UNIT-1

INTRODUCTION TO SOFTWARE ENGINEERING

Computer Software : is the product that software professionals build & then support over the long term.

- Software products could be of two types
  - Generic products
  - Customised products

- Generic products : are produced by a development organisation & sold on open markets to any customer who is able to buy them.
  Ex: softwares for PC's such as databases, word processors and project management tools.
- Customized products : are developed by software contractors for the customers according to the customer requirements.
  Ex: Control systems for electronic devices, systems written to support a particular business process & air traffic control system

SOFTWARE ENGINEERING :

Def:- The establishment & use of sound (strong or best) engineering principles in order to obtain economically software that is reliable and works efficiently on real machines

IEEE defination :

The application of systematic, disciplined, quantifiable approach to the development, operation and maintence of S/W

- Software Engineering is an engineering discipline which is concerned with all aspects of software production.
- Software : it is a set of instructions or programs, instructing a computer to do a specific task.
  - Software is a generic term used to describe computer programs
  - scripts, applications, programs are all terms often used to describe software.

The Evolving Role of a software:-

- Today, software takes a dual role, It is being used as a product and also as a vehicle for delivering the product.
- As a product it will be used in various applications.
- As a vehicle for delivering the product, software acts as the basis for the control of the computer.

Example: Operating systems, Networking softwares for Communication software tools, and environment.

Software delivers the most important product of our time information. It transforms personal data.

Example: An individual's financial transactions. It provides a gateway to world wide information networks Ex: (Internet) to access information.

- The software has seen many changes since its inception.
- Computer industry has been delivered exponential improvement in price, performance, but the problems with software have not been decreasing.

As per the IBM report 31% projects cancelled before they are completed. 53% over-run their cost estimations. for every 100 projects 94 restarts.

Some Software failures:-

A major problem of software industry is its inability to develop a bug-free software. If software developers are asked to certify that the developed software is a bug-free s/w, no software would have been developed till now.

Some of the software failures that are occured in the recent and past are:

Software failures:

① The Y2K problem: It was simply the ignorance of first two digits. The four digit date format, like 1964 was shorted to two digit format, like 64. The developers couldnot visualize the problem of 2000. Later millions of rupees have been spent to handle that problem.

② American "Patriot Missile": were using as a defence for Iraqui missiles. The patriot missiles failed several times to hit Iraqui missiles and hit soldiers and 28 U.S. soldiers were killed. The problems was software bug in the guidance system of the missiles.

③ Arine-5 space rocket was destroyed after 39 sec. of its launch. The reason was an overflow error while converting the velocity of the rocket from 64-bit format to 16 bit format.

④ Many companies have experienced failures in their accounting system due to the faults in software itself. The failures ranged from producing wrong information to the complete system crashing.

- Every other Engineer feels that things are improving day by day and they are sure that they are developing good quality products day by day.
- But many software engineers believe that software quality is not improving and it is getting worse.

Software: programs + data structures + documents

Software is a set of programs when executed provides desired features, functions and performance.

- Data structures that are used to write the programs.
- Documents that describe the operation and use of programs.

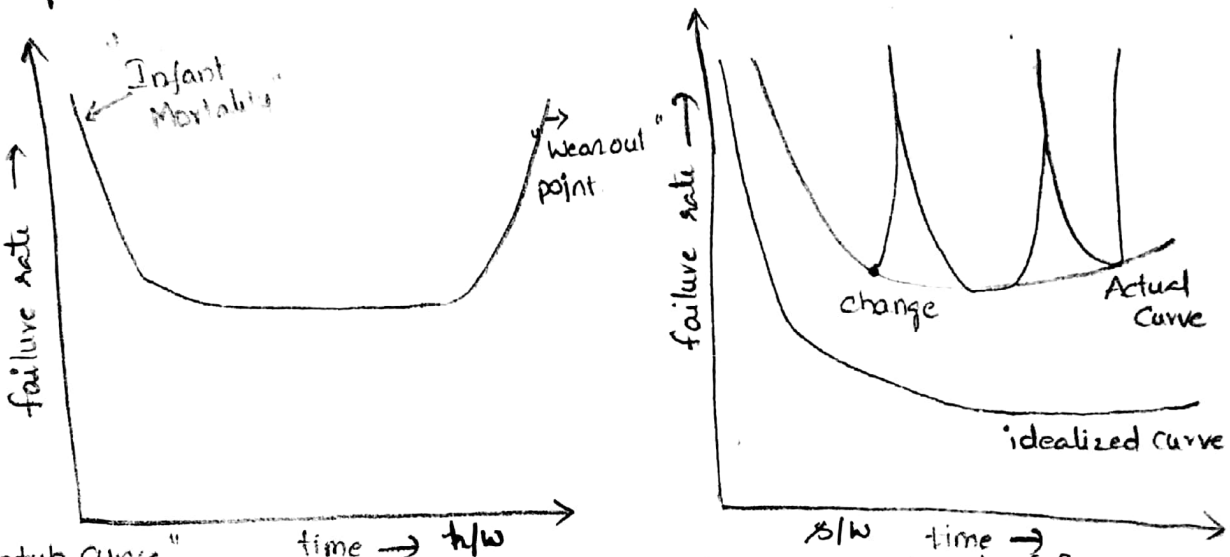
Software characteristics :-

. Software is a logical thing rather than a physical element.

(i) "Software is developed (or) Engineered; it is not manufactured in the classical sence."

– Some similarities may exist in development of software and manufacturing of hardware but both are fundamentally different. In both activities, high quality is achieved through good design, but manufacturing phase can introduce quality problems in hardware which are modifications are not exist for software. Both dependent on people, but the relationship between people applied and work acomplished is different. Both require construction of a "product" but approaches are different.

(ii) Software does not "wear out".



– Bathtub curve"

– If we take failure curve for hardware, at the begining stage failure rate will be very high due to manufacturing defects and those will be corrected and follow a steady state for a certain period of time and starts wear out later due to some environmental effects such as dust, temperature, vibrations etc... But s/w is a logical thing so it should be like the idealized curve. but in s/w we will keep on making changes which will cause again errors. and so on.

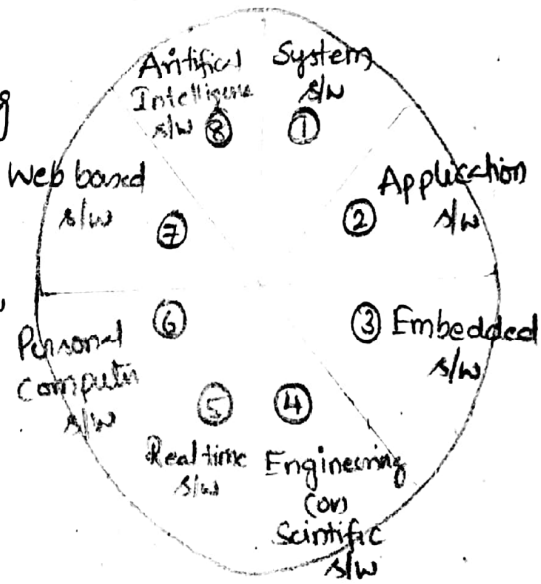
(iii) Although ... Construction ...

(iii) Although industry is moving towards component-based construction, software continues to be custom built.
- We get hardware product by assembling the hardware components but every software project must be custom built.

The Changing Nature of software: [Applications of software]

① System software: software that is developed to service other programs (s/w) Example: operating systems, compilers, editors, drivers, networking software etc... some system software processes complex but determinate data (compilers, editors). some system applications process largely indeterminate data (os, networking s/w)

② Application software: This software consists of standalone programs designed to process bussiness applications. [bussiness software]. It could be pay roll, file monitoring system, employee management, account management etc... Management information system, enterprise resource planning and such other software are some examples of bussiness s/w.



③ Embedded software: It resides within a product (or) system and control various functions of the product. It handles hardware components and also called as Intelligent s/w Embedded s/w is limited and will be designed for a particular device (or) product.
Ex: Software for automobiles, aircraft signalling system, lift system. Control unit of power generation plants etc...

* Softwar

③ Engineering or Scientific software: This software is characterised by the engineering scientific applications rang from astronomy to volcanology, from automative stress analysis to space shuttle orbital dynamics and from molecular biology to automated manufacturing. Huge Computing is required here to process it.
Ex: Computer Aided Design, System Simulation, MATLABs and some interactive applications.

④ Real time software:- Software that is developed to monitor, control and analyze the real world events as they occur.
Ex: Software required for weather forecasting which tells the status of temp, humidity, and other environmental parameters to forecast weather

⑤ Personal Computer software:- The s/w used in personal computers
Ex: word processors, computer graphics, multimedia and animating tools, Database management, computer games etc...

⑥ Web-based software:- The s/w that is related to web applications.
Ex: CGI, HTML, Java, Perl, DHTML etc...

⑦ Artificial-Intelligence software:- AI software makes use of non-numerical algorithms to solve complex problems that are not amenable (co-operative) to compute and straight forward analysis. This is the s/w designed for robotics, image & voice recognition, expert systems, artificial neural networks, theorem proving and game playing.
etc...

* Software Myths:

S/w Myths :- Beliefs about s/w & the process used to build it.

Different kinds of people i.e, managers, customers & s/w Practitioners may have myths.

1) Management Myths :-

Myth: "s/w manager thinks that if they have a book that is full of standards & Procedures for building s/w, won't that provide his s/w practitioners with everything they need to know."

Reality: The book of standards may very well exist, but is it used already? Are the s/w practitioners aware of its existence? Does it reflect modern s/w engineering practice? Is it complete? Is it adaptable? Like this many questions will arise but the answer for many of the questions is "NO".

Myth : "If they get behind schedule, they can add more Programmers and ~~code~~ catch up."

Reality: s/w developers is not a mechanistic process like h/w manufacturing Here, adding people to a late s/w Project makes it still late. Because if the people are added, people who are working must spend some time educating newcomers about that project, there by reducing the amount of time spent on productive development effort. People can be added but only in a planned & well-coordinated manner.

Myth: "If they[manager] decide to outsource the s/w Project to a third party, they can just release & let that Organization build it."

Reality: If an Organization that is going to outsource the

Project doesn't understand how to manage & control s/w projects internally, it will always struggle when it outsource s/w projects.

2, Customer Myths:

Myth: "A general statement of objectives is sufficient to begin writing programs - we can fill in the details later."

Reality: An ambiguous statement of objectives may cause disaster sometimes. so unambiguous requirements are developed only through effective & continuous communication b/w customer and developer.

Myth: "Project requirements continually change, but change can be easily accommodated because s/w is flexible."

Reality: It is true that s/w requirements change, but the impact of change varies with the time at which it is introduced. When the requirement changes are requested early (before design (or) code has been started) cost impact is relatively small but as the time passes the cost impact grows rapidly.

3, Practitioner Myths:

Person who works for the development of the s/w project.

Myth: "Once we write the program & get it to work, our job is done."

Reality: Actually, the sooner you begin writing code, the longer it'll take you to get done. Industry data indicate that 60% to 80% of all effort expended on s/w 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 s/w quality assurance

mechanism "the formal technical review" can be applied from the beginning of a project. s/w reviews are a "quality filter" that are more effective than testing for finding certain classes of s/w errors.

Myth: "The only deliverable work product for a successful Project is the working Program."

Reality: A working Program is only one part of a s/w configuration that includes many elements. Documentation Provides a foundation for successful engineering & it is the guidance for s/w support.

Myth: "s/w Engineering will make us create voluminous & unnecessary documentation & will always slow us down."

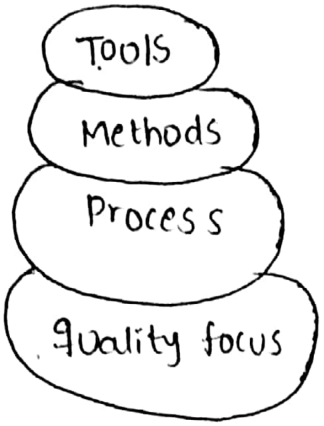
Reality: s/w Engineering is not about creating documents It is about creating quality. Better quality leads to reduced rework and reduced rework results in fast delivery times.

In this manner, the managers, customers & s/w Practitioners have their own beliefs about s/w & the Process that is used to build s/w. So these people should come out of their myths to work for the development of good quality s/w Project.

* Software Engineering - A layered technology:

Software Engineering: The establishment & use of sound (strong or best) engineering principles in order to obtain economically s/w that is reliable and works efficiently on real machines. [OR]

IEEE def:
The application of a systematic, disciplined, quantifiable approach to the development, Operation & maintenance of s/w. [Application of Eng to s/w]



(i) S/w engineering is a layered technology.

(ii) Any engineering approach [including s/w eng] must rest on an organizational commitment to quality.

(iii) The bedrock that supports s/w engineering is a quality focus. Whatever the product we are going to develop, our focus should be on the quality of the Product.

(iv) The foundation for s/w engineering is the process layer. S/w eng process is the glue that holds the technology layers together and enables timely development of computer s/w. Process defines a framework that must be established for effective delivery of s/w eng technology.

(v) S/w eng, methods provide the technical "how to's" for building s/w. Methods encompass many tasks that include communication, requirements, analysis, design modelling, Program construction, testing & support.

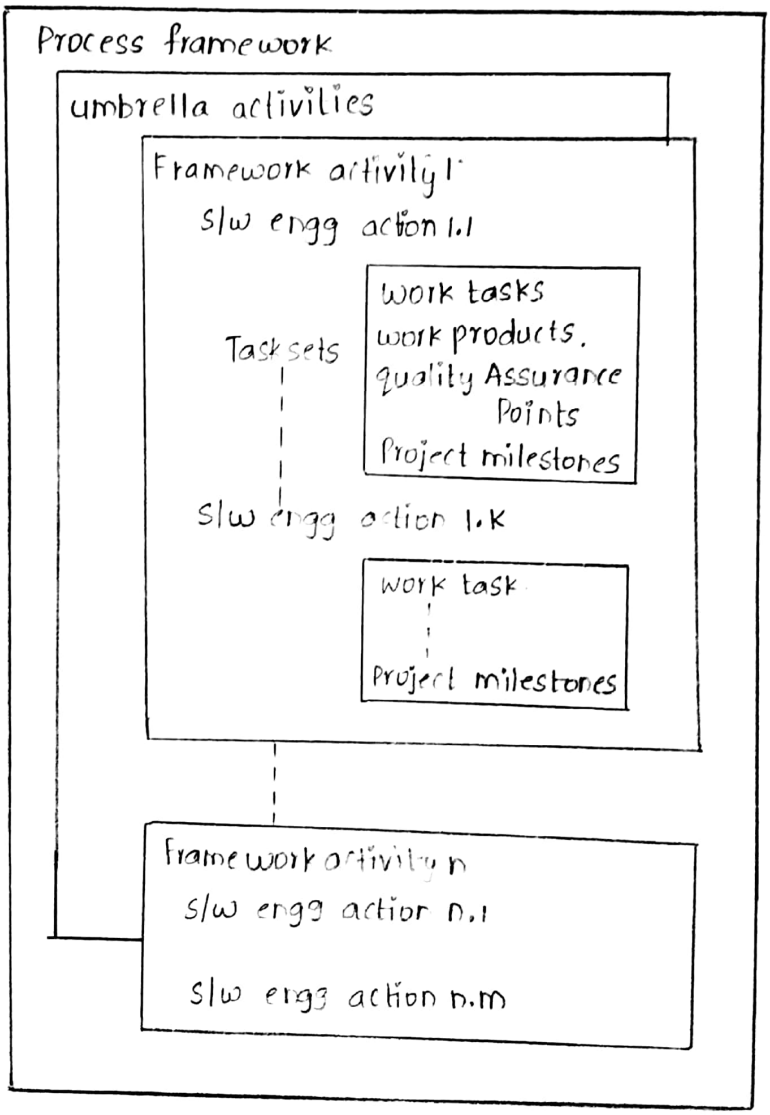
(vi) S/w eng, tools provide automated (or) Semiautomated support for the process & methods. when tools are integrated so that information created by one tool can be used by another, CASE [a system for the support of s/w development] is established. [Computer Aided s/w eng,]

Process framework:

A process framework establishes the foundation for a complete s/w process by identifying a small no; of framework activities that are applicable to all s/w projects and also, the process framework combines a set of umbrella activities that are applicable across the entire s/w process.

"Process Framework → Framework activities + umbrella activities"

s/w process:



- Each framework activity is populated by a set of s/w engineering actions.

s/w Engg action → A collection of related tasks that produces a major s/w engineering work.

Eg: Design

- Each s/w Engg action is populated with individual work tasks that accomplish some part of the work.

A generic process framework that is applicable to any kind of projects consists of the following "framework activities"

1. Communication: Involves heavy communication with customer encompasses requirements gathering & other related activities
2. Planning: Establishes a plan for s/w Engineering work. It describes the technical tasks to be conducted, risks that can occur, resources that will be required, work products to be produced and a work schedule.
3. Modeling: This activity encompasses the analysis and design of the product that is to be developed.
4. Construction: combines code generation (manual or automated) and the testing that is required to uncover the errors in the code.
5. Deployment: Includes the delivery of the product to the customer and getting the feedback after evaluation by customer.

*"Umbrella activities" that are also applicable throughout s/w process include:

1. S/w Project tracking & control: Allows the s/w team to asses project play and take necessary action to maintain schedule.
2. Risk management: Assesses the risks that may effect the Outcome of the project or quality of the product.
3. S/w quality assurance: Defines and conducts the activities required to ensure s/w quality.
4. Formal technical reviews: Assess the s/w engg work products to uncover and remove errors before going to the next action(or) activity.
5. Measurement: Defines and collects the process, Project and Product metrics.
6. S/w configuration Management: Manages the effects of change throughout s/w process.
7. Reusability management: Establishes mechanisms to achieve reusable components.
8. Work Product preparation & production: Encompasses the activities required to create work products such as models, documents tags, forms and lists.

framework activities are used combinely with umbrella activities in any kind of s/w projects i.e., from development of small programs to creation of large web applications & engineering of large complex computer-based systems.

→ For eg, from the generic process framework, consider the modeling framework activity. This "modeling" framework activity is composed of two s/w engg actions: Analysis & Design.

→ Analysis encompasses of set of work tasks such as requirement gathering, elaboration, negotiation, specification & validation that lead to the creation of analysis model.

→ Design encompasses the work tasks such as data design, architectural design, interface design, component-level design that create design model.

→ Each s/w engg action is represented by a no; of different task set. Each **task set** is a collection of s/w engg work tasks, related work products, quality assurance points & project milestones.

→ The taskset that best accommodates the needs of the product and the characteristics of the team is chosen i.e., the s/w engg action ~~occurs~~ can be adopted to the specific needs of the s/w project and the characteristics of the team.

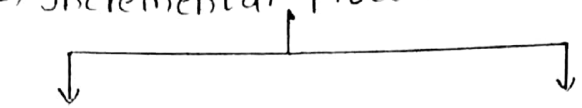
Taskset: Actual work to be done to accomplish the objectives of the s/w engg action.

Eg: Requirements gathering → s/w engg action occurs during communication activity.

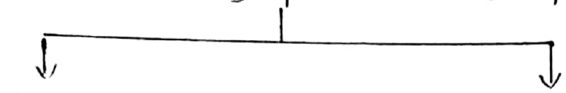
→ For a small project, task set for requirements gathering can be small & for complex project, many things will be considered & the task set will be relatively big.

Process Models:

1) Waterfall model
2) Incremental process model.



3) Evolutionary process model.



4) Unified process model.

Process model: It defines a distinct set of activities, actions, tasks, milestones and work products that are required to develop high-quality s/w. It provides a way for s/w engg work.

->Every process model follows a generic process frame work that encompasses the following framework activities: communication, planning, modeling, construction & Deployment.

* Waterfall Model:

We call it as "Waterfall model" because its diagramatic representation resembles a cascade of waterfalls. We can also call this model as "classic life cycle" or "linear sequential" model

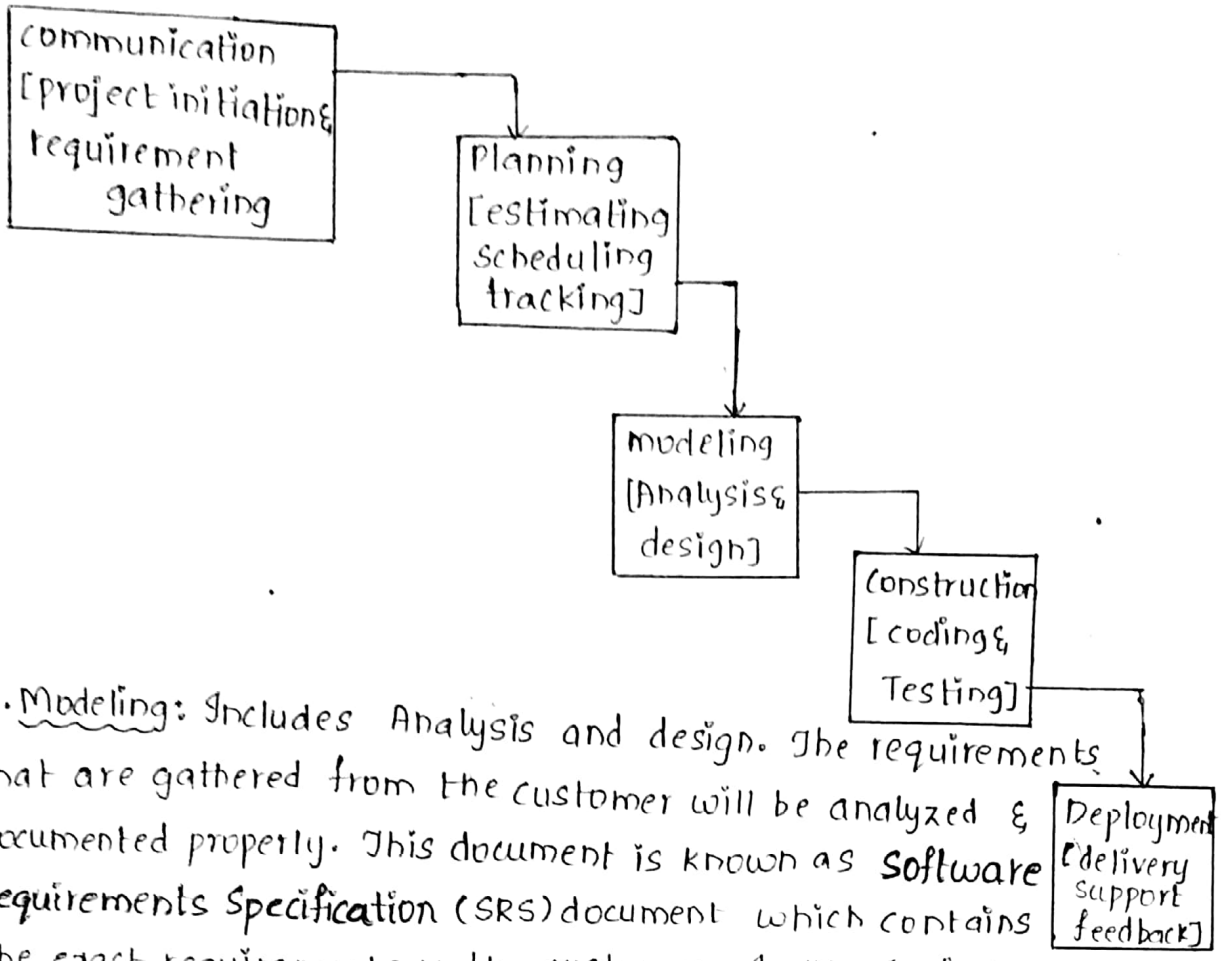
This model has 5 phases: communication, planning, modeling, construction, deployment. The phases always occur in sequential manner and donot overlap.

-> The developer must complete each phase before the next phase begins.

Communication: The goal of this phase is to understand the exact requirements of the customer. This activity is usually executed together with the customer, so heavy communication btwn the customer & developer is needed. The requirements describe "what" of a system, not the "how".

Planning: This activity establishes a plan for s/w engg work. It describes the technical tasks to be conducted, the risks that are likely, the resources that will be required, the work products to be produced & a work schedule. And also cost will be estimated

based on the requirements.



3. Modeling: Includes Analysis and design. The requirements that are gathered from the customer will be analyzed & documented properly. This document is known as **Software Requirements Specification** (SRS) document which contains the exact requirements of the customer. In "Design" phase the requirements specifications are transformed into a structure that is suitable for implementation in some programming language. Here overall s/w architecture is defined and high-level, low-level design work is performed. This SDD should be sufficient to begin the coding.

4. Construction: Includes coding and testing. If the design is completed, coding will be done in some programming language. We can use code generation tools in this coding phase. After the completion of the coding it will be tested and we can use different testing tools to uncover the errors.

5. Deployment: After completion of the testing, if the product is working efficiently, it will be delivered to the customer. After that customer evaluates the product & sends the feedback to the organization that developed the product. After that the s/w maintainence is the very important task that will be done by the organization that developed the product. s/w maintaince is a broad activity that includes error correction, enhancements& optimization.

Problems of Waterfall model:

1. It is difficult to define all the requirements at the beg of the project.
2. This model is not suitable for accomodating any change.
3. Working slw is not available till the end of the project.
4. It doesnot suit well to large projects.
5. Real projects are rarely sequential.

Due to these problems, the application of waterfall model should be limited to situations where the requirements and their implementation are well understand.

Eg: If an organization has experience in developing accounting systems then building a new accounting system based on existing designs could be easily managed with this model.

* Incremental Process Models:

These models are effective when the requirements are well defined and then no confusion about the functionality of the end product.

-> In these models, after every cycle, a useable product is given to the customer. Every new cycle will have additional functionality.

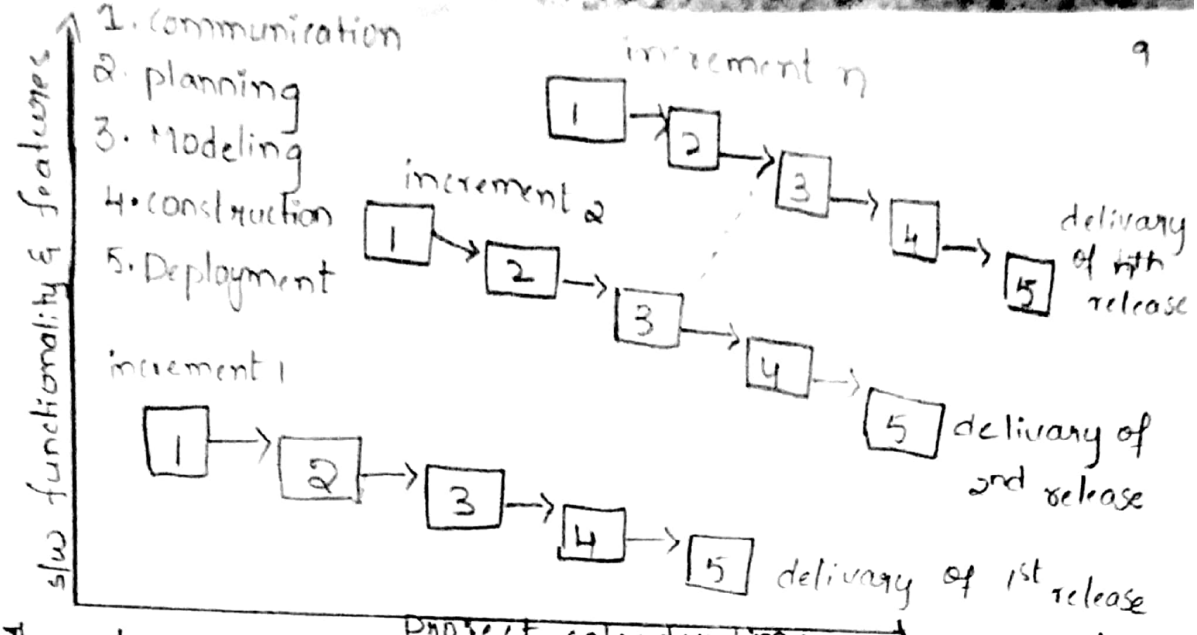
-> Eg: In the university automation slw, library automation module may be delivered in the 1st phase and examination automation module in the 2nd phase & so on.

-> These incremental process models are better when atleast some working products need to be delivered quickly for the use.

Incremental model / Iterative Enhancement model:

-> This model has the same phases as the waterfall model, but these phases may be conducted in several cycles.

-> A useable product is released at the end of each cycle, with each release providing additional functionality.



→ The phases communication, planning, Modeling, construction & deployment are same in every cycle as in waterfall Model.

→ The aim of waterfall & prototyping models is the delivary of a complete, operational & good quality product.

→ But this model delivers an operational quality product at each release that satisfies only a subset of the customer's requirements.

→ The complete product is divided into releases, and the developer delivers the product release by release as shown in figure.

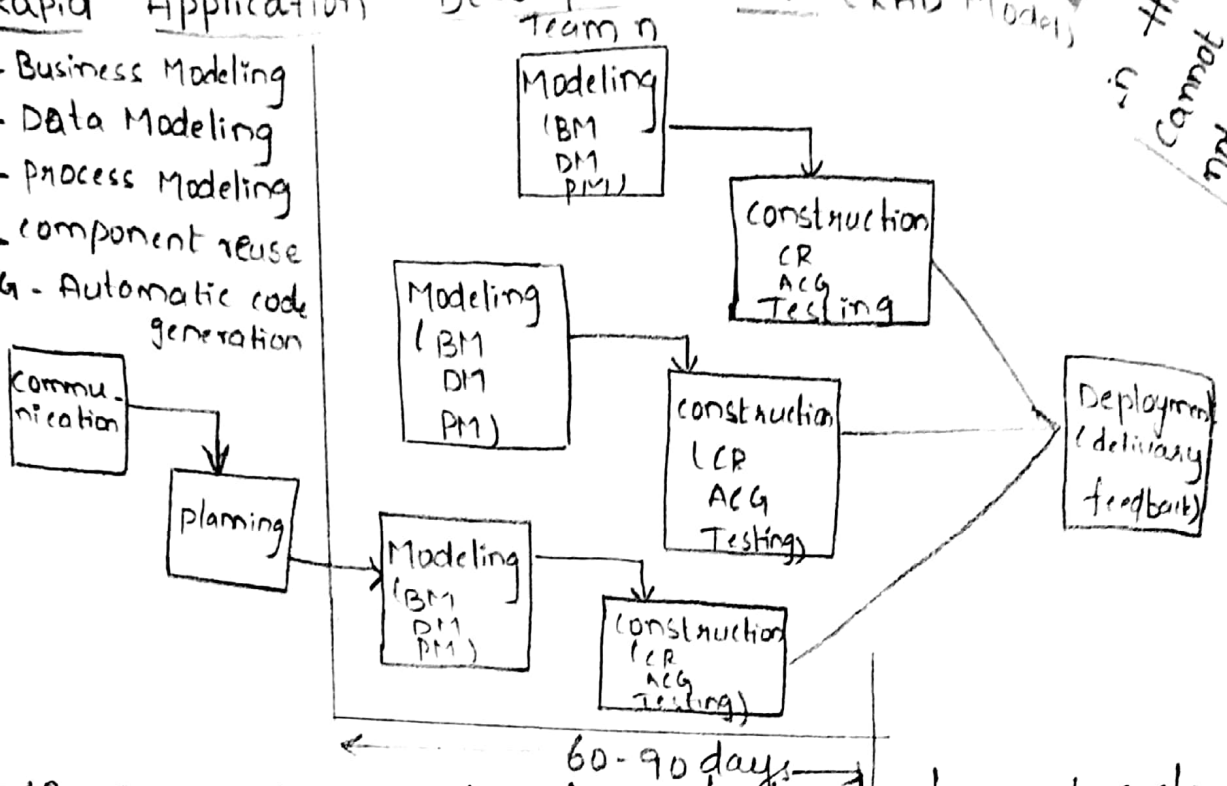
→ At each release, customer gets an Operational quality product that does a portion of what is required. The customer is able to do some useful work after first release.

→ For Eg:- Word processing s/w developed using this model might deliver basic file management Editing at first release; more sophisticated editing & document production functions at 2nd release; Spelling & grammer checking after 3rd release; and advanced page layout capability in the next release like that

→ With this model, 1st release may be available with in few weeks (or) months and the customer can do some useful work with the product. But the customer has to wait for some years to get recode a product using waterfall & (or) Prototyping models

Rapid Application Development Model (RAD Model)

BM - Business Modeling
DM - Data Modeling
PM - process Modeling
CR - component reuse
ACG - Automatic code generation



→ This RAD model consists of a short development cycle
→ RAD model is "high-speed" adoption of waterfall model, in which rapid development is achieved by using a component-based construction approach.
→ If the reauirements are well understand is project scope is limited, RAD process enables the development team to create a "fully functionally" system" with in a very short time period (for eg: 60 to 90 days)
→ RAD model also follows the ~~fifth~~ five genric framework activities communication → to understand the business reauirements from the customer planning is very important in this model because multiple s/w teams work in parallel on different systems [component-based] Modeling here encomphasises 3 more phases Business modeling, data modeling & process modeling. construction in this model use reusable components & automatic code generation & testing tools Deployment Establishes a basis for subsequent iterations.

In this model, the user involvment is more & if the user cannot be involved throughout the life cycle, this may not be an appropriate model

Draw backs of RAD Model:-

i) For large projects, RAD requires sufficient human resources to create the right number of RAD teams.

ii) Highly specialized & skilled developers are expected & such developers may not be available

iii) It may not be appropriate when technical risks are high. (for eg:- when a new application makes heavy use of new technology).

iv) It may not be effective, if the system can not be properly modularised

Evalutionary Process Models → prototyping model
→ spiral model

-> In these models, the same phases as defined for waterfall model occur but in a cyclic fashion. This model however, differs from incremental models in the sense that requires a useable product at the end of each cycle.

-> For eg:- in a simple database application, one cycle might implement graphical user interface (GUI); another file manipulation; another queries; & another updates. But customer can not work with these individuals. customers needs complete End product here to work with. Here we go for Evalutionary development models which provide a End product quickly but with less quality & can be implemented again with good quality later, after knowing complete requirments.

→These models are useful projects using new technol
that is not well understood. Also useful for complex
projects in which reauirments are not well understood
at the beginning

Prototyping Model

→ Some times, a customer defines a set of general objectives for s/w but he does not know, what Exactly he needs, he can not identify detailed input, & output reauirments. Also the developer may not sure of Efficiency of an algorithm & the adaptability of os. In these situations prototyping model is very useful.

→ In this model, first a working prototype is developed instead of actual s/w. The working prototype is developed as per the current available reauirments. This working prototype has less functional capabilities, low reliability & untested performance (i.e low performance)

→ The developers use this prototype to refine ther reauirements and prepare the final reauirments specificati -on document. Because the working prototype has been Evaluated by the customer is customer gives the feedback to developers that helps to remove uncertainities in the reauirements.

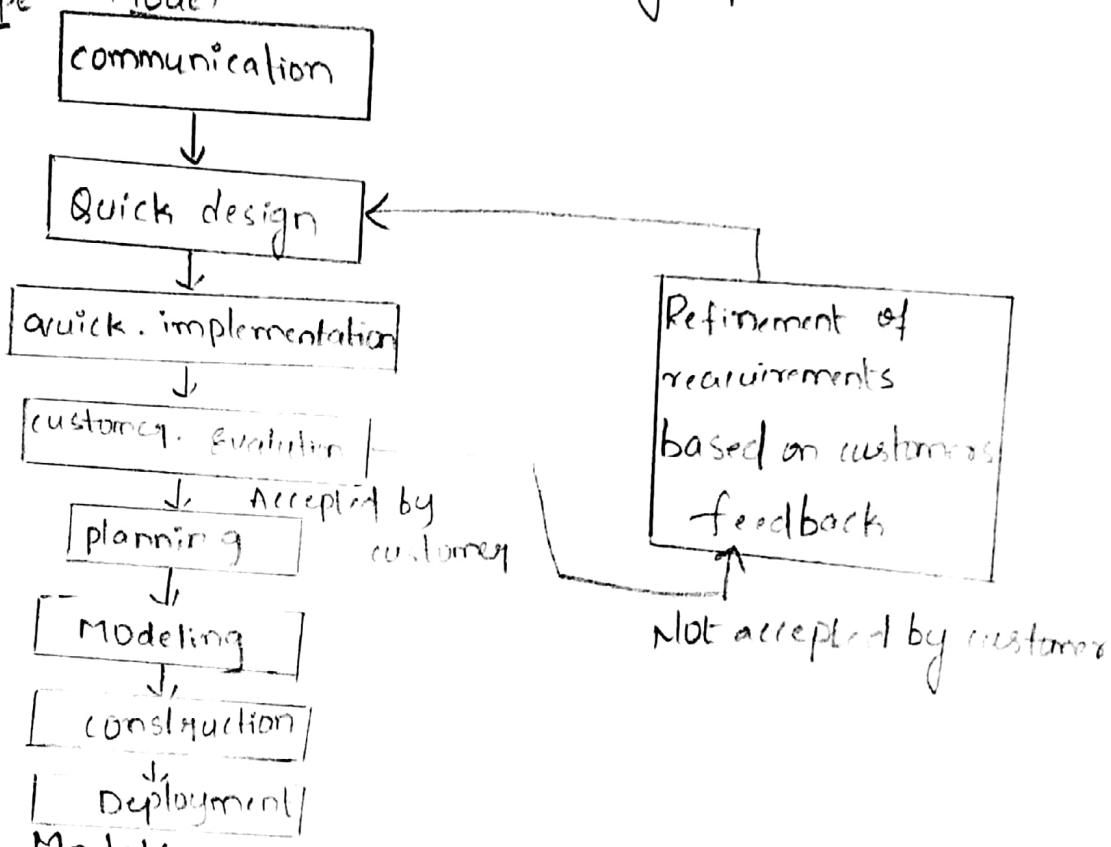
→ The prototype may be a useable program, but it is not suitable as the final s/w product because of its poor performance, low reliability and very low auality

→ The developers should develop the prototype as early as possible to speed up the s/w development process.

→ After the finalization & s/w reauirement specification document the prototype is discarded (i.e the code for prototype is thrown away). The only use of the prototyp is to determine the customer's real needs.

-> The Experiance gathered from developing the prototype helps in developing actual s/w
-> After getting the final reaquirements, the same phases as in waterfall model will be followed.
-> This model produces maintainable & aquality s/w. But this model reaquires Extensive participation & involvments of the customers which is not always possible.

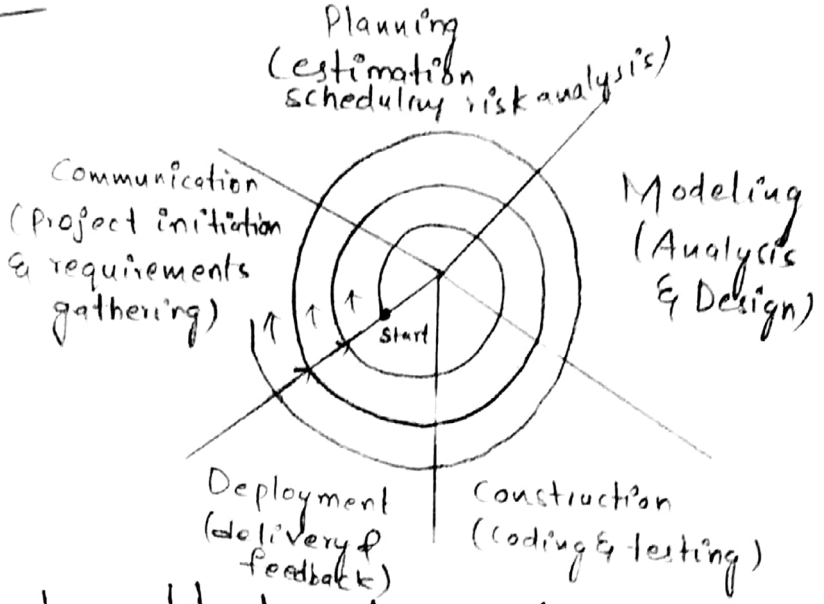
Prototype Model



Spiral Model:-

->This is an evolutionary s/w process model that combines the iterative nature of prototyping with the controlled aspects of waterfall model.
->using the spiral model, s/w is developed in series of Evalutionary releases. During early iterations, a working prototype might be released. During later, iterations, increasingly more complete versions of the s/w product are released.

Spiral Model



-> Spiral model also follows the same generic framework activities as in waterfall model: communication, planning, modeling, construction & deployment each of the framework activities represent one segment of the spiral path as shown in figure.

-> As the evolutionary process begins, the s/w team performs activities around the spiral in clockwise direction, begining at the center

-> The first circuit in the spiral in clockwise might result in progressively more sophisticated versions of the software.

-> Each pass through the planning region results in adjustments to the project plan. cost & schedule are adjusted based on feedback delivered from the customer after delivary of the prototype.

-> The first circuit around the spiral might represent a "concept developed project" which starts at center & continues for multiple iterations to get the product from until the concept development is completed. Next it may take some more iteration to get the product from concept which is the "new product development project".

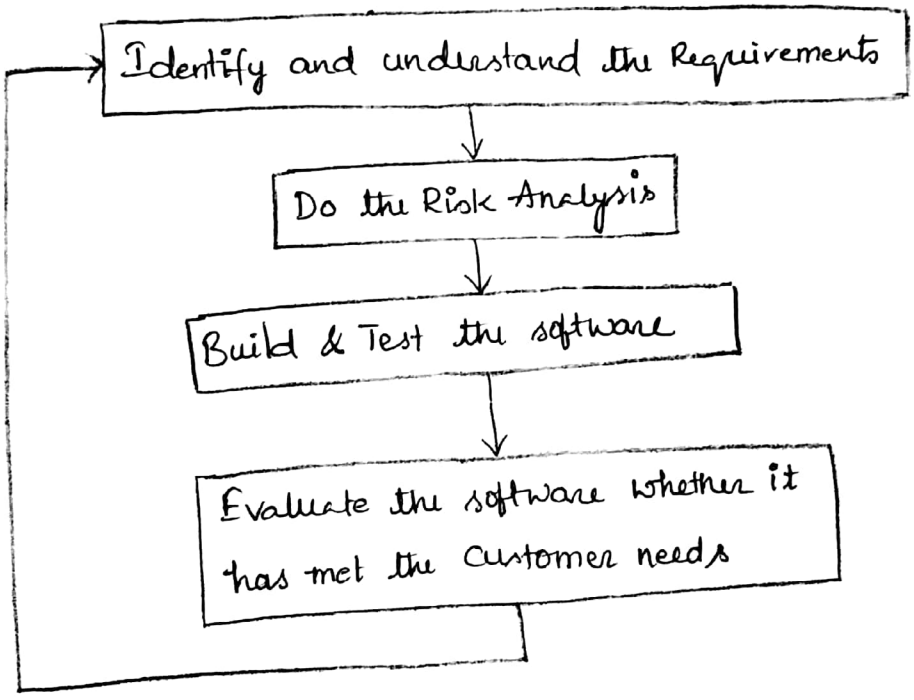
Next it may take some more iterations to get a quality product by making enhancements & it is called as "product Enhancement project"

-> The spiral model is a realistic approach to the development of large-scale systems & software

-> The main advantage of this model is "risk reduction mechanism" which will be focused mainly in this model to reduce risks using prototyping

-> But this model depends much on risk assesment expertise for success.

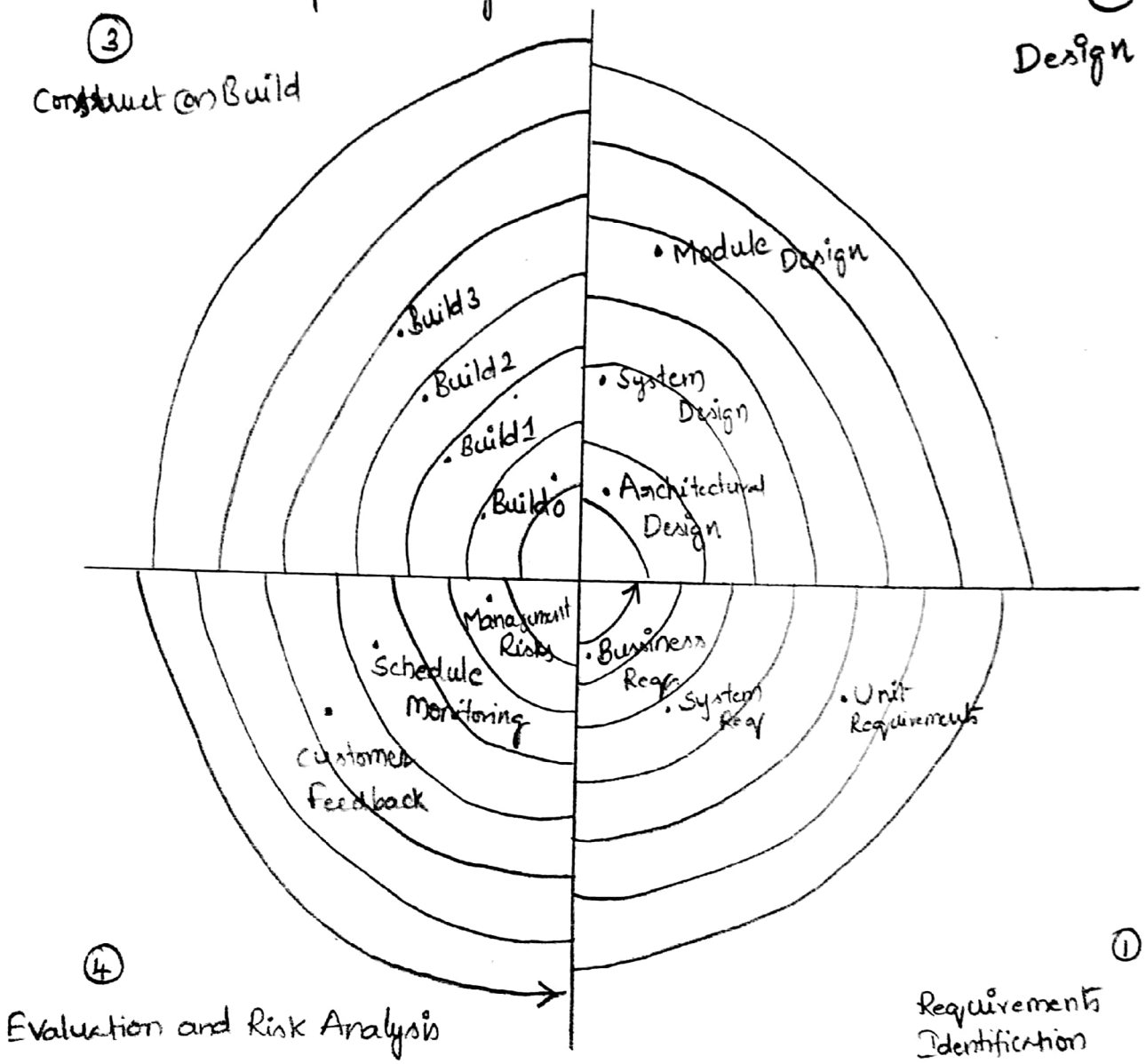
-> To Explain in simpler terms, the steps involved in Spiral Model :-



Applications :-

1. When there is a budget constraints and risk evaluation is important.
2. for Medium to High-Risk projects.

3. Long term projects because of potential changes to econ... priorities as the requirements change with the time.
4. When the requirements are complex and need some evalu... to get clarity.
5. Significant changes are expected in the product during the development cycle.



Drawbacks:-

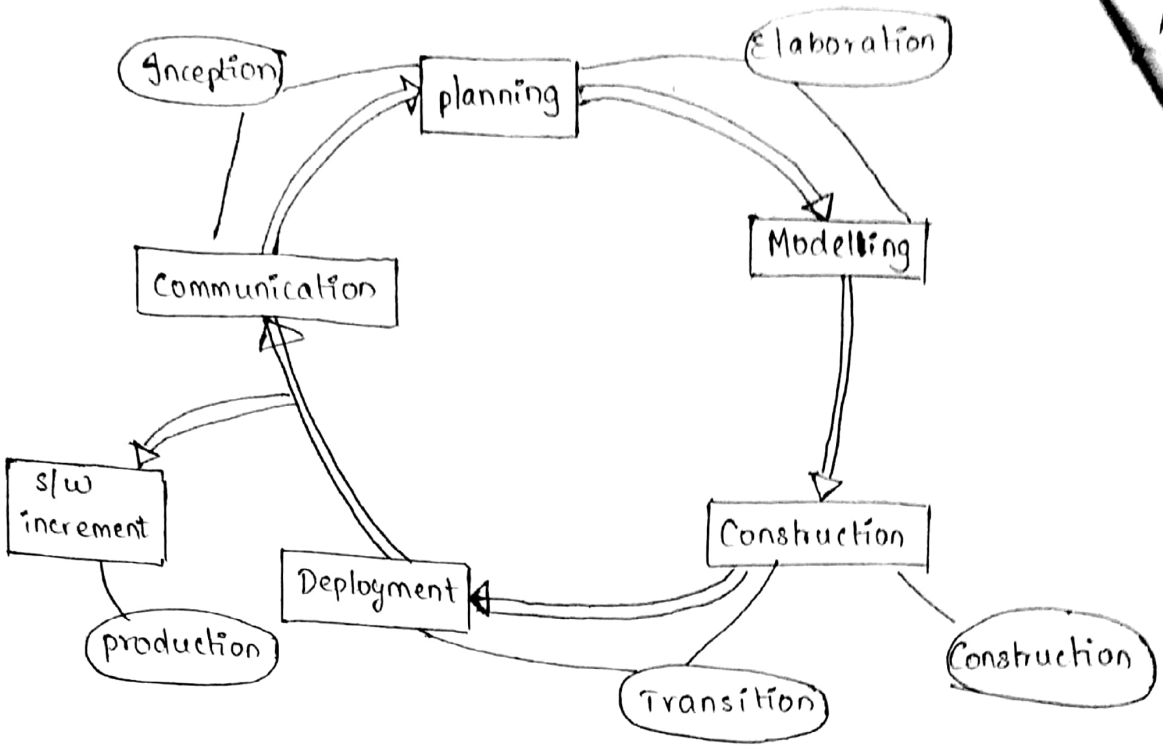
1. Management is more complex
2. End of project may not be known early.
3. Not suitable for small (or) Low risk projects
4. Large no. of intermediate stages require excessive documentation.

The unified Process model:-

- The unified process is a "use-case driven, architecture-centric, iterative and incremental" s/w process designed as frame-work for UML methods and tools.
- The unified process shows the importance of customer communication and efficient methods for describing the customer's view of a system (i.e the use-case). It also emphasizes the role of s/w architecture and helps the architect focus on right goals i.e understandability, adaptability to future changes and reuse. This model also suggests a process flow that is iterative and incremental, providing the evolutionary nature which is essential in modern s/w development.
- In the past, people were working much on procedure-oriented technologies. But in the recent past i.e from 1980's onwards Object-oriented (OO) methods and programming languages got widespread usage. So to use these object-oriented ~~languages~~ technologies in a systematic manner, 3 persons namely Jacobson, Rumbaugh and Booch developed a unified modeling language (UML). Later UML became an industry standard for object-oriented s/w development. UML provides the necessary technology to support object-oriented s/w engineering s/w engineering using UML, which is unified process.
- Today, the unified process and UML are widely used on object-oriented projects of all kinds.

phases of unified process:-

- There are 5 phases involved in unified process:
  - Inception, Elaboration, Construction, Transition, production.



— The inception phase encompasses both customer communication and planning activities. In this phase, by collaborating with the customer and end-users, business requirements for the s/w are identified, a rough architecture and a plan for the iterative, incremental nature of the project is developed. In this model, business requirements are described through a set of preliminary use-cases. A use-case describes ~~through a set of~~ sequence of actions that are performed by an actor (eg: a person or a system). use-cases help to identify the scope of the project and provide a basis for project planning. Architecture is this phase is just a brief description of the features and the functions of the s/w to be developed. Later the architecture will be expanded in detail. planning ~~identifier~~ identifies resources, assesses major risks, defines a schedule, and establishes a basis for the phases that are to be applied as the s/w increment is developed.

- The elaboration phase encompasses the planning and modeling activities of the generic process model. Elaboration refines and expands the preliminary use-cases that were developed as part of the inception phase and expands the architectural representation to include five different of the s/w → the use-case model, the analysis model, the design model, the implementation model and the deployment model.
- In some cases, elaboration creates an "executable architectural baseline" which is similar to the prototype (not same). In this phase, also the plan is carefully reviewed at the end of this phase ensure that scope, risks and delivery dates remain reasonable modifications to the plan may be made at this time
- ~~The construction phase of the generic process model.~~
- The construction phase of UP (unified process) is identical to the construction activity of the generic process model. using the architectural model as input, the s/w components that make each use-case operational for end-users are developed in this phase. For this, analysis and design models should be completed.
- As the s/w components are being implemented, unit test are designed and executed for each. In addition, integration activities (component assembly and integration testing) are conducted.
- The transition phase of the UP encompasses the latter stages of the generic construction activity of the first part of generic deployment activity. Here s/w is given to end-users for beta testing and user feed back reports both defects and necessary changes.

- Also the s/w team creates necessary support information such as user-manuals, installation produces that is required for the release. At the end of transition phase, the s/w increment becomes a useable s/w release
- The production phase of the UP is similar to the deployment activity of the generic process model. During this phase, ongoing use of s/w is monitored, support for the system is provided and defect reports and requests for changes are submitted and evaluated.
- UP phases do not occur in a sequence, but with staggered concurrently

Unified model work products:-

Inception phase → vision document, initial use-case model, initial risk assessment, project plan, business model etc.,

Elaboration phase → complete use-case model, additional requirements, Analysis model, s/w architecture description, Executable architectural prototype, preliminary design model, revised requirements list, revised project plan, preliminary user manual etc.,

Construction phase → Design model, s/w components, integrated s/w increment, Test plan and procedure, Test cases, support document user manuals, installation manuals etc.,

Transition phase → Delivered s/w increment, Beta test reports, general user feed back.

production phase → NO work products but the s/w support is provided