

# 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 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.

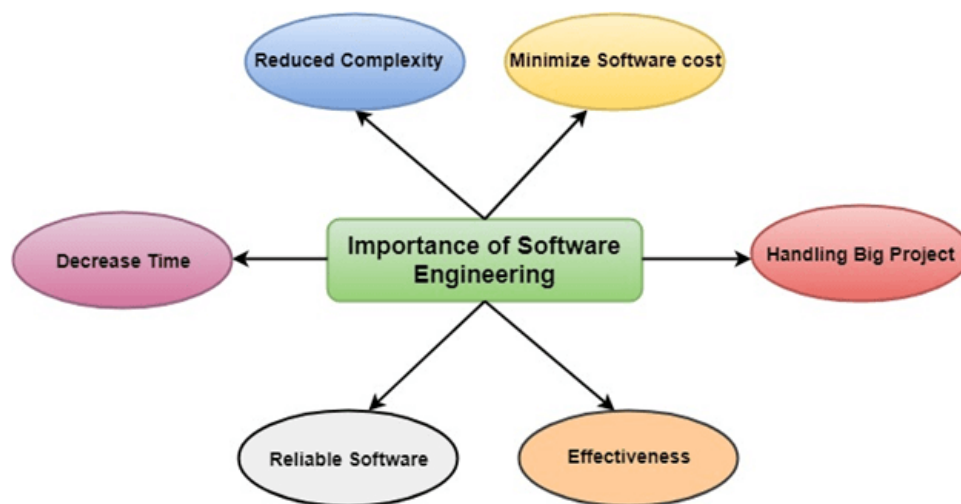
Sound knowledge of fundamentals of computer science.

Intelligence.

Ability to work in a team

Discipline, etc.

## Importance of Software Engineering



**The importance of Software engineering is as follows:**

1. **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.

2. **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.
3. **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.
4. **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.
5. **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.

6. **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 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:

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. After each stage is defined, it is "signed off" and development goes onto the following stage.
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.

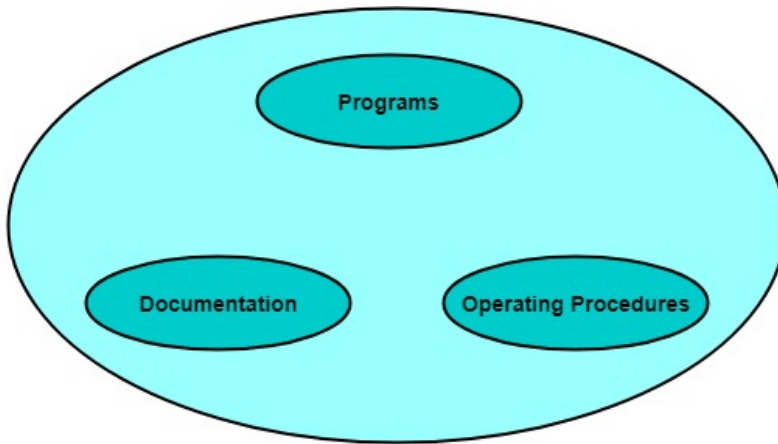
## Software Crisis

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.

## Program vs. Software

Software is more than programs. Any program is a subset of software, and it becomes software only if documentation & operating procedures manuals are prepared.

There are three components of the software as shown in fig:

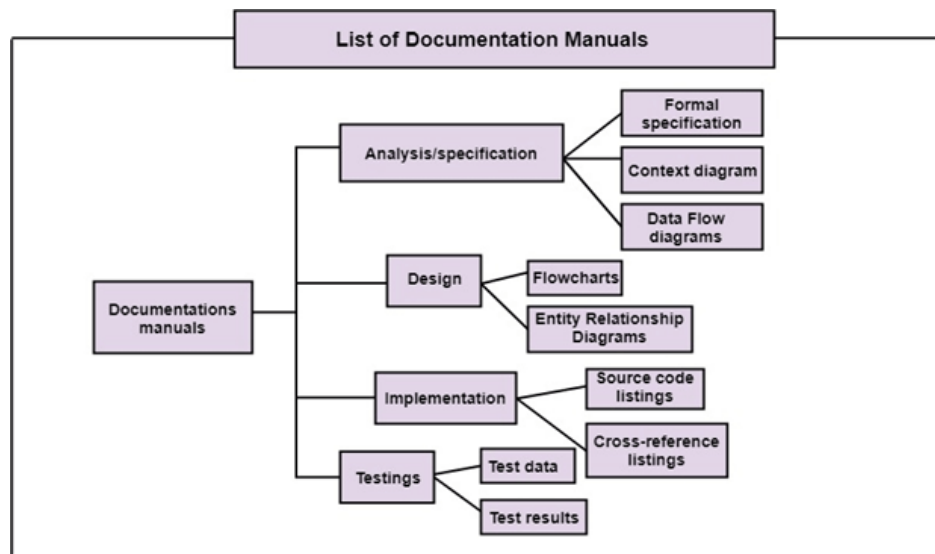


Software= Program + Documentation + Operating Procedures

Fig:Components of Software

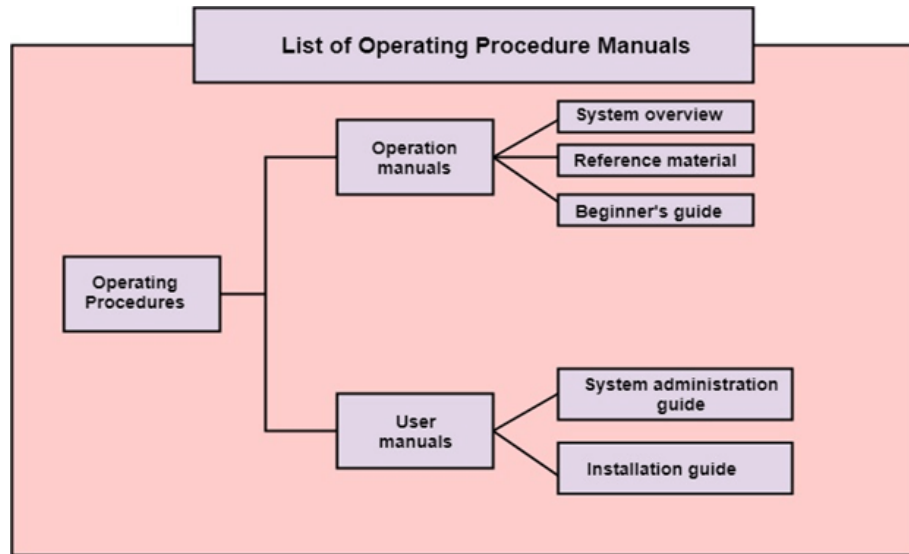
**1. Program:** Program is a combination of source code & object code.

**2. Documentation:** Documentation consists of different types of manuals. Examples of documentation manuals are: Data Flow Diagram, Flow Charts, ER diagrams, etc.



**3. Operating Procedures:** Operating Procedures consist of instructions to set up and use the software system and instructions on how react to the system failure. Example of operating system procedures manuals is: installation guide, Beginner's guide, reference guide, system administration guide, etc.





## 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. It also captures the structure in which these methods are to be undertaken.

In other words, 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. Thus, no element which life cycle model is followed, the essential activities are contained in all life cycle models though the action may be carried out in distinct orders in different life cycle models. During any life cycle stage, more than one activity may also be carried out.

## Need of SDLC

The development team must determine a suitable life cycle model for a particular plan and then observe to it.

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. Otherwise, it would point to chaos and project failure. This problem can be defined by using an example. Suppose a software development issue is divided into various parts and the parts are assigned to the team members. From then on, suppose the team representative is allowed the freedom to develop the roles assigned to them in whatever way they like. It is possible that one representative might start writing the code for his part, another might choose to prepare the test documents first, and some other engineer might begin with the design phase of the roles assigned to him. This would be one of the perfect methods for project failure.

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 Cycle represents the process of developing software. SDLC framework includes the following steps:



The stages of SDLC are as follows:

## **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.

**For Example,** A client wants to have an application which concerns money transactions. In this method, the requirement has to be precise like what kind of operations will be done, how it will be done, in which currency it will be done, etc.

Once the required function is done, an analysis is complete with auditing the feasibility of the growth of a product. In case of any ambiguity, a signal is set up for further discussion.

Once the requirement is understood, the SRS (Software Requirement Specification) document is created. The developers should thoroughly follow this document and also should be reviewed by the customer for future reference.

## **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.

## **Software Engineering | Software Evolution**

**Software Evolution** is a term which refers to the process of developing software initially, then timely updating it for various reasons, i.e., to add new features or to remove obsolete functionalities etc. The evolution process includes fundamental

activities of change analysis, release planning, system implementation and releasing a system to customers.

The cost and impact of these changes are assessed to see how much system is affected by the change and how much it might cost to implement the change. If the proposed changes are accepted, a new release of the software system is planned. During release planning, all the proposed changes (fault repair, adaptation, and new functionality) are considered.

A design is then made on which changes to implement in the next version of the system. The process of change implementation is an iteration of the development process where the revisions to the system are designed, implemented and tested.

**The necessity of Software evolution:** Software evaluation is necessary just because of the following reasons:

a) **Change in requirement with time:** With the passes of time, the organization's needs and modus Operandi of working could substantially be changed so in this frequently changing time the tools(software) that they are using need to change for maximizing the performance.

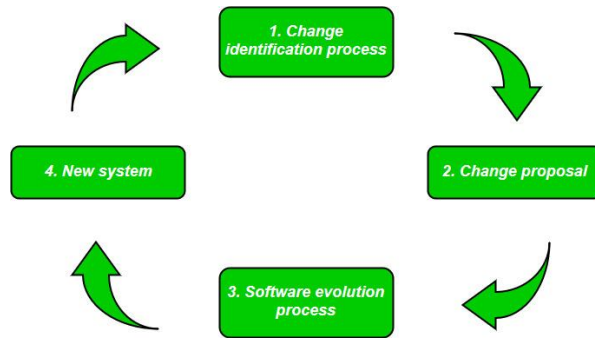
b) **Environment change:** As the working environment changes the things(tools) that enable us to work in that environment also changes proportionally same

happens in the software world as the working environment changes then, the organizations need reintroduction of old software with updated features and functionality to adapt the new environment.

**c) Errors and bugs:** As the age of the deployed software within an organization increases their preciseness or impeccability decrease and the efficiency to bear the increasing complexity workload also continually degrades. So, in that case, it becomes necessary to avoid use of obsolete and aged software. All such obsolete Softwares need to undergo the evolution process in order to become robust as per the workload complexity of the current environment.

**d) Security risks:** Using outdated software within an organization may lead you to at the verge of various software-based cyberattacks and could expose your confidential data illegally associated with the software that is in use. So, it becomes necessary to avoid such security breaches through regular assessment of the security patches/modules are used within the software. If the software isn't robust enough to bear the current occurring Cyber attacks so it must be changed (updated).

**e) For having new functionality and features:** In order to increase the performance and fast data processing and other functionalities, an organization need to continuously evolve the software throughout its life cycle so that stakeholders & clients of the product could work efficiently.



### Laws used for Software Evolution:

1. **Law of continuing change:**

This law states that any software system that represents some real-world reality undergoes continuous change or become progressively less useful in that environment.

2. **Law of increasing complexity:**

As an evolving program changes, its structure becomes more complex unless effective efforts are made to avoid this phenomenon.

3. **Law of conservation of organization stability:**

Over the lifetime of a program, the rate of development of that program is approximately constant and independent of the resource devoted to system development.

4. **Law of conservation of familiarity:**

This law states that during the active lifetime of the program, changes made in the successive release are almost constant.

# Software Engineering | Changing Nature of Software

The software is instruction or computer program that when executed provide desired features, function, and performance. A data structure that enables the program to adequately manipulate information and document that describe the operation and use of the program.

## **Characteristic of software:**

There is some characteristic of software which is given below:

1. Functionality
2. Reliability
3. Usability
4. Efficiency
5. Maintainability
6. Portability

## **Changing Nature of Software:**

Nowadays, seven broad categories of computer software present continuing challenges for software engineers .which is given below:

### **1. System Software:**

System software is a collection of programs which are written to service



other programs. Some system software processes complex but determinate, information structures. Other system application process largely indeterminate data. Sometimes when, the system software area is characterized by the heavy interaction with computer hardware that requires scheduling, resource sharing, and sophisticated process management.

**2. Application Software:**

Application software is defined as programs that solve a specific business need. Application in this area process business or technical data in a way that facilitates business operation or management technical decision making. In addition to convention data processing application, application software is used to control business function in real time.

**3. Engineering and Scientific Software:**

This software is used to facilitate the engineering function and task. however modern application within the engineering and scientific area are moving away from the conventional numerical algorithms. Computer-aided design, system simulation, and other interactive applications have begun to take a real-time and even system software characteristic.

**4. Embedded Software:**

Embedded software resides within the system or product and is used to implement and control feature and function for the end-user and for the system itself. Embedded software can perform the limited and esoteric function or provided significant function and control capability.

**5. Product-line Software:**

Designed to provide a specific capability for use by many different

customers, product line software can focus on the limited and esoteric marketplace or address the mass consumer market.

#### **6. Web Application:**

It is a client-server computer program which the client runs on the web browser. In their simplest form, Web apps can be little more than a set of linked hypertext files that present information using text and limited graphics. However, as e-commerce and B2B application grow in importance. Web apps are evolving into a sophisticated computing environment that not only provides a standalone feature, computing function, and content to the end user.

#### **7. Artificial Intelligence Software:**

Artificial intelligence software makes use of a nonnumerical algorithm to solve a complex problem that is not amenable to computation or straightforward analysis. Application within this area includes robotics, expert system, pattern recognition, artificial neural network, theorem proving and game playing.

## **What is legacy software?**

Legacy software is software that has been around a long time and still fulfills a business need. It is mission critical and tied to a particular version of an operating system or hardware model (vendor lock-in) that has gone end-of-life. Generally the lifespan of the hardware is shorter than that of the software. As time goes on, the hardware gets harder to maintain but is kept because it is installed and (for now) working and has proven too complex and/or expensive to replace

# Software Engineering | Capability maturity model (CMM)

CMM was developed by the Software Engineering Institute (SEI) at Carnegie Mellon University in 1987.

- It is not a software process model. It is a framework that is used to analyze the approach and techniques followed by any organization to develop software products.
- It also provides guidelines to further enhance the maturity of the process used to develop those software products.
- It is based on profound feedback and development practices adopted by the most successful organizations worldwide.
- This model describes a strategy for software process improvement that should be followed by moving through 5 different levels.
- Each level of maturity shows a process capability level. All the levels except level-1 are further described by Key Process Areas (KPA's).

## Shortcomings of SEI/CMM:

- It encourages the achievement of a higher maturity level in some cases by displacing the true mission, which is improving the process and overall software quality.
- It only helps if it is put into place early in the software development process.

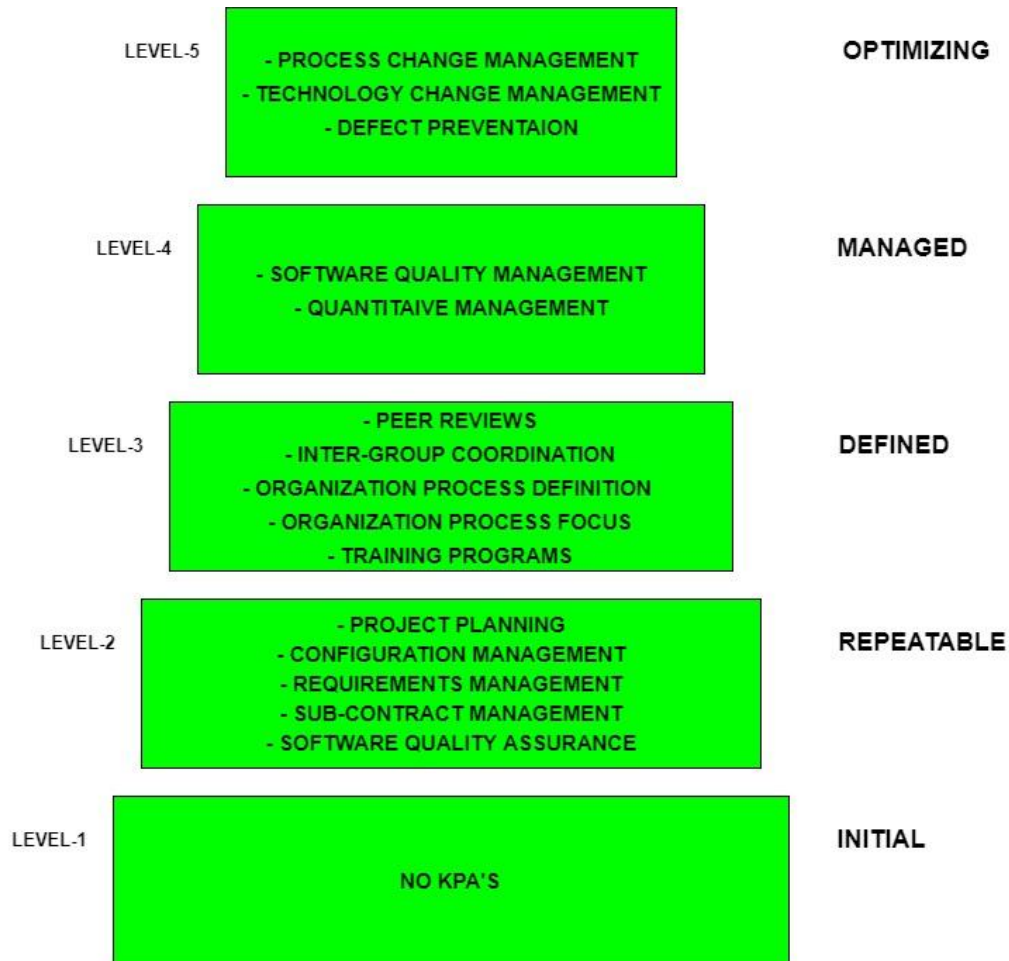
- It has no formal theoretical basis and in fact is based on the experience of very knowledgeable people.
- It does not have good empirical support and this same empirical support could also be constructed to support other models.

### **Key Process Areas (KPA's):**

Each of these KPA's defines the basic requirements that should be met by a software process in order to satisfy the KPA and achieve that level of maturity.

Conceptually, key process areas form the basis for management control of the software project and establish a context in which technical methods are applied, work products like models, documents, data, reports, etc. are produced, milestones are established, quality is ensured and change is properly managed.

The 5 levels of CMM are as follows:



### Level-1: Initial –

- No KPA's defined.
- Processes followed are Adhoc and immature and are not well defined.
- Unstable environment for software development.
- No basis for predicting product quality, time for completion, etc.

### Level-2: Repeatable –

- Focuses on establishing basic project management policies.
- Experience with earlier projects is used for managing new similar natured projects.
- **Project Planning-** It includes defining resources required, goals, constraints, etc. for the project. It presents a detailed plan to be followed systematically for the successful completion of good quality software.
- **Configuration Management-** The focus is on maintaining the performance of the software product, including all its components, for the entire lifecycle.
- **Requirements Management-** It includes the management of customer reviews and feedback which result in some changes in the requirement set. It also consists of accommodation of those modified requirements.
- **Subcontract Management-** It focuses on the effective management of qualified software contractors i.e. it manages the parts of the software which are developed by third parties.
- **Software Quality Assurance-** It guarantees a good quality software product by following certain rules and quality standard guidelines while developing.

### **Level-3: Defined –**

- At this level, documentation of the standard guidelines and procedures takes place.
- It is a well-defined integrated set of project-specific software engineering and management processes.

- **Peer Reviews-** In this method, defects are removed by using a number of review methods like walkthroughs, inspections, buddy checks, etc.
- **Intergroup Coordination-** It consists of planned interactions between different development teams to ensure efficient and proper fulfillment of customer needs.
- **Organization Process Definition-** Its key focus is on the development and maintenance of the standard development processes.
- **Organization Process Focus-** It includes activities and practices that should be followed to improve the process capabilities of an organization.
- **Training Programs-** It focuses on the enhancement of knowledge and skills of the team members including the developers and ensuring an increase in work efficiency.

#### **Level-4: Managed –**

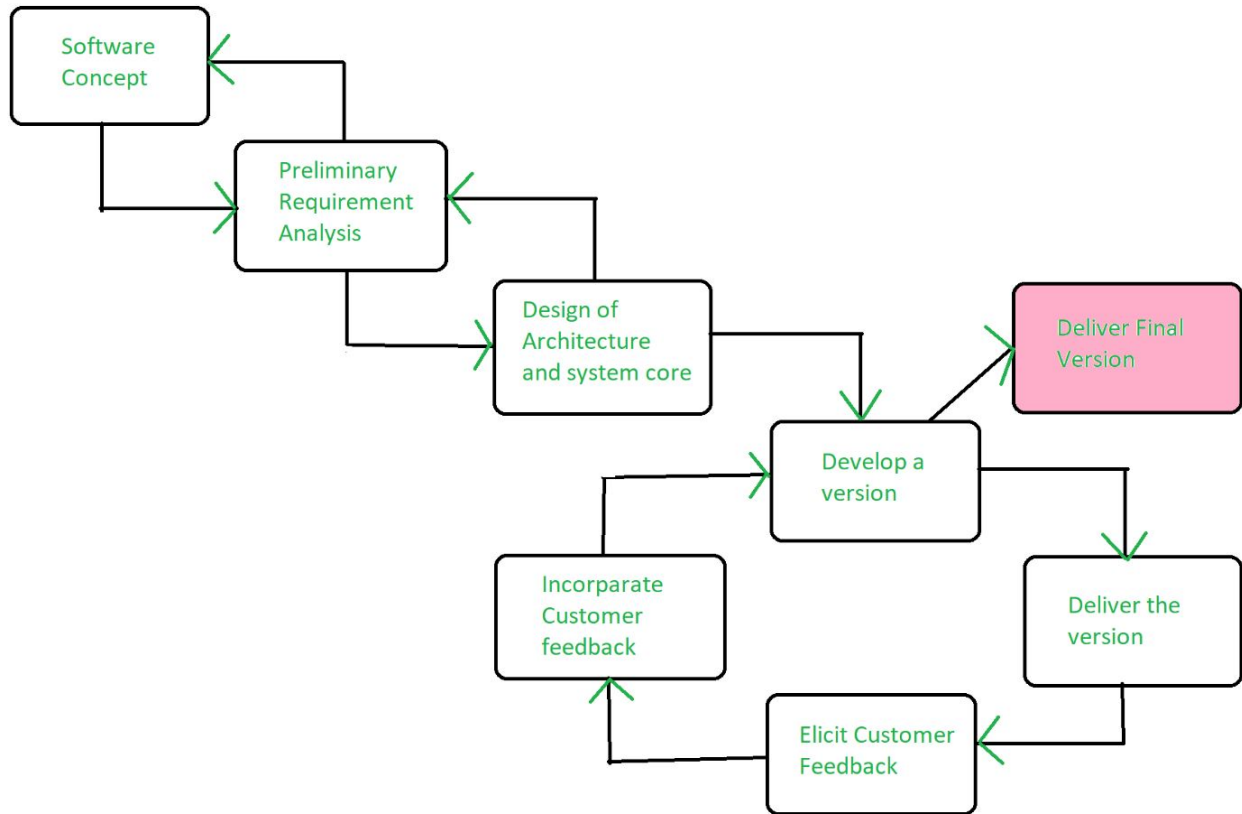
- At this stage, quantitative quality goals are set for the organization for software products as well as software processes.
- The measurements made help the organization to predict the product and process quality within some limits defined quantitatively.
- **Software Quality Management-** It includes the establishment of plans and strategies to develop quantitative analysis and understanding of the product's quality.
- **Quantitative Management-** It focuses on controlling the project performance in a quantitative manner.

## Level-5: Optimizing –

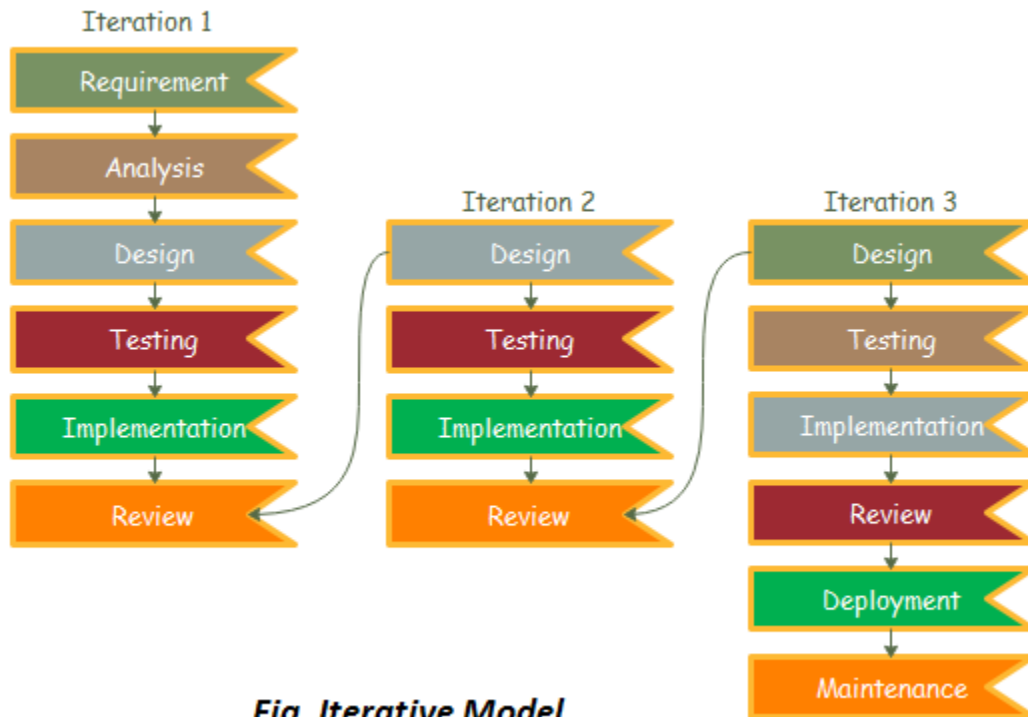
- This is the highest level of process maturity in CMM and focuses on continuous process improvement in the organization using quantitative feedback.
- Use of new tools, techniques, and evaluation of software processes is done to prevent recurrence of known defects.
- **Process Change Management-** Its focus is on the continuous improvement of the organization's software processes to improve productivity, quality, and cycle time for the software product.
- **Technology Change Management-** It consists of the identification and use of new technologies to improve product quality and decrease product development time.
- **Defect Prevention-** It focuses on the identification of causes of defects and prevents them from recurring in future projects by improving project-defined processes.



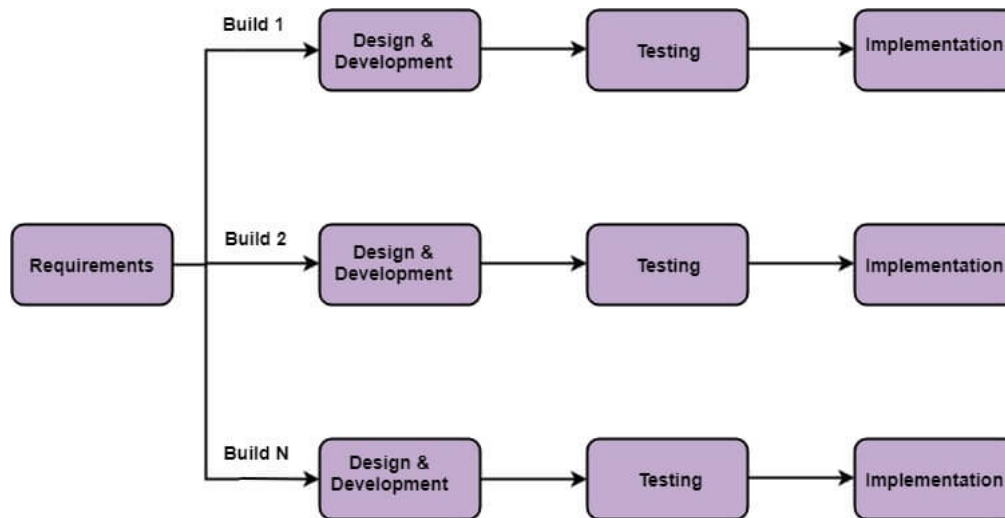
# Evolutionary Model



## Iterative Model

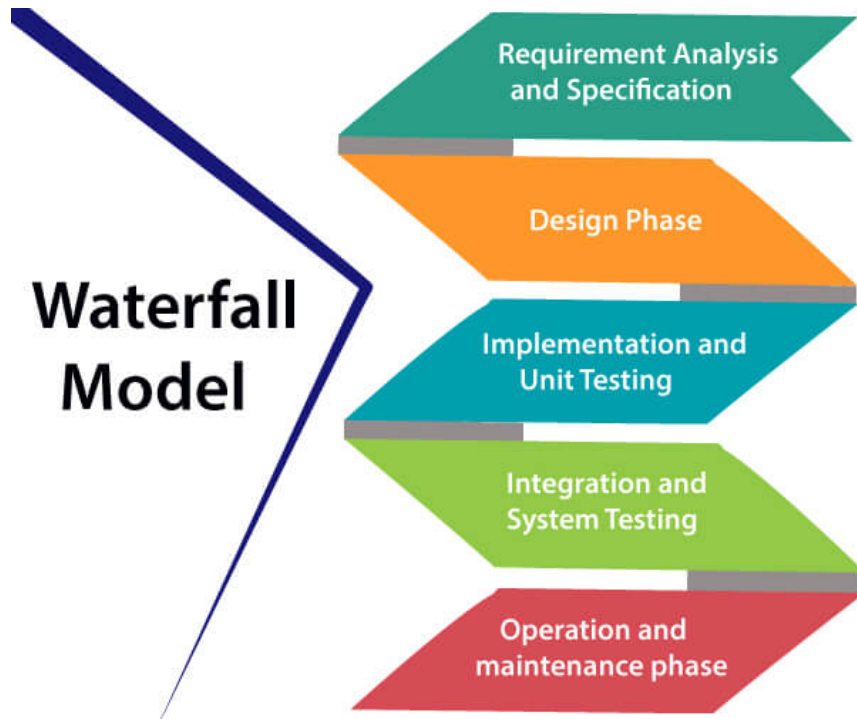


## Incremental Model

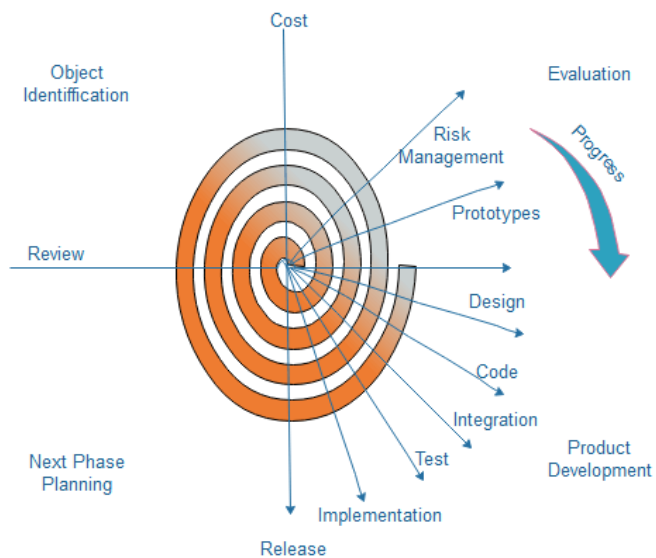


**Fig: Incremental Model**

## Waterfall Model



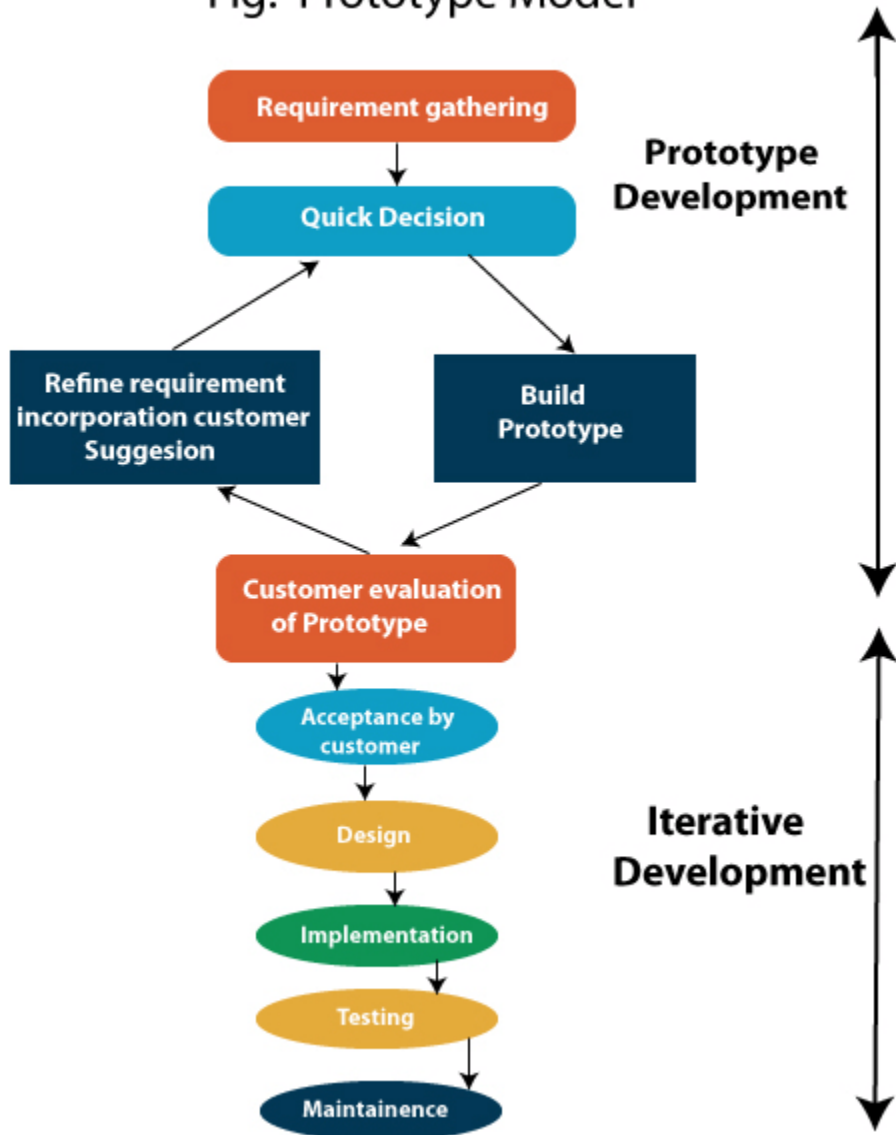
## Spiral Model



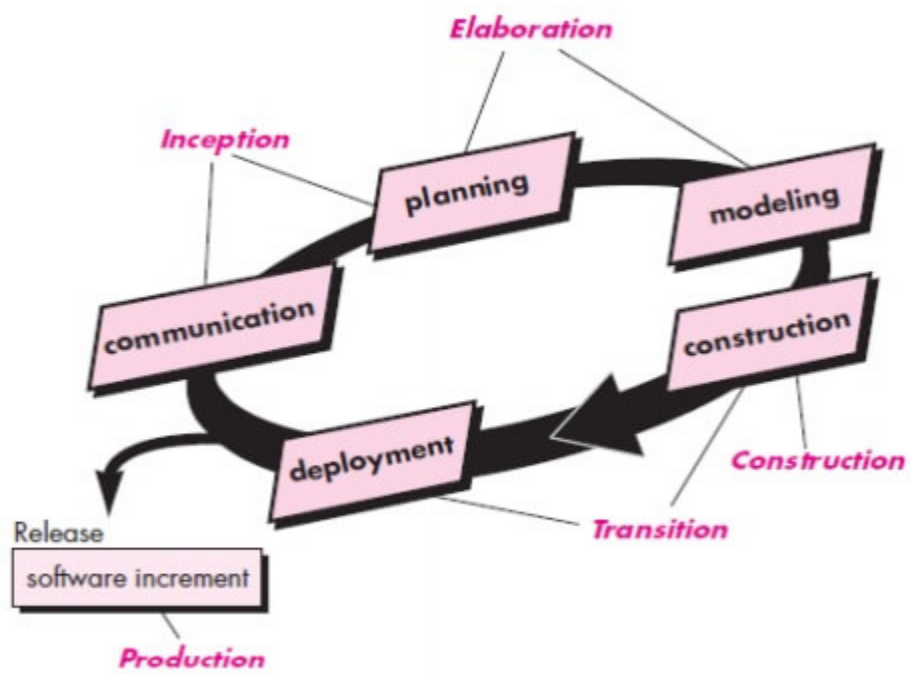
**Fig. Spiral Model**

# Prototype Model

Fig: Prototype Model



# Unified Process Model



# Functional vs Non Functional Requirements

**Functional Requirements:** These are the requirements that the end user specifically demands as basic facilities that the system should offer. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements.

**Non-functional requirements:** These are basically the quality constraints that the system must satisfy according to the project contract. The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements.

## Functional Requirements

A functional requirement defines a system or its component.

It specifies "What should the software system do?"

## Non Functional Requirements

A non-functional requirement defines the quality attribute of a software system.

It places constraints on "How should the software system fulfill the functional requirements?"

Functional requirement is specified by User.

Non-functional requirement is specified by technical peoples e.g. Architect, Technical leaders and software developers.

It is mandatory.

It is not mandatory.

It is captured in use case.

It is captured as a quality attribute.

Defined at a component level.

Applied to a system as a whole.

Helps you verify the functionality of the software.

Helps you to verify the performance of the software.

Functional Testing like System, Integration, End to End, API testing, etc are done.

Non-Functional Testing like Performance, Stress, Usability, Security testing, etc are done.

Usually easy to define.

Usually more difficult to define.

### Example

- 1) Authentication of user whenever he/she logs into the system.
- 2) System shutdown in case of a cyber attack.
- 3) A Verification email is sent to user whenever he/she registers for the first time on some software system.

### Example

- 1) Emails should be sent with a latency of no greater than 12 hours from such an activity.
- 2) The processing of each request should be done within 10 seconds
- 3) The site should load in 3 seconds when the number of simultaneous users are  $> 10000$

## User Interface requirements

UI is an important part of any software or hardware or hybrid system. A software is widely accepted if it is -

- easy to operate
- quick in response
- effectively handling operational errors
- providing simple yet consistent user interface

User acceptance majorly depends upon how user can use the software. UI is the only way for users to perceive the system. A well performing software system must also be equipped with attractive, clear, consistent and responsive user interface. Otherwise the functionalities of software system can not be used in convenient way. A system is said be good if it provides means to use it efficiently. User interface requirements are briefly mentioned below -

- Content presentation
- Easy Navigation
- Simple interface
- Responsive
- Consistent UI elements
- Feedback mechanism



- Default settings
- Purposeful layout
- Strategic use of color and texture.
- Provide help information
- User centric approach
- Group based view settings.

# System Requirements

## What Does System Requirements Mean?

System requirements are the configuration that a system must have in order for a hardware or software application to run smoothly and efficiently. Failure to meet these requirements can result in installation problems or performance problems. The former may prevent a device or application from getting installed, whereas the latter may cause a product to malfunction or perform below expectation or even to hang or crash. System requirements are also known as minimum system requirements.

## Techopedia Explains System Requirements

For packaged products, system requirements are often printed on the packaging. For downloadable products, the system requirements are often indicated on the download page. System requirements can be broadly classified as functional requirements, data requirements, quality requirements and constraints. They are often provided to consumers in complete detail. System requirements often indicate the minimum and the recommended configuration. The former is the most basic requirement, enough for a product to install or run, but performance is not guaranteed to be optimal. The latter ensures a smooth operation.

Hardware system requirements often specify the operating system version, processor type, memory size, available disk space and additional peripherals, if any, needed. Software system requirements, in addition to the aforementioned requirements, may also specify additional software dependencies (e.g., libraries, driver version, framework version). Some hardware/software manufacturers provide an upgrade assistant program that users can download and run to determine whether their system meets a product's requirements.

# Requirement Engineering

The process to gather the software requirements from client, analyze and document them is known as requirement engineering.

The goal of requirement engineering is to develop and maintain sophisticated and descriptive 'System Requirements Specification' document.

## Requirement Engineering Process

It is a four step process, which includes –

- Feasibility Study
- Requirement Gathering
- Software Requirement Specification
- Software Requirement Validation

Let us see the process briefly -

### Feasibility study

When the client approaches the organization for getting the desired product developed, it comes up with rough idea about what all functions the software must perform and which all features are expected from the software.

Referencing to this information, the analysts does a detailed study about whether the desired system and its functionality are feasible to develop.

This feasibility study is focused towards goal of the organization. This study analyzes whether the software product can be practically materialized in terms of implementation, contribution of project to organization, cost constraints and as per values and objectives of the organization. It explores technical aspects of the project and product such as usability, maintainability, productivity and integration ability.

The output of this phase should be a feasibility study report that should contain adequate comments and recommendations for management about whether or not the project should be undertaken.

### Requirement Gathering

If the feasibility report is positive towards undertaking the project, next phase starts with gathering requirements from the user. Analysts and engineers communicate with the client and end-users to know their ideas on what the software should provide and which features they want the software to include.

## Software Requirement Specification

SRS is a document created by system analyst after the requirements are collected from various stakeholders.

SRS defines how the intended software will interact with hardware, external interfaces, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, Security, Quality, Limitations etc.

The requirements received from client are written in natural language. It is the responsibility of system analyst to document the requirements in technical language so that they can be comprehended and useful by the software development team.

SRS should come up with following features:

- User Requirements are expressed in natural language.
- Technical requirements are expressed in structured language, which is used inside the organization.
- Design description should be written in Pseudo code.
- Format of Forms and GUI screen prints.
- Conditional and mathematical notations for DFDs etc.

## Software Requirement Validation

After requirement specifications are developed, the requirements mentioned in this document are validated. User might ask for illegal, impractical solution or experts may interpret the requirements incorrectly. This results in huge increase in cost if not nipped in the bud. Requirements can be checked against following conditions -

- If they can be practically implemented
- If they are valid and as per functionality and domain of software
- If there are any ambiguities
- If they are complete
- If they can be demonstrated

## Software Requirements Characteristics

Gathering software requirements is the foundation of the entire software development project. Hence they must be clear, correct and well-defined.

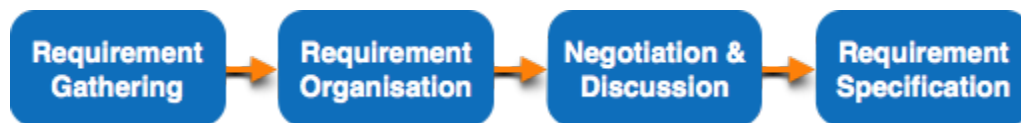
A complete Software Requirement Specifications must be:

- Clear

- Correct
- Consistent
- Coherent
- Comprehensible
- Modifiable
- Verifiable
- Prioritized
- Unambiguous
- Traceable
- Credible source
- Complete

## Requirement Elicitation Process

Requirement elicitation process can be depicted using the following diagram:



- Requirements gathering - The developers discuss with the client and end users and know their expectations from the software.
- Organizing Requirements - The developers prioritize and arrange the requirements in order of importance, urgency and convenience.
- Negotiation & discussion - If requirements are ambiguous or there are some conflicts in requirements of various stakeholders, if they are, it is then negotiated and discussed with stakeholders. Requirements may then be prioritized and reasonably compromised.

The requirements come from various stakeholders. To remove the ambiguity and conflicts, they are discussed for clarity and correctness. Unrealistic requirements are compromised reasonably.

- Documentation - All formal & informal, functional and non-functional requirements are documented and made available for next phase processing.

### Technique

- Interviews
- Questionnaires
- Task analysis
- Domain Analysis

- Brainstorming
- Prototyping
- Observation

## Requirements Validation Techniques

**Requirements validation** is the process of checking that requirements defined for development, define the system that the customer really wants. To check issues related to requirements, we perform requirements validation. We usually use requirements validation to check error at the initial phase of development as the error may increase excessive rework when detected later in the development process.

In the requirements validation process, we perform a different type of test to check the requirements mentioned in the [Software Requirements Specification \(SRS\)](#), these checks include:

- Completeness checks
- Consistency checks
- Validity checks
- Realism checks
- Ambiguity checks
- Verifiability

The output of requirements validation is the list of problems and agreed on actions of detected problems. The lists of problems indicate the problem detected

during the process of requirement validation. The list of agreed action states the corrective action that should be taken to fix the detected problem.

There are several techniques which are used either individually or in conjunction with other techniques to check to check entire or part of the system:

1. **Test case generation:**

Requirement mentioned in SRS document should be testable, the conducted tests reveal the error present in the requirement. It is generally believed that if the test is difficult or impossible to design than, this usually means that requirement will be difficult to implement and it should be reconsidered.

2. **Prototyping:**

In this validation techniques the prototype of the system is presented before the end-user or customer, they experiment with the presented model and check if it meets their need. This type of model is generally used to collect feedback about the requirement of the user.

3. **Requirements Reviews:**

In this approach, the SRS is carefully reviewed by a group of people including people from both the contractor organisations and the client side, the reviewer systematically analyses the document to check error and ambiguity.

4. **Automated Consistency Analysis:**

This approach is used for automatic detection of an error, such as nondeterminism, missing cases, a type error, and circular definitions, in requirements specifications.

First, the requirement is structured in formal notation then CASE tool is

used to check in-consistency of the system, The report of all inconsistencies is identified and corrective actions are taken.

#### 5. Walk-through:

A walkthrough does not have a formally defined procedure and does not require a differentiated role assignment.

- Checking early whether the idea is feasible or not.
- Obtaining the opinions and suggestion of other people.
- Checking the approval of others and reaching an agreement.

## System Modeling

**System modeling** is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system. It is about representing a system using some kind of graphical notation, which is now almost always based on notations in the **Unified Modeling Language (UML)**. Models help the analyst to understand the functionality of the system; they are used to communicate with customers.

Models can explain the system from **different perspectives**:

- An **external** perspective, where you model the context or environment of the system.
- An **interaction** perspective, where you model the interactions between a system and its environment, or between the components of a system.
- A **structural** perspective, where you model the organization of a system or the structure of the data that is processed by the system.
- A **behavioral** perspective, where you model the dynamic behavior of the system and how it responds to events.

Five types of UML diagrams that are the most useful for system modeling:

- **Activity** diagrams, which show the activities involved in a process or in data processing.
- **Use case** diagrams, which show the interactions between a system and its environment.
- **Sequence** diagrams, which show interactions between actors and the system and between system components.

- **Class** diagrams, which show the object classes in the system and the associations between these classes.
- **State** diagrams, which show how the system reacts to internal and external events.

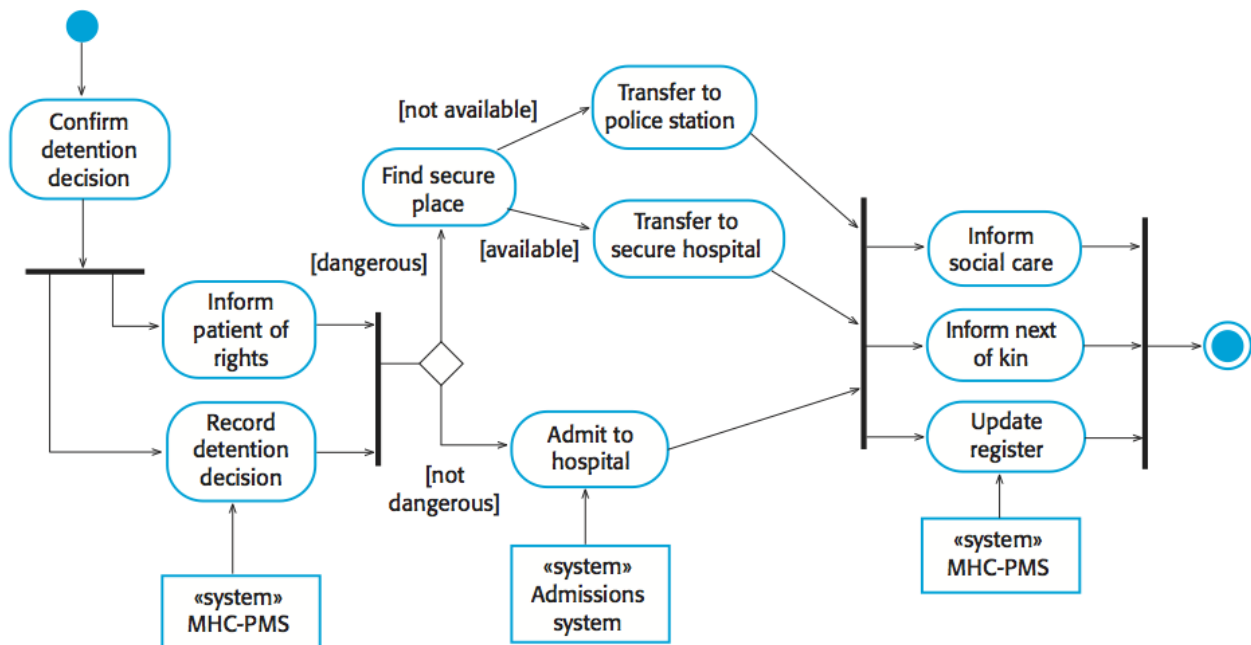
## Context and process models

**Context models** are used to illustrate the operational context of a system - they show what lies outside the system boundaries. Social and organizational concerns may affect the decision on where to position system boundaries. Architectural models show the system and its relationship with other systems.

**System boundaries** are established to define what is inside and what is outside the system. They show other systems that are used or depend on the system being developed. The position of the system boundary has a profound effect on the system requirements. Defining a system boundary is a political judgment since there may be pressures to develop system boundaries that increase/decrease the influence or workload of different parts of an organization.

Context models simply show the other systems in the environment, not how the system being developed is used in that environment. **Process models** reveal how the system being developed is used in broader business processes. UML activity diagrams may be used to define business process models.

The example below shows a UML **activity diagram** describing the process of involuntary detention and the role of MHC-PMS (mental healthcare patient management system) in it.





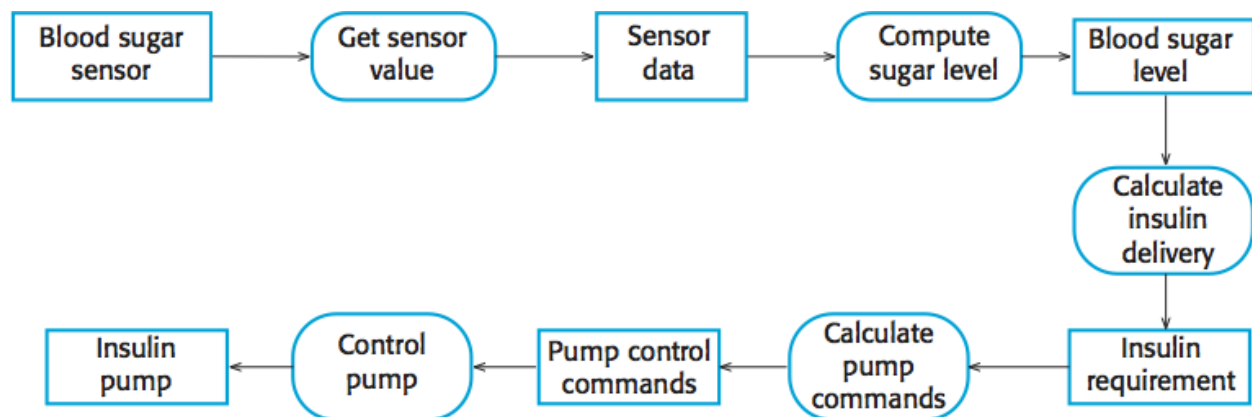
## Behavioral models

---

**Behavioral models** are models of the dynamic behavior of a system as it is executing. They show what happens or what is supposed to happen when a system responds to a stimulus from its environment. Two types of stimuli:

- Some **data** arrives that has to be processed by the system.
- Some **event** happens that triggers system processing. Events may have associated data, although this is not always the case.

Many business systems are data-processing systems that are primarily driven by data. They are controlled by the data input to the system, with relatively little external event processing. **Data-driven models** show the sequence of actions involved in processing input data and generating an associated output. They are particularly useful during the analysis of requirements as they can be used to show end-to-end processing in a system. Data-driven models can be created using UML **activity diagrams**:



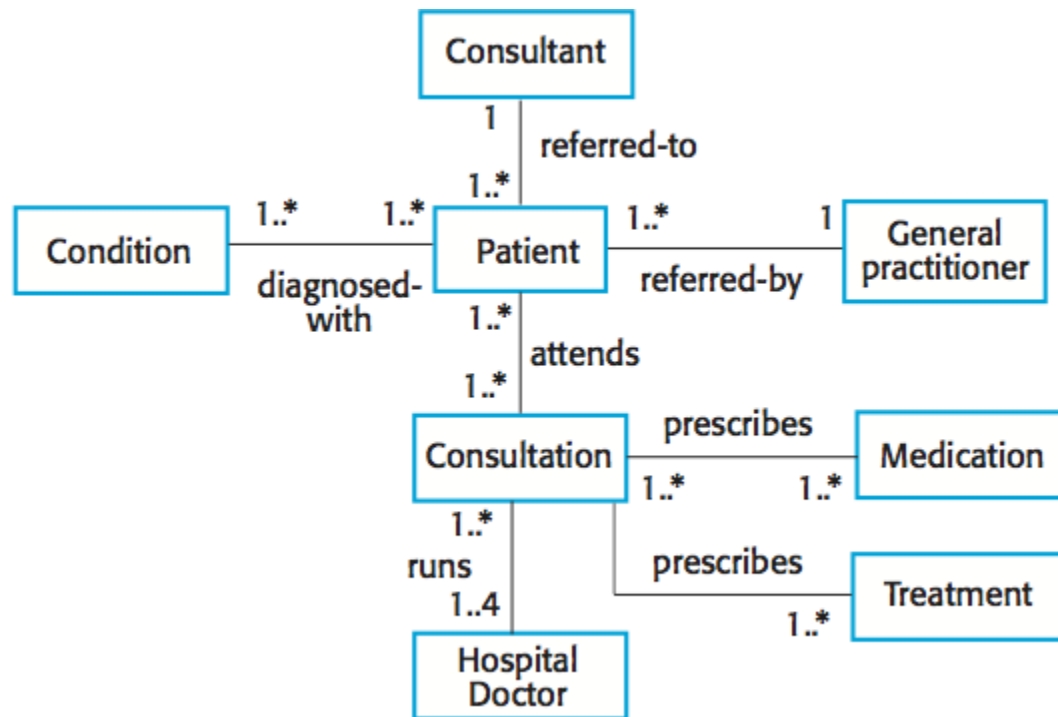
## Structural models

---

**Structural models** of software display the organization of a system in terms of the components that make up that system and their relationships. Structural models may be **static** models, which show the structure of the system design, or **dynamic** models, which show the organization of the system when it is executing. You create structural models of a system when you are discussing and designing the system architecture.

UML **class diagrams** are used when developing an object-oriented system model to show the classes in a system and the associations between these classes. An object class can be thought of as a general definition

of one kind of system object. An association is a link between classes that indicates that there is some relationship between these classes. When you are developing models during the early stages of the software engineering process, objects represent something in the real world, such as a patient, a prescription, doctor, etc.



## Interaction models

Types of interactions that can be represented in a model:

- Modeling **user interaction** is important as it helps to identify user requirements.
- Modeling **system-to-system interaction** highlights the communication problems that may arise.
- Modeling **component interaction** helps us understand if a proposed system structure is likely to deliver the required system performance and dependability.

**Use cases** were developed originally to support requirements elicitation and now incorporated into the UML. Each use case represents a discrete task that involves external interaction with a system. Actors in a use case may be people or other systems. Use cases can be represented using a UML use case diagram and in a more detailed textual/tabular format.

Simple use case diagram:

