24CSA512-Software Engineering and Design Patterns

Unit I

Software Engineering – Introduction - Software Classification - Layered Technology - Software Process –Practice – SDLC - Generic Process Model, Process Assessment.

Unit II

Perspective Models - Agile Process Models – Scrum and Extreme Programming (XP) - Requirements Analysis - Unified Modelling Language – Design Engineering – Test Engineering.

Unit III

Introduction: What Is a Design Pattern? Describing Design Patterns, The Catalogue of Design Patterns, Organizing the Catalogue, How Design Patterns Solve Design Problems, How to Select a Design Pattern, How to Use a Design Pattern.

Unit IV

Creational Patterns: Abstract Factory, Factory Method, Singleton Structural Patterns: Adapter, Bridge, Composite, Decorator, Façade, and Proxy. Behavioral Patterns: Command, Iterator, Observer and Template Method. Case study on design patterns.

UNIT-I

Unit I

Software Engineering – Introduction - Software Classification - Layered Technology - Software Process –Practice – SDLC - Generic Process Model, Process Assessment.

- Computer software is a product or program code developed by software engineers.
- The applications of computer software are: Telecommunication, military, medical sciences, online shopping, office products, IT industry etc.
- A Software consists of data and the related documents.
- The software is the key element in all computer based systems and products.
- The main purpose behind software engineering is to give a framework for building a software with best quality.

- Software is defined as a collection of programs, documentation and operating procedures.
- The **Institute of Electrical and Electronic Engineers (IEEE)** defines software as a ‘collection of computer programs, procedures, rules and associated documentation and data.’

- ***Software Engineering*** *is a systematic, disciplined, quantifiable study and approach to the design, development, operation, and maintenance of a software system.*
- *The establishment and use of sound engineering principles in order to obtain economical software that is reliable and works efficiently on real machines.*
- *Software engineering is a systematic and disciplined approach towards the development of the software operation and maintenance.*
- *Software engineering is an engineering branch associated with the development of software product using well-defined scientific principles, methods and procedures.*

- 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

Characteristics of a software

- Software should achieve a good quality in design and meet all the specifications of the customer.
- Software does not wear out i.e. it does not lose the material.
- Software should be inherently complex.
- Software must be efficient i.e. the ability of the software to use system resources in an effective and efficient manner.
- Software must be integral i.e. it must prevent from unauthorized access to the software or data.

- **Objectives of Software Engineering:**

1. **Maintainability –**
   It should be feasible for the software to evolve to meet changing requirements.
2. **Efficiency –**
   The software should not make wasteful use of computing devices such as memory, processor cycles, etc.
3. **Correctness –**
   A software product is correct if the different requirements as specified in the SRS document have been correctly implemented.
4. **Reusability –**
   A software product has good reusability if the different modules of the product can easily be reused to develop new products.
5. **Testability –**
   Here software facilitates both the establishment of test criteria and the evaluation of the software with respect to those criteria.
6. **Reliability –**
   It is an attribute of software quality. The extent to which a program can be expected to perform its desired function, over an arbitrary time period.
7. **Portability –**
   In this case, the software can be transferred from one computer system or environment to another.
8. **Adaptability –**
   In this case, the software allows differing system constraints and the user needs to be satisfied by making changes to the software.
9. **Interoperability** – Capability of 2 or more functional units to process data cooperatively.

- **Program vs Software Product:**

1. A program is a set of instructions that are given to a computer in order to achieve a specific task whereas software is when a program is made available for commercial business and is properly documented along with its licensing.

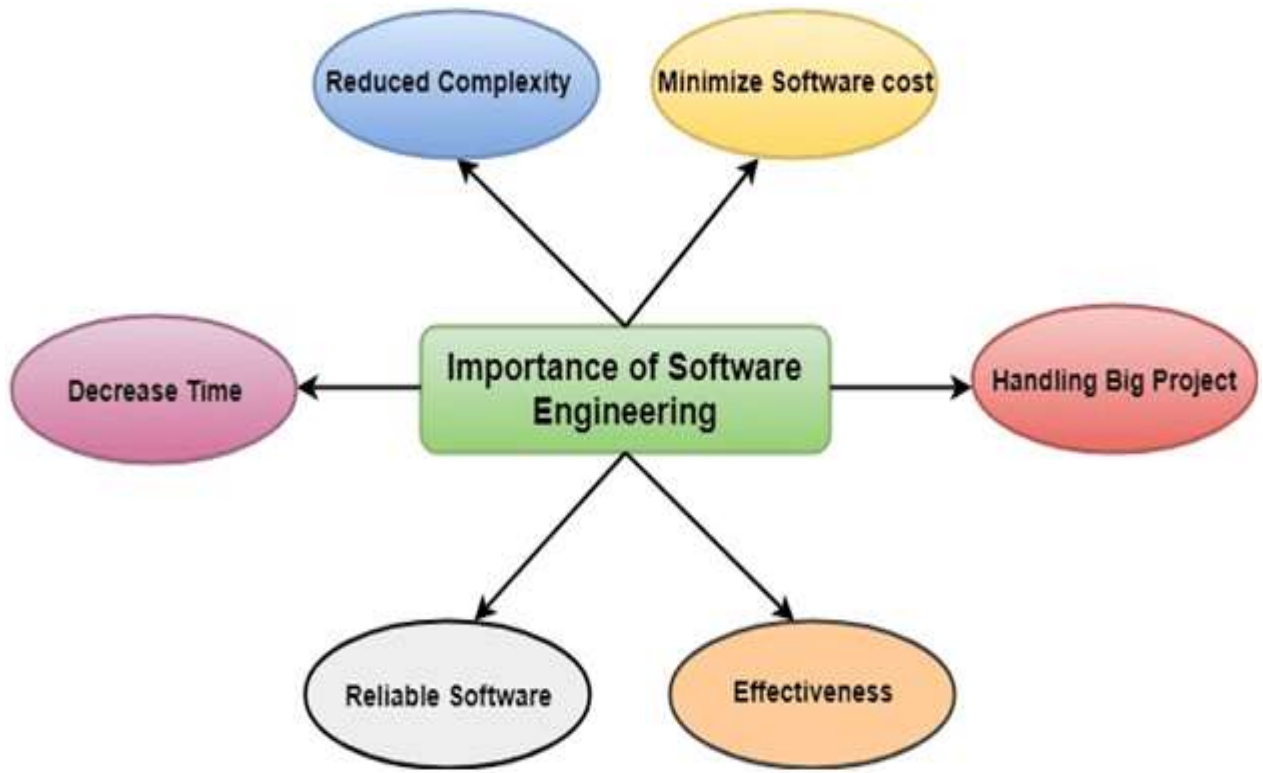
Software=Program+documentation+licensing.

A program is one of the stages involved in the development of the software, whereas a software development usually follows a life cycle, which involves the feasibility study of the project, requirement gathering, development of a prototype, system design, coding, and testing.

- **Classification of Software**

- **On the basis of application:**

1. **System Software –**
   System Software is necessary to manage the computer resources and support the execution of application programs. Software like operating systems, compilers, editors and drivers etc., come under this category. A computer cannot function without the presence of these. Operating systems are needed to link the machine dependent needs of a program with the capabilities of the machine on which it runs. Compilers translate programs from high-level language to machine language.
2. **Networking and Web Applications Software –**
   Networking Software provides the required support necessary for computers to interact with each other and with data storage facilities. The networking software is also used when software is running on a network of computers (such as World Wide Web). It includes all network management software, server software, security and encryption software and software to develop web-based applications like HTML, PHP, XML, etc.
3. **Embedded Software –**
   This type of software is embedded into the hardware normally in the Read Only Memory (ROM) as a part of a large system and is used to support certain functionality under the control conditions. Examples are software used in instrumentation and control applications like washing machines, satellites, microwaves etc.


Reduced Complexity
Minimize Software cost
Decrease Time
Importance of Software Engineering
Handling Big Project
Reliable Software
Effectiveness

- Software 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.
- Process models may contain activities, which are part of the software process, software product, and the roles of people involved in software engineering.

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

1. **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.
2. **Evolutionary development:** This method interleaves the activities of specification, development, and validation. An initial system is rapidly developed from a very abstract specification.
3. **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.
4. **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.

Software Crisis

1. **Size:** Software is becoming more expensive and more complex with the growing complexity and expectation out of software. For example, the code in the consumer product is doubling every couple of years.
2. **Quality:** Many software products have poor quality, i.e., the software products defects after putting into use due to ineffective testing technique. For example, Software testing typically finds 25 errors per 1000 lines of code.
3. **Cost:** Software development is costly i.e. in terms of time taken to develop and the money involved. For example, Development of the FAA's Advanced Automation System cost over $700 per lines of code.
4. **Delayed Delivery:** Serious schedule overruns are common. Very often the software takes longer than the estimated time to develop, which in turn leads to cost shooting up. For example, one in four large-scale development projects is never completed.

Software Engineering - Layered technology

- Software engineering is a fully layered technology.
- To develop a software, we need to go from one layer to another.
- All these layers are related to each other and each layer demands the fulfillment of the previous layer.

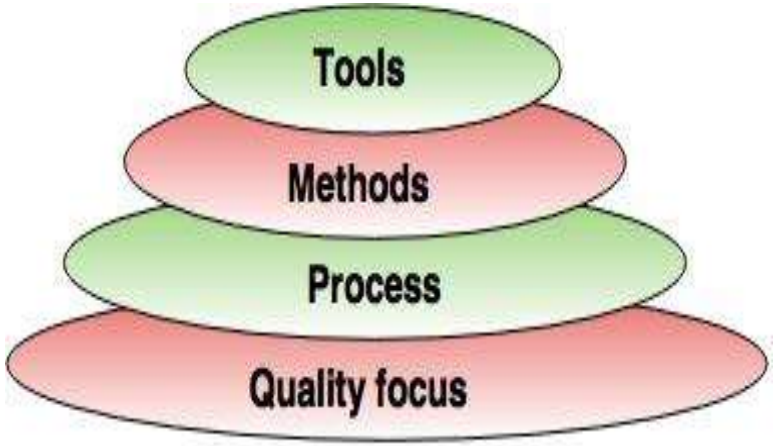


Fig. - Software Engineering Layers

- **The layered technology consists of:**

  1. Quality focus

The characteristics of good quality software are:

Correctness of the functions required to be performed by the software.

- Maintainability of the software
- Integrity i.e. providing security so that the unauthorized user cannot access information or data.
- Usability i.e. the efforts required to use or operate the software.

- **2. Process**

- It is the base layer or foundation layer for the software engineering.
- The software process is the key to keep all levels together.
- It defines a framework that includes different activities and tasks.
- In short, it covers all activities, actions and tasks required to be carried out for software development.

- **3. Methods**
- The method provides the answers of all 'how-to' that are asked during the process.
- It provides the technical way to implement the software.
- It includes collection of tasks starting from communication, requirement analysis, analysis and design modelling, program construction, testing and support.
- **4. Tools**
- The software engineering tool is an automated support for the software development.
- The tools are integrated i.e the information created by one tool can be used by the other tool.
- **For example:** The Microsoft publisher can be used as a web designing tool.

Software Process Framework

- The process of framework defines a small set of activities that are applicable to all types of projects.
- The software process framework is a collection of task sets.
- Task sets consist of a collection of small work tasks, project milestones, work productivity and software quality assurance points.

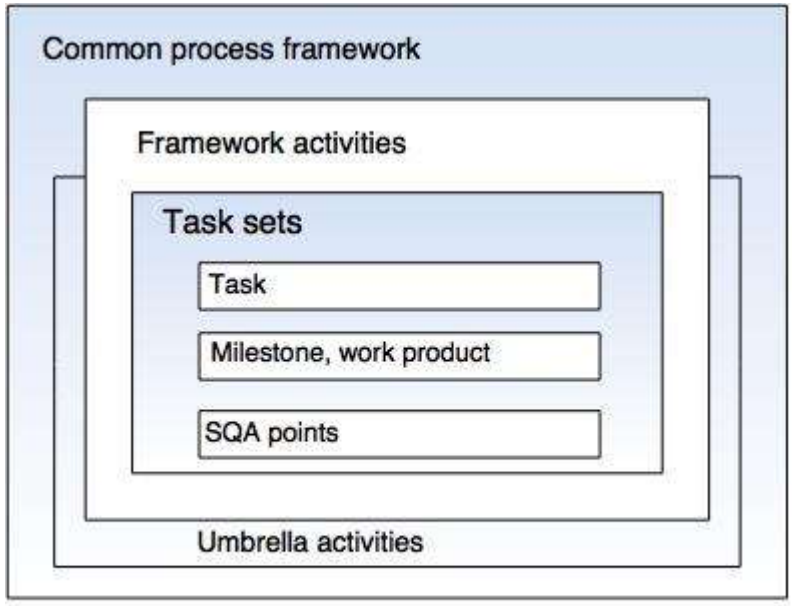


Fig.- A software process framework

Umbrella activities

- **Typical umbrella activities are:**

  1. Software project tracking and control
- In this activity, the developing team accesses project plan and compares it with the predefined schedule.
- If these project plans do not match with the predefined schedule, then the required actions are taken to maintain the schedule.
- **2. Risk management**
- Risk is an event that may or may not occur.
- If the event occurs, then it causes some unwanted outcome. Hence, proper risk management is required.

- **3. Software Quality Assurance (SQA)**
- SQA is the planned and systematic pattern of activities which are required to give a guarantee of software quality.
  For example, during the software development meetings are conducted at every stage of development to find out the defects and suggest improvements to produce good quality software.
- **4. Formal Technical Reviews (FTR)**
- FTR is a meeting conducted by the technical staff.
- The motive of the meeting is to detect quality problems and suggest improvements.
- The technical person focuses on the quality of the software from the customer point of view.

- **5. Measurement**
- Measurement consists of the effort required to measure the software.
- The software cannot be measured directly. It is measured by direct and indirect measures.
- Direct measures like cost, lines of code, size of software etc.
- Indirect measures such as quality of software which is measured by some other factor. Hence, it is an indirect measure of software.
- **6. Software Configuration Management (SCM)**
- It manages the effect of change throughout the software process.

- **7. Reusability management**
- It defines the criteria for reuse the product.
- The quality of software is good when the components of the software are developed for certain application and are useful for developing other applications.
- **8. Work product preparation and production**
- It consists of the activities that are needed to create the documents, forms, lists, logs and user manuals for developing a software.

- A software process is a collection of various activities.

There are five generic process framework activities:

1.Communication

2.Planning

3.Modeling

4.Construction

5.Deployment

- **1. Communication:**
  The software development starts with the communication between customer and developer.

  2. Planning:
  It consists of complete estimation, scheduling for project development and tracking.

  3. Modeling:
- Modeling consists of complete requirement analysis and the design of the project like algorithm, flowchart etc.
- The algorithm is the step-by-step solution of the problem and the flow chart shows a complete flow diagram of a program.

- **4. Construction:**
- Construction consists of code generation and the testing part.
- Coding part implements the design details using an appropriate programming language.
- Testing is to check whether the flow of coding is correct or not.
- Testing also check that the program provides desired output.

- **5. Deployment:**
- Deployment step consists of delivering the product to the customer and take feedback from them.
- If the customer wants some corrections or demands for the additional capabilities, then the change is required for improvement in the quality of the software.

Software Development Life Cycle (SDLC)

- A software life cycle model (also termed process model) is a pictorial and diagrammatic representation of the software life cycle.
- A life cycle model represents all the methods required to make a software product transit through its life cycle stages.
- A life cycle model maps the various activities performed on a software product from its inception to retirement. Different life cycle models may plan the necessary development activities to phases in different ways.
- During any life cycle stage, more than one activity may also be carried out.

Need of SDLC

- Without using an exact life cycle model, the development of a software product would not be in a systematic and disciplined manner. When a team is developing a software product, there must be a clear understanding among team representative about when and what to do.
- A software life cycle model describes entry and exit criteria for each phase. A phase can begin only if its stage-entry criteria have been fulfilled. So without a software life cycle model, the entry and exit criteria for a stage cannot be recognized. Without software life cycle models, it becomes tough for software project managers to monitor the progress of the project.

SDLC Cycle


SDLC
(Software Development life Cycle)
Requirement Analysis
Defining
Designing
Coding
Testing
Deployment
Maintenance

- **Stage1: Planning and requirement analysis**
- Requirement Analysis is the most important and necessary stage in SDLC.
- The senior members of the team perform it with inputs from all the stakeholders and domain experts or SMEs in the industry.
- Planning for the quality assurance requirements and identifications of the risks associated with the projects is also done at this stage.
- Business analyst and Project organizer set up a meeting with the client to gather all the data like what the customer wants to build, who will be the end user, what is the objective of the product. Before creating a product, a core understanding or knowledge of the product is very necessary.

- **Stage2: Defining Requirements**
- Once the requirement analysis is done, the next stage is to certainly represent and document the software requirements and get them accepted from the project stakeholders.
- This is accomplished through "SRS"- Software Requirement Specification document which contains all the product requirements to be constructed and developed during the project life cycle.

- **Stage3: Designing the Software**
- The next phase is about to bring down all the knowledge of requirements, analysis, and design of the software project. This phase is the product of the last two, like inputs from the customer and requirement gathering.

- **Stage4: Developing the project**
- In this phase of SDLC, the actual development begins, and the programming is built. The implementation of design begins concerning writing code. Developers have to follow the coding guidelines described by their management and programming tools like compilers, interpreters, debuggers, etc. are used to develop and implement the code.

- **Stage5: Testing**
- After the code is generated, it is tested against the requirements to make sure that the products are solving the needs addressed and gathered during the requirements stage.
- During this stage, unit testing, integration testing, system testing, acceptance testing are done.

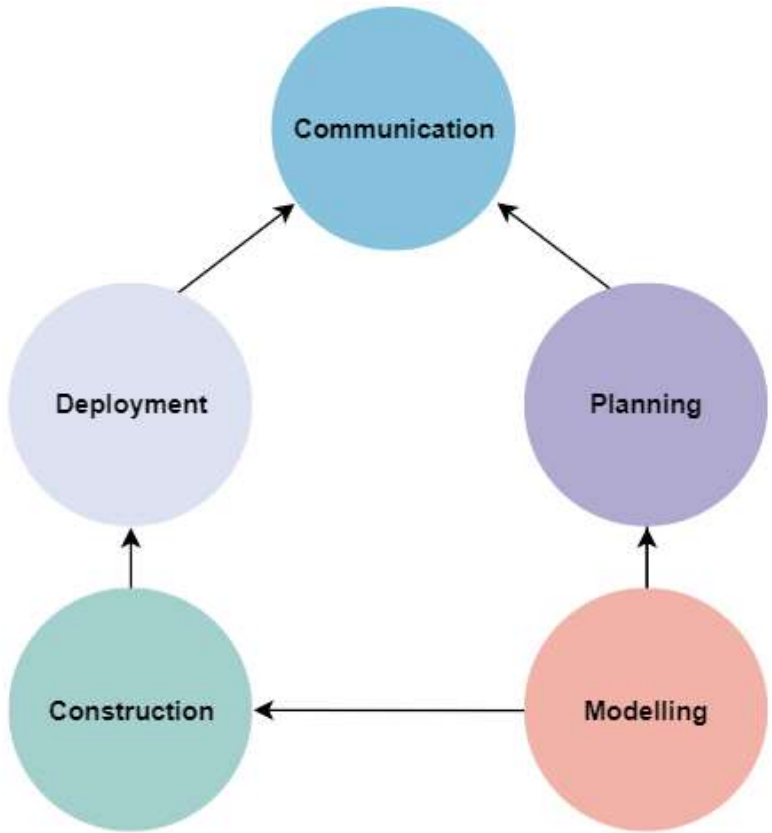
- **Stage6: Deployment**
- Once the software is certified, and no bugs or errors are stated, then it is deployed.
- Then based on the assessment, the software may be released as it is or with suggested enhancement in the object segment.
- After the software is deployed, then its maintenance begins.

- **Stage7: Maintenance**
- Once when the client starts using the developed systems, then the real issues come up and requirements to be solved from time to time.
- This procedure where the care is taken for the developed product is known as maintenance.

Generic Process Model

- The **generic process model** is an abstraction of the software development process. It is used in most software since it provides a base for them.
- The generic process model encompasses the following five steps:

1. Communication
2. Planning
3. Modelling
4. Construction
5. Deployment


Communication
Deployment
Planning
Construction
Modelling

- **Communication**
- In this step, we communicate with the clients and end-users.
- We discuss the requirements of the project with the users.
- The users give suggestions on the project. If any changes are difficult to implement, we work on alternative ideas.
- **Planning**
- In this step, we plan the steps for project development. After completing the final discussion, we report on the project.
- Planning plays a key role in the software development process.
- We discuss the risks involved in the project.

- **Modelling**
- In this step, we create a model to understand the project in the real world. We showcase the model to all the developers. If changes are required, we implement them in this step.
- We develop a practical model to get a better understanding of the project.
- **Construction**
- In this step, we follow a procedure to develop the final product.
- If any code is required for the project development, we implement it in this phase.
- We also test the project in this phase.

- **Deployment**
- In this phase, we submit the project to the clients for their feedback and add any missing requirements.
- We get the client feedback.
- Depending on the feedback form, we make the appropriate changes.

Process Assessment and Improvement

- Software Process Assessment is a disciplined and organized examination of the software process which is being used by any organization bases the on the process model.
- **Types of Software Assessment :**
- **Self Assessment :** This is conducted internally by the people of their own organisation.
- **Second Party assessment:** This is conducted by an external team or people of the own organisation are supervised by an external team.
- **Third Party assessment:**

- In an ideal case Software Process Assessment should be performed in a transparent, open and collaborative environment.
- This is very important for the improvement of the software and the development of the product.
- The results of the Software Process Assessment are confidential and are only accessible to the company.
- The assessment team must contain at least one person from the organization that is being assessed.

Software Process Maturity Assessment:

- The scope of Software Process Assessment includes many components like it should cover all the processes in the organisation, a selected subset of the software process or a specific project. The idea of process maturity serves as the foundation for the majority of standard-based process evaluation methodologies.
- Process maturity is important when the organisation intended to embark on a long term improvement strategy.

- Software processes are assessed to ensure their ability to control the cost, time and quality of software. Assessment is done to improve the software process followed by an organization.
- Software Process Improvement (SPI) Cycle includes:
- Process measurement
- Process analysis
- Process change

- Different approaches towards process assessment include
- **CMM (Capability Maturity Model) and CMMI (Capability Maturity Model Integration)**

- CMM was developed by SEI (Software Engineering Institute) and evolved into CMMI later. It is an approach based on which an organization’s process maturity is determined.

- CMM’s **five** maturity levels:

1. **Initial Level**: Processes are not organized and the success of a project depends only on the competence of the individual working on it. May not be able to repeat past successes in future projects. The probability of exceeding the estimated cost and schedule is high.
2. **Repeatable Level**: In this level, successes of the past could be repeated because the organization uses project management techniques to track cost and schedule. Management according to a documented plan helps in the improved process.
3. **Defined Level**: Organization’s set of standard processes are defined and are slightly modified to incorporate each project demands. This provides consistency throughout the works of the organization.
4. **Managed Level**: Management of processes using quantitative techniques improves performance. Processes are assessed through data collection and analysis.
5. **Optimizing Level**: Processes are monitored and improved through feedback from current work. Innovative techniques are applied to cope with changing business objectives and the environment.

- **CMMI** maturity levels include:
- Initial.
- Managed.
- Defined.
- Quantitatively Managed.
- Optimized.

- **CMMI** capability levels include:
- **Level 0: Incomplete** – Incomplete processes are processes that are not performed or partially performed.
- **Level 1: Performed** – Specific goals are satisfied by processes and yet certain objectives related to quality, cost and schedule are not met. Useful work can be done.
- **Level 2: Managed** – Cost, quality and schedule are managed and processes are monitored by management techniques.
- **Level 3: Defined** – It includes management and additionally follow the organization's specified set of standard processes which are altered for each project.
- **Level 4: Quantitatively Managed** – Statistical and quantitative techniques are used for the management of processes.
- **Level 5: Optimized –** It focuses on continuous improvement of Quantitatively Managed process through innovations and nature of processes.

- **Standard CMMI Appraisal Method for Process Improvement (SCAMPI)**
- It is a method used by Software Engineering Institute (SEI) for providing quality ratings with respect to Capability Maturity Model Integration (CMMI). Assessment includes five phases initiating, diagnosing, establishing, acting and learning. The appraisal process includes preparation, on-site activities, findings and ratings, final reporting etc.

- **CMM Based Appraisal for Internal Process Improvement (CBA IPI)**
- It is an SEI CMM(Capability Maturity Model) based assessment method that provides diagnostics, enables and encourages an organization to understand its maturity. It gives the organization an insight into its software development capability by assessing the strength and weakness of the current process.
- **SPICE(ISO/IEC15504)**
- This standard is one of the joint mission of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). They assist organizations in developing an objective evaluation of the effectiveness of a software process and related business management functions.

- It consists of six levels as:
- Not performed.
- Performed informally.
- Planned and tracked.
- Well defined.
- Quantitatively controlled.
- Continuously improved.

- **ISO 9001:2000 for software**
- It is applied to organizations aiming to improve the overall quality of product, process and services. They evaluate the ability of an organization to consistently provide products that meet customer requirements. Here the main aim of the organization should be enhancing customer satisfaction.
- It follows **Plan Do Check Act** (PDCA) cycle which includes:
- **Planning** by defining the processes and their needs required to develop a better-quality product.
- **Doing** necessary actions according to the plan.
- **Checking** whether the actions for ensuring quality according to requirements are fulfilled.
- **Acting** on activities that are used to improve processes in the organization.