UNIT - I

INTRODUCTION, SOFTWARE PROCESS.

INTRODUCTION:

1) Professional software development
   - Software engineering
   - Software engineering diversity
   - Software engineering and web
2) Software engineering ethics.

Professional software development.

Software development is a professional activity where s/w is developed for specific business purposes. It is maintained and changed throughout its life.

A professionally developed s/w system is often more than a single program. It is group of seperate programs & configuration files. It also includes system documentation.

Software figure 1.1 frequently asked qstns about s/w. refer page 6. (Text)

* → Software engineers are concerned with developing software products (ie s/w which can be sold to the customers).

→ There are two kinds of s/w products.

a) Generic product (b) Customized (bespoke) product

Generic product:- This are stand-alone systems that are marketed and sold to any customer who wishes to buy them.

Examples: Software for PC's such as database, word processors, project management tools, CAD s/ws, software for specific markets such as appointments systems for dentists etc.

Customized (or bespoke) products: These are systems that are commissioned by a specific customer to meet their own needs.
Examples: Embedded control systems, air traffic control system s/w, traffic monitoring systems.

The important difference between these types of software is that
In generic products the organization that develop the s/w controls the s/w specification ie the changes and decisions are made by the software developer
In customized products specification & decision of what software should do ie what changes are required for software are decided by the customer.

Essential attributes of good professional software

Product characteristics	Description
1) Maintainability	:→ Software should be written in such a way that it can evolve to meet the changing needs to customers.
2) Dependability and security	:- Software dependability includes a range of characteristics including reliability

security, and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.

Efficiency : Software should not make wasteful use of system resource such as memory and processor cycles. Efficiency therefore includes responsiveness Processing time, memory utilization etc.

Acceptability: Software must be acceptable to the type of users for which it is designed ie it must be understandable, usable and compatible with other systems that they use.

Software engineering.

Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.

In the above defination there are two key phrases:

1) Engineering discipline:→ Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.

All aspects of software production: Software engineering is not just concerned with the technical processes but also concerned with project management & development of tools, methods etc, to support software production.

Importance of Software engineering:

Software engineering is important for two reasons

1. More and more, individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.
2. It is usually cheaper in the long run to use software engineering methods and techniques for long run software systems rather than just write the programs as if it was personal programming project.

Software process activity.

A software process is a sequence of activities that leads to the production of a software product. These activities are

1. Software specification: where customers and engineers define the software that is to be produced and the constraints on its operation

2. Software development: where the software is designed and programmed.

3. Software validation: where the software is checked to ensure that it is what the customer requires.

4. Software evolution: where the software is modified to reflect changing customer and market requirements.

General issues that affect most software.

1. Heterogeneity:
Increasingly, systems are required to operate as distributed systems across networks that include different types of computer and mobile devices.

2. Business and social change
Business and society are changing quickly as emerging economies develop and new technologies become available. They need to be able to change their existing software and to rapidly develop new software.

3. Security and trust:
As software is intertwined with all aspects of our lives, it is essential that we can trust that software. ie we have to make sure that malicious users cannot attack our software and that information security is maintained.

Software engineering ethics → moral principles that govern a person's behaviour or the conducting of an activity

Software engineering involves wider responsibilities than simply the application of technical skills.

- Software engineering must behave in an honest and ethically responsible way.
- Ethical behaviour is more than upholding the laws but it
- Ethical behaviour involves set of principals that are morally correct.

Issues of professional responsibility are

1) **Confidentiality** :→ Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality has been signed.

2) **Competence** :→ Engineers should not misrepresent their level of competence. ie they should not accept work which is outwith their competence.

3) **Intellectual property rights** :→ Engineers should be aware of intellectual property such as patents, copyrights etc. They should be careful that their intellectual property of employers and clients is protected.

4) **Computer misuse** :- s/w engineers should not use their technical skills to misuse other people's computers.

ACM/IEEE Code of Ethics. (Association for Computing Machinery. Institute of Electrical & Electronic Engineers)

Professional societies and institutions have an important role in setting ethical standards.

Organization such as ACM IEEE (Institute of electrical & electron etc publish a code of professional conduct or engin code of ethics.

Rationale (a set of reasons or a logical basis for a course of action or belief) for the code of ethics.

(Text book page 15, 16)

The code contains eight Principles related to the behaviour of and decisions made by professional s/w engineers.

1. PUBLIC :→ S/w engineers shall act consistently with the public interest.
2. CLIENT AND EMPLOYER :→ S/w engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. PRODUCT : S/w Engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT : Software engineers shall maintain integrity and independence in their professional judgement.
5. MANAGEMENT :- S/w engineering managers & leaders shall subscribe to and promote an ethical approach to the mgmt of s/w development & maintenance.
6. PROFESSION :→ S/w engineers shall advance the integrity & reputation of the Profession consistent with

7. COLLEAGUES :→ S/W engineers shall be fair to colleagues.

8. SELF :→ S/W engineers shall participate in life long learning.

Ethical dilemmas (difficult situation or a problem).

1) Disagreement in principle with the policies of senior mgmt.
2) Your employer acts in an unethical way
3) Participation in the development of military weapons systems or nuclear systems.

Software Processes

Software process models :- is a simplified representation of a s/w process.
- The waterfall model
- Incremental development
- Reuse-oriented software engineering

Process activities
- Software specification
- Software design and implementation
- Software validation

Coping with Change.
- Prototyping
- Incremental Delivery
- Bohm's Spiral Model.

The Software Process:

A software process is a structured set of activities required to develop a software system.

There are many different types of software process but all include 4 activities that are fundamental to s/w engineer.

1. Software specification :-> defining what the system should do.
2. Design and implementation - defining the organization of the system and implementing the systems.
3. Validation :- checking that it does what the customer wants.
4. Evolution :- changing the system in response to changing customer needs.

Plan driven and agile processes.

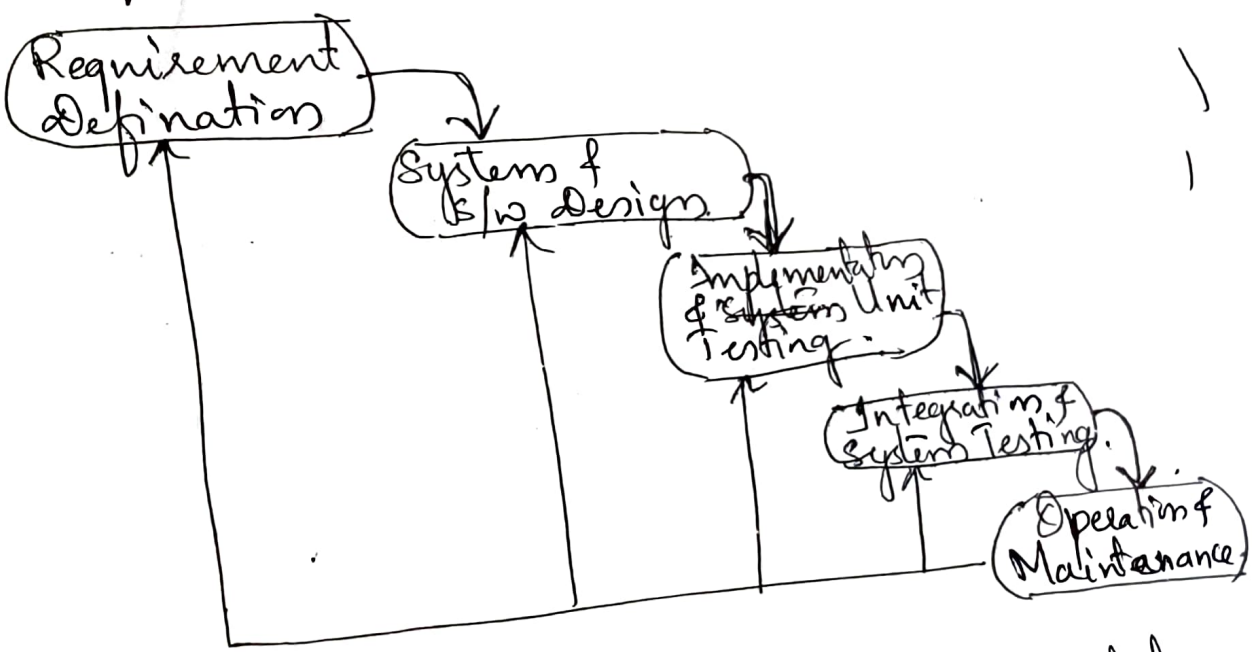
Plan driven procss are the processes where all of the procss activities are planned in advance and progress is measured against this plan.

In agil processes planning is incremental and it is easier to change the procss to reflect changing customer requirements.

Most of the s/w processes include both plan-driven and agile approaches.

Software procss models

1. Waterfall model :



Water fall model is a plan-driven model
- Seperate and distinct phases of specification and development.

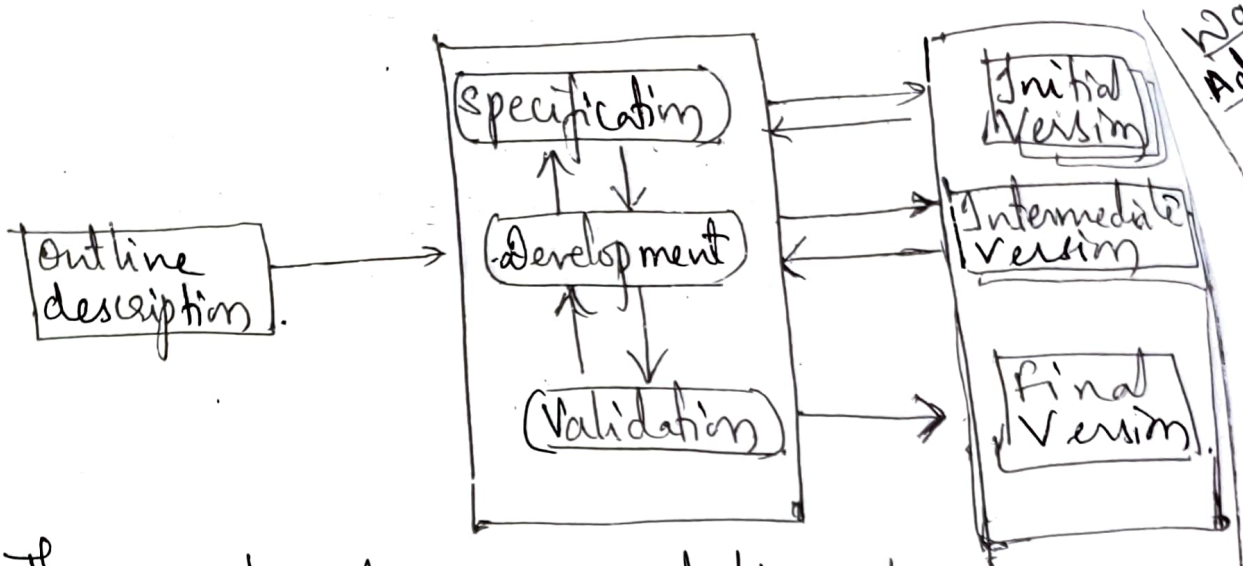
The seprate phases of waterfall model are
- Requirements and definition: services, constraints & goals are established by consultation with system user
- System and software design: requirements collected are given to the design
- Implementation and Unit Testing: each model is implemented tested so that each model meet its requirements
- Integration and System Testing.
- Operation and Maintenance: This is a longest life cycle phase. when the errors correction, errors which were not discovered in earlier stages of the life cycle

The main deawback of waterfall model is the difficulty of accommodating the changes made after the procer is underway.
ie Commitments must be made at the early stage in the procer.
∴ this model is only appropriate when the requirements are well understood and changes will be fairly limited during the design procer.

→ Waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.

Incremental Development

Incremental development is based on the idea of developing an initial implementation and evolving it through several versions until an adequate system has been developed.
Specification, development and validation are interleaved. May be plan driven or agile.



→ The cost of accommodating changing customer requirements is reduced.
ie the amount of documentation that has to be redone is much less than is required with the waterfall model

→ It is easier to get customer feedback on the development work that has been done.
ie customer can comment on the s/w and see how much has been implemented

→ More flexible.

→ Customer can be able to use & gain value from the s/w earlier than is possible with a waterfall process

→ Customer feedback is recived after the delivery of each component.

Drawback.

→ The process is not visible..

→ Systems structure tends to degrade as new increments are added.

→ ~~More expensive~~ ~~components~~ ~~to waterfall model~~

Waterfall Model

Advantages

- easy to understand
- easy to manage
- fewer production issues
- better budget mgmt.
- Sequential method.

Disadvantages.

- Not flexible.
- doesn't handle unexpected risks
- Not a good well. for complex or long-term projects. requirements are not understood
- Difficult to capture all requirements at the initial stage
- Iterative (Interleaved) method.

- A waterfall model is a sequential project management methodology where one phase completely finishes before the next phase begins.

Example: A case study of incremental model:
Suppose we want to develop a web-based social network with the following functionality
a) The user should log into the system and can send or accept the friend request
How can we use the incremental model in this scenario.

Soln:
We need to convert the system into seperate components.
Component 1: Sign up & log in
Component 2: Send friend request.
Component 3: Accept friend request.

- Now when we start one activity with component 1 (signup & login). This component undergoes the phase of requirements gathering and analysis design & implementation, deployment & maintaining. When this component is ready, it will be delivered to the customer.

→ After

| Signup & login | → component 1 is ready after first increment.

→ After that, we add or increment another component 2 that sends friend request. This component undergoes the phases of req gathering & analysis, design & implem, deployment & ~~analysis~~ maintainance. When this component is ready it is delivered to custom.

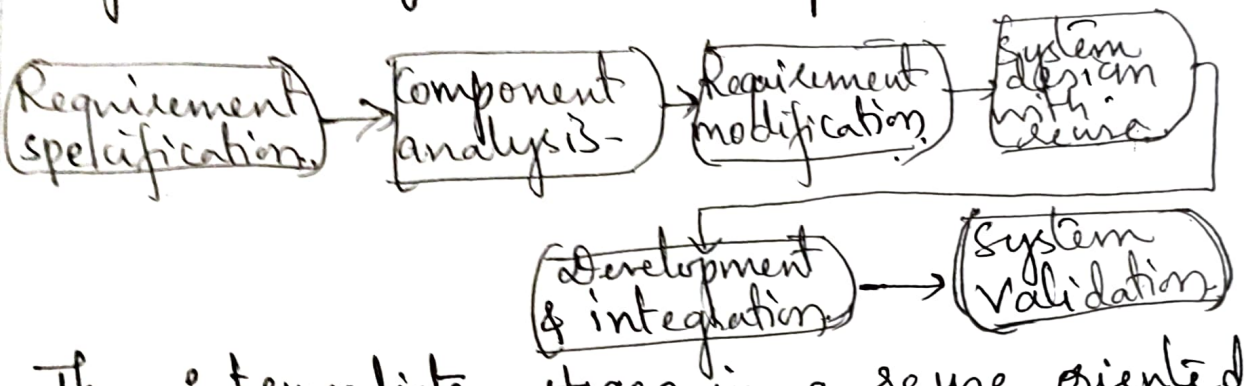
| signup & login | → Component 1 is ready for first increment.
| Send friend reqr | → Component 2 is ready for 2nd int.

→ After that, we add 3 componen ie accept friend reqn. that again undergoes the phases req. gath & analysis design & implm deploymment & maintainance. When component is ready it is delivered to the cus.

| Signup & login |
| Send friend req |
| Accept frn req |

Reuse - oriented software engineering.

- Many software projects reuse code that are required & modify them as needed and make use in their system.
- Reuse oriented approaches rely on a large base of reusable s/w components and an integrating framework for the composition of these components.



The intermediate stages in a reuse oriented process are different. These stages are

1. Component analysis: Given the requirements specification a search is made for components to implement that specification. Usually there is no exact match & the components that may be used only provide some of the functionality required.
2. Requirement modification: During this stage, the requirements are analyzed using information about the components that have been discovered. They are then modified to reflect the available components. When modification are impossible, then component analysis activity may be re-entered to search for alternate solutions.
3. System design with reuse: The framework of the system is designed or an existing framework is reused.
4. Development and integration: Systems that cannot be externally procured is developed, & the components & COTS systems are integrated to create the new system.

Process activities

- Software specification
- Software Design and Implementation
- Software Validation.
- Software Evolution.

The four basic process activities of specification, development, validation & evolution are organized differently in different development processes.
In waterfall model they are organized in sequence whereas in increamental develpt they are interleaved.

1) Software Specification or Requirement Engineering

Software specification or requirements engineering is the process of understanding and defining what services are required from the system & understanding system's operation & development.
Four main activities in the requirements engineering process:

a) **Feasibility study**: An estimate is made of weather the identified user needs may be satisfied using current s/w & h/w technology. It also considers weather the ~~system~~ proposed system is cost effective from all point of view. The result should inform the decision of weather

Requirement elicitation and analysis: This is the process of deriving the system requirements through observation of existing systems, etc. This can help to understand and develop more than one system.

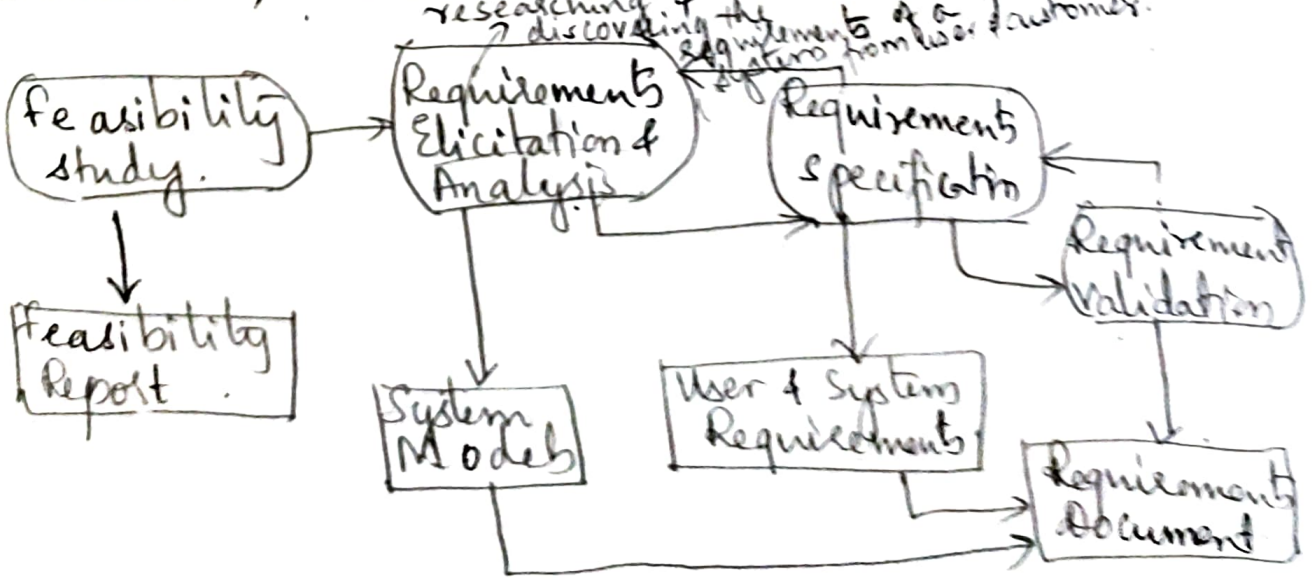
c) Requirement specification: Requirement specification is the activity of translating the information gathered during the analysis activity into a document that defines a set of requirements. Two types of requirements are included in this document → (i) User requirement (ii) System requirement

User requirement :- These are the abstract statements of the system requirements for the customer & end users.

System requirements: These are a more detailed description of the functionality to be provided.

d) Requirement validation:

Checking the validity of the requirements for realism, consistency and completeness. During this process if errors are discovered in document, then it must be modified & corrected.



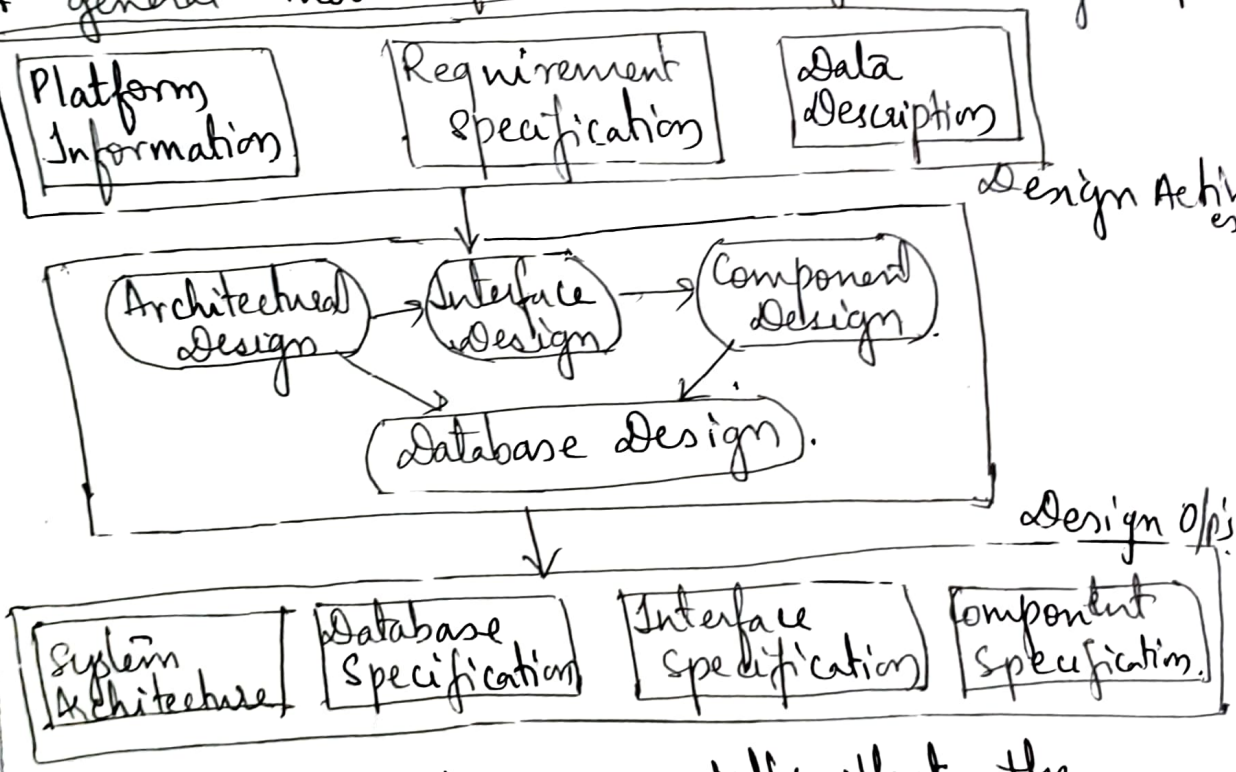
2. Software design and Implementation:

The process of converting the system specification into an executable system.

Software design is a description of the structure of the s/w to be implemented.

Implementation: translate this structure into an executable program

A general model of the design process



→ The above diagram tells that the design process is sequential. Infact design process shows inputs to Design process, process activities, and the documents produced as outputs from this process.

→ Platform Information will specify the environment in which the s/w will execute

→ requirement specification is description of the functionality the s/w must provide.

→ If the system uses the existing data

then the description of that data must be provided or specified included in platform specification otherwise. data description must be an input to design process

→ Design activities may be interleaved. Some of the design process may include database & some do not. Depending upon the the system being developed design process vary.

In the above figure shows four activities that is part of design process.

→ Architectural design, where you identify the overall structure of the system.

Interface design, where you define the interfaces between system components. It should be unambiguity

Component design, where you take each system component and design how it will operate.

Database design where you design the system data structure and how these are to be represented in a database.

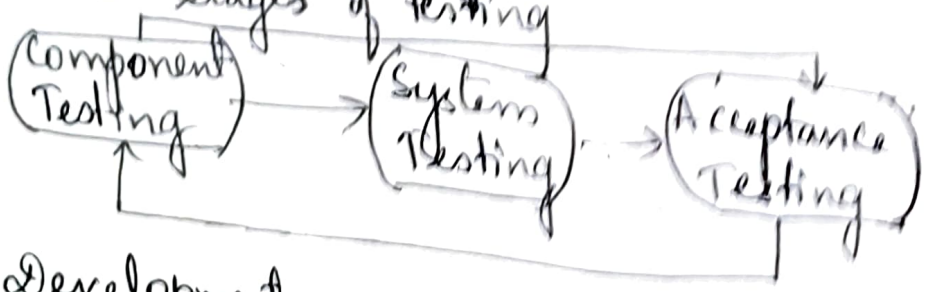
These activities lead to set of design outputs which is shown in above figure.

3 Software Validation

Verification and validation (V & V) tells that a system conforms to its specification and meets the requirements of the system customers.

Validation & Verification involves checking & reviews process and system testing.

Three-stages of testing



1) Development or component Testing
- Individual components are tested independently
- components can be simple entities such as functions, or objects etc.

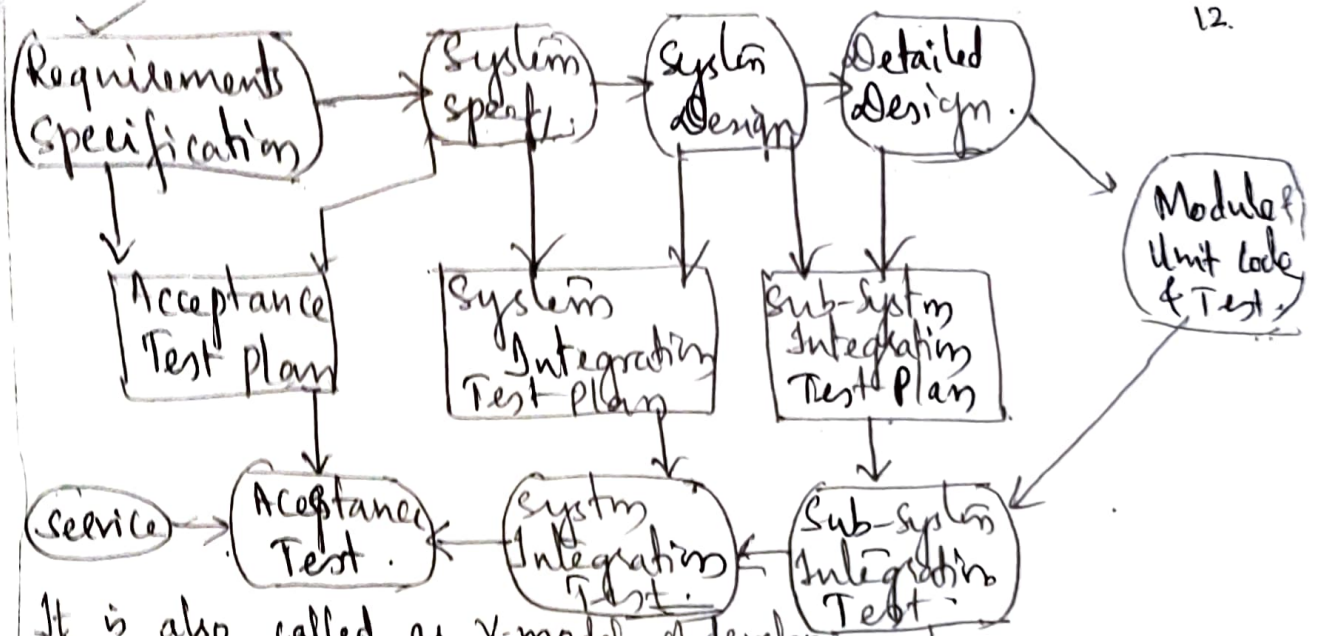
2) System Testing
- System components are integrated to create a complete system.
- Testing of the system as a whole.

3) Acceptance Testing
- This is the final stage in the testing process before system is accepted.
- Testing for customer data to check that the system meets the customer's needs.

→ Testing phases in a plan driven software process

When a plan driven s/w process is used, testing is done by a set of test plans.

- A team of testers who are independent



It is also called as V-model of development.

→ There are two type of testing
alpha testing and beta testing.

→ Acceptance testing is sometimes called alpha-testing.

→ Custom systems are developed for a single client. A-testing process continues until the system developer & the client agree that the delivered system is an acceptable

Ex: Generic products: Customized products such as Library s/w etc

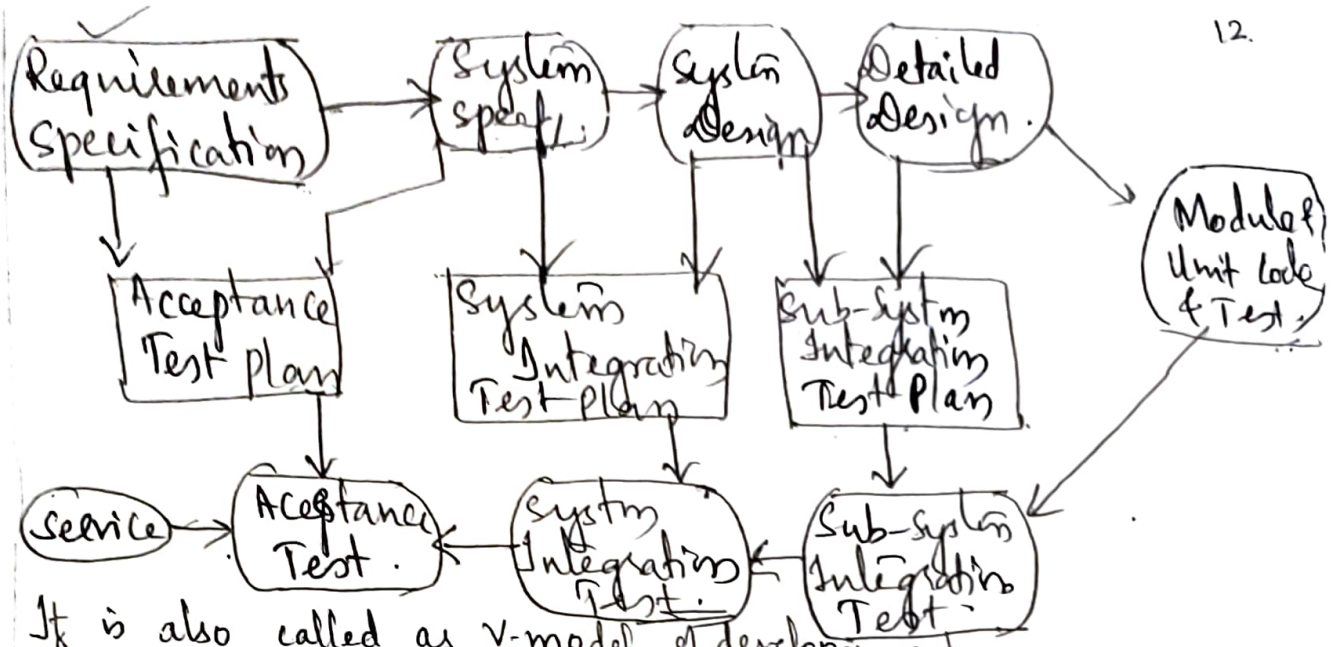
→ When a system is to be marketed as a s/w product, a testing process called 'beta testing' is used. Beta testing involves delivering a system to number of possible customers who agree to use them. They report the problems to the system developer. After this feedback, system is modified & released for further beata testing or for general sale

Ex: Generic product such as microwave, washing m/c.

4) Software evolution (maintenance)

Software is flexible and can change.

- Changes can be made any time during or after the system development.
- S/w evolution (maintenance) some times costlier than the s/w development.
- The below diagram shows where software is continually changed over its lifetime in response to changing requirements &



It is also called as V-model of development.

→ There are two type of testing.
alpha testing and beta testing.

→ Acceptance testing is sometimes called alpha-testing.

→ Custom systems are developed for a single client. α-testing process continues until the system developer & the client agree that the delivered system is an acceptable

Generic products Ex: Customized products such as Library s/w etc

→ When a system is to be marketed as a s/w product, a testing process called 'beta testing' is used. Beta testing involves delivering a system to number of potential customers who agree to use them. They report the problems to the systems developer. After this feedback, the system is modified & released for further beata testing or for general sale

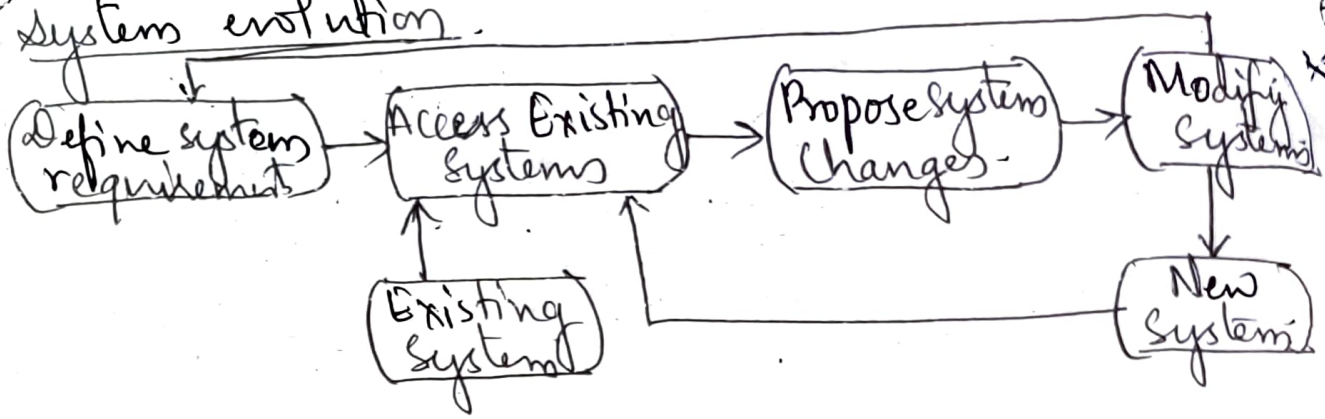
Ex: Generic products such as microwave, washing m/c.

4) Software evolution (maintenance)

Software is flexible and can change.

- Changes can be made any time during or after the systems development.
- S/w evolution (maintenance) some times costlier than the s/w development.
- The below diagram shows where software is continually changed over its lifetime in response to changing requirements &

System evolution.



Copying with change

"Continuing Change" — A system must be continually adapted or it becomes progressively less satisfactory".

→ Change is inevitable in all large s/w projects
- Business process changes lead to new & changed system requirements.
- As new technologies open up, new design & implementation emerge.
- Therefore what ever software process model is used it has to accomodate to the s/w being developed.

→ Change leads to rework so the costs of change include both rework & redesign. ie Change adds to the cost of s/w development.

There are two related approaches that can be used to reduce the cost of rework.

① Change avoidance: where the s/w process includes activities that can anticipate (predict) possible changes before significant rework is required.

for example prototype system may be developed to show some of key features of the system to customers.

(2) Change tolerance: where the change is designed so that changes can be accepted with at relatively low cost.
- This can be done normally through incremental developmental, so that small part of the system may have to be altered to incorporate the change.

Two way of copying with change & changing system requirements
① System Proptotyping ② Incremental delivery.

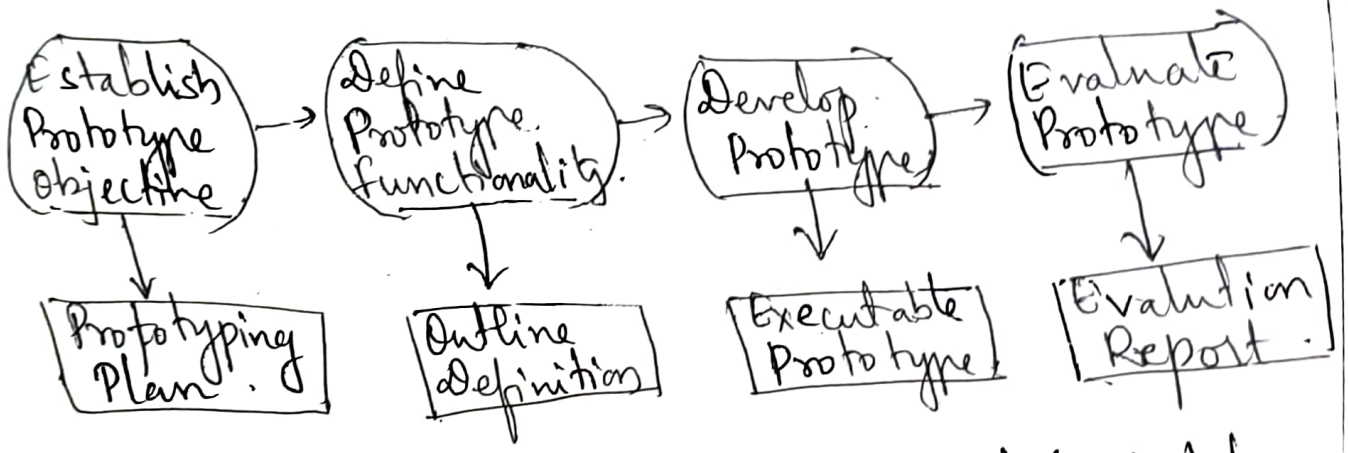
System Proptotyping is where a version of the system or part of system is developed quickly to check the customer's requirements. This will help to avoid supports change avoidance as it allows users to experiment with the system before delivery.

Incremental delivery, where system increments are delivered to the customers for comment & experimentation. This supports both change avoidance and change tolerance.

Prototyping:
A prototype is an initial version of a s/w system that is used to demonstrate concepts, try out design options.

Prototype can be used in

- requirement engineering proces, to help with elicitation (researching & discovering the requirements) & validation of system requirements.
- In design proces, a prototype can be used to explore design options & develop a UI design.



In the above figure a proces model for prototype is developed.

1) The Objective of prototype should be made explicit from start of the proces. This may be to develop a system to validate functional of system req. or to develop system to demonstrate feasib. of application to managers. ∴ one prototype can meet all objective.

2) Define prototype functionality: This stage tells you what functionality to include and what not of the prototype system. for example you may not include non-functional requirements such as memory utilization, response time. This helps to reduce the cost → Error checking & recovery may not be included in the prototype.

3) Develop the prototype: → May be based on Rapid prototyping languages or tools.

Final stage is to evaluate prototype.
Provision must be made for user training
the prototype objective & should be used to
derive a plan for evalution.

Throw-away prototype.

Prototype should be discarded after development as
they are not good basis for a production
system.

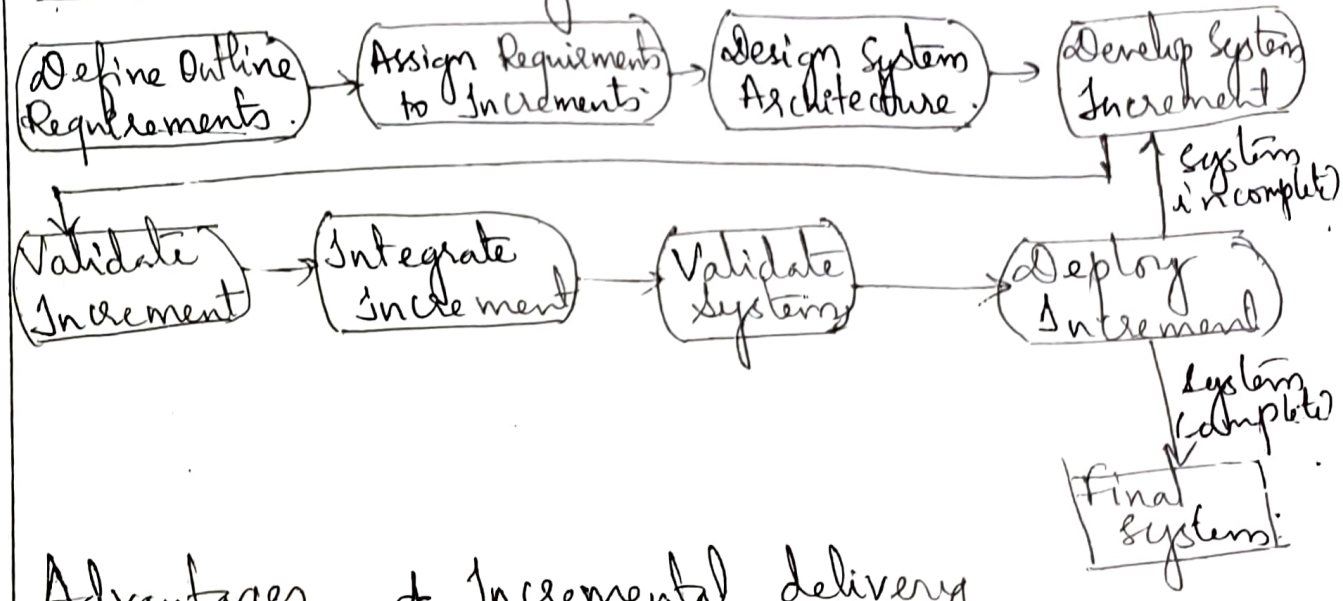
- It may be impossible to tune the system to
meet non-functional requirements such as
performance, security, robustness, & reliability
requirement which were ignored during prototype
development.
- Prototypes are normally undocumented because
rapid change during development.
- The changes made during prototype development
will probably have degraded the system
structure.
- The prototype probably may not meet
organisational quality standards.

Incremental delivery.

- Rather than deliver the system as a
single delivery the development and
delivery is broken down into increments
with each increment.

- User requirements are prioritised and the highest priority requirements are included in early increments.
- The requirements for the service are defined in detail and delivered at first increment. Once the development of a process or increment is started further requirements for latter stages can take place but requirements changes for the current increment are not accepted.

Incremental delivery



Advantages of Incremental delivery.

- Customers can use the early increments as prototypes and gain experience that informs their requirements for later system increments.
- Customers do not have to wait until the entire system is delivered before they can gain value from it. The first increment satisfies their most critical requirements so they can use the s/w immediately.

- Highest priority service are delivered first and increments then integrated so customers are less likely to encounter s/w failures in the most important parts of the system.

Incremental delivery problems.

- Most system require a set of basic facilities that are used by different parts of the system. As requirements are not defined in detail until an increment is implemented, it is hard to identify common facilities the are needed by all increments.
- Iterative development can also be difficult when a replacement system is being developed.

- Case Study refer Text

Boehm's spiral model

- Process is represented as a spiral rather than as a sequence of activities with backtracking.
- Each loop in the spiral represents a phase in the process. Thus, the innermost loop might be concerned with system feasibility, the next loop with requirement definition, next loop with system design & so on
- Spiral model combines change avoidance & change tolerance, it includes risks management activities to reduce risks through out the process

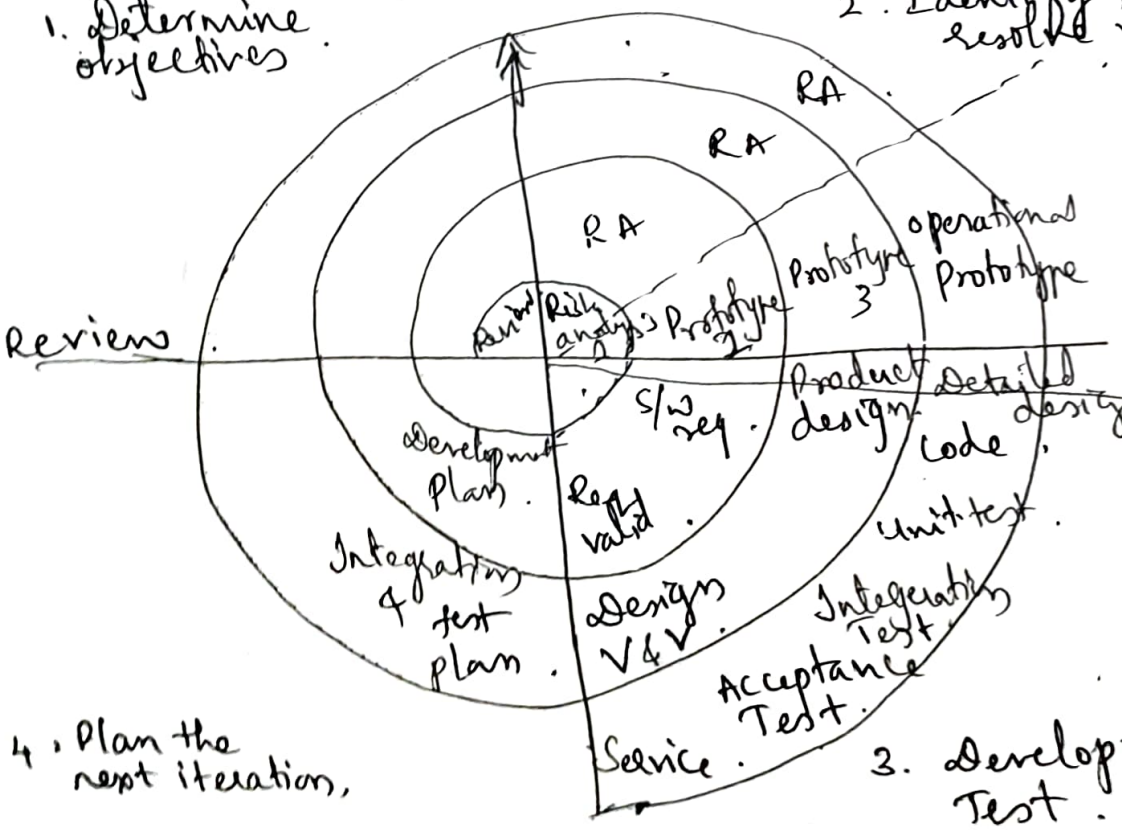
Each loop in the spiral is split into 4 sectors

1. Objective setting: Specific objectives for the phase are identified. Constraints on the process & product are identified & a detailed management plan is drawn up. Project risks are identified & alternative strategies are planned to those risk.
2. Risk assessment (Identify & resolve risk) and reduction: for each of the identified project risks, a detailed analysis is carried out. Steps are taken to reduce the risk. for ex:- if there is a risk at requirement are inappropriate, a prototype may be developed.

At the end of this quadrant, Prototype is built for the best possible solution.

3. Development and validation:- A development model for the system is chosen which can be any of the generic models. The identified features are developed & verified through testing for ex. throwaway prototyping may be the best development process if interface risk are more OR if the identified risk is subsystem integration then waterfall model may be the best model to use.

4. Planning :- The project is reviewed & a decision made whether to continue with a further loop of the spiral.



Advantages of Spiral model

- More emphasis placed on risk analysis hence avoidance of risk is enhanced.
- Good for large & critical projects.
- Additional functionality can be added at a later date.
- Customer satisfaction

Disadvantages of Spiral model.

- Can be costly model to use.
- Risk analysis requires highly specific expertise.
- Project's success is highly dependent on the risk analysis phase.
- Not suitable for smaller projects.

When to use Spiral - Model

- When costs and risk evaluation is important.
- For medium to high risk projects.
- Long term projects.
- Users are unaware of their needs.
- Requirements are complex
- Significant changes are expected (research & exploration).