# Q1. Explain V-model with a neat diagram for planning and validation process.

**Ans-** The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as **Verification and Validation model**.

The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a highly-disciplined model and the next phase starts only after completion of the previous phase.

V-Model - Design

Under the V-Model, the corresponding testing phase of the development phase is planned in parallel. So, there are Verification phases on one side of the ‘V’ and Validation phases on the other side. The Coding Phase joins the two sides of the V-Model.

The following illustration depicts the different phases in a V-Model of the SDLC.



V-Model - Verification Phases

There are several Verification phases in the V-Model, each of these are explained in detail below.

Business Requirement Analysis

This is the first phase in the development cycle where the product requirements are understood from the customer’s perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement. This is a very important activity and needs to be managed well, as most of the customers are not sure about what exactly they need. The **acceptance test design planning** is done at this stage as business requirements can be used as an input for acceptance testing.

System Design

Once you have the clear and detailed product requirements, it is time to design the complete system. The system design will have the understanding and detailing the complete hardware and communication setup for the product under development. The system test plan is developed based on the system design. Doing this at an earlier stage leaves more time for the actual test execution later.

Architectural Design

Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken. The system design is broken down further into modules taking up different functionality. This is also referred to as **High Level Design (HLD)**.

The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood and defined in this stage. With this information, integration tests can be designed and documented during this stage.

Module Design

In this phase, the detailed internal design for all the system modules is specified, referred to as **Low Level Design (LLD)**. It is important that the design is compatible with the other modules in the system architecture and the other external systems. The unit tests are an essential part of any development process and helps eliminate the maximum faults and errors at a very early stage. These unit tests can be designed at this stage based on the internal module designs.

Coding Phase

The actual coding of the system modules designed in the design phase is taken up in the Coding phase. The best suitable programming language is decided based on the system and architectural requirements.

The coding is performed based on the coding guidelines and standards. The code goes through numerous code reviews and is optimized for best performance before the final build is checked into the repository.

Validation Phases

The different Validation Phases in a V-Model are explained in detail below.

Unit Testing

Unit tests designed in the module design phase are executed on the code during this validation phase. Unit testing is the testing at code level and helps eliminate bugs at an early stage, though all defects cannot be uncovered by unit testing.

Integration Testing

Integration testing is associated with the architectural design phase. Integration tests are performed to test the coexistence and communication of the internal modules within the system.

System Testing

System testing is directly associated with the system design phase. System tests check the entire system functionality and the communication of the system under development with external systems. Most of the software and hardware compatibility issues can be uncovered during this system test execution.

Acceptance Testing

Acceptance testing is associated with the business requirement analysis phase and involves testing the product in user environment. Acceptance tests uncover the compatibility issues with the other systems available in the user environment. It also discovers the non-functional issues such as load and performance defects in the actual user environment.

V- Model ─ Application

V- Model application is almost the same as the waterfall model, as both the models are of sequential type. Requirements have to be very clear before the project starts, because it is usually expensive to go back and make changes. This model is used in the medical development field, as it is strictly a disciplined domain.

The following pointers are some of the most suitable scenarios to use the V-Model application.

* Requirements are well defined, clearly documented and fixed.
* Product definition is stable.
* Technology is not dynamic and is well understood by the project team.
* There are no ambiguous or undefined requirements.
* The project is short.

V-Model - Pros and Cons

The advantage of the V-Model method is that it is very easy to understand and apply. The simplicity of this model also makes it easier to manage. The disadvantage is that the model is not flexible to changes and just in case there is a requirement change, which is very common in today’s dynamic world, it becomes very expensive to make the change.

The advantages of the V-Model method are as follows −

* This is a highly-disciplined model and Phases are completed one at a time.
* Works well for smaller projects where requirements are very well understood.
* 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.

The disadvantages of the V-Model method are as follows −

* High risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing.
* Once an application is in the testing stage, it is difficult to go back and change a functionality.
* No working software is produced until late during the life cycle.

# Q2. Explain System Software, application software with examples.

**ANS-**

As we know that software is a set of instructions or programs instructing a computer to do specific tasks. Software is basically a generic term used to describe computer programs. In general Scripts, applications, programs and a set of instructions are all terms often used to describe software.

Now the basis of language in which software is developed and platform which is required for its execution we can classified software as in two divisions which are System software and Application software. Following are some basic differences between System software and Application software.

| **Sr. No.** | **Key** | **System Software.** | **Application Software.** |
| --- | --- | --- | --- |
| 1 | Definition | System Software is the type of software which is the interface between application software and system. | On other hand Application Software is the type of software which runs as per user request. It runs on the platform which is provide by system software. |
| 2 | Development Language | In general System software are developed in low level language which is more compatible with the system hardware in order to interact with. | While in case of Application software high level language is used for their development as they are developed as some specific purpose software. |
| 3 | Usage | System software is used for operating computer hardware. | On other hand Application software is used by user to perform specific task. |
| 4 | Installation | System software are installed on the computer when operating system is installed. | On other hand Application software are installed according to user’s requirements. |
| 5 | User interaction | As mentioned in above points system software are specific to system hardware so less or no user interaction available in case of system software. | On other hand in application software user can interacts with it as user interface is available in this case. |
| 6 | Dependency | System software can run independently. It provides platform for running application software. | On other hand in application software can’t run independently. They can’t run without the presence of system software.. |
| 7 | Examples | Some examples of system software’s are compiler, assembler, debugger, driver, etc. | On other hand some examples of application software’s are word processor, web browser, media player, etc. |

# Q3. Describe waterfall model.

**ANS**

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.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

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.

Waterfall Model - Disadvantages

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows −

* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
* It is difficult to measure progress within stages.
* Cannot accommodate changing requirements.
* Adjusting scope during the life cycle can end a project.
* Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

# Define Software. Also write its characteristics. 4M

Computer software is the product that software professionals build and then support over the long term. It encompasses programs that execute within a computer of any size and architecture, content that is presented as the computer programs execute, and descriptive information in both hard copy and virtual forms that encompass virtually any electronic media.

(OR)

Software is: (1) instructions (computer programs) that when executed provide desired features, function, and performance; (2) data structures that enable the programs to adequately manipulate information, and (3) descriptive information in both hard copy and virtual forms that describes the operation and use of the programs.

# Characteristics:

* 1. Software is developed or engineered; it is not manufactured in the classical sense.
  2. Software doesn’t “wear out.”
  3. Although the industry is moving toward component-based construction, most software continues to be custom built.

# What are the drawbacks of spiral model ? 6M Ans. The drawbacks of spiral model are :

* 1. It is based on customer communication. If the communication is not proper then the software product that gets developed will not be the up to the mark.
  2. It demands considerable risk assessment. If the risk assessment is done properly then only the successful product can be obtained.

# 2a.Explain whether it is possible to combine two software models? If so provide an example?4M

**A)** Yes you can combine different process models example could be Spiral model it consist of incremental and waterfall model used hand to hand also waterfall model is implemented iteratively.

Yes it is possible to combine process models. It is the spiral process model. This model of development combines the features of the prototyping model and the waterfall model. Let us consider the case study of the baggage-handling system at the Denver International Airport (DIA) again. Initially, DIA had intended that each individual airline would be responsible for building its own baggage-handling system. However, as the construction of the airport progressed, a larger vision emerged for the inclusion of an airport-wide integrated baggage-handling system that could provide a major improvement in the efficiency of luggage delivery and no one gave any thought to risk assessment. The essential concept of the Spiral Model is to minimize risks by the repeated use of prototypes and other means.

# 2b) umbrella activities occur throughout the software process. Do you think they are applied? 6M

1. **Umbrella Activities**
   * Software project management
   * Formal technical reviews
   * Software quality assurance
   * Software configuration management
   * Work product preparation and production
   * Reusability management
   * Measurement
   * Risk management

## ****Yes! In major cases, the umbrella activities are applied evenly across the process.****

**Mostly, when dealing with the control progress, quality checks, changes made and, in going through the risk factors of a software, the umbrella activities are spread evenly throughout the process, as these contents are needed to be check at every step of the development stage.**

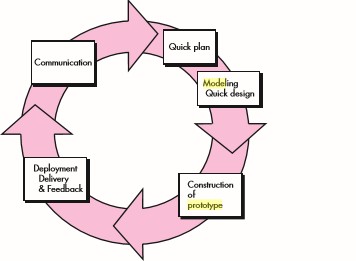
# 3a) What process adaptations are required if the prototype will evolve into a deliverable? 5M

**A)** There are three process adaptations that are required if the prototype will evolve into a deliverable system or product. The processes are more rigorous design rules and SQA procedures must be applied from the beginning, the prototype must be designed with extensibility in mind, and then it becomes the framework for extensions that will cause it to evolve into a production system.

# 3b) Illustrate the prototype paradigm in process models? 5M

**A)** Often, a customer defines a set of general objectives for software, but does not identify detailed requirements for functions and features. A prototyping paradigm may

offer the best approach. The prototyping paradigm assists you and other stakeholders to better understand what is to be built when requirements are fuzzy. The prototyping paradigm begins with communication. A prototyping iteration is planned quickly, and modeling (in the form of a “quick design”) occurs. A quick design focuses on a representation of those aspects of the software that will be visible to end users (e.g., human interface layout or output display



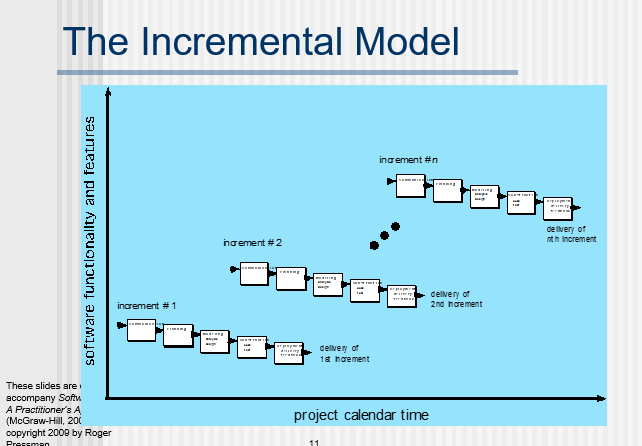
# 4a) Distinguish between personal software process model and Team software process model? 5M

A)The personal software process is a disciplined data driven software development process that is designed to help software engineers understand and to improve their personal software process performance. TSP helps engineers to improve their estimation and planning skills and to reduce the number of defects in their work. This enables them to make commitments that they can keep and to manage the quality of their projects.

# 4b) Explain why incremental development is the most effective approach for developing business software systems? Why is this model less approach for real- time systems engineering? 5M

A)Business software systems usually complex, software intensive, and frequently being changes when business goals or processes are changed. So incremental development is better.

Real-time systems usually involve many hardware components which are not easy to change and cannot be incremental. Also real-time systems usually safety critical which needed be built based on well planned process.



# 5a) Examine how agility principles are modeled successfully in software industry? 5M

1. The principles underlying agile development are:
   1. Individual and interactions over processes and tools. By taking advantages of individual skills and ability and by ensuring that the development team knows what each other are doing, the overheads of formal communication and process assurance are avoided. This means that the team can focus on the development of working software.
   2. Working software over comprehensive documentation. This contributes to accelerated development because time is not spent developing, checking and managing documentation. Rather, the programmer’s time is focused on the development and testing of code.
   3. Customer collaboration over contract negotiation. Rather than spending time developing, analyzing and negotiating requirements to be included in a system contract, agile developers argue that it is more effective to get feedback from customer’s directly during the development about what is required. This allows useful functionality to be developed and delivered earlier than would be possible if contracts were required.
   4. Responding to change over following a plan. Agile developers argue (rightly) that being responsive to change is more effective than following a plan-based process because change is inevitable whatever process is used. There is significant overhead in changing plans to accommodate change and the inflexibility of a plan means that work may be done that is later discarded.

# b)Explain how the Formal method models will yield a detect free software?5M

The formal methods model offers the promise of defect-free software. Yet, concern about its applicability in a business environment has been voiced:

* The development of formal models is currently quite time consuming and expensive.
* Because few software developers have the necessary background to apply formal methods, extensive training is required. It is difficult to use the models as a communication mechanism for technically unsophisticated customers.The development of a formal specification provides insights and an understanding of the software requirements and software design
  + Clarify customers’ requirements
  + Reveal and remove ambiguity, inconsistency and incompleteness
  + Facilitate communication of requirement or design
  + Provides a basis for an elegant software design

# 6 a). What is the IEEE definition of software engineering? Compare IEEE definition with seminal definition? 4M

1. The IEEE definition:
   * *Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in (1).*

The seminal definition:

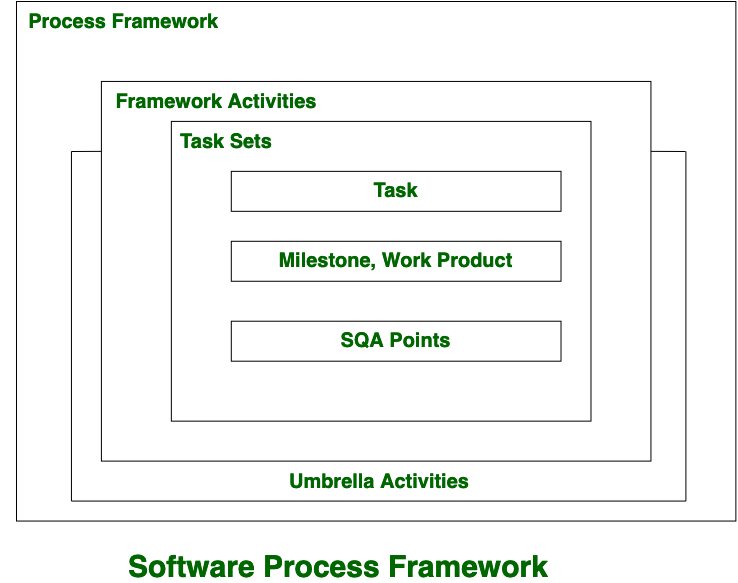
* + *[Software engineering is] the establishment and use of sound*
  + *engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.*

# Explain the role of risk analysis in an evolutionary process model like spiral model? 6M

The spiral model is a realistic approach to the development of large-scale systems and software. Because software evolves as the process progresses, the developer and customer better understand and react to risks at each evolutionary level. The spiral model uses prototyping as a risk reduction mechanism but, more important, enables you to apply the prototyping approach at any stage in the evolution of the product. It maintains the systematic stepwise approach suggested by the classic life cycle but incorporates it into an iterative framework that more realistically reflects the real world. The spiral model demands a direct consideration of technical risks at all stages of the project and, if properly applied, should reduce risks before they become problematic. But like other paradigms, the spiral model is not a panacea. It may be difficult to convince customers (particularly in contract situations) that the evolutionary approach is controllable. It demands considerable risk assessment expertise and relies on this expertise for success. If a major risk is not uncovered and managed, problems will undoubtedly occur.

# Illustrate various layers of Software Process Framework in software engineering

**Framework** is a Standard way to build and deploy applications. **Software Process Framework** is a foundation of complete software engineering process. Software process framework includes all set of umbrella activities. It also includes number of framework activities that are applicable to all software projects.



A generic process framework encompasses five activities which are given below one by one:

1. **Communication:**  
   In this activity, heavy communication with customers and other stakeholders, requirement gathering is done.
2. **Planning:**  
   In this activity, we discuss the technical related tasks, work schedule, risks, required resources etc.
3. **Modeling:**  
   Modelling is about building representations of things in the ‘real world’.In modelling activity, a product’s model is created in order to better understanding and requirements.
4. **Construction:**  
   In software engineering, construction is the application of set of procedures that are needed to assemble the product. In this activity, we generate the code and test the product in order to make better product.
5. **Deployment:**  
   In this activity, a complete or non-complete products or software are represented to the customers to evaluate and give feedback. on the basis of their feedback we modify the products for supply better product.

**Umbrella activities include:**

* Risk management
* Software quality assurance(SQA)
* Software configuration management(SCM)
* Measurement
* Formal technical reviws(FTR)

# Software changes are common after the first version have been put into use. Suggest a few ways to build software to stop deterioration due to change.

It is fact that several modern applications and the software’s have gained the capacity to eliminate the data or content before the presentation of the application in front of the end user. This is considered under the deterioration due to the changes of application information. the following are some ways which can stop the deterioration:

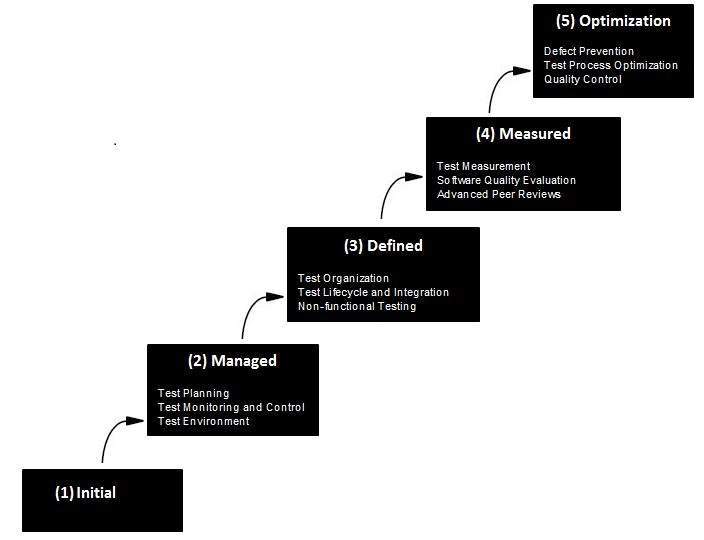
The applications of software must be maintainable. These applications must have capacity engineered to a degree changes as this can be build easily when the application is in growth stages.

The chances of deterioration can be minimal through allow automatic updates to be built in

# Explain Capability Maturity model

The Software Engineering Institute (SEI) Capability Maturity Model (CMM) specifies an increasing series of levels of a software development organization. The higher the level, the better the software development process, hence reaching each level is an expensive and time-consuming process.

Levels of CMM



* **Level One :Initial** - The software process is characterized as inconsistent, and occasionally even chaotic. Defined processes and standard practices that exist are abandoned during a crisis. Success of the organization majorly depends on an individual effort, talent, and heroics. The heroes eventually move on to other organizations taking their wealth of knowledge or lessons learnt with them.
* **Level Two: Repeatable** - This level of Software Development Organization has a basic and consistent project management processes to track cost, schedule, and functionality. The process is in place to repeat the earlier successes on projects with similar applications. Program management is a key characteristic of a level two organization.
* **Level Three: Defined** - The software process for both management and engineering activities are documented, standardized, and integrated into a standard software process for the entire organization and all projects across the organization use an approved, tailored version of the organization's standard software process for developing,testing and maintaining the application.
* **Level Four: Managed** - Management can effectively control the software development effort using precise measurements. At this level, organization set a quantitative quality goal for both software process and software maintenance. At this maturity level, the performance of processes is controlled using statistical and other quantitative techniques, and is quantitatively predictable.
* **Level Five: Optimizing** - The Key characteristic of this level is focusing on continually improving process performance through both incremental and innovative technological improvements. At this level, changes to the process are to improve the process performance and at the same time maintaining statistical probability to achieve the established quantitative process-improvement objectives.

# List the merits of using incremental model in Software Development Process.

**Advantages of Incremental model:**

* Generates working software quickly and early during the software life cycle.
* This model is more flexible – less costly to change scope and requirements.
* It is easier to test and debug during a smaller iteration.
* In this model customer can respond to each built.
* Lowers initial delivery cost.
* Easier to manage risk because risky pieces are identified and handled during it’d iteration.

# Is software engineering applicable when WebApps are built? If so, how might it be modified to accommodate the unique characteristics of WebApps

Yes, software engineering is applicable, when WebApps are built because it is a layered technology and consists of Tools, Methods, Process, and A quality focus.

WebApps exhibit some unique attributes like, Network intensiveness, concurrency, availability, security etc.

These unique characteristics of WebApps can be accommodated through a generic process framework for software engineering. It encompasses five activities and they are modified to accommodate the WebApps.

Modified process framework activities for WebApps are:

1. Customer communication (analysis/formulation) - Setting the goals, objectives and scope of the first increment.

2. Planning - fine grained estimates and schedule for the first increment, coarser estimates for subsequent increments.

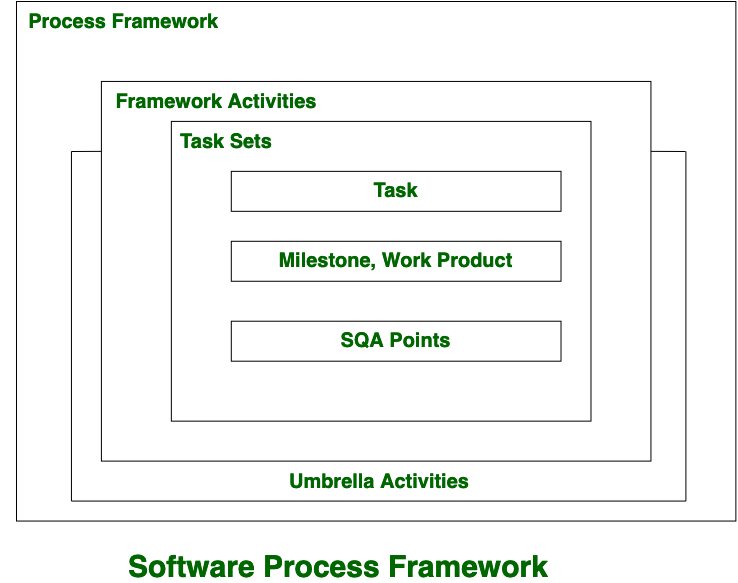
3. Modeling - Establishes requirements and identifies content items, content design, architectural design, navigational design, and interface design.

4. Construction - page generation and testing, merging of content and technical designs to produce executable web pages that are exercised to uncover errors.

5. Deliver and evaluation - Increment is reviewed and changes required by customer are integrated into the next increment.

# Describe how the framework activities, actions and tasks that occur within each activity are organized

**Framework** is a Standard way to build and deploy applications. **Software Process Framework** is a foundation of complete software engineering process. Software process framework includes all set of umbrella activities. It also includes number of framework activities that are applicable to all software projects.



A generic process framework encompasses five activities which are given below one by one:

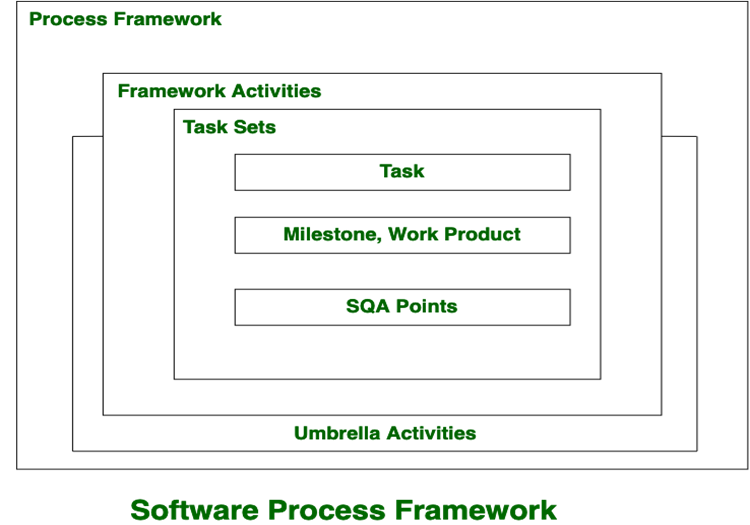
1. **Communication:**  
   In this activity, heavy communication with customers and other stakeholders, requirement gathering is done.
2. **Planning:**  
   In this activity, we discuss the technical related tasks, work schedule, risks, required resources etc.
3. **Modeling:**  
   Modelling is about building representations of things in the ‘real world’.In modelling activity, a product’s model is created in order to better understanding and requirements.
4. **Construction:**  
   In software engineering, construction is the application of set of procedures that are needed to assemble the product. In this activity, we generate the code and test the product in order to make better product.
5. **Deployment:**  
   In this activity, a complete or non-complete products or software are represented to the customers to evaluate and give feedback. on the basis of their feedback we modify the products for supply better product.

**Umbrella activities include:**

* Risk management
* Software quality assurance(SQA)
* Software configuration management(SCM)
* Measurement
* Formal technical reviws(FTR)

# Describe a process framework in your own words. When we say that framework activities are applicable to all projects, does this mean that the same work tasks are applied for all projects, regardless of size and complexity? Explain.

**Framework** is a Standard way to build and deploy applications. **Software Process Framework** is a foundation of complete software engineering process. Software process framework includes all set of umbrella activities. It also includes number of framework activities that are applicable to all software projects.



A generic process framework encompasses five activities which are given below one by one:

1.     **Communication:**  
In this activity, heavy communication with customers and other stakeholders, requirement gathering is done.

2.     **Planning:**  
In this activity, we discuss the technical related tasks, work schedule, risks, required resources etc.

3.     **Modeling:**  
Modelling is about building representations of things in the ‘real world’.In modelling activity, a product’s model is created in order to better understanding and requirements.

4.     **Construction:**  
In software engineering, construction is the application of set of procedures that are needed to assemble the product. In this activity, we generate the code and test the product in order to make better product.

5.     **Deployment:**  
In this activity, a complete or non-complete products or software are represented to the customers to evaluate and give feedback. on the basis of their feedback we modify the products for supply better product.

**Umbrella activities include:**

·        Risk management

·        Software quality assurance(SQA)

·        Software configuration management(SCM)

·        Measurement

·        Formal technical reviws(FTR)

# Distinguish between desktop and web applications

## Web Applications

* Deployment & up-gradation for web based application requires deployment on single set of server machines.
* Web application can be accessible from anywhere through internet, there is no restriction on location.
* Web applications are platform independent, they can work on multiple types of platforms with the requirement of web browser.
* Web applications are at higher security risk as they are designed to increase accessibility.
* Web application heavily on internet connectivity, for there operation.

## Desktop Applications

* Deployment & up-gradation is done on individual client machines.
* They can be only accessible on the machines which they are deployed
* Need to be developed separately for different platform machines
* Desktop applications have better authorization & administrators for better control.
* Desktop applications doesn’t require any internet connection for there operations (Some applications only require internet connectivity at the time of updations).

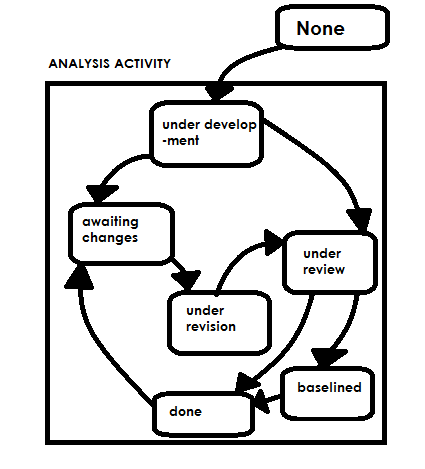
# Explain concurrent process models

Concurrent Process model is an evolutionary process model in software engineering. The term concurrent mean “done at the same time”.

If we take waterfall model as an example, you will not know the activities going on in each phase, only after the phase is over, you get a work product or a document.

Suppose we want to know the state of activities of the design phase and at the same time we want to know about testing phase activities. Which ones are complete and which are not?

The concurrent process model shows the current state of activities, tasks and their associated states that remain in different phases.

Concurrent Process Model

For example,

‘Design Phase’ may be at an ‘awaiting state’ and ‘customer communication’ is in ‘under revision’ state. The customer wants some changes to the design, then ‘communication’ goes to ‘awaiting changes’ and ‘design’ goes to the under-development stage again.

The benefit of this model is that project managers know each phase is what state and why.

**Main Point** –  Maintain information about each phase at the activity level.

# Distinguish between Personal software Process (PSP) model and Team software Process (TSP) Model.

**Personal software process :-**

* It is a systematic application development method intended to help engineers understand and enhance their output by applying professionalism to the way they build software and monitoring their expected and actual code creation.
* It indicates developers how to control the value of their assets, what to draw up a sound plan as well as how to make promises. This also provides them the evidence to explain their proposals.
* The personal software method focuses on entities to enhance their results. It consists of methods, types and techniques that orient programmers in their technical work.

**Team software process :-**

* The goal of the TSP is to enhance the quality and efficiency of the entire team application development project, in addition to helping us help meet the expense and timeline obligations of creating software.
* TSP is designed for use in education environments, concentrating on the process of creating a project management team, creating team goals, assigning team tasks, and several other collaboration activities.
* Team software relies on a community of people and seeks to improve team performance.

# Differentiate between software engineering and system engineering

* Software engineering highly focuses on implementing quality software while system engineers highly concern about the users and domains.
* Software engineering includes in computer science or computer based engineering background while system engineering may covers a broader education area includes Engineering, Mathematics and Computer science.).
* Software engineers focus solely on software components while system engineering deals with a substantial amount of physical component of computers.
* Software Engineering deals with designing and developing software of the highest quality, while Systems Engineering is the sub discipline of engineering, which deals with the overall management of engineering projects during their life cycle.
* Software engineering techniques such as use-case modeling and configuration management are being used in the systems engineering process.

# Provide three examples of software projects that would be amenable to the waterfall model.

Waterfall model is a sequential approach to software development that begins with customer specification requirements and progresses through planning, modeling, construction and deployment

The waterfall model is appropriate for projects with the following characteristics:

(1) The problem is well understood (requirements are well-defined);

(2) The delivery date is realistic;

(3) it's unlikely that major changes in requirements will be requested as the project proceeds

Communication  project initiation, requirements gathering

Planning  Estimating, scheduling, tracking

Modeling  Analysis and design

Construction  code and test

Deployment  delivery, support and feedback.

Example software project – (1) transaction Maintenance system (TMS)

(2) Us Department of defense

(3) NASA

(1) Transaction Maintain once system

Phase 1. Preliminary investigation: Aim of this phase is not develop system but to

investigate the problem.

Phase 2. Requirement analysis: This phase is concerned about the collection of

requirements. Output of this phase is SRS document.

Phase 3. System design: (1) Algorithm

(2) Data structure

(3) software architecture

(4) Interface design

Phase 4. coding

Phase 5. Integration & Testing

Phase 6. Implementation & maintain once.

In similar lines, us department of defense and NASA uses the waterfall model in an extensive way. As waterfall model is a sequential approach and many increments can be applied after the evolution of software in this model, it is well suited for large and long-term projects.

Along with these software projects, many government projects follow waterfall model.

# “Requirements change so much. After all, don’t people know what they want”- Justify

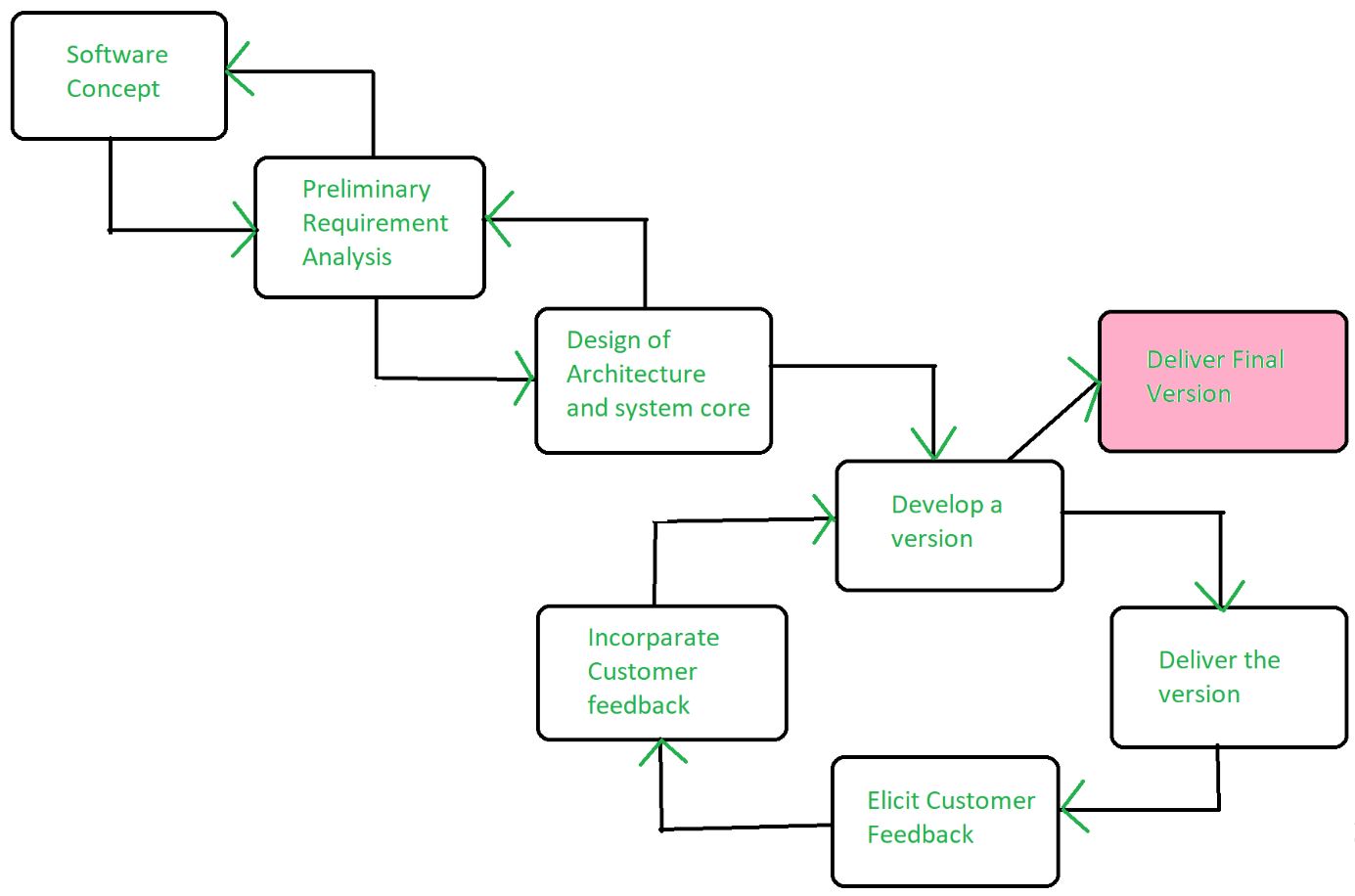
It is difficult to predict in advance which requirement will persist and which will change, It is difficult to predict how customer priorities will change as the project proceeds, and it is difficult for customers to verbalize their software needs until they see a working prototype.

# Summarize the evolutionary development model and its problems in detail

**Evolutionary model** is a combination of Iterative and Incremental model of software development life cycle. Delivering your system in a big bang release, delivering it in incremental process over time is the action done in this model. Some initial requirements and architecture envisioning need to be done.

It is better for software products that have their feature sets redefined during development because of user feedback and other factors. The Evolutionary development model divides the development cycle into smaller, incremental waterfall models in which users are able to get access to the product at the end of each cycle.

Feedback is provided by the users on the product for the planning stage of the next cycle and the development team responds, often by changing the product, plan or process. Therefore, the software product evolves with time.  
All the models have the disadvantage that the duration of time from start of the project to the delivery time of a solution is very high. Evolutionary model solves this problem in a different approach.



Evolutionary model suggests breaking down of work into smaller chunks, prioritizing them and then delivering those chunks to the customer one by one. The number of chunks is huge and is the number of deliveries made to the customer. The main advantage is that the customer’s confidence increases as he constantly gets quantifiable goods or services from the beginning of the project to verify and validate his requirements. The model allows for changing requirements as well as all work in broken down into maintainable work chunks.

**Application of Evolutionary Model:**

1. It is used in large projects where you can easily find modules for incremental implementation. Evolutionary model is commonly used when the customer wants to start using the core features instead of waiting for the full software.
2. Evolutionary model is also used in object oriented software development because the system can be easily portioned into units in terms of objects.

**Advantages:**

* In evolutionary model, a user gets a chance to experiment partially developed system.
* It reduces the error because the core modules get tested thoroughly.

**Disadvantages:**

* Sometimes it is hard to divide the problem into several versions that would be acceptable to the customer which can be incrementally implemented and delivered.

# Provide three examples of software projects that would be amenable to the prototyping model. Specify

Software applications that are relatively easy to prototype almost always involve human-‐machine interaction and/or heavy computer graphics. Other applications that are sometimes amenable to prototyping are certain classes of mathematical algorithms, subset of command driven systems and other applications where results can be easily examined without real-‐time interaction. Applications that are difficult to prototype include control and process control functions, many classes of real-‐time applications and embedded software.

# State Hooker’s Principles

**Hooker's General SE Principles**

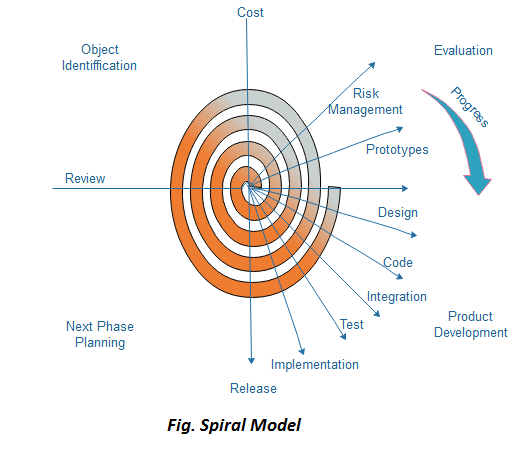
* The Reason It All Exists.
* KISS (Keep It Simple, Stupid!)
* Maintain the Vision.
* What You Produce, Others Will Consume.
* Be Open to the Future.
* Plan Ahead for Reuse.
* Think!

# Explain Boehm’s spiral model of software engineering process and its draw backs

## Spiral Model

The spiral model, initially proposed by Boehm, is an evolutionary software process model that couples the iterative feature of prototyping with the controlled and systematic aspects of the linear sequential model. It implements the potential for rapid development of new versions of the software. Using the spiral model, the software is developed in a series of incremental releases. During the early iterations, the additional release may be a paper model or prototype. During later iterations, more and more complete versions of the engineered system are produced.

**The Spiral Model is shown in fig:**



**Each cycle in the spiral is divided into four parts:**

**Objective setting:** Each cycle in the spiral starts with the identification of purpose for that cycle, the various alternatives that are possible for achieving the targets, and the constraints that exists.

**Risk Assessment and reduction:** The next phase in the cycle is to calculate these various alternatives based on the goals and constraints. The focus of evaluation in this stage is located on the risk perception for the project.

**Development and validation:** The next phase is to develop strategies that resolve uncertainties and risks. This process may include activities such as benchmarking, simulation, and prototyping.

**Planning:** Finally, the next step is planned. The project is reviewed, and a choice made whether to continue with a further period of the spiral. If it is determined to keep, plans are drawn up for the next step of the project.

The development phase depends on the remaining risks. For example, if performance or user-interface risks are treated more essential than the program development risks, the next phase may be an evolutionary development that includes developing a more detailed prototype for solving the risks.

The **risk-driven** feature of the spiral model allows it to accommodate any mixture of a specification-oriented, prototype-oriented, simulation-oriented, or another type of approach. An essential element of the model is that each period of the spiral is completed by a review that includes all the products developed during that cycle, including plans for the next cycle. The spiral model works for development as well as enhancement projects.

## When to use Spiral Model?

* When deliverance is required to be frequent.
* When the project is large
* When requirements are unclear and complex
* When changes may require at any time
* Large and high budget projects

## Advantages

* High amount of risk analysis
* Useful for large and mission-critical projects.

## Disadvantages

* Can be a costly model to use.
* Risk analysis needed highly particular expertise
* Doesn't work well for smaller projects.

# Demonstrate software development life cycle

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.



A typical Software Development Life Cycle consists of the following stages −

### Stage 1: Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

### Stage 2: Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an **SRS (Software Requirement Specification)** document which consists of all the product requirements to be designed and developed during the project life cycle.

### Stage 3: Designing the Product Architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

### Stage 4: Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

### Stage 5: Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

### Stage 6: Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

## SDLC Models

There are various software development life cycle models defined and designed which are followed during the software development process. These models are also referred as Software Development Process Models". Each process model follows a Series of steps unique to its type to ensure success in the process of software development.

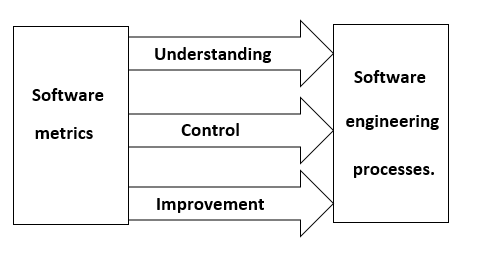
Following are the most important and popular SDLC models followed in the industry −

* Waterfall Model
* Iterative Model
* Spiral Model
* V-Model
* Big Bang Model

Other related methodologies are Agile Model, RAD Model, Rapid Application Development and Prototyping Models.

# Describe the types of measurements that an individual software engineer is asked to make and how those measurements can be used to improve personal effectiveness

Measurement are a key element for controlling software engineering processes. By controlling it meant that one access the status of the process, observe the trends to what is likely to happen and take corrective action for modifying our practices. Measurements improve our processes by modifying the activities based on different measures –



**Understanding –** metrics help in making the aspect of processes more visible, thereby giving better understanding of relationships among the activities and entities they **Control –** using baselines, goals we can predict what is likely to happen and make appropriate changes in process to help meet the goals. **Improvement –** By taking corrective actions and make appropriate changes, we can improve a product. Based on the analysis of a project, a process can be improved.

**Classification of software metrics.**

The software metrics can be classified as follows:

**1. Product vs. process metrics**. Software metrics may be broadly classified as either product metrics or process metrics. Product metrics are measures of the software product at any stage of its development, from requirements to installed system. Product metrics may measure the complexity of the software design, the size of the final program of the number of pages of documentation produced. Process metrics are measures of the software development process, such as the overall development time, type of methodology used, or average level of experience of programming staff.

**2. Objective vs. subjective metrics.** Objective measures should always result in identical values for a given metric, as measured by two or more qualified observers. For subjective measures, even qualified observers may measure different values for a given metric. For example, for product metrics, the size of product measured in line of code is an objective measure. In process metrics, the development time is an example of objective measure, where the level of a programmer’s experience is likely to be a subjective measure.

**3. Primitive vs. computed metrics.** Primitive metrics are those metrics that can be directly observed, such as the program size in LOC, the number of defects observed in unit testing, or the total development time for the project. Computed metrics are those that cannot be directly observed but are computed in some way from other metrics. For example, productivity metrics like LOC produced per person month or product quality like defects per thousand LOC.

**4. Private vs. public metrics.** This classification is based on the use of different types to process data. It is natural that individual software engineers might be sensitive to the use of metrics collet on an individual basis, and therefore these data should be private to individuals and serve as an indicator for individual only. Examples of private metrics include defect rates and errors found during development.

Public metrics assimilate information that originally was private to individual’s and teams. Project level defect rate effort, calendar times and related data are collected and evaluated in an attempt to uncover indicators that can improve organizational process performance.

# Umbrella activities occur throughout the software process. Do you think they are applied evenly across the process, or are some concentrated in one or more framework activities?

**Yes! In major cases, the umbrella activities are applied evenly across the process.**

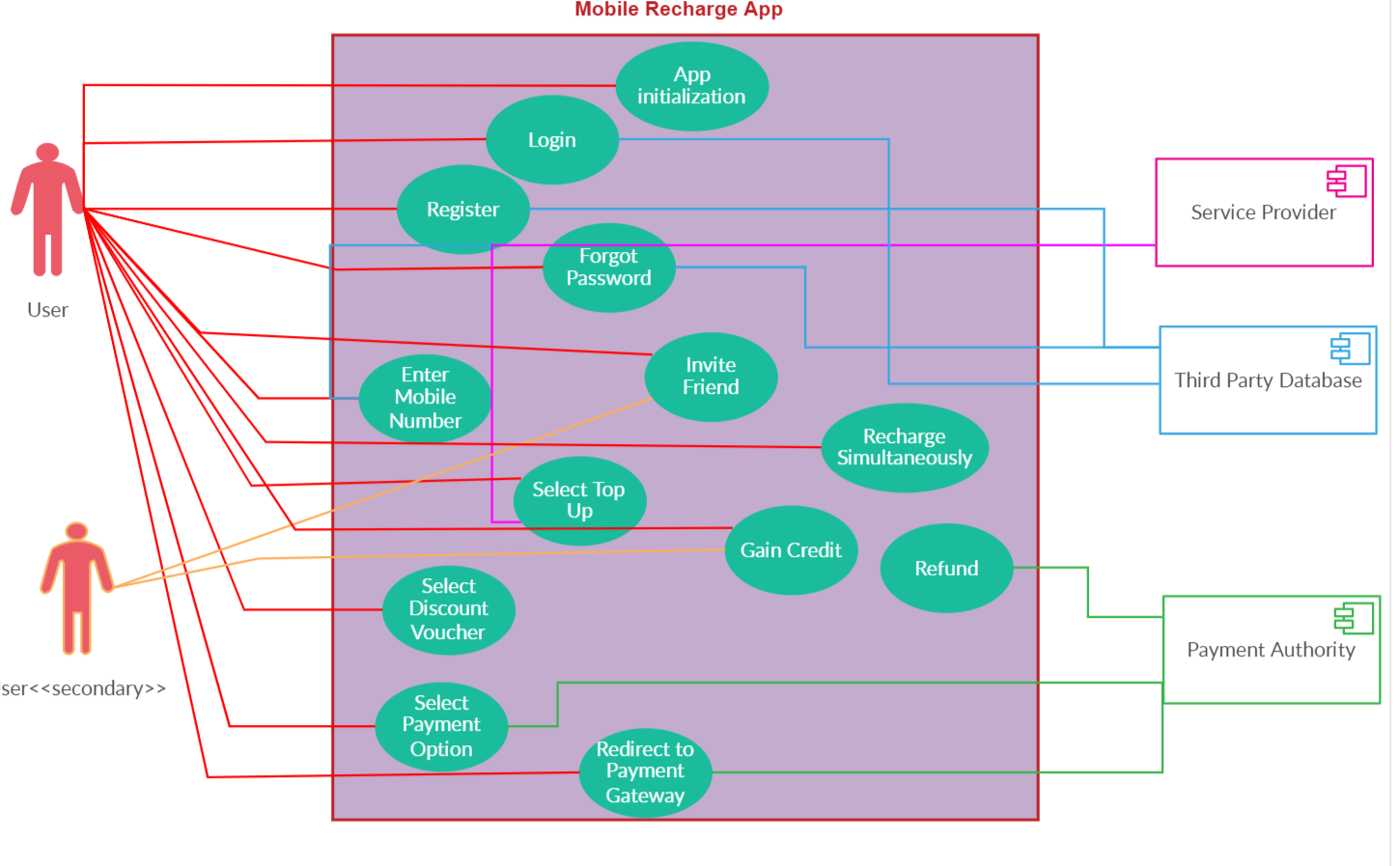
**Mostly, when dealing with the control progress, quality checks, changes made and, in going through the risk factors of a software, the umbrella activities are spread evenly throughout the process, as these contents are needed to be check at every step of the development stage.**

* Due to this, even the same processes of different projects too differ to each other on a slight edge.
* For this, the engineering mode on the software should be quick, optimized and adaptable. meeting all the needs of a problem given by the client.
* When the adoptions are done of a process, the activities may be concentrated on one or more frameworks too.

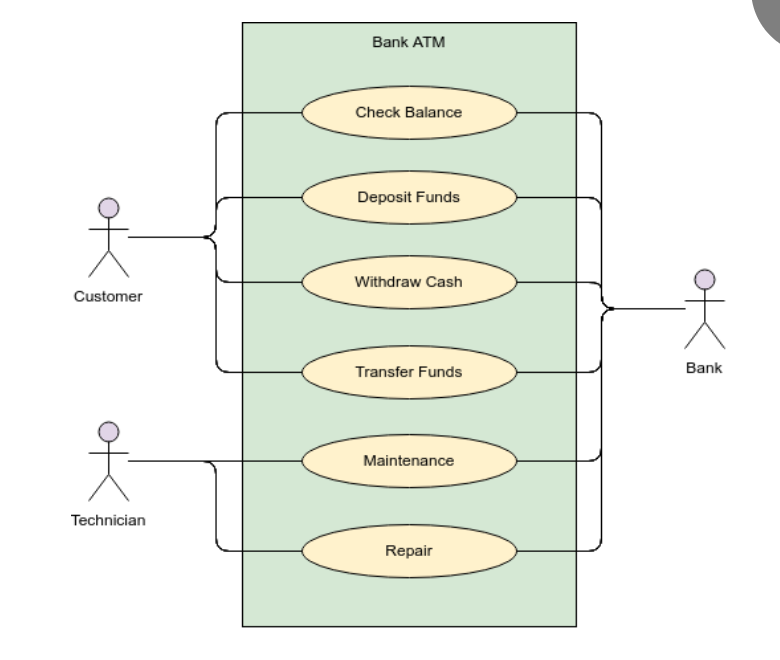
# Explain customer myths and their realities with a set of examples?

 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.

# Draw use case diagram for Online Mobile Recharge application and explain it



# Draw use case diagram for ATM application and explain it

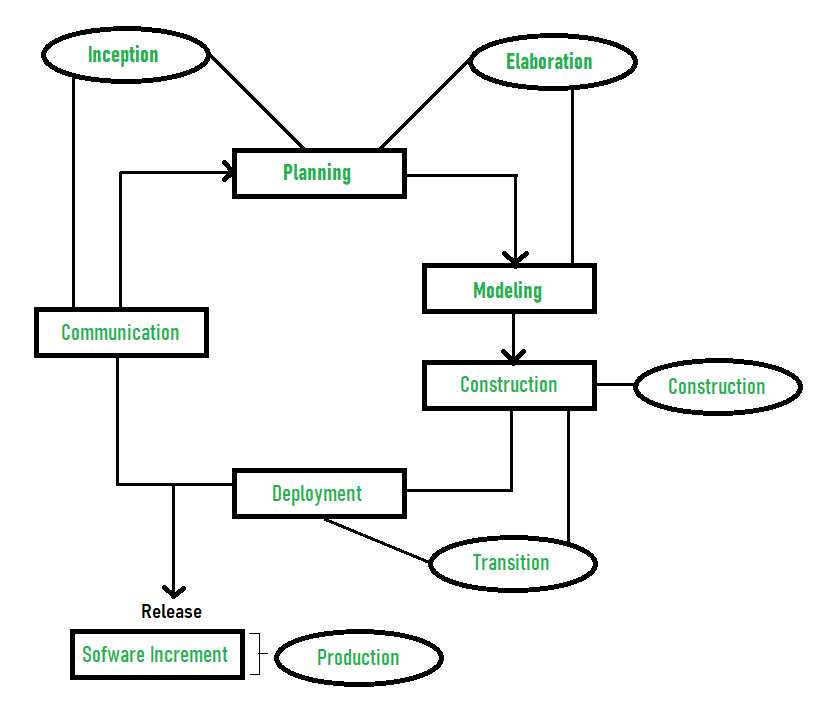


# Explain unified process model with a diagram

**Rational Unified Process (RUP)** is a software development process for object-oriented models. It is also known as the Unified Process Model. It is created by Rational corporation and is designed and documented using UML (Unified Modeling Language). This process is included in IBM Rational Method Composer (RMC) product. IBM (International Business Machine Corporation) allows us to customize, design, and personalize the unified process.

RUP is proposed by Ivar Jacobson, Grady Bootch, and James Rambaugh. Some characteristics of RUP include use-case driven, Iterative (repetition of the process), and Incremental (increase in value) by nature, delivered online using web technology, can be customized or tailored in modular and electronic form, etc. RUP reduces unexpected development costs and prevents wastage of resources.

**Phases of RUP :**  
There are total five phases of life cycle of RUP:



1. **Inception –**
   * Communication and planning are main.
   * Identifies Scope of the project using use-case model allowing managers to estimate costs and time required.
   * Customers requirements are identified and then it becomes easy to make a plan of the project.
   * Project plan, Project goal, risks, use-case model, Project description, are made.
   * Project is checked against the milestone criteria and if it couldn’t pass these criteria then project can be either cancelled or redesigned.
2. **Elaboration –**
   * Planning and modeling are main.
   * Detailed evaluation, development plan is carried out and diminish the risks.
   * Revise or redefine use-case model (approx. 80%), business case, risks.
   * Again, checked against milestone criteria and if it couldn’t pass these criteria then again project can be cancelled or redesigned.
   * Executable architecture baseline.
3. **Construction –**
   * Project is developed and completed.
   * System or source code is created and then testing is done.
   * Coding takes place.
4. **Transition –**
   * Final project is released to public.
   * Transit the project from development into production.
   * Update project documentation.
   * Beta testing is conducted.
   * Defects are removed from project based on feedback from public.
5. **Production –**
   * Final phase of the model.
   * Project is maintained and updated accordingly.

# Explain Practitioner myths.

**Developer Myths**. Developers often want to be artists (or artisans), but the software development craft is becoming an engineering discipline. However myths remain:

* *The job is done when the code is delivered.*

Commercially successful software may be used for decades. Developers must continually maintain such software: they add features and repair bugs. Maintenance costs predominate over all other costs; maintenance may be 70% of the development costs. This myth is true only for *shelfware* --- software that is never used, and there are no customers for next release of a shelfware product.

* *Project success depends solely on the quality of the delivered****program***.

Documentation and software configuration information is very important to the quality. After functionality, maintainability, see the preceding myth, is of critical importance. Developers must maintain the software and they need good design documents, test data, etc to do their job.

* *You can't assess software quality until the program is running.*

There are *static* ways to evaluate quality without running a program. Software reviews can effectively determine the quality of requirements documents, design documents, test plans, and code. Formal (mathematical) analyses are often used to verify safety critical software, software security factors, and very-high reliability software.

# Outline Management myths in software

***1) 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.

# ”Prototype model is important in software development”- Justify

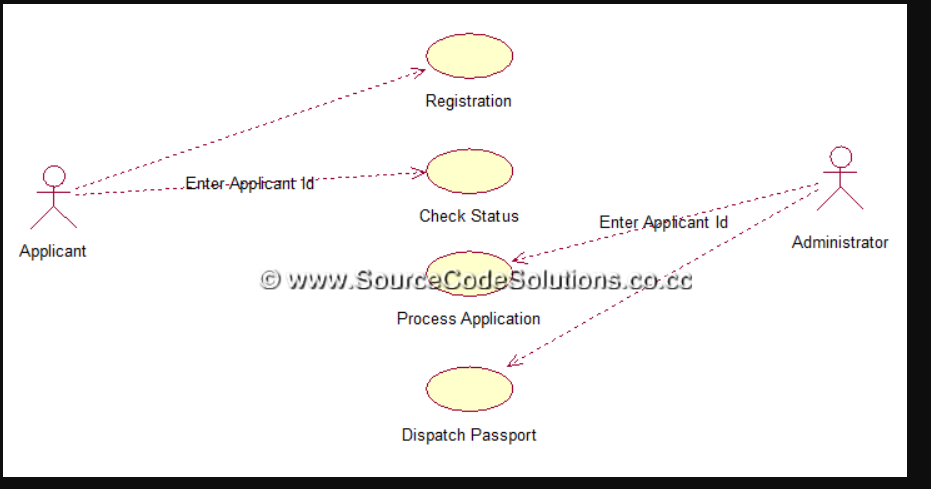
**Prototyping Model** is a software development model in which prototype is built, tested, and reworked until an acceptable prototype is achieved. It also creates base to produce the final system or software. It works best in scenarios where the project's requirements are not known in detail. It is an iterative, trial and error method which takes place between developer and client.

**Advantages of the Prototyping Model**

Here, are important pros/benefits of using Prototyping models:

* Users are actively involved in development. Therefore, errors can be detected in the initial stage of the software development process.
* Missing functionality can be identified, which helps to reduce the risk of failure as Prototyping is also considered as a risk reduction activity.
* Helps team member to communicate effectively
* Customer satisfaction exists because the customer can feel the product at a very early stage.
* There will be hardly any chance of software rejection.
* Quicker user feedback helps you to achieve better software development solutions.
* Allows the client to compare if the software code matches the software specification.
* It helps you to find out the missing functionality in the system.
* It also identifies the complex or difficult functions.
* Encourages innovation and flexible designing.
* It is a straightforward model, so it is easy to understand.
* No need for specialized experts to build the model
* The prototype serves as a basis for deriving a system specification.
* The prototype helps to gain a better understanding of the customer's needs.
* Prototypes can be changed and even discarded.
* A prototype also serves as the basis for operational specifications.
* Prototypes may offer early training for future users of the software system.

# Draw Use case diagram for passport automation application and explain it.



Use Case diagrams identify the functionality provided by the system (use cases), the users who interact with the system (actors), and the association between the users and the functionality. Use Cases are used in the Analysis phase of software development to articulate the high-level requirements of the system. The primary goals of Use Case diagrams include:

* Providing a high-level view of what the system does.
* Identifying the users ("actors") of the system.
* Determining areas needing human-computer interfaces.

**System Functions**  
• Secure Registration of information by the Applicants.  
• Schedule the applicants an appointment for manual verification of original documents.  
• Panel for Passport Application Status Display by the Administrator.  
• SMS and Mail updates to the applicants by the administrator.  
• Administrator can generate reports from the information and is the only authorized personnel to add the eligible application information to the database.

# Differentiate between analysis and design in SDLC

Below is the topmost comparison between System Analysis vs Design

|  |  |
| --- | --- |
| **System Analysis** | **System design** |
| This is a post process which is used when the design is complete and ready for analysis. | This process starts from scratch and utilizes all your new ideas. |
| The analysis includes processing, execution, bug fixing, and making reports | The design includes the collection of raw data, requirement, need, and planning. |
| Basically, it resembles removing the error from the existing model. | Errors may come while creating but couldn’t fix at the start. |
| The time consumed in analyzing is less than the time consumed by design | In design, the maximum time goes before the initiation of work. |
| The software’s used is different than the tool used for designing. | The tool used is different as the work performed by them is different. |
| The system required for analysis should have a higher configuration. | Design PC is even good enough but the less configuration would sustain for smaller projects and even can do for big if done in parts. |
| Multiple screens are required for analysis to check and fix the bug. | The single but large screen is more than enough in this case as just detailing is the main job. |
| Pre-planning is not a compulsion but yes discussion on results does matter. | Pre-planning is the most important task and discussion about pre-planning is done. |
| To become a system analyzer you need to have post graduation in your studies then you can make growth in your career. | Design member needs to have a diploma or graduation degree with him to ensure his growth in the design field. |
| The analysis shows the depth of study you have in a particular field. | In design, your overall knowledge with concepts is required with your existing education. |
| The analysis makes you answerable for that project as you have a habit to clear errors from it. | Design can make you answerable only in part of the structure and dimensional parameters. |

# Why does an iterative process make it easier to manage change?

The software team manages change by focusing on a defined increment and postponing any changes until the next increment. All agile process models are iterative/incremental.

An iterative process make it easier to manage changes. Since each iteration is a mini – project, the project team addresses, to some extent, all the risks associated with the project as a whole each time it builds an increment of the system. As risks become greater, as delays occur, and as the environment become more unstable. The team is able to make necessary adjustments on a relatively small scale and propagate those adjustments across the entire project. An iterative process has greater flexibility to change the plan. Hence it is easier to manage the changes.

# Explain how does a software engineer proceed after gathering requirements for software development

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