanding, using, maintaining, and evolving software systems throughout their lifecycle.

3. Operating Procedures –

- Set of step-by-step instructions compiled by an organization to help workers carry out complex routine operations.
- Operating procedures in software engineering are *documented guidelines outlining the steps and protocols for deploying, maintaining, and managing software systems.*
- They ensure consistency, reliability, and efficiency in software operations, covering tasks like installation, configuration, troubleshooting, and regular maintenance activities.

Software Characteristics

1. **Software is developed or engineered; it is not manufactured in the classical sense:**

Although some similarities exist between software development and hardware manufacturing, few activities are fundamentally different.

In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems than software.

2. **The software doesn't "wear out.":**

Hardware components suffer from the growing effects of many other environmental factors. Stated simply, the hardware begins to wear out.

Software is not susceptible to the environmental maladies that cause hardware to wear out.

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

Software Characteristics

There are no software spare parts.

Every software failure indicates an error in design or in the process through which the design was translated into machine-executable code. Therefore, the software maintenance tasks that accommodate requests for change involve considerably more complexity than hardware maintenance. However, the implication is clear—the software doesn't wear out. But it does deteriorate.

3. The software continues to be custom-built:

A software part should be planned and carried out with the goal that it tends to be reused in various projects.

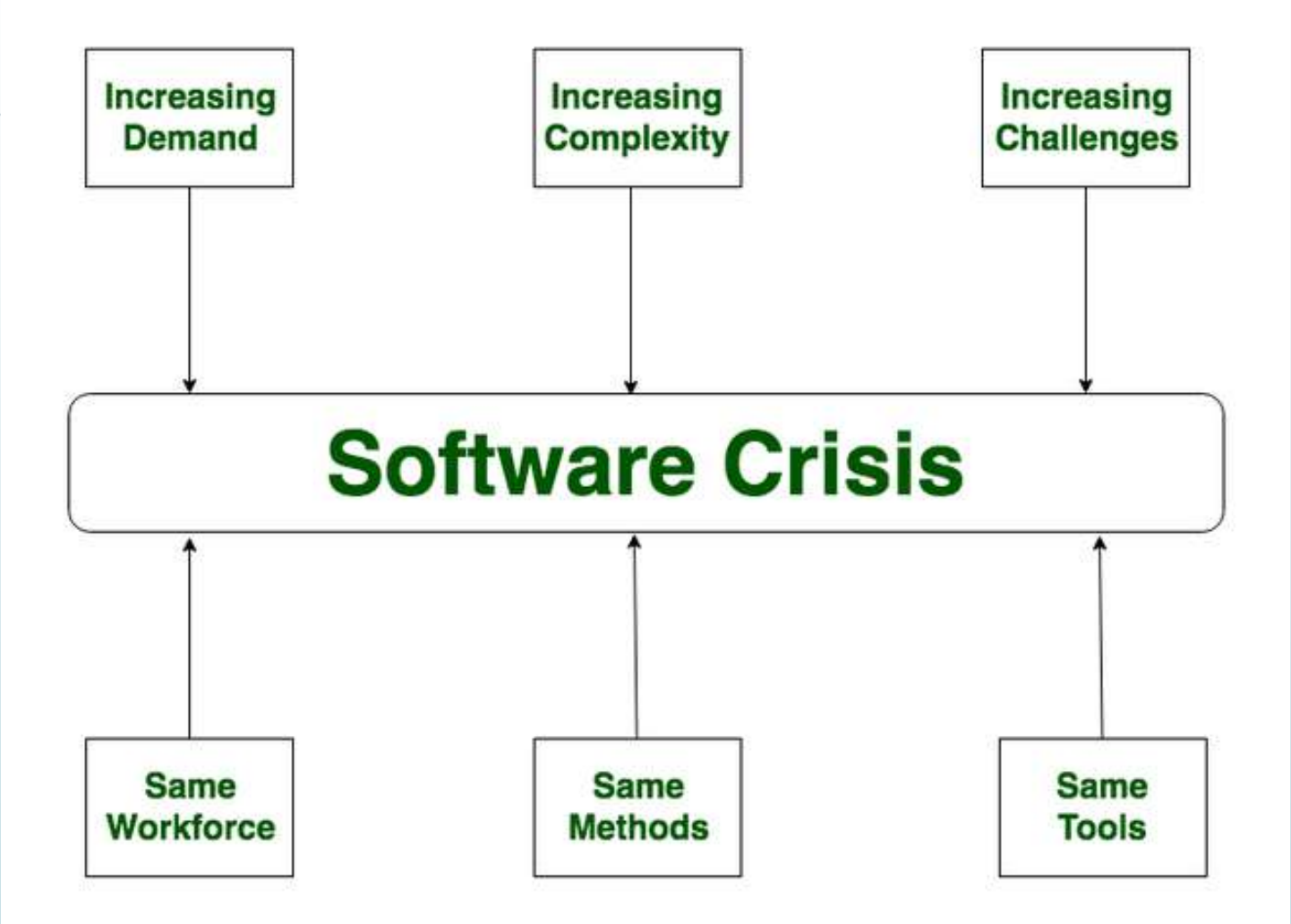
Current reusable segments encapsulate the two pieces of information and the preparation that is applied to the information, empowering the programmer to make new applications from reusable parts.

In the hardware world, component reuse is a natural part of the engineering process.

Software Crisis

Software Crisis is a term used in computer science for the difficulty of writing useful and efficient computer programs in the required time. The software crisis was due to using the same workforce, same methods, and same tools even though rapidly increasing software demand, the complexity of software, and software challenges. With the increase in software complexity, many software problems arise because existing methods were insufficient.

If we use the same workforce, same methods, and same tools after the fast increase in software demand, software complexity, and software challenges, then there arise some issues like software budget problems, software efficiency problems, software quality problems, software management, and delivery problems, etc. This condition is called a Software Crisis.


Increasing Demand
Increasing Complexity
Increasing Challenges
Software Crisis
Same Workforce
Same Methods
Same Tools

Causes of Software Crisis:

- The cost of owning and maintaining software was as expensive as developing the software.
- At that time Projects were running overtime.
- At that time Software was very inefficient.
- The quality of the software was low quality.
- Software often did not meet user requirements.
- The average software project overshoots its schedule by half.
- At that time Software was never delivered.
- Non-optimal resource utilization.
- Challenging to alter, debug, and enhance.
- The software complexity is harder to change.

Factors Contributing to Software Crisis:

- Poor project management.
- Lack of adequate training in software engineering.
- Less skilled project members.
- Low productivity improvements.

Solution of Software Crisis:

There is no single solution to the crisis. One possible solution to a software crisis is Software Engineering because software engineering is a systematic, disciplined, and quantifiable approach. For preventing software crises, there are some guidelines:

- Reduction in software over budget.
- The quality of the software must be high.
- Less time is needed for a software project.
- Experienced and skilled people working on the software project.
- Software must be delivered.
- Software must meet user requirements.

MAIN ATTRIBUTES OF SOFTWARE ENGINEERING

EFFICIENCY

Efficiency in software refers to how well the software uses system resources (like CPU, memory, disk, and network) to perform its tasks.

Example: A web browser that loads pages quickly and consumes minimal RAM is considered efficient.

RELIABLITY

Reliability refers to the software's ability to consistently perform its intended functions without failure under specified conditions over time.

Example: Banking software that processes transactions without errors, even during peak loads, is considered reliable.

REUSABILITY

Reusability refers to software's ability to use existing modules in new applications without modification.

Example: A library of reusable functions for processing user data can be integrated into multiple projects without rewriting the code.

MAINTAINABILITY

Maintainability is the ease with which software can be updated, fixed, or improved to meet new requirements or correct defects.

Example: A software where developers can quickly add new features or fix bugs is maintainable.

Thank You