Unit 1
Software Engineering

Prepared By

Abhishek Kesharwani

Assistant Professor ,United College of Engineering and Research

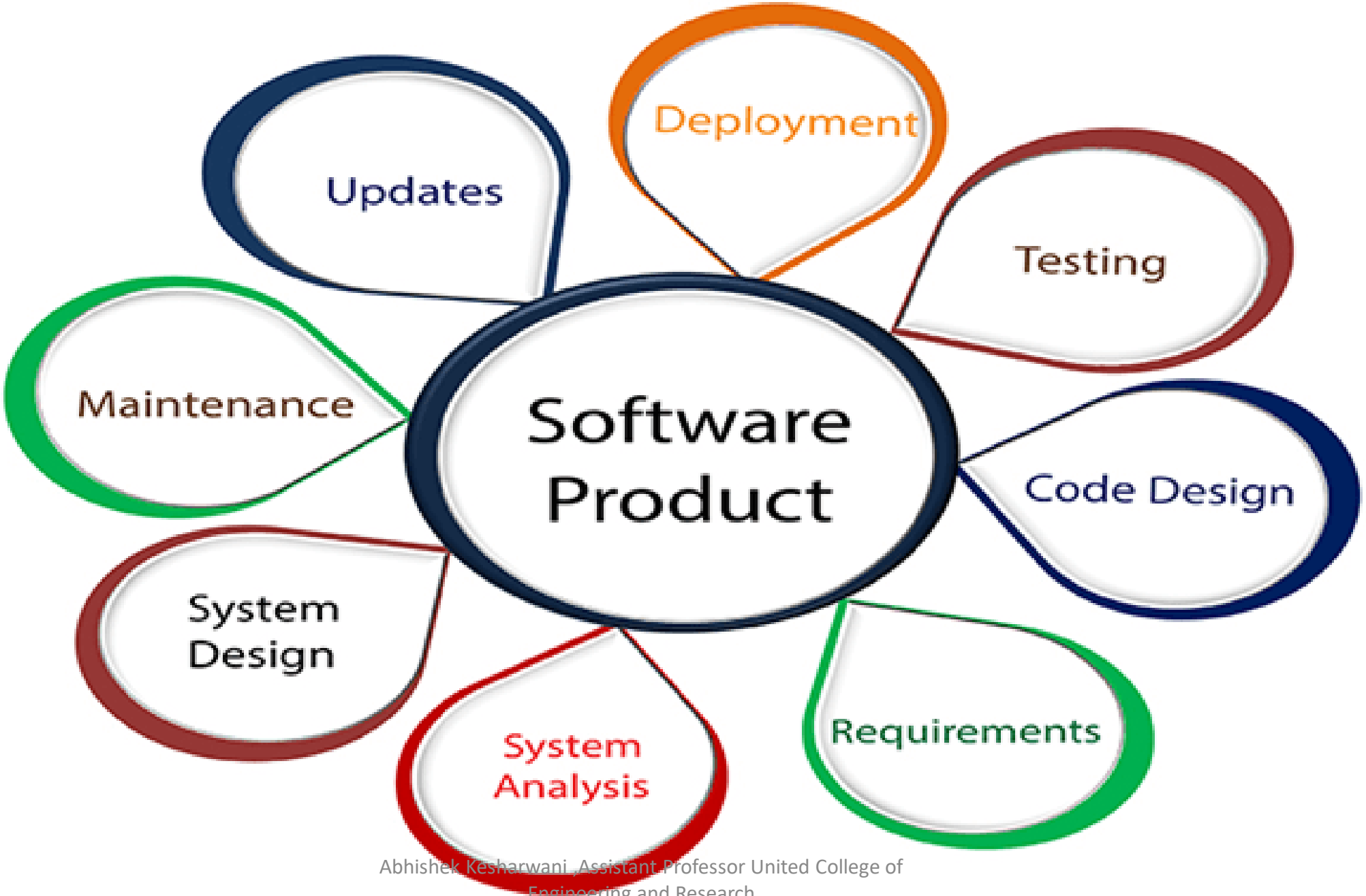
Index

- Introduction to Software Engineering
- Software Components
- Software Characteristics
- Software Crisis
- Software Engineering Processes
- Similarity and Differences from Conventional Engineering Processes
- Software Quality Attributes.

Introduction to 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**.


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

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.

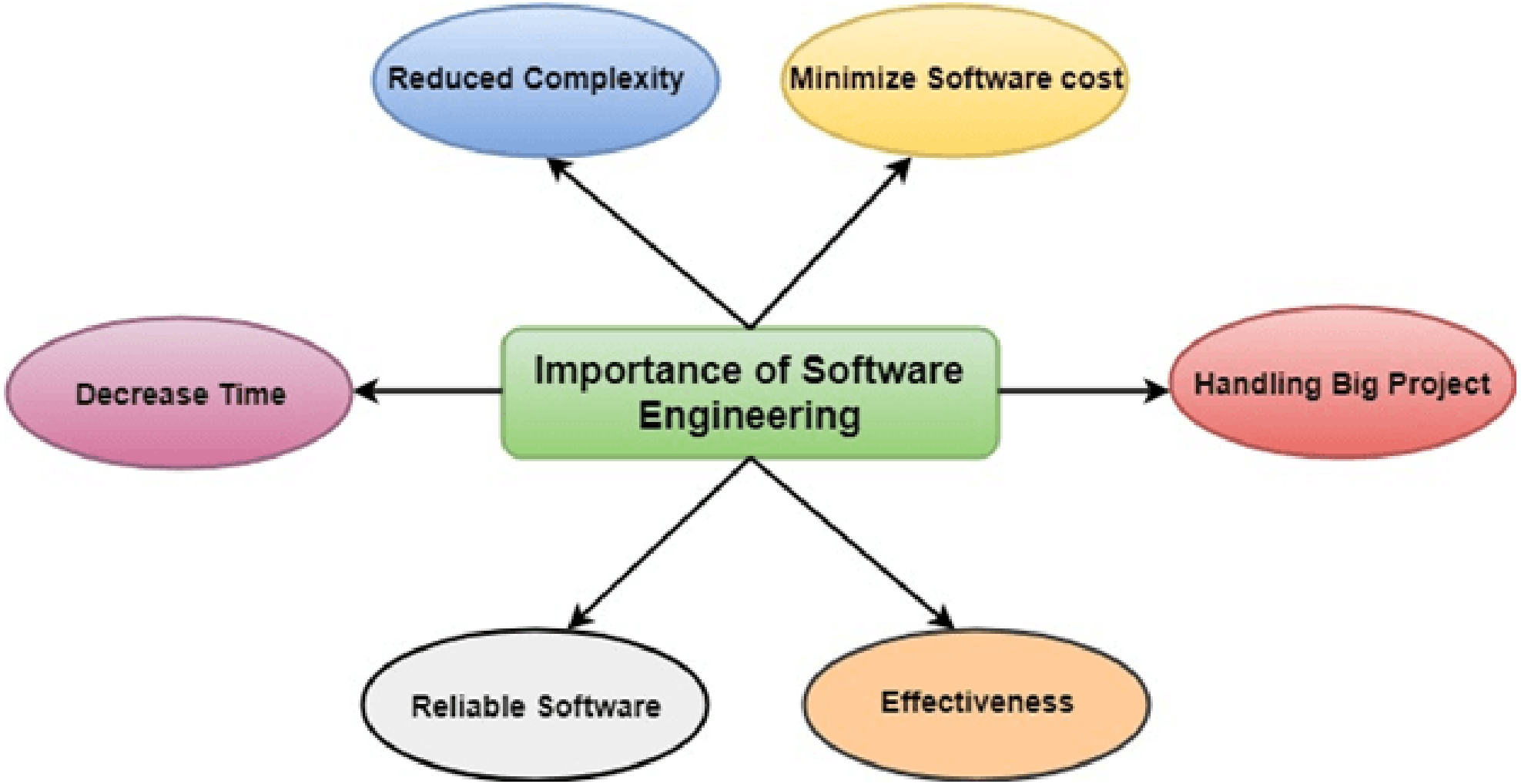
- **Dynamic Nature:** The continually growing and adapting nature of programming hugely depends upon the environment in which the client works. If the quality of the software is continually changing, new upgrades need to be done in the existing one.
- **Quality Management:** Better procedure of software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

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 hard work 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 engineering method.

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.

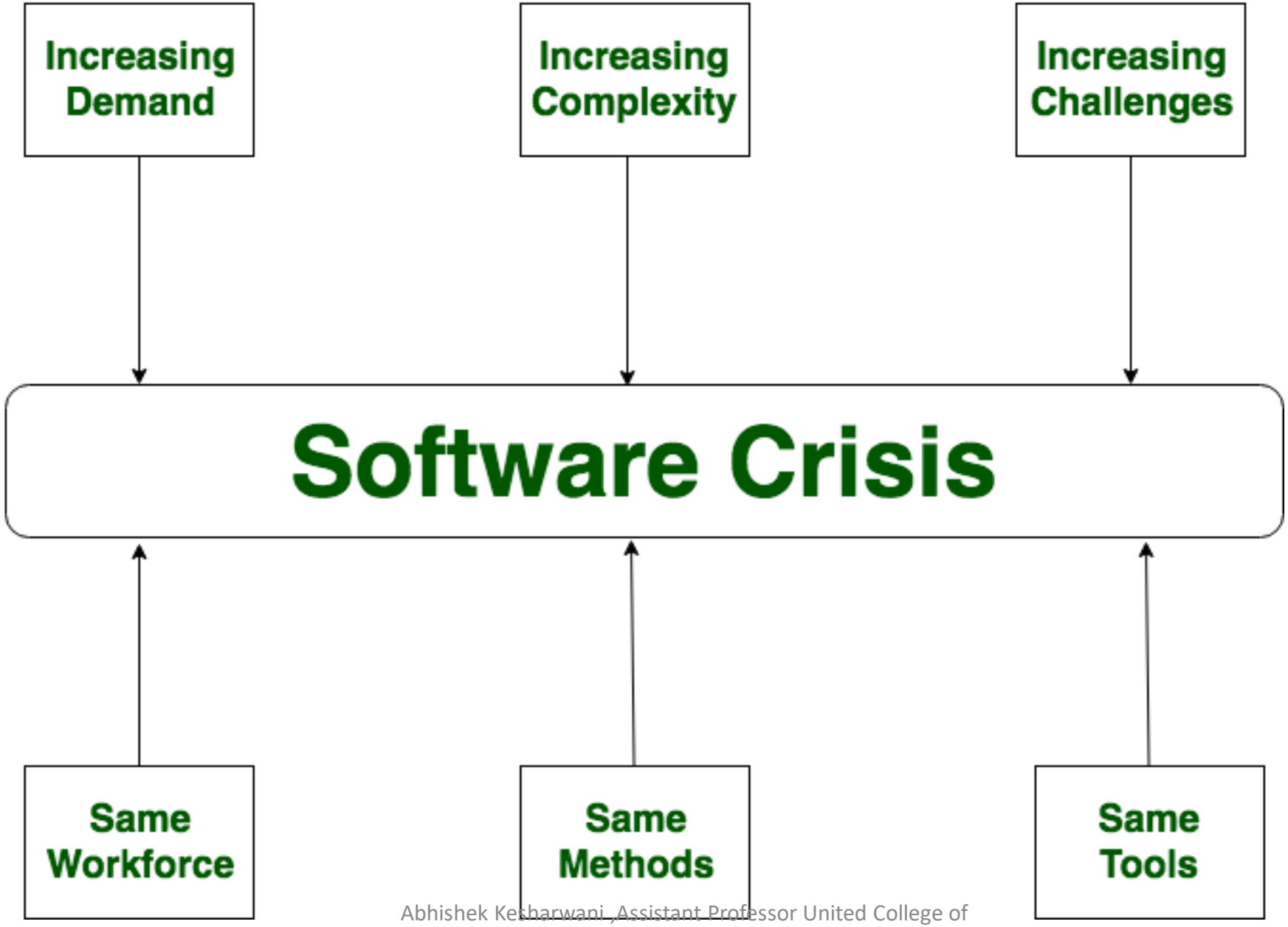
Effectiveness:

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.

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 .software crisis was due to using same workforce, same methods, same tools even though rapidly increasing in software demand, complexity of software and software challenges.
- With increase in the complexity of software, many software problems arise because existing methods were insufficient.

- If we will use same workforce, same methods and same tools after fast increasing in software demand, software complexity and software challenges, then there arise some problems like software budget problem, software efficiency problem, software quality problem, software managing and delivering problem etc. This condition is called 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 was running over-time
- At that time Software was very inefficient
- The quality of software was low quality
- Software often did not meet requirements
- The average software project overshoots its schedule by half
- At that time Software was never delivered

Solution of Software Crisis

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

- Reduction in software over-budget
- The quality of software must be high
- Less time needed for software project
- Experience working team member on software project
- Software must be delivered

Questions Asked in Different Software Companies

1. **What are the characteristics of the software? [TCS Interview]**
2. **What are the various categories of software? [Wipro Interview]**
3. **What are the challenges in software? [Wipro Interview]**

Software Engineering Processes

- The term **software** specifies to the set of computer programs, procedures and associated documents (Flowcharts, manuals, etc.) that describe the program and how they are to be used.
- A software process is the set of activities and associated outcome that produce a software product. Software engineers mostly carry out these activities.

These are four key process activities, which are common to all software processes. These activities are:

1. **Software specifications:** The functionality of the software and constraints on its operation must be defined.
2. **Software development:** The software to meet the requirement must be produced.
3. **Software validation:** The software must be validated to ensure that it does what the customer wants.
4. **Software evolution:** The software must evolve to meet changing client needs.

The Software Process Model

- A software process model is a specified definition of a software process, which is presented from a particular perspective.
- Models, by their nature, are a simplification, so a software process model is an abstraction of the actual process, which is being described.
- Process models may contain activities, which are part of the software process, software product, and the roles of people involved in software engineering.

Some examples of the types of software process models that may be produced are:

1. A workflow model: This shows the series of activities in the process along with their inputs, outputs and dependencies. The activities in this model perform human actions.

2. A dataflow or activity model: This represents the process as a set of activities, each of which carries out some data transformations. It shows how the input to the process, such as a specification is converted to an output such as a design. The activities here may be at a lower level than activities in a workflow model. They may perform transformations carried out by people or by computers.

3. A role/action model: This means the roles of the people involved in the software process and the activities for which they are responsible.

There are several various general models or paradigms of software development:

- **The waterfall approach:** This takes the above activities and produces them as separate process phases such as requirements specification, software design, implementation, testing, and so on. After each stage is defined, it is "signed off" and development goes onto the following stage.
- **Evolutionary development:** This method interleaves the activities of specification, development, and validation. An initial system is rapidly developed from a very abstract specification.

- **Formal transformation:** This method is based on producing a formal mathematical system specification and transforming this specification, using mathematical methods to a program. These transformations are 'correctness preserving.' This means that you can be sure that the developed programs meet its specification.
- **System assembly from reusable components:** This method assumes the parts of the system already exist. The system development process target on integrating these parts rather than developing them from scratch.

Difference between Software Engineering process and Conventional Engineering Process

S.No.	Software Engineering Process	Conventional Engineering Process
1.	Software Engineering Process is a process which majorly involves computer science, information technology and discrete mathematics.	Conventional Engineering Process is a process which majorly involves science, mathematics and empirical knowledge.
2.	It is mainly related with computers, programming and writing codes for building applications.	It is about building cars, machines, hardware, buildings etc.
3.	In Software Engineering Process construction and development cost is low.	In Conventional Engineering Process construction and development cost is high.

S.No.	Software Engineering Process	Conventional Engineering Process
4.	It can involve the application of new and untested elements in software projects.	It usually applies only known and tested principles to meet product requirements.
5.	In Software Engineering Process, most development effort goes into building new designs and features.	In Conventional Engineering Process, most development efforts are required to change old designs.
6.	It majorly emphasize on quality.	It majorly emphasize on mass production.

Software Quality Attributes

Software Quality Attributes are features that facilitate the measurement of **performance of a software product** by Software Testing professionals, and include attributes such as availability, interoperability, correctness, reliability, learnability, robustness, maintainability, readability, extensibility, testability, efficiency, and portability.

High scores in Software Quality Attributes enable software architects to guarantee that a software application will perform as the specifications provided by the client.

1. Availability

This attribute is indicative as to whether an application will execute the tasks it is assigned to perform.

Availability also includes certain concepts that relate to software security, **performance**, integrity, reliability, dependability, and confidentiality. In addition, top-notch availability indicates that a software-driven system will repair any operating faults so that service outage periods would not exceed a specific time value.

2. Interoperability

- Software-driven systems could be required to communicate and act in tandem to solve certain tasks. Interoperability describes the ability of two systems to engage in the exchange of information via certain interfaces.
- Therefore, **Software Quality Assurance** engineers must examine the interoperability attribute in terms of both syntactic and semantic interoperability.

3. Performance

- This attribute pertains to the ability of a software-driven system to conform to timing requirements. From a testing point of view, it implies that Software Testing engineers must check whether the system responds to various events within defined time limits. These events may occur in the form of clock events, process interruptions, messages, and requests from different users, and others.

4. Testability

- Software testability indicates how well a software-driven system allows Software Testing professionals to conduct tests in line with predefined criteria.
- This attribute also assesses the ease with which Software Quality Assurance engineers can develop test criteria for a said system and its various components.
- Engineers can assess the **testability of a system** by using various techniques such as encapsulation, interfaces, patterns, low coupling, and more.

5. Security

- This attribute measures the ability of a system to arrest and block malicious or unauthorized actions that could potentially destroy the system.
- The attribute assumes importance because security denotes the ability of the system to protect data and defend information from unauthorized access.
- Security also includes authorization and authentication techniques, protection against **network attacks**, data encryption, and such other risks. It is imperative for Software Testing Companies and professionals to regularly conduct updated security checks on systems.

6. Usability

- Every software-driven system is designed for ease of use to accomplish certain tasks.
- The attribute of usability denotes the ease with which users are able to execute tasks on the system; it also indicates the kind of user support provided by the system.
- The most well-known principle for this property is KISS (Keep It Simple Stupid). In addition, Software Quality Assurance engineers must test software to check whether it supports different accessibility types of control for people with disabilities.
- Usability has a critical and long standing bearing on the commercial fortunes of a software application or package.

7. Functionality

- This attribute determines the conformity of a software-driven system with actual requirements and specifications.
- Most Software Testing professionals view this attribute as crucial and a foremost requirement of a modern application, and would therefore advocate the **performance of tests** that assess the desired functionality of a system in the initial stages of Software Testing initiatives.

Questions Asked in Different Software Companies

1. **Define Software process. [TCS Interview]**