By Shubh Thakare +919595939264

# Unit 1. Introduction to Software Engineering.

## Introduction to Software Engineering :

### 1.The evolving role of software

### Software Engineering

#### Software

**Software is**

- ▶ ↓A information transformer - Producing, managing, acquiring, Modifying, displaying or transmitting information that is simple/ complex which is satisfies the users needs & makes the user world better.
- ▶ ↓A software is either a collection of programs which will transfer the data from one place to another

#### Engineering

- ▶ It is systematic and technical approach which tells you the methods to carry out the work.
- ▶ It can de defined as Applying something technical method or algorithm in a systematic way.

### Software Engineering

◆ Software engineering is the establishment and use of sound engineering principles (applying some software methods) in order to obtain economically software that is reliable(not working properly) and works efficiently on real machines.

◆ Software engineering is a discipline that integrates process methods and tools for the development of computer system.

## Software Engineering Layers

- Process layer : It is glue that holds the technology layers together and enables timely. In other way we can say the technology we are using produce accurate result.
- Methods : Provides the technical (which platform) how- to's for building software. It is a set of basic principles that govern each area of technology.
- Tools: Provides automated and semi automated support for process and method e.g MS excel tool is used to applying some formula's.

## Software characters

- Software is developed or engineered, it is not manufactured in the classical sense.
- Software does not wear out. (design and develop is available every where it is product) car example
- Although industry is moving towards component based assembling most software continues to be custom built.

# Software Application

- System software -
- Real time software
- Business software
- Engineering and scientific software e.g CAD, CAM
- Embedded software
- Personal computer software
- Web based software
- Artificial intelligence software
- Apps Software iOS,Andriod

## Q) DIFFERENCE BETWEEN PROGRAM AND SOFTWARE.

| PROGRAM | SOFTWARE |
|---|---|
| 1) Small in size.<br>2) Authors himself is user-soul.<br>3) Single developer.<br>4) Adopt development.<br>5) Lack proper interface.<br>6) Large proper documentation. | 1) Large in size.<br>2) Large number.<br>3) Team developer.<br>4) Systematic development.<br>5) Well define interface.<br>6) Well documented. |

By Shubh Thakare +919595939264

# Software process model

## what is software process model ?

- A process model describes the sequence of phases for the entire lifetime of a product . Therefore it is sometimes also called product life cycle. This covers everything from the initial commercial ideas until the final DE installation or disassembling of the product after its use
- Each process model follows a particular life cycle in order to ensure success in process of software development.
- Three main phases concept phase implementation phase maintenance phase

## Need of software project management

By Shubh Thakare +919595939264

# Software Management Activities

**Software project management comprises of a number of activities, which contains planning of project, deciding scope of software product, estimation of cost in various terms, scheduling of tasks and events, and resource management. Project management activities may include:**

- Project Planning
- Scope Management
- Project Estimation

Why software engineering:-

- 1) In the late 1960's hardware price were falling but software price rising.
- 2) Many software projects failed.
- 3) Large software project required large development loams.
- 4) Many software project late and over budget.
- 5) Complicity of software project is increased.

6) Demand for new software on the market.

**By Shubh Thakare +919595939264**

# THE EVOLVING ROLE OF SOFTWARE

Today, software takes on a dual role.

It is a product and, at the same time, the vehicle for delivering a product.

As a product, it delivers the computing potential embodied by computer hardware or, more broadly, a network of computers that are accessible by local hardware. Whether it resides within a cellular phone or operates inside a mainframe computer, software is an information transformer—producing, managing, acquiring, modifying, displaying, or transmitting information that can be as simple as a single bit or as complex as a multimedia presentation.

As the vehicle used to deliver the product, software acts as the basis for the control of the computer (operating systems), the communication of information (networks), and the creation and control of other programs (software tools and environments).

Software delivers the most important product of our time—information. Software transforms personal data (e.g., an individual's financial transactions) so that the data can be more useful in a local context; it manages business information to enhance competitiveness; it provides a gateway to worldwide information networks (e.g., Internet) and provides the means for acquiring information in all of its forms. The role of computer software has undergone significant change over a time span of little more than 50 years.

Dramatic improvements in hardware performance, profound changes in computing architectures, vast increases in memory and storage capacity, and a wide variety of exotic input and output options have all precipitated more sophisticated and complex computer-based systems. The lone programmer of an earlier era has been replaced by a team of software specialists, each focusing on one part of the technology required to deliver a complex application. And yet,

the same questions asked of the lone programmer are being asked when modern computer-based systems are built:

1)Why does it take so long to get software finished?

2)Why are development costs so high?

3)Why can't we find all the errors before we give the software to customers?

4)Why do we continue to have difficulty in measuring progress as software is being developed?

By Shubh Thakare +919595939264

## 2. Changing Nature of Software {S/W Application}

# Changing Nature of Software :

The nature of software has changed a lot over the years.

1.System software: Infrastructure software come under this category like compilers, operating systems, editors, drivers, etc. Basically system software is a collection of programs to provide service to other programs.

2. Real time software: These software are used to monitor, control and analyze real world events as they occur. An example may be software required for weather forecasting. Such software will gather and process the status of temperature, humidity and other environmental parameters to forcast the weather.

3. Embedded software: This type of software is placed in “Read-Only- Memory (ROM)”of the product and control the various functions of the product. The product could be an aircraft, automobile, security system, signalling system, control unit of power plants, etc. The embedded software handles hardware components and is also termed as intelligent software .

4. Business software : This is the largest application area. The software designed to process business applications is called business software. Business software could be payroll, file monitoring system, employee management, account management. It may also be a data warehousing tool which helps us to take decisions based on available data. Management information system, enterprise resource planning (ERP) and such other software are popular examples of business software.

5. Personal computer software :The software used in personal computers are covered in this category. Examples are word processors, computer graphics, multimedia and animating tools, database management, computer games etc. This is a very upcoming area and many big organisations are concentrating their effort here due to large customer base.

6. Artificial intelligence software: Artificial Intelligence software makes use of non numerical algorithms to solve complex problems that are not amenable to computation or straight forward analysis. Examples are expert systems, artificial neural network,signal processing software etc.

7. Web based software: The software related to web applications come under this category. Examples are CGI, HTML, Java, Perl, DHTML etc.

## 3.Software myths.

Software myths are misleading attitudes that have caused serious problems for managers and technical people alike. Software myths propagate misinformation and confusion. There are three kinds of software myths:

***Management myths:*** Managers with software responsibility are often under pressure to maintain budgets, keep schedules from slipping, and improve quality. Following are the management myths:

Myth: We already have a book that's full of standards and procedures for building software, won't that provide my people with everything they need to know? Reality: The book of standards may very well exist, but isn't used. Most software practitioners aren't aware of its existence. Also, it doesn't reflect modern software engineering practices and is also complete.

Myth: My people have state-of-the-art software development tools, after all, we buy them the newest computers. Reality: It takes much more than the latest model mainframe, workstation, or PC to do high-quality software development. Computer-aided software engineering (CASE) tools are more important than hardware for achieving good quality and productivity, yet the majority of software developers still do not use them effectively.

Myth: If we get behind schedule, we can add more programmers and catch up (sometimes called the Mongolian horde concept). Reality: Software development is not a mechanistic process like manufacturing. As new people are added, people who were working must spend time educating the newcomers, thereby reducing the amount of time spent on productive development effort. People can be added but only in a planned and well-coordinated manner.

Myth: If I decide to outsource the software project to a third party, I can just relax and let that firm build it. Reality: If an organization does not understand how to manage and control software projects internally, it will invariably struggle when it outsources software projects.

***2) Customer myths*:** Customer myths lead to false expectations (by the customer) and ultimately, dissatisfaction with the developer. Following are the customer myths:

Myth: A general statement of objectives is sufficient to begin writing programs-we can fill in the details later. Reality: A poor up-front definition is the major cause of failed software efforts. A formal and detailed description of the functions, behavior, performance, interfaces, design constraints, and validation criteria is essential.

Myth: Project requirements continually change, but change can be easily accommodated because software is flexible. Reality: It is true that software requirements change, but the impact of change varies with the time at which it is introduced. When changes are requested during software design, the cost impact grows rapidly. Resources have been committed and a design framework has been established. Change can cause heavy additional costs. Change, when requested after software is in production, can be much more expensive than the same change requested earlier.

## 3) Practitioner's myths: Practitioners have following myths:

Myth: Once we write the program and get it to work, our job is done. Reality: Industry data indicate that between 60 and 80 percent of all effort expended on software will be expended after it is delivered to the customer for the first time

Myth: Until I get the program "running" I have no way of assessing its quality. Reality: One of the most effective software quality assurance mechanisms can be applied from the inception of a project—the formal technical review.

**1-Definition Phase -** It is the base of Definition phase. The experts get the knowledge about "What".

- Information needed for processing.
- Which functions are required.
- Expectations about the capacity.
- Interface which is established.
- Area of the validation.

This phase defines all the expectations depending on the standard of the software Engineering. It contains three steps.

- Analysis of system
- Planning of project
- Requirement Analysis

**2-Development phase -** Focus point of development phase is "How". After the explanation of "What" it turn to "How". Various type of question raised in developer mind that how to design the data structure and Architecture of software, Procedural detail how to

implemented and how design convert in a programming language and testing of software how to perform. Three special steps always taken in this phase which are

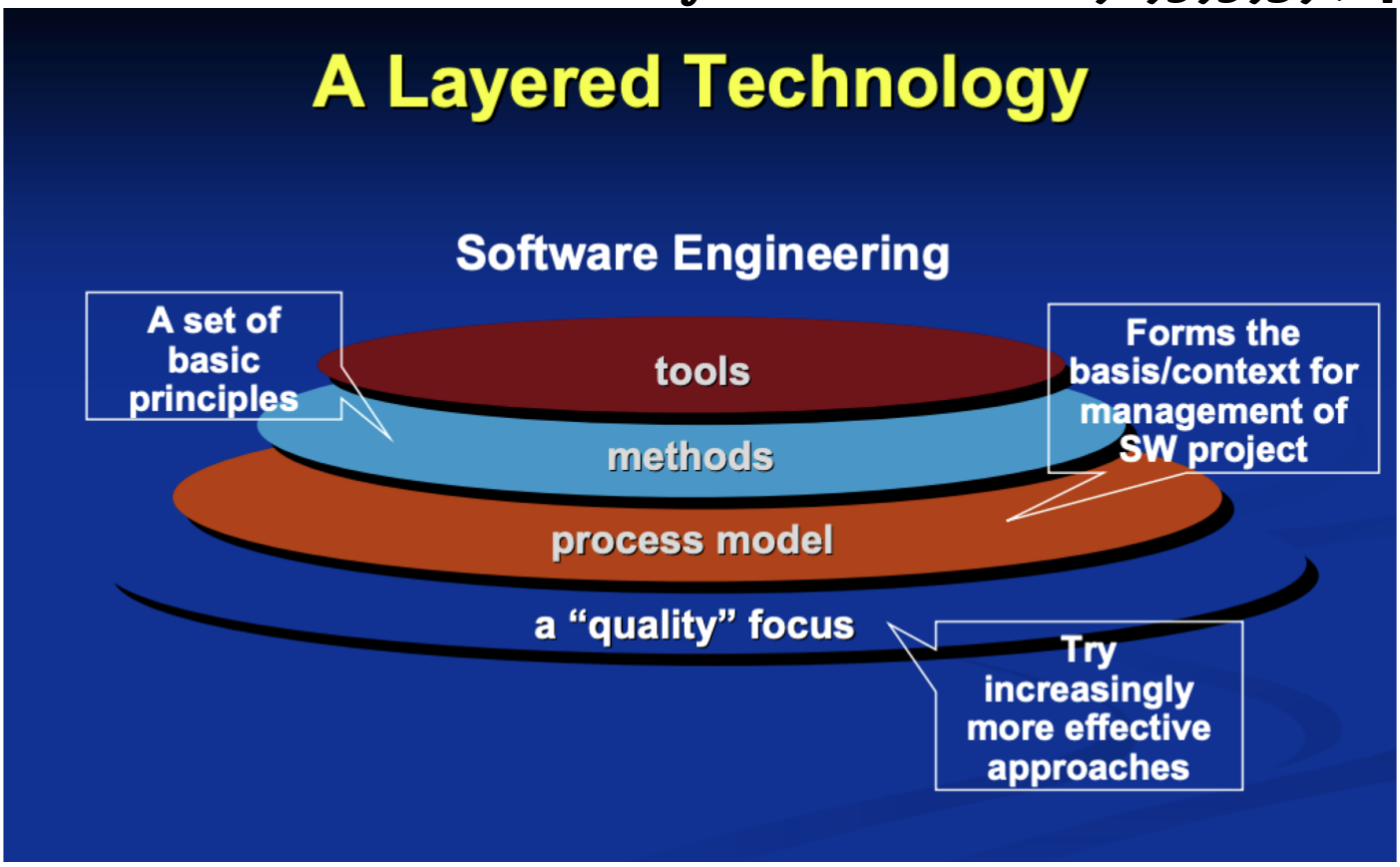
- Design of software
- Coding
- testing of software system

**3-Maintenance phase -** The main focus of maintenance phase is change which cause is correction of errors, adaption of new idea, According to the needs of software after change in customer mood.

## A Generic view of process :

## 1.Software engineering- A layered technology

By Shubh Thakare +919595939264

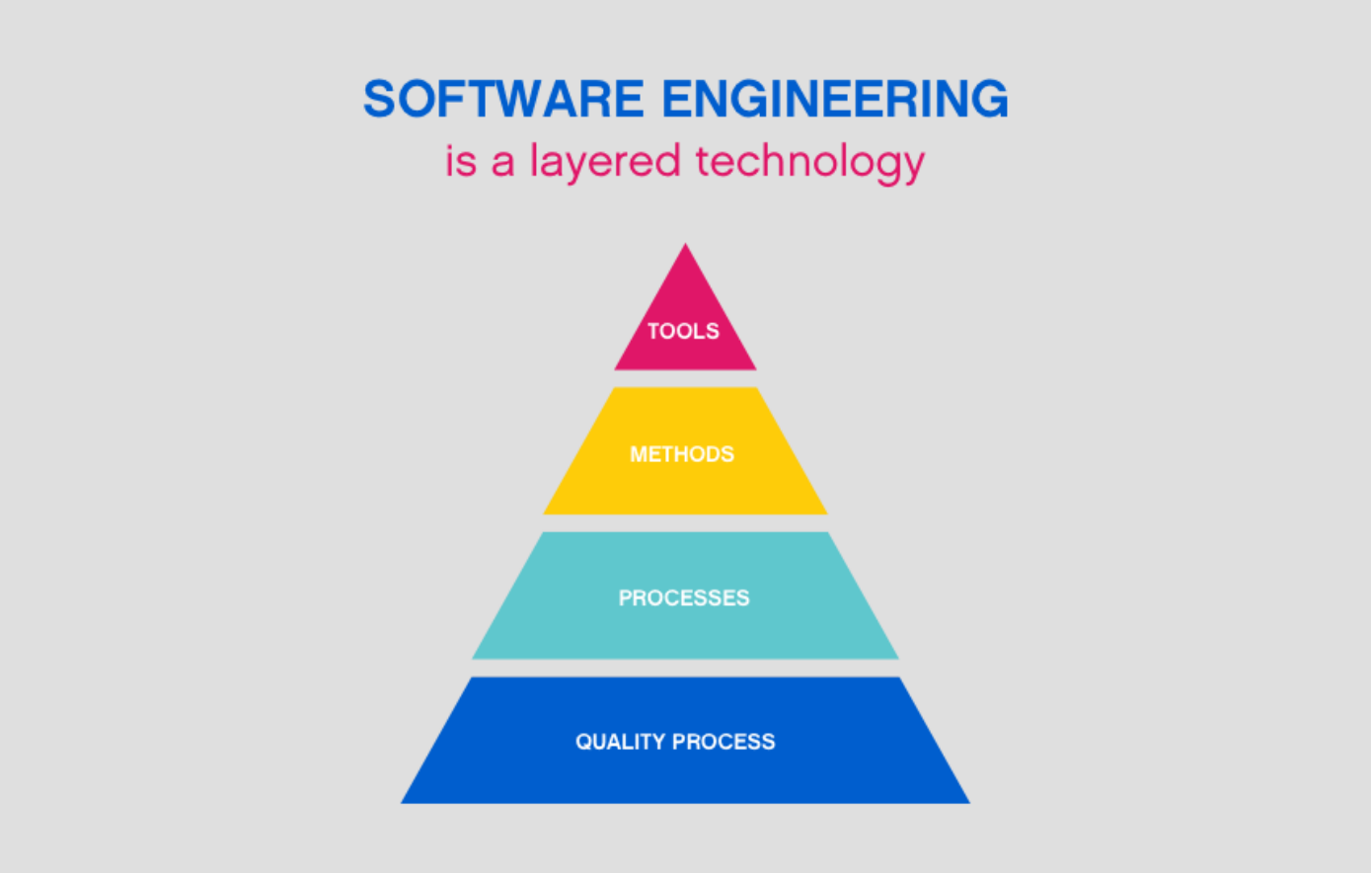



# Software Engineering: A Layered Approach

Over the past years, technological advancements have resulted in some revolutionary breakthroughs. You can now order clothes, a cab or even food from your smartphone, while on the phone and en route your next adventure. And that is only an example of how software has grown to provide humanity with endless opportunities. But with software becoming more complex there was soon a need to advance from basic programming principles to that of software engineering in order to tackle those projects more efficiently.

More complex software products require more than basic programming skills, as they are designed to be critical systems for businesses. Software engineering teams are required to analyze user needs and then go on to design, implement and test their end product to make sure it satisfies those needs through the use of programming languages. This process is systematic, disciplined and quantifiable. According to the IEEE, software engineering is “the application of a systematic disciplined, quantifiable approach to the development, operation and maintenance of software; that is, the application of engineering to software”.In a nutshell, software engineering is a systematic approach used by professionals to develop complex software end products within a specified time and budget. This approach is often viewed as layered.

### The 4 Layers Of Software Engineering




**Tools**: This layer contains automated or semi-automated tools that offer support for the framework and the method each software engineering project will follow.

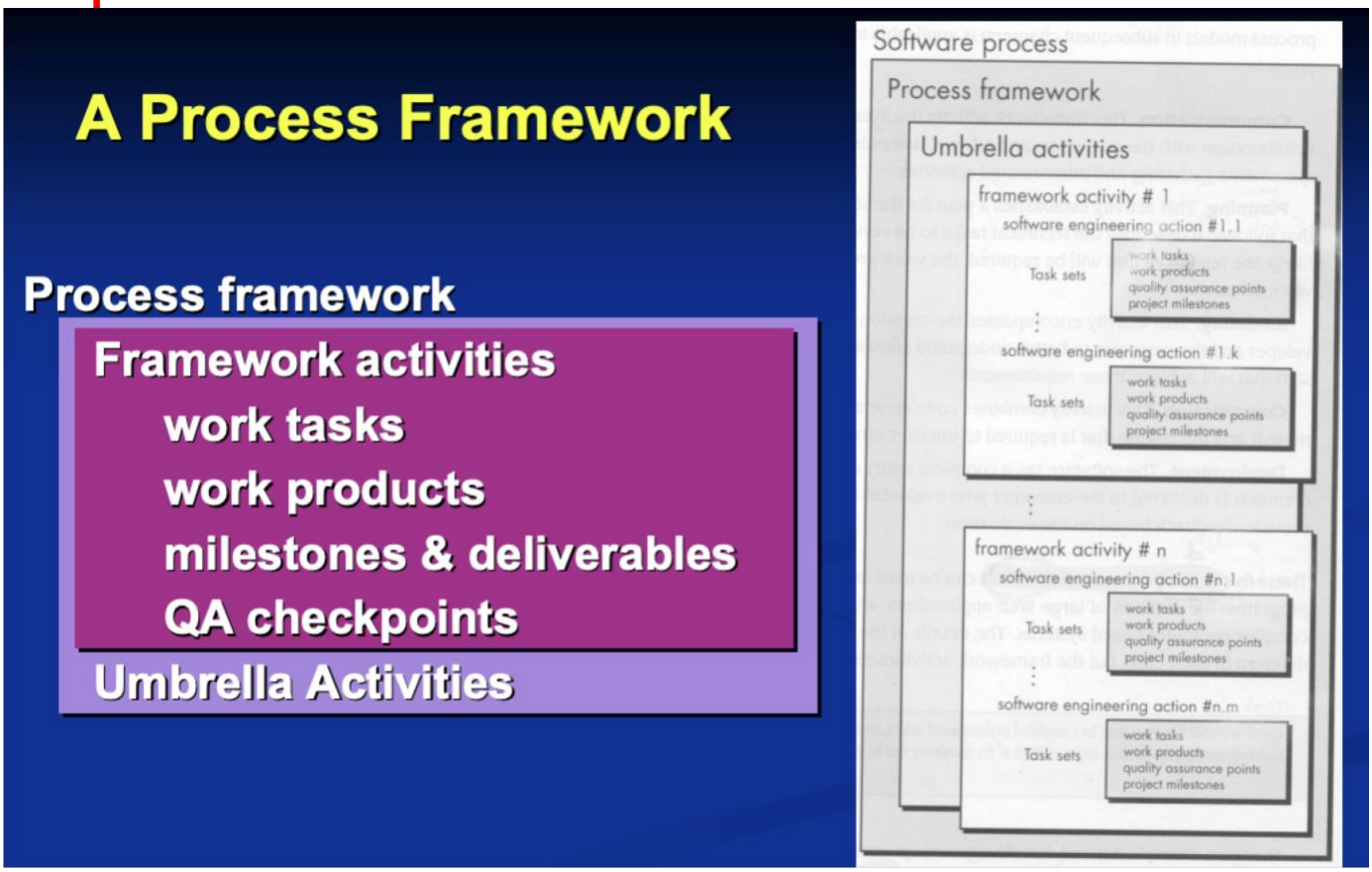
**Method**: This layer contains the methods, the technical knowledge and “how-tos” in order to develop software.

**Process**: This layer consists of the framework that must be established for the effective delivery of software.

**A Quality Focus**: This layer is the fundamental layer for software engineering. As stated above it is of great importance to test the end product to see if it meets its specifications. Efficiency, usability, maintenance and reusability are some of the requirements that need to be met by new software.

Having Tools, Methods and Processes laid out from the beginning of any software engineering process makes it an easier task for both developers and project managers to check the quality of the end product and deliver a more complex software on time by staying on budget.

## 2.A process framework



By Shubh Thakare +919595939264

# Framework Activities

- Communication
- Planning
- Modeling
  - Analysis of requirements
  - Design
- Construction
  - Code generation
  - Testing
- Deployment

# Umbrella Activities

- Software project management
- Risk management
- Software quality assurance
- Formal technical reviews
- Software configuration management
- Work product preparation and production
- Reusability management

# 2 PROCESS MODELS

Building computer software is an iterative social learning process, and the outcome, something that Baetjer [Bae98] would call "software capital," is an embodiment of knowledge collected, distilled, and organized as the process is conducted.

But what exactly is a software process from a technical point of view? Within the context of this book, we define a *software process* as a framework for the activities, actions, and tasks required to build high-quality software. Is "process" synonymous with "software engineering"? The answer is yes and no. A software process defines the approach that is taken as software is engineered. But software engineering also encompasses technologies that populate the process—technical methods and automated tools.

More important, software engineering is performed by creative, knowledgeable people who should adapt a mature software process so that it is appropriate for the products that they build and the demands of their marketplace.

## THE SOFTWARE PROCESS

In this part of *Software Engineering: A Practitioner's Approach,* you'll learn about the process that provides a framework for software engineering practice. These questions are addressed in the chapters that follow:

- What is a software process?
- What are the generic framework activities that are present in every software process?
- How are processes modeled, and what are process patterns?
- What are the prescriptive process models, and what are their strengths and weaknesses?
- Why is *agility* a watchword in modern software engineering work?
- What is agile software development, and how does it differ from more traditional process models?

Once these questions are answered, you'll be better prepared to understand the context in which software engineering practice is applied.

# The Process Model: Adaptability

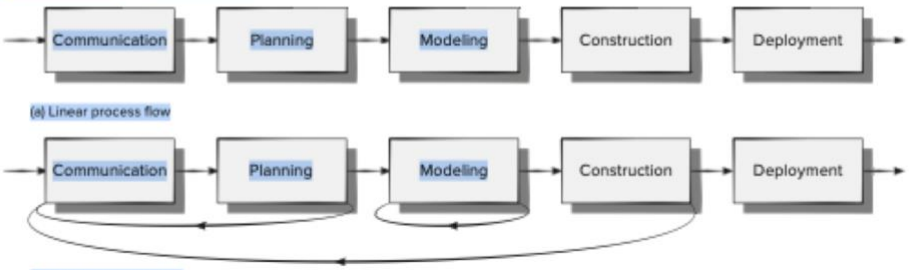
- the framework activities will always be applied on every project ... BUT
- the tasks (and degree of rigor) for each activity will vary based on:
  - the type of project
  - characteristics of the project
  - common sense judgment; concurrence of the project team



risk management, quality assurance, configuration management, technical reviews, and others—are applied throughout the process.

You should note that one important aspect of the software process has not been discussed yet. This aspect—called *process flow*—describes how the framework activities and the actions and tasks that occur within each framework activity are organized with respect to sequence and time. It is illustrated in Figure 2.2.

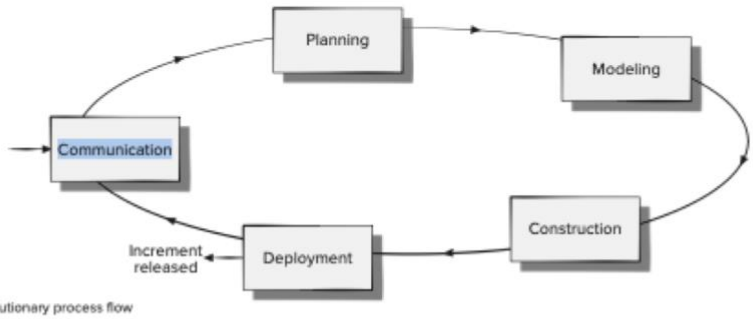
**FIGURE 2.2** Process flow



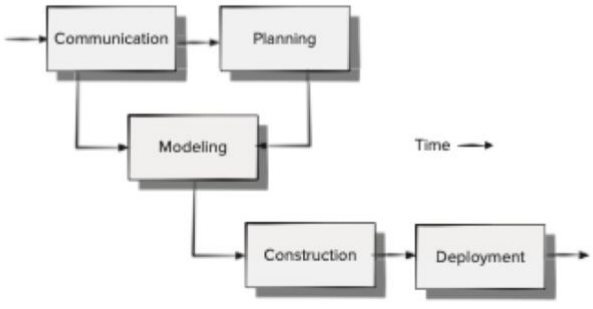

(a) Linear process flow




(b) Iterative process flow




(c) Evolutionary process flow




(d) Parallel process flow

By Shubh Thakare +919595939264

# 3.The Capability Maturity Model Integration (CMMI)

## What is CMMI? A model for optimizing development processes

**The Capability Maturity Model Integration (CMMI) helps organizations streamline process improvement, encouraging a productive, efficient culture that decreases risks in software, product and service development.**

## What is CMMI?

The Capability Maturity Model Integration (CMMI) is a process and behavioral model that helps organizations streamline process improvement and encourage productive, efficient behaviors that decrease risks in software, product and service development.

The CMMI was developed by the Software Engineering Institute at Carnegie Mellon University as a process improvement tool for projects, divisions or organizations. The DoD and U.S. Government helped develop the CMMI, which is a common requirement for DoD and U.S. Government software development contracts. The CMMI is currently administered by the CMMI Institute, which was purchased by the ISACA in 2016.

## CMMI model

The CMMI starts with an appraisal process that evaluates three specific areas: process and service development, service establishment and management, and product and service acquisition. It's designed to help improve performance by providing businesses with everything they need to consistently develop better products and services.

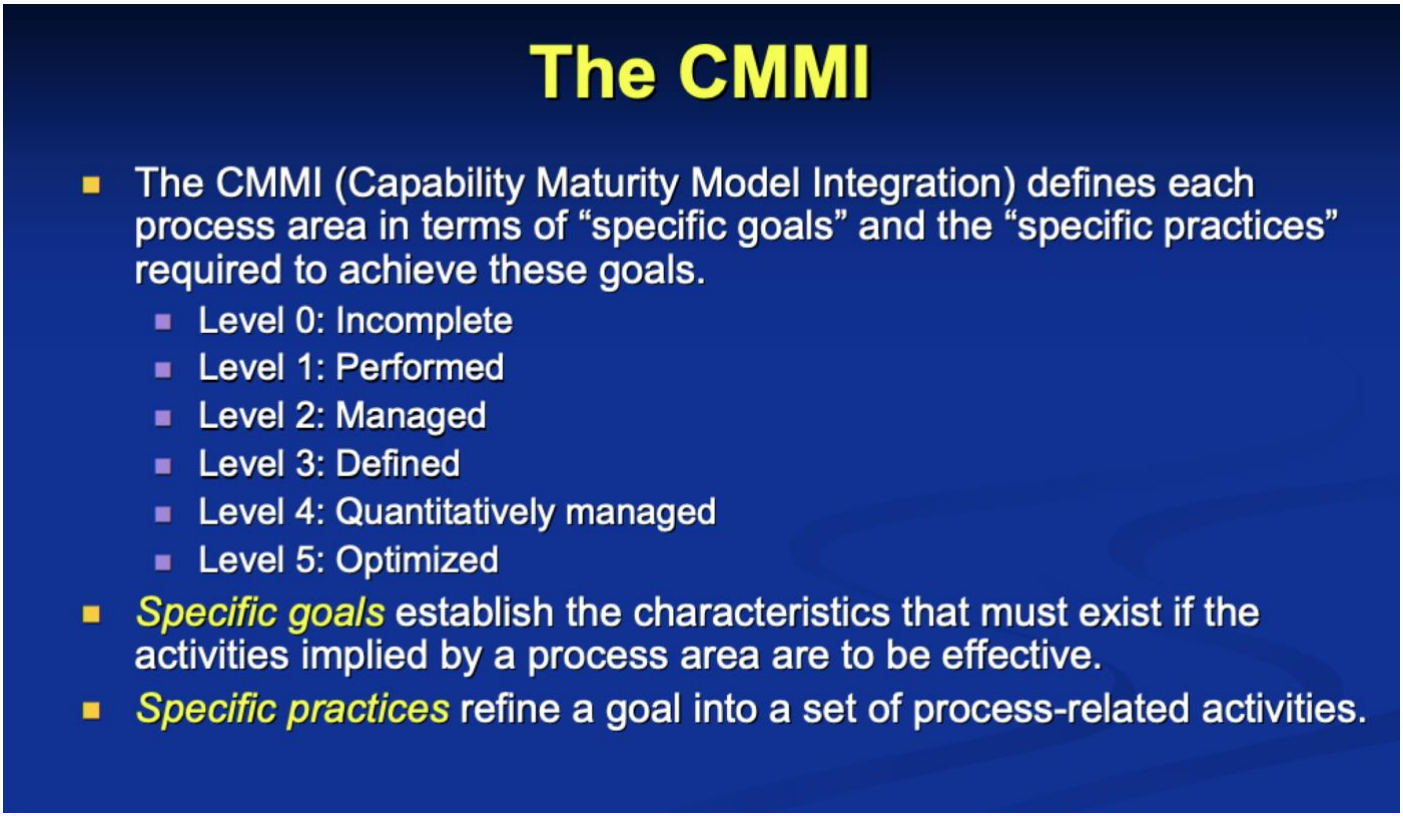
But the CMMI is more than a process model; it's also a behavioral model. Businesses can use the CMMI to tackle the logistics of improving performance by developing measurable benchmarks, but it can also create a structure for encouraging productive, efficient behavior throughout the organization.

## Evolution of CMMI

The CMMI was developed to combine multiple business maturity models into one framework. It was born from the Software CMM model developed between 1987 and 1997. CMMI Version 1.1 was released in 2002, followed by Version 1.2 in 2006 and Version 1.3 in 2010; V1.3 is currently being replaced by V2.0, which will be released in phases starting March 2018.

In its first iteration as the Software CMM, the model was tailored to software engineering. The latest version of CMMI is more abstract and generalized, allowing it to be applied to hardware, software and service development across every industry.

Every iteration of the CMMI aims to be easier for businesses to understand and use than the last, and each model is designed to be more cost-effective and easier to integrate or deploy. It encourages businesses to focus on quality over quantity by establishing benchmarks for vetting vendors and suppliers, identifying and resolving process issues, minimizing risk and building a corporate culture that will support the CMMI model.




## 4.Process patterns

**Process Patterns in Software Engineering**

As the software team moves through the software process they encounter problems. It would be very useful if solutions to these problems were readily available so that problems can be resolved quickly. Process-related problems which are encountered during software engineering work, it identifies the encountered problem and in which environment it is found, then it will suggest proven solutions to problem, they all are described by process pattern. By solving problems a software team can construct a process that best meets needs of a project.

**Uses of the process pattern :** At any level of abstraction, patterns can be defined. They can be used to describe a problem and solution associated with framework activity in some situations. While in other situations patterns can be used to describe a problem and solution associated with a complete process model.

**Template :**

- **Pattern Name –** Meaningful name must be given to a pattern within context of software process (e.g. Technical Reviews).
- **Forces –** The issues that make problem visible and may affect its solution also environment in which pattern is encountered.

**Type :** It is of three types :

- **Stage pattern –** Problems associated with a framework activity for process are described by stage pattern. Establishing Communication might be an example of a staged pattern. This pattern would incorporate task pattern Requirements Gathering and others.
- **Task-pattern –** Problems associated with a software engineering action or work task and relevant to successful software engineering practice (e.g., Requirements Gathering is a task pattern) are defined by task-pattern.
- **Phase pattern –** Even when the overall flow of activities is iterative in nature, it defines sequence of framework activities that occurs within process. Spiral Model or Prototyping might be an example of a phase pattern.

**Process patterns** can be defined as the set of activities, actions, work tasks or work products and similar related behaviour followed in a software development life cycle.

Process patterns can be more easily understood by dividing it into terms, Process which means the steps followed to achieve a task and patterns which means the recurrence of same basic features during the lifecycle of a process. Thus in a more universal term process patterns are common or general solution for a complexity.

Typical Examples are:

- Customer communication (a process activity).
- Analysis (an action).
- Requirements gathering (a process task).
- Reviewing a work product (a process task).
- Design model (a work product).

# 5.Process assessment

A software process assessment is a disciplined examination of the software processes used by an organization, based on a process model. The assessment includes the identification and characterization of current practices, identifying areas of strengths and weaknesses, and the ability of current practices to control or avoid significant causes of poor (software) quality, cost, and schedule.

A software assessment (or audit) can be of three types.
A self-assessment (first-party assessment) is performed internally by an organization's own personnel.
A second-party assessment is performed by an external assessment team or the organization is assessed by a customer.
A third-party assessment is performed by an external party or (e.g., a supplier being assessed by a third party to verify its ability to enter contracts with a customer).

Software Process Maturity Assessment
The scope of a software process assessment can cover all the processes in the organization, a selected subset of the software processes, or a specific project. Most of the standard-based process assessment approaches are invariably based on the concept of process maturity.
When the assessment target is the organization, the results of a process assessment may differ, even on successive applications of the same method. There are two reasons for the different results. They are,
The organization being investigated must be determined. For a large company, several definitions of organization are possible and therefore the actual scope of appraisal may differ in successive assessments.
Even in what appears to be the same organization, the sample of projects selected to represent the organization may affect the scope and outcome.

Software Process Assessment Cycle
According to Paulk and colleagues (1995), the CMM-based assessment approach uses a six-step cycle. They are −
Select a team - The members of the team should be professionals knowledgeable in software engineering and management.
The representatives of the site to be appraised complete the standard process maturity questionnaire.

The assessment team performs an analysis of the questionnaire responses and identifies the areas that warrant further exploration according to the CMM key process areas.

The assessment team conducts a site visit to gain an understanding of the software process followed by the site.

The assessment team produces a list of findings that identifies the strengths and weakness of the organization's software process.

The assessment team prepares a Key Process Area (KPA) profile analysis and presents the results to the appropriate audience.

## 6.Personal and team process models.

### What Is Personal Software Process (PSP)?

- The Personal Software Process (PSP) shows engineers how to
  - manage the quality of their projects
  - make commitments they can meet
  - improve estimating and planning
  - reduce defects in their products

PSP emphasizes the need to record and analyze the types of errors you make, so you can develop strategies eliminate them.

By Shubh Thakare +919595939264

# What Is Team Software Process (TSP)?

- The Team Software Process (TSP), along with the Personal Software Process, helps the high-performance engineer to

- ensure quality software products
- create secure software products
- improve process management in an organization

# Personal & Team Process Models

- The Personal Software Process (PSP) shows engineers how to
  - manage the quality of their projects
  - make commitments they can meet
  - improve estimating and planning
  - reduce defects in their products
- PSP emphasizes the need to record and analyze the types of errors you make, so you can develop strategies eliminate them.

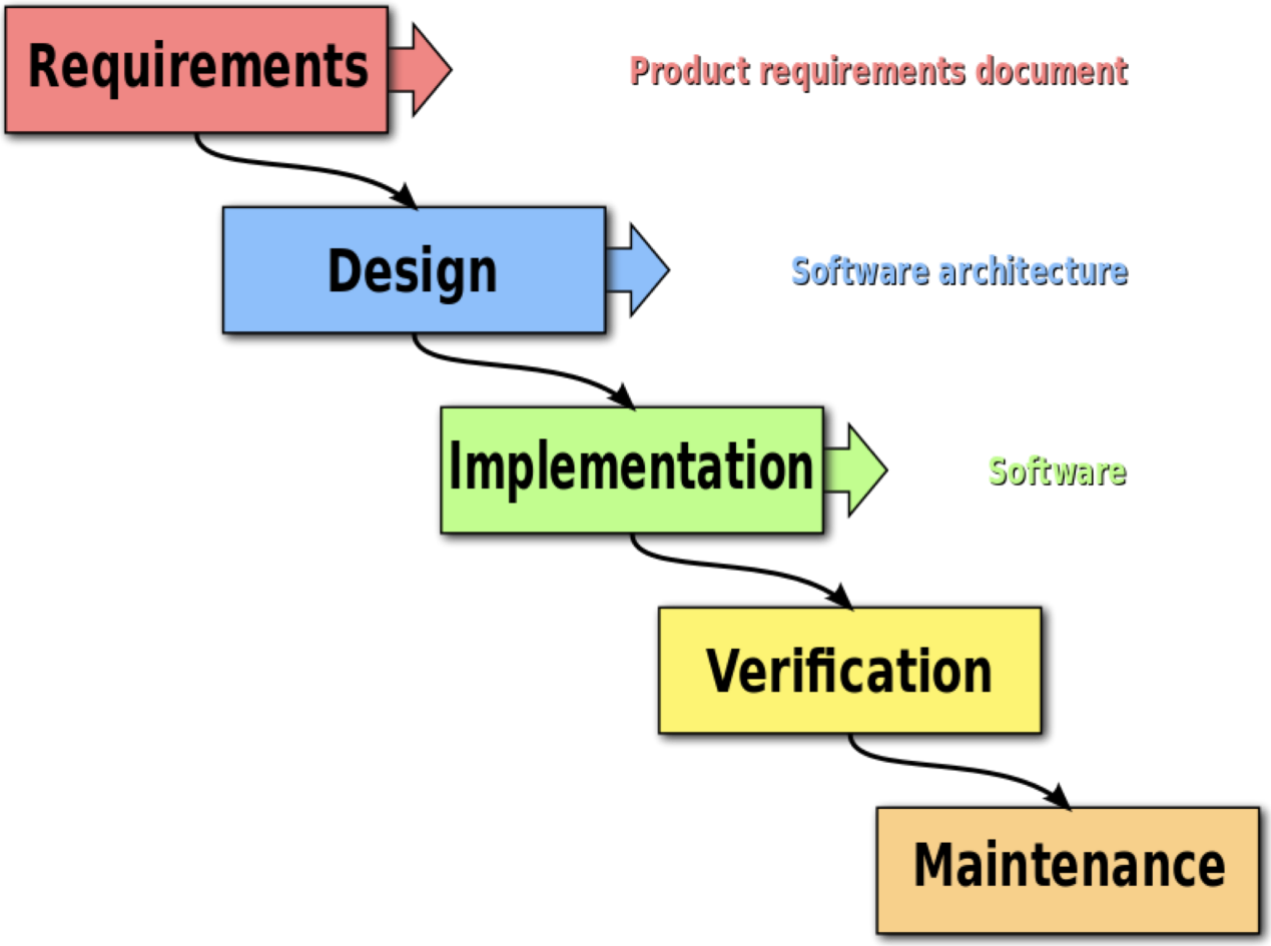
- PSP model Framework Activities
  - Planning – isolates requirements and based on these develops both size & resource estimates. A defect estimate is made.
  - High level Design – external specification of all components. All issues are recorded and tracked.
  - High level Design Review- formal verification to uncover errors
  - Development- metrics are maintained for all important tasks & work results.
  - Postmortem- using measures & metrics collected effectiveness of process is determined an improved.

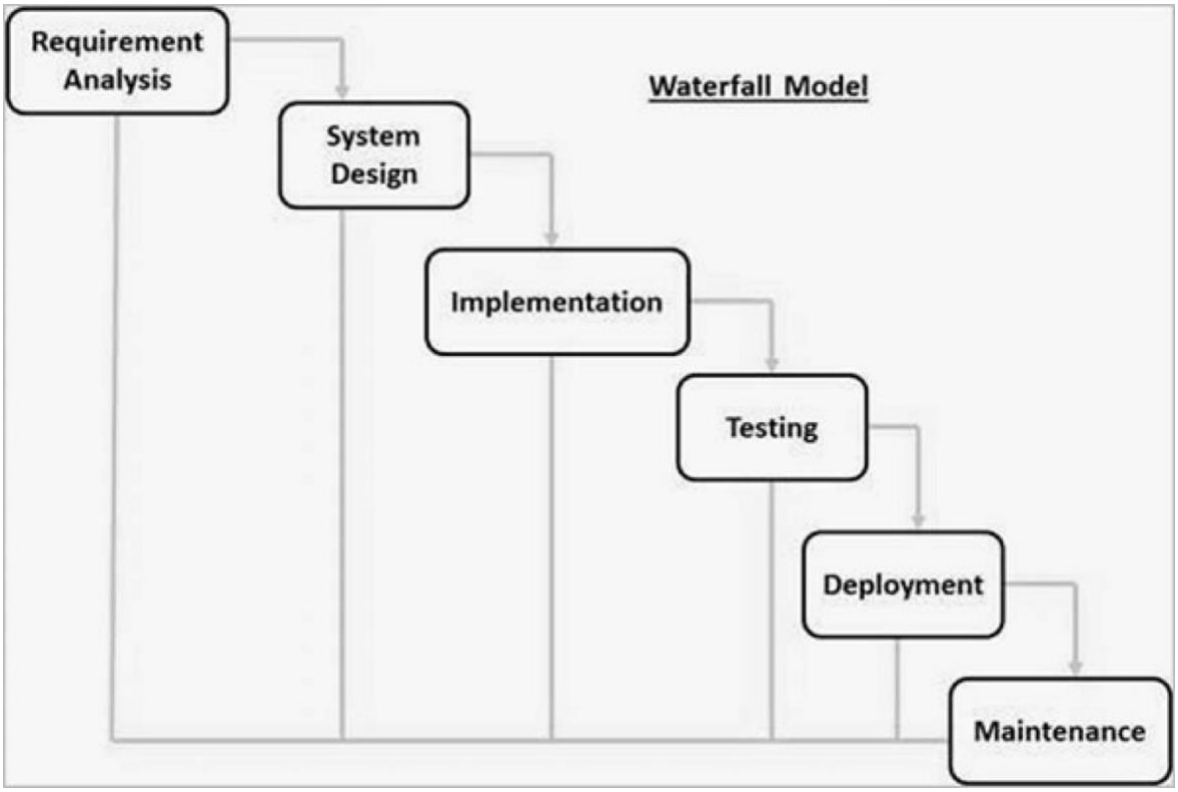
- The Team Software Process (TSP), along with the Personal Software Process, helps the high-performance engineer to
  - ensure quality software products
  - create secure software products
  - improve process management in an organization
- TSP Framework Activities
  - Launch high level design
  - Implementation
  - Integration
  - Test
  - postmortem

By Shubh Thakare +919595939264

# Process Models

## 1.The waterfall model



The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

Waterfall Model - Design

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

**The following illustration is a representation of the different phases of the Waterfall Model.**

The sequential phases in Waterfall model are −

- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

Maintenance − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better

versions are released. Maintenance is done to deliver these changes in the customer environment.

## Waterfall Model - Application

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are −

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.

The project is short.

## Waterfall Model - Advantages

The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Some of the major advantages of the Waterfall Model are as follows −

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

## 2.Incremental process models

# Incremental Model

Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.

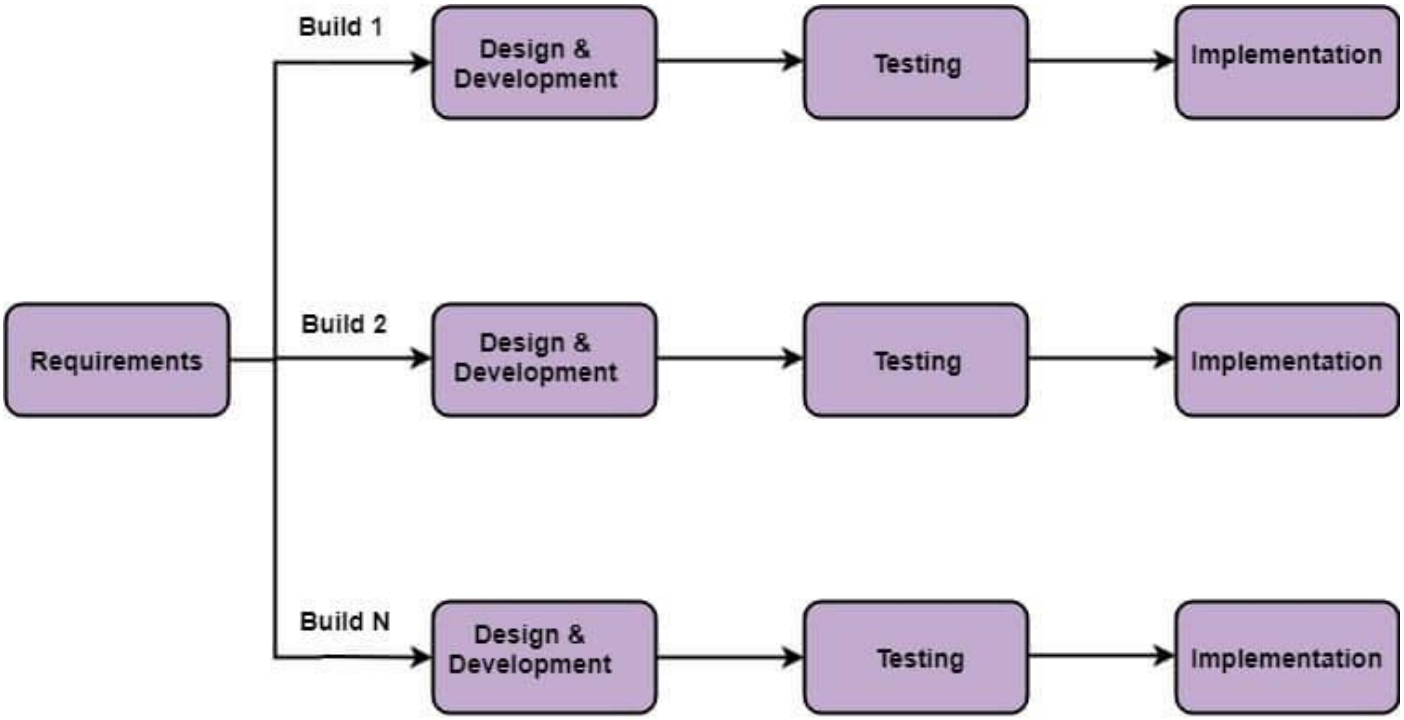



Fig: Incremental Model

## The various phases of incremental model are as follows:

**1. Requirement analysis:** In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

**2. Design & Development:** In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

**3. Testing:** In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task.

**4. Implementation:** Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

## When we use the Incremental Model?

- When the requirements are superior.
- A project has a lengthy development schedule.
- When Software team are not very well skilled or trained.
- When the customer demands a quick release of the product.
- You can develop prioritized requirements first.

## Advantage of Incremental Model

- Errors are easy to be recognized.
- Easier to test and debug
- More flexible.
- Simple to manage risk because it handled during its iteration.
- The Client gets important functionality early.

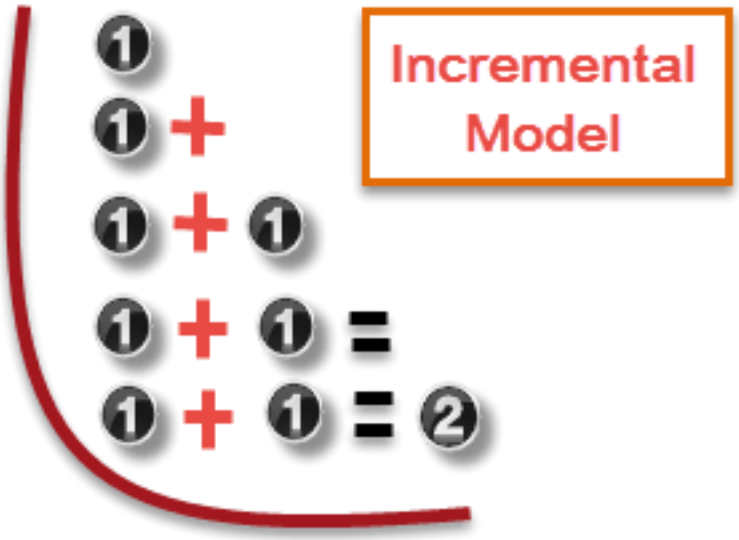
## Disadvantage of Incremental Model

- Need for good planning
- Total Cost is high.
- Well defined module interfaces are needed.

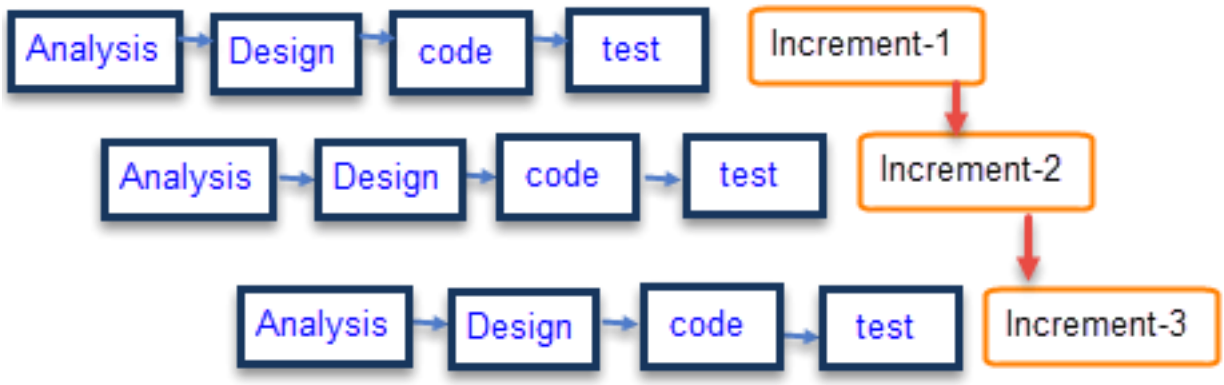
### What is Incremental Model?

Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle. Incremental development is done in steps from analysis design, implementation, testing/verification, maintenance.

**By Shubh Thakare +919595939264**




Each iteration passes through the **requirements, design, coding and testing phases**. And each subsequent release of the system adds function to the previous release until all designed functionality has been implemented.




Incremental Model

The system is put into production when the first increment is delivered. The first increment is often a core product where the basic requirements are addressed, and supplementary features are added in the next increments. Once the core product is analyzed by the client, there is plan development for the next increment.

## Characteristics of an Incremental module includes

- System development is broken down into many mini development projects
- Partial systems are successively built to produce a final total system
- Highest priority requirement is tackled first

- Once the requirement is developed, requirement for that increment are frozen

| Incremental Phases | Activities performed in incremental phases |
|---|---|
| **Requirement Analysis** | • Requirement and specification of the software are collected |
| **Design** | • Some high-end function are designed during this stage |
| **Code** | • Coding of software is done during this stage |
| **Test** | • Once the system is deployed, it goes through the testing phase |

## When to use Incremental models?

- Requirements of the system are clearly understood
- When demand for an early release of a product arises
- When software engineering team are not very well skilled or trained
- When high-risk features and goals are involved
- Such methodology is more in use for web application and product based companies

## Advantages and Disadvantages of Incremental Model

| Advantages | Disadvantages |
|---|---|
| • The software will be generated quickly during the software life cycle | • It requires a good planning designing |
| • It is flexible and less expensive to change requirements and scope | • Problems might cause due to system architecture as such not all requirements collected up front for the entire software lifecycle |
| • Throughout the development stages changes can be done | • Each iteration phase is rigid and does not overlap each other |
| • This model is less costly compared to others | • Rectifying a problem in one unit requires correction in all the units and consumes a lot of time |

- A customer can respond to each building

- Errors are easy to be identified

## 3.Evolutionary process models

- Evolutionary models are iterative type models.
- They allow to develop more complete versions of the software.

**Following are the evolutionary process models.**

1. The prototyping model
2. The spiral model
3. Concurrent development model

### 1. The Prototyping model

- Prototype is defined as first or preliminary form using which other forms are copied or derived.
- Prototype model is a set of general objectives for software.
- It does not identify the requirements like detailed input, output.
- It is software working model of limited functionality.
- In this model, working programs are quickly produced.

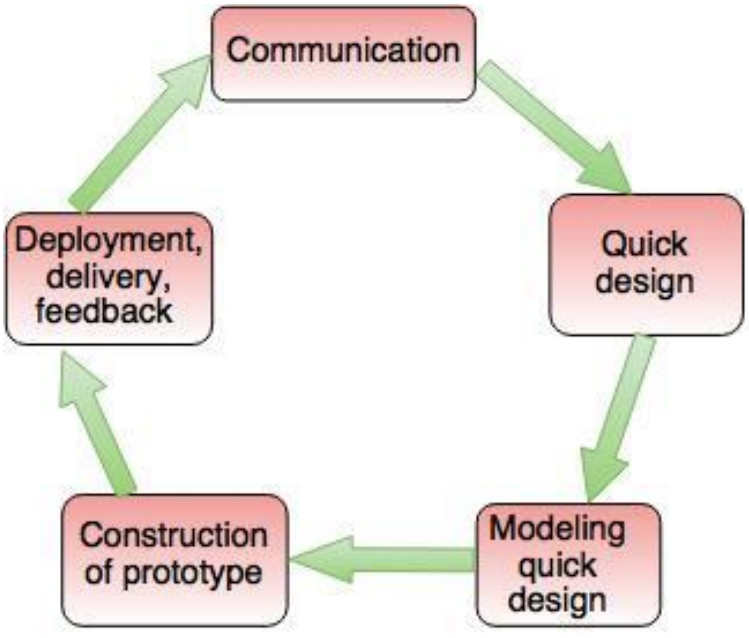



**Fig. - The Prototyping Model**

**The different phases of Prototyping model are:**

**1. Communication**

In this phase, developer and customer meet and discuss the overall objectives of the software.

**2. Quick design**

- Quick design is implemented when requirements are known.
- It includes only the important aspects like input and output format of the software.
- It focuses on those aspects which are visible to the user rather than the detailed plan.
- It helps to construct a prototype.

**3. Modeling quick design**

- This phase gives the clear idea about the development of software because the software is now built.
- It allows the developer to better understand the exact requirements.

**4. Construction of prototype**

The prototype is evaluated by the customer itself.

**5. Deployment, delivery, feedback**

- If the user is not satisfied with current prototype then it refines according to the requirements of the user.
- The process of refining the prototype is repeated until all the requirements of users are met.
- When the users are satisfied with the developed prototype then the system is developed on the basis of final prototype.

**Advantages of Prototyping Model**

- Prototype model need not know the detailed input, output, processes, adaptability of operating system and full machine interaction.
- In the development process of this model users are actively involved.
- The development process is the best platform to understand the system by the user.
- Errors are detected much earlier.
- Gives quick user feedback for better solutions.
- It identifies the missing functionality easily. It also identifies the confusing or difficult functions.

**Disadvantages of Prototyping Model:**

- The client involvement is more and it is not always considered by the developer.
- It is a slow process because it takes more time for development.
- Many changes can disturb the rhythm of the development team.
- It is a thrown away prototype when the users are confused with it.

## 2. The Spiral model

- Spiral model is a risk driven process model.
- It is used for generating the software projects.
- In spiral model, an alternate solution is provided if the risk is found in the risk analysis, then alternate solutions are suggested and implemented.
- It is a combination of prototype and sequential model or waterfall model.
- In one iteration all activities are done, for large project's the output is small.

**The framework activities of the spiral model are as shown in the following figure.**

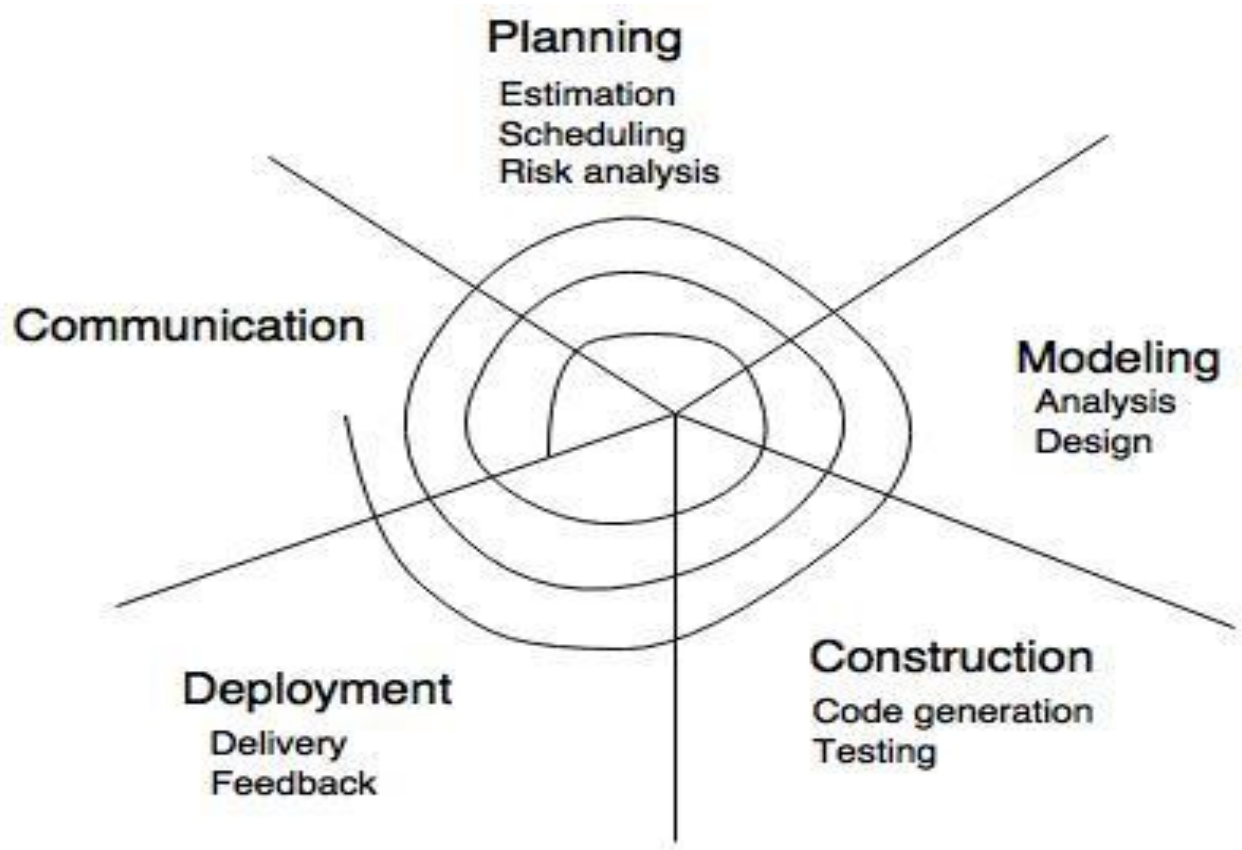



**Fig. - The Spiral Model**

**NOTE:** The description of the phases of the spiral model is same as that of the process model.

**Advantages of Spiral Model**

- It reduces high amount of risk.
- It is good for large and critical projects.
- It gives strong approval and documentation control.
- In spiral model, the software is produced early in the life cycle process.

**Disadvantages of Spiral Model**

- It can be costly to develop a software model.
- It is not used for small projects.

## 3. The concurrent development model

- The concurrent development model is called as concurrent model.
- The communication activity has completed in the first iteration and exits in the awaiting changes state.
- The modeling activity completed its initial communication and then go to the underdevelopment state.
- If the customer specifies the change in the requirement, then the modeling activity moves from the under development state into the awaiting change state.
- The concurrent process model activities moving from one state to another state.

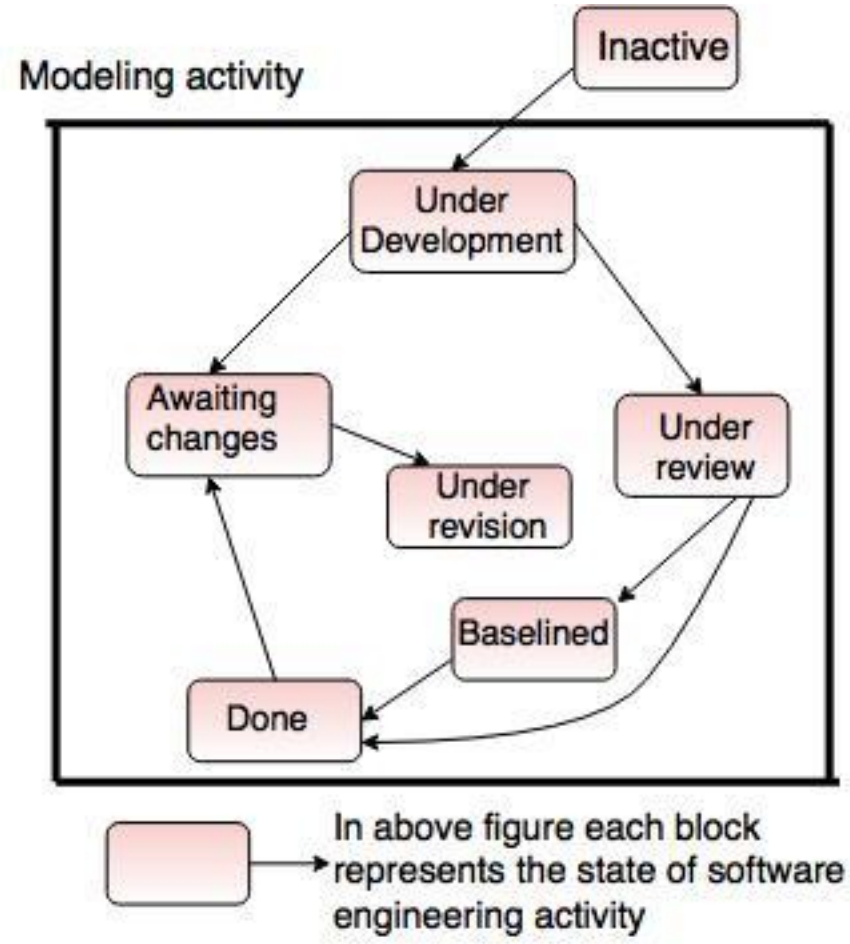



**Fig. - One element of the concurrent process model**

**Advantages of the concurrent development model**

- This model is applicable to all types of software development processes.
- It is easy for understanding and use.
- It gives immediate feedback from testing.

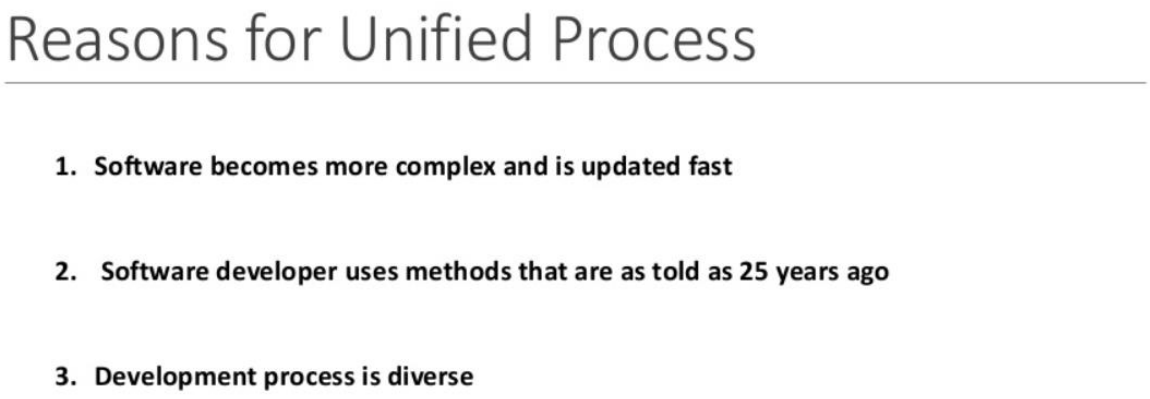
- It provides an accurate picture of the current state of a project.

**Disadvantages of the concurrent development model**

- It needs better communication between the team members. This may not be achieved all the time.
- It requires to remember the status of the different activities.

## 4.The Unified process

UML, the unified modeling language, was developed to support their work. UML
contains a robust notation for the modeling and development of object-oriented systems and has became a de facto industry standard for modeling software of all types. UML is used throughout Part Two of this book to represent both requirements
and design models. Appendix 1 presents an introductory tutorial and a list of recommended
books for those who are unfamiliar with basic UML notation and modeling rules.



# What does Unified Process do?

1. Provides guidance to the order of team's activities
2. Integrates team's work and individual's work
3. Specifies artifacts
4. Offers criteria for monitoring and measuring

# Lifecycle of Unified Process

- Each cycle concludes with a product release to customers
- Each cycle consist of four phases:

1. Inception
2. Elaboration
3. Construction
4. Transition

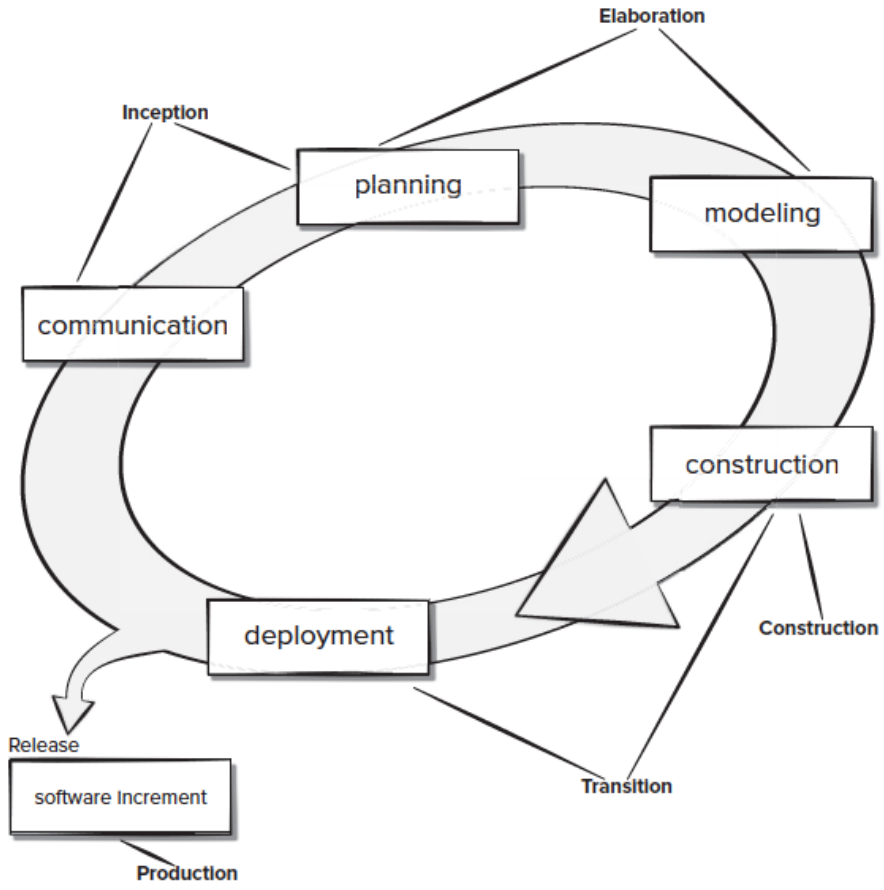
# UNIFIED PROCESS WORK PRODUCTS

- Tasks which are required to be completed during different phases
- Inception Phase
  *Vision document *Initial Use-Case model *Initial Risk assessment *Project Plan
- Elaboration Phase
  *Use-Case model *Analysis model *Software Architecture description *Preliminary design model *Preliminary model

# UNIFIED PROCESS WORK PRODUCTS

- Construction Phase
  *Design model
  *System components
  *Test plan and procedure
  *Test cases
  *Manual
- Transition Phase
  *Delivered software increment
  *Beta test results
  *General user feedback

The Unified Process




*Inception*—the first and the shortest phase in the project. It is used to prepare basis for the project, including preparation of business case, establishing project scope and setting boundaries, outlining key

requirements, and possible architecture solution together with design tradeoffs, identifying risks, and development of initial project plan—schedule with main milestones and cost estimates. If the inception phase lasts for too long, it is like an indicator stating that the project vision and goals are not clear to the stakeholders. With no clear goals and vision the project most likely is doomed to fail. At this scenario it is better to take a pause at the very beginning of the project to refine the vision and goals. Otherwise it could lead to unnecessary make-overs and schedule delays in further phases.

- *Elaboration*—during this phase the project team is expected to capture a majority of system's requirements (e.g., in the form of use cases), to perform identified risk analysis and make a plan of risk management to reduce or eliminate their impact on final schedule and product, to establish design and architecture (e.g., using basic class diagrams, package diagrams, and deployment diagrams), to create a plan (schedule, cost estimates, and achievable milestones) for the next (construction) phase.
- *Construction*—the longest and largest phase within Unified Process. During this phase, the design of the system is finalized and refined and the system is built using the basis created during elaboration phase. The construction phase is divided into multiple iterations, for each iteration to result in an executable release of the system. The final iteration of construction phase releases fully completed system which is to be deployed during transition phase, and
- *Transition*—the final project phase which delivers the new system to its end-users. Transition phase includes also data migration from legacy systems and user trainings.

Each phase and its iteration consists of a set of predefined activities. The Unified Process describes work activities as disciplines—a discipline is a set of activities and related artifacts in one subject area (e.g., the activities within requirements analysis). The disciplines described by Unified Process are as follows [107]:

- *Business modeling*—domain object modeling and dynamic modeling of the business processes,
- *Requirements*—requirements analysis of system under consideration. Includes activities like writing use cases and identifying nonfunctional requirements,
-

*Analysis and design*—covers aspects of design, including the overall architecture,

•

*Implementation*—programming and building the system (except the deployment),

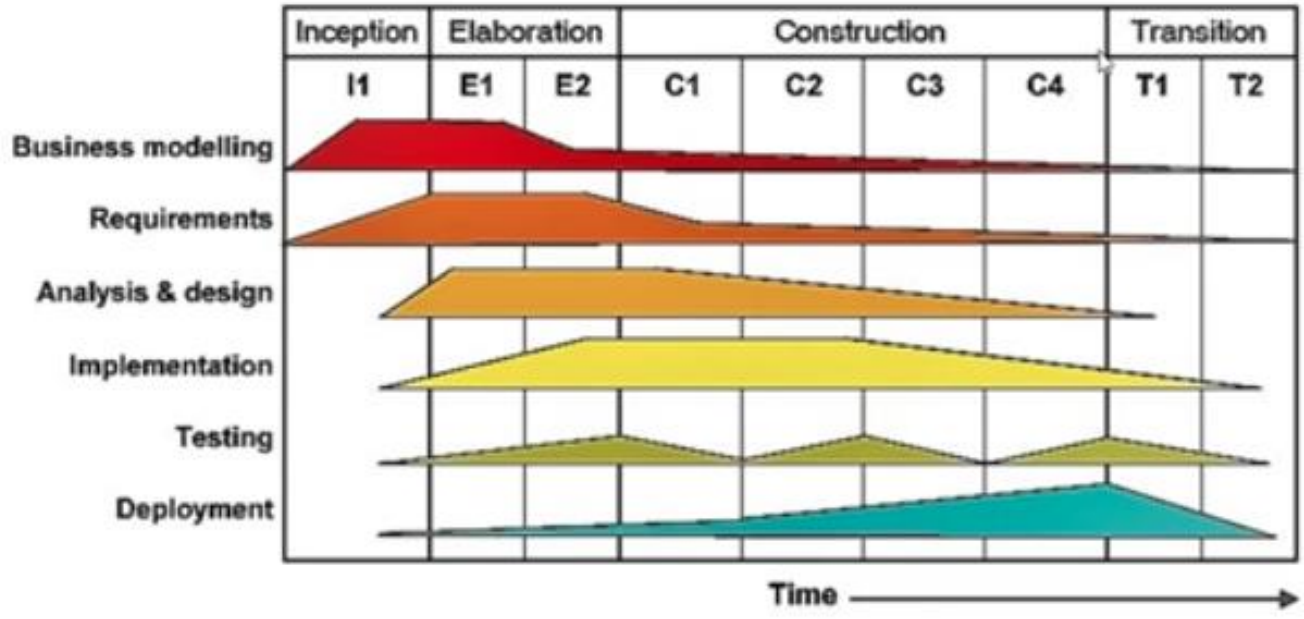
•

*Test*—involves testing activities such as test planning, development of test scenarios, alpha and beta testing, regression testing, acceptance testing, and

•

*Deployment*—the deployment activities of developed system.

## Unified Process with Fig.



By Shubh Thakare +919595939264

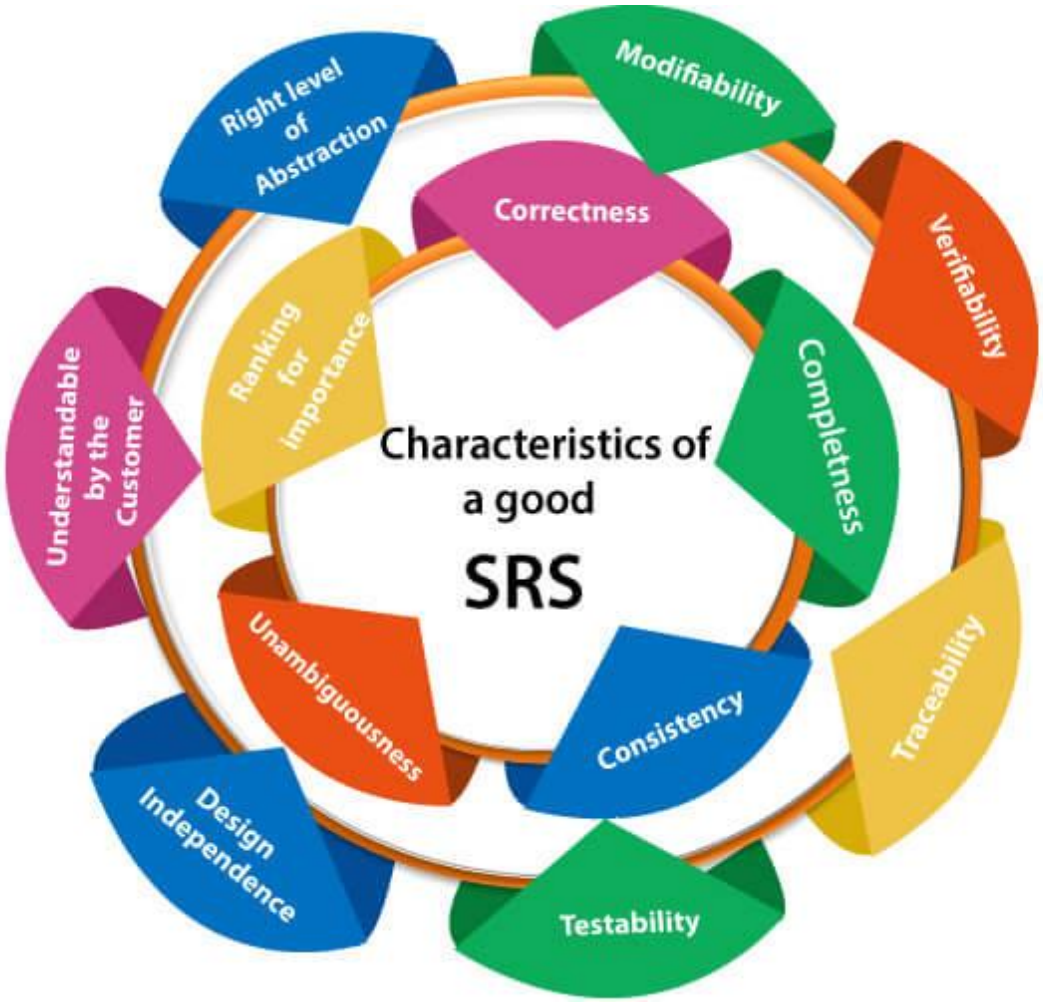
# Requirement Engineering

## What is a Requirement?

The requirements are the high-level descriptions about a particular system services, constraints or to a detailed specification that are generated during the requirements gathering process.

## Requirement Types:

- **User Requirements -** It is a detailed description in natural language along with diagrams of the services the system provides and its operational constraints. It is usually developed by end users.
- **System requirements -** It is a structured document detailing the descriptions of the system's functions, services and operational constraints.
- **Functional Requirements -** It describes the services of the system, how the system should react to particular inputs and how the system should behave in definite situations.
- **Non-functional Requirements -** It describes the attributes of the system.
- **Domain Requirements -** Requirements that arises from the domain of the application and that reflect characteristics of that domain. It can be either functional or non-functional specifications.



**By Shubh Thakare +919595939264**

# 1.Functional and non-functional requirements

## What is a Functional Requirement?

In software engineering,

a functional requirement defines a system or its component. It describes the functions a software must perform. A function is nothing but inputs, its behavior, and outputs. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.

## What is Non-Functional Requirement?

A non-functional requirement defines the quality attribute of a software system. They represent a set of standards used to judge the specific operation of a system. Example, how fast does the website load?

A non-functional requirement is essential to ensure the usability and effectiveness of the entire software system. Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

Non-functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement.

## Example of Functional Requirements

- The software automatically validates customers against the ABC Contact Management System
- The Sales system should allow users to record customers sales
- The background color for all windows in the application will be blue and have a hexadecimal RGB color value of 0x0000FF.
- Only Managerial level employees have the right to view revenue data.
- The software system should be integrated with banking API
- The software system should pass Section 508 accessibility requirement.

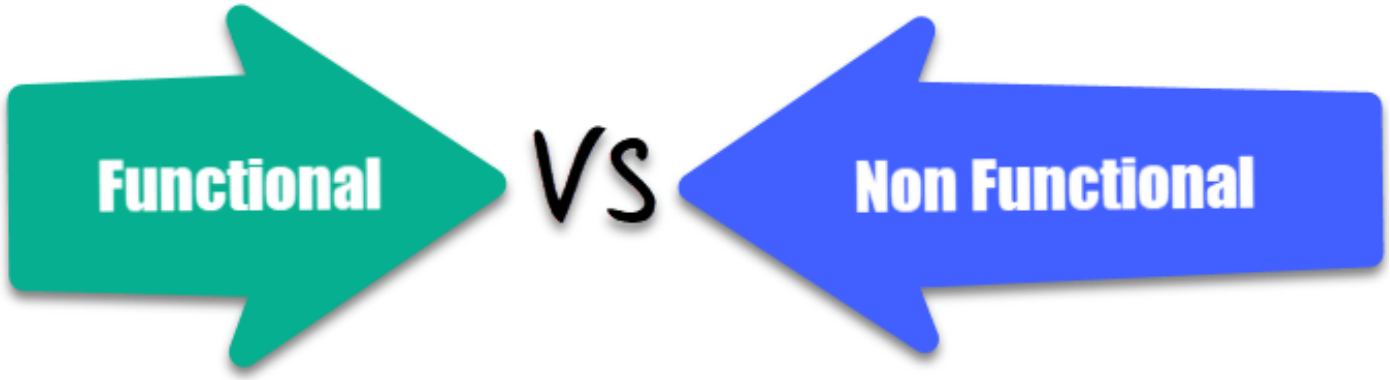
By Shubh Thakare +919595939264

## Examples of Non-functional requirements

Here, are some examples of non-functional requirement:

1. Users must change the initially assigned login password immediately after the first successful login. Moreover, the initial should never be reused.
2. Employees never allowed to update their salary information. Such attempt should be reported to the security administrator.
3. Every unsuccessful attempt by a user to access an item of data shall be recorded on an audit trail.
4. A website should be capable enough to handle 20 million users with affecting its performance
5. The software should be portable. So moving from one OS to other OS does not create any problem.
6. Privacy of information, the export of restricted technologies, intellectual property rights, etc. should be audited.

## Functional vs Non Functional Requirements




| Parameters | Functional Requirement | Non-Functional Requirement |
|---|---|---|
| What it is | Verb / Functions | Attributes |
| Requirement | It is mandatory | It is non-mandatory |
| Capturing type | It is captured in use case. | It is captured as a quality attribute. |
| End-result | Product feature | Product properties |
| Capturing | Easy to capture | Hard to capture |
| Objective | Helps you verify the functionality of the software. | Helps you to verify the performance of the software. |
| Area of focus | Focus on user requirement | Concentrates on the user's expectation. |
| Documentation | Describe what the product does | Describes how the product works |
| Type of Testing | Functional Testing like System, Integration, End to End, API testing, etc. | Non-Functional Testing like Performance, Stress, Usability, Security testing, etc. |

| Parameters | Functional Requirement | Non-Functional Requirement |
|---|---|---|
| Test Execution | Test Execution is done before non-functional testing. | After the functional testing |
| Product Info | Product Features | Product Properties |

## Advantages of Functional Requirement

Here, are the pros/advantages of creating a typical functional requirement document-

- Helps you to check whether the application is providing all the functionalities that were mentioned in the functional requirement of that application
- A functional requirement document helps you to define the functionality of a system or one of its subsystems.
- Functional requirements along with requirement analysis help identify missing requirements. They help clearly define the expected system service and behavior.
- Errors caught in the Functional requirement gathering stage are the cheapest to fix.
- Support user goals, tasks, or activities for easy project management
- Functional requirement can be expressed in Use Case form or user story as they exhibit externally visible functional behavior.

## Advantages of Non-Functional Requirement

Benefits/pros of Non-functional testing are:

- The nonfunctional requirements ensure the software system follow legal and compliance rules.
- They ensure the reliability, availability, and performance of the software system
- They ensure good user experience and ease of operating the software.
- They help in formulating security policy of the software system.

## 2.User 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
- Strategical use of color and texture.
- Provide help information
- User centric approach
- Group based view settings.

## 3.System requirements

# System Requirements

System requirements is a statement that identifies the functionality that is needed by a system in order to satisfy the customer's requirements.

[1] System requirements are a broad and also narrow subject that could be implemented to many items. Whether discussing the system requirements for certain computers, software, or the business processes from a broad view point. Also, taking it down to the exact hardware or coding that runs the software. System requirements are the most effective way of meeting the user needs and reducing the cost of implementation.

[2] System requirements could cause a company to save a lot of money and time, and also can cause a company to waste money and time. They are the first and foremost important part of any project, because if the system requirements are not fulfilled, than the project is not complete.

### What Are System Requirements

System requirements are a broad and also narrow detailed statement that the customer makes in order to achieve their requirements. The statement should clearly explain what the customer exactly wants and how they want it. A customer's need might be to satisfy a contract, solve a problem, achieve an objective, meet a standard, or to meet any other guidelines of the project.

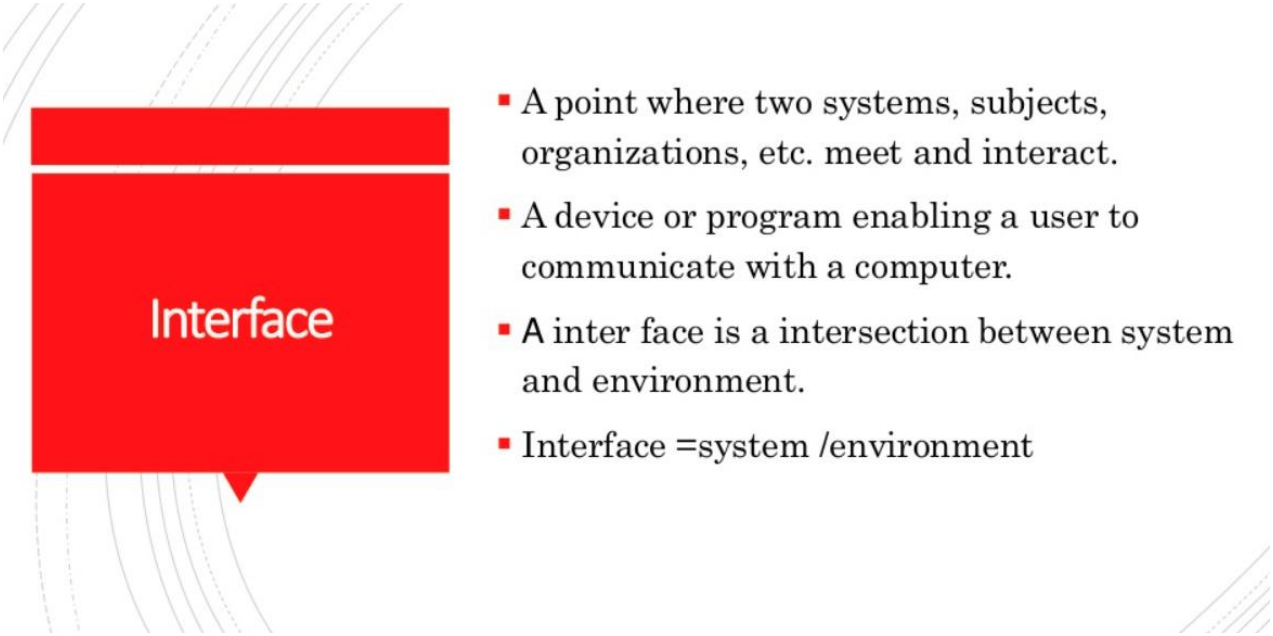
[1] System requirements will always vary depending on the project and no two systems would have identical requirements. For instance, if two companies were wanting to build a wireless phone. Company A could have requirements for the phone to be different colors, light weight, and transparent.

2}Company B could have touch screen, all black, and also to be in-between 1 lbs. – 1.20 lbs. for their requirements. These examples are just the simple constraints for a product. There could be thousands of other requirements for the wireless phone. Including, chip maker, what products to use, and cost of course. There is no such thing as a correct amount of system requirements. This can vary from product to product or solution to solution.

A flying plane could have millions of requirements, compared to a bicycle that could have a hundred requirements. As well, the requirements for solving the issue of hand washing would be a lot less than the requirements for sending a person to Mars.

[3] Yet, having too many system requirements for a project could cause the cost and time to dramatically increase. On the other side of that, having to few requirements might cause your system to not work correctly, or last as long as you like. Thus making the requirements the most important aspect of a system. See sample picture below for customer requirement.

## 4.Interface specification



By Shubh Thakare +919595939264

## Specification

- A Specification is a agreement Between the produce of the services Consumer of that services

## Interface Specification

- All software systems must operate with existing systems that have already been implemented and installed in an environment.
- If the new system and existing systems must work together, the interfaces of existing systems have to be precisely specified.
- These specifications should be defined early in the process and included in the requirements document.

## Types of Interface Specification

- There are three types of Interface specification:
- **Procedural interfaces.**
- **Data structures.**
- **Representations of data.**
- **Message passing interface.**

**By Shubh Thakare +919595939264**

## Procedural interfaces

**Procedural interfaces** where existing programs or sub-systems offer a range of services that are accessed by calling interface procedures. In simple words it Is used for calling the existing programs by the new programs These interfaces are sometimes called Application Programming Interfaces (APLs).

## Data structures

- **Data structures** that are passed from one sub-system to another. Graphical data models are the best notations for this type of description

## Representations of data

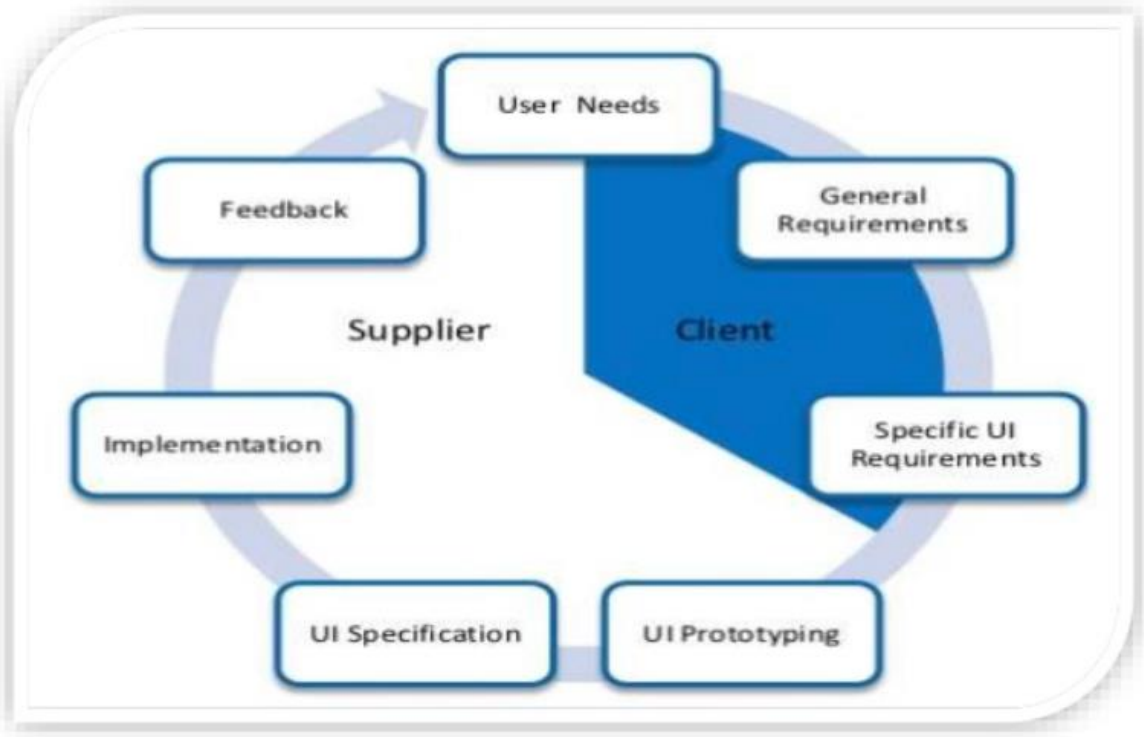
- **Representations of data** (such as the ordering of bits) that have been established for an existing sub-system. These interfaces are most common in embedded, real-time system. Some programming languages such as Ada (although not Java) support this level of Specification.

By Shubh Thakare +919595939264

Message passing interface

- Sub system requesting service from other sub systems.

# Interface Specification Cycle




## 5.The software requirements document

Without a map (or well, Google Maps!), it can get tough traveling to a new city or country. You wouldn't know what transport to take or which direction to travel in, making it almost impossible to reach your destination.

Similarly, in software development, you are highly unlikely to create the right product without proper documentation of software requirements.

Documentation ensures that the **software development team** or other stakeholders are on the same page regarding what needs to be built and are fully aware of the goal, scope, functional requirements, challenges, and budget regarding the software. However, as much as creating software is exciting, documenting its requirements can be boring and tiresome.

These documents are often long, text-heavy, and full of technical jargon, making them very difficult to understand. This makes them highly vulnerable to misinterpretations and can thus, lead to disastrous results.

To avoid costly design mistakes, product managers and software developers often use pre-made templates that keep the documentation process to the point and easy to understand.

Before we introduce you to our awesome software requirements documentation template, let's take a quick look at what exactly is a software requirements document and what are the things one needs to include in these documents.

## What is a Software Requirements Document? – Definition

A software requirements document (also known as software requirements specifications) is a document that describes the intended use-case, features, and challenges of a software application.

These documents are created before the project has started development in order to get every stakeholder on the same page regarding the software's functionality.

Software requirements are written up by the tech team depending on the project they are working on. As non-technical colleagues, clients, and partners get involved it's important to ensure that everyone is on the same page and agree with the scope, budget, and goal of the project.

## Why a Software Requirements Document is Important?

Software requirement documents provide an important map of the product being built, the features that will be included, and much more.

This roadmap helps to keep the technical and non-technical team on the same wavelength as to what the expectations are. It helps to ensure that the product is built meeting the needs whether it's for internal purposes, for users or clients.

**By Shubh Thakare +919595939264**

# What You Should Include in Your Software Requirements Document?

A typical software requirements document should involve the following details:

**Software Requirements Document**

**1. Introduction**

**1.1 Purpose:** Set the expectations for the outcome of the product.

**1.2 Intended Audience:** Who is the software for? Who is the end-user? Will the software be used internally at a company or externally?

**1.3 Intended Use:** What is the software for? What problem is it solving?

**1.4 Scope:** Explain the scope of the software. What are the main goals and objectives? How do they relate to the company's goals?

**1.5 Definitions and Acronyms:** Provide an overview of any definitions the reader should understand before reading on.

**2. Overall Description:** Describe what you are building and for who.

**2.1 User Needs:** Explain the user needs for this software.

**2.2 Assumptions and Dependencies:** What assumptions are you making that could cause an error in your approach? Is the project reliant on any other factors that could affect the development of the software?

**3. System Features and Requirements**

**3.1 Functional Requirements:** Take time to define the functional requirements that are essential for the software to build.

**3.2 External Interface Requirements:** Are there any UX and UI requirements that you must keep in mind as you build?

**3.3 System Features:** What features are required for the software to even work.

**3.4 Non-functional Requirements:** Are there any non-functional requirements that you need to address (i.e. budget, team, etc.)