# Unit - I

DELTA Pg No. Date / /

## Historical Evolution of software engineering

The size of the present software system is increasing tremendously. The major problem of software is that they usually over run their cost or poor quality. Therefore the purpose of software engeneering is to understand the proper strategy practices and development cycles. So that we can product quality products that are within budget and satisfy the requirement of the user.

## Software Crisis –

(i) Schedule and cost estimates were often grossly inaccurate.
(ii) Productivity of programmers could not keep up with demand.
(iii) Poor quality software was produced.

In other words, the software crisis is characterized by an inability to develop software on time, on budget and within requirements. As a result, the delivered software systems are :
(i) Completely unsatisfactory.
(ii) Extremely late.
(iii) For over the budget.



Sign.

(iv) Poor suited for intended users of system.

Reasons of Software Crisis:—

(i) Lack of communication between software developers and users.
(ii) Increase in cost of software compare to hardware.
(iii) Increase in the size of software.
(iv) Increased complexity of program area.
(v) Project management problem.
(vi) Lack of understanding of the problem and its environment.
(vii) High optimistic estimates regarding software development time and cost.
(viii) Duplication of efforts due to absent of automation in the most of the software development.

Real time Scenerio:—

31% of project cancelled before its completion.

There are 94% project restart.

DELTA Pg No.
Date / /

- History of software engineering :-

| Year | Before 1970 | 1971-1980 | 1981-1990 | 1990-date |
|---|---|---|---|---|
| Style | Programming any way | Programming in small scale | Program-ming in large | Mega Program-ming |
| Characteristic Problem | Small Program | Algorithmk Programming | Interface management System Structure | Distributed infor-mation System |
| Data issues | Representing structure and Sym-bolic information | Data Structure and types | long lived data basis Symbolic as well as numeric | Multi-media database |
| Control issues | Elementary under-standing of control Flows | Programs execute once & terminate | Program assemblies execute continually | Concurren-cy and distribu-tion |
| Specification issues | Mnemonics, precise use of prose | Simple I/O specification | System with complex specifi-cation. | Safety Critical Systems |
| State Space | State not well under-stood apart from control | Small, Simple state space | large, struc-tured state space | Enormous |
| Management focus | None | Individual effort | Team efforts | Software Quality Process improve-ment |
| Tool methods | Assemblers | Programming language, Compiler & linker | System maintenance Environments integrated tools, docu-ments | OO Frame-work |

My NoteBook

Sign

Software –

Software is a program along with proper documentation (requirement, testing, coding and design) and installation guide user manuals terms and conditions licence etc.

Program –

Set of instruction is called program which perform some useful task.

Characteristics of software –

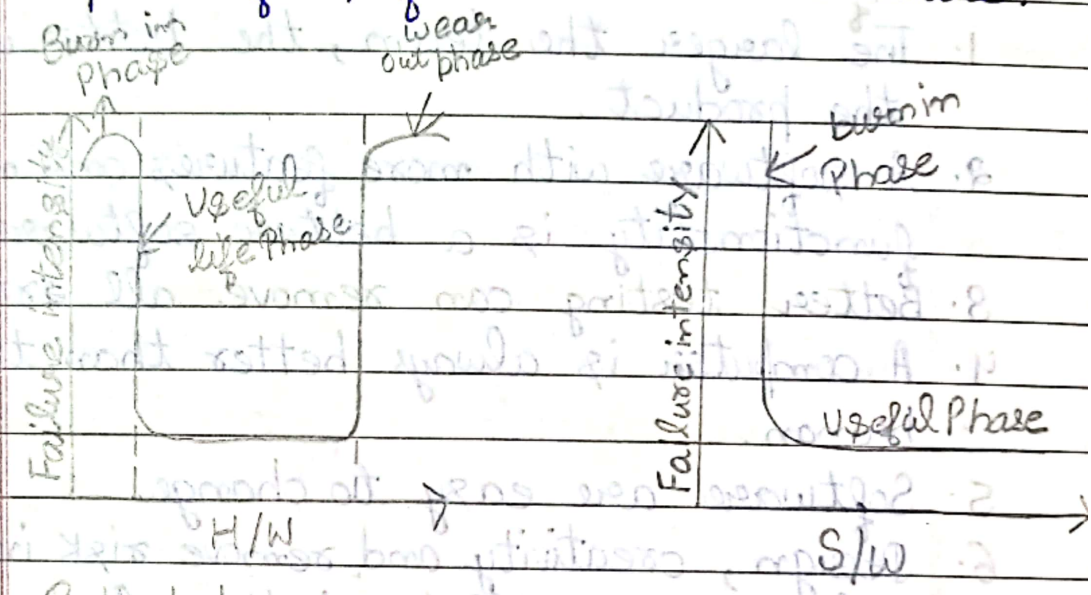
1. Functionality = It means whether software is providing solution to the requirements which ~~were~~ were return in SRS. (Software requirement specifies)

2. User friendly / convenience or usability = It means whether the software is easily comfortable or usable by clients.

3. Efficiency = It means whethers software is providing desire level of functionality of performance by using minimum resources.

4. Maintainability = Software must be maintainable that is either to enhance functionality or to make it adaptable to to some new platform or to remove some errors or bugs.

5. Portability = It must be portable with respect of software and hardware.




Both tub-curve

In hardware, there are 3 phases :-

1. Burn Phase = In this process, where we after proper learning a hardware product will work for approximate amount of time which may vary phase to phase.

2. Wear out Phase = Because wear and tear in hardware parts the failure intensity again increases. There is every time

production cost in hardware. While in software, there is no wear out phase the more we use it the more reliable it will. There are single in software and then maintenance cost continue in the software.

Software with :-

1. The larger the team, the better will be the product.
2. A software with more features and more functionality is a better software.
3. Better testing can remove all errors.
4. A computer is always better than the human.
5. Software are easy to change.
6. Design, creativity and remove risk in this field human will be better always.
7. A larger ~~then~~ team can improve or remove delays in the project.
8. A good mature software company can predict cost and time.
9. Reusing a software with maintenence increase reliability.

Goals :-

1. Realistic
2. Achievable
3. Challangable

Goal must be achievable, challangable also.

Goals must be divided into short term goals and mile stones so that we get important feedback and we can re-align ourself to the correct path.

After goal setting and after proper planning we must ensure we are working with a same plan.

Major problems in current software development

- In proper understanding of users requirement users keep on changing requirement or we don't do proper.
- Multiplicity of software development model, company chooses wrong model for some projects.
- Selections of wrong tools or wrong technology for some project.
- Not enough time and not enough money Most of the time client asked a decent job in sufficient time and cost.
- Wrong motivational technique - sometimes software companies give restricted environment to employ to achieve fall standard or CMM level.

DELTA Pg No.
Date / /

# Software development life cycle

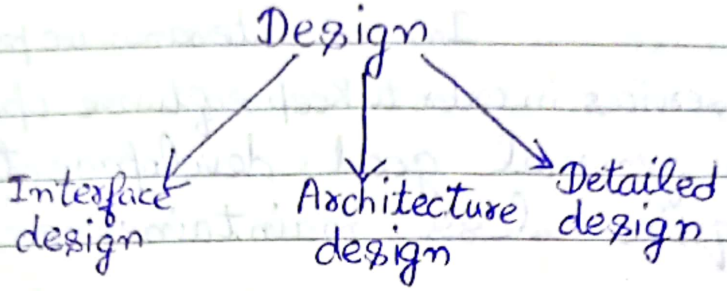
(i) Requirement analysis / feasibility study –

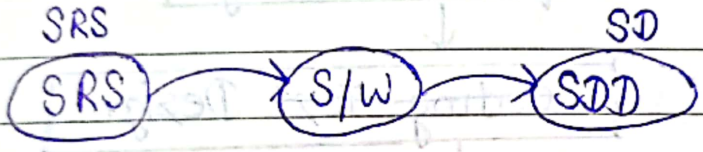
Try to understand what are the exact requirement of customer, then we document them properly. Here we try to find more and more information about every aspect of project. In the last, we will generate a document called SRS (Software requirement specifier) which tell precisely "What it is to be done" without discussing how.
In feasibility, we study financial technically to search an alternative method.

(ii) Design –

In designing phase, input and output in (SDD) software design document.

Here we resolve every technical aspect of the project starting from interface design architectural design to detailed design




(iii) Coding –

Coding is purely technical. In coding we translate a design into particular programming language.

(iv) Testing –

Testing include a error, bug, failar Testing is a process for searching bugs and faults. Testing is never complete so we must have proper schedule time and budget team testing plan etc.

Testing have some limitation where we stop testing.



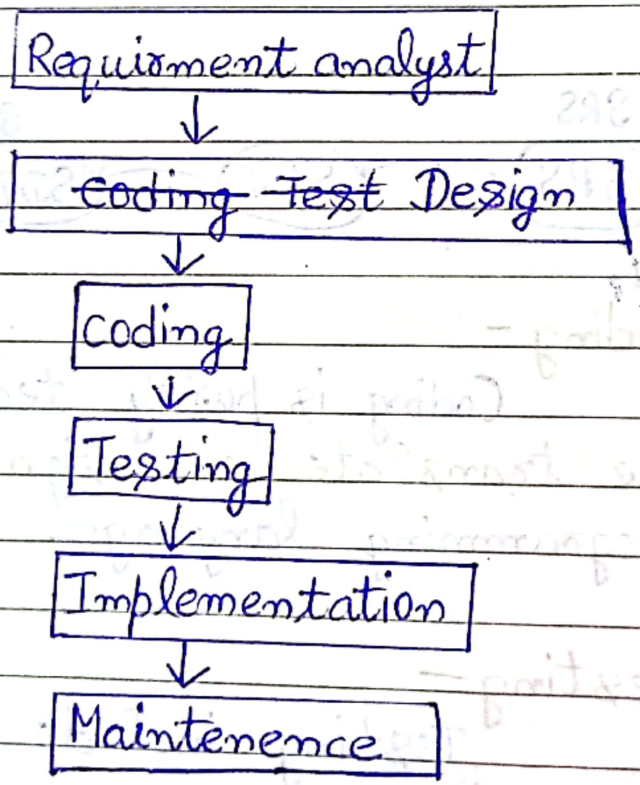
Sign.

(v) Implementation –

A implementationwe takecare of hardware capability of a computer on which software is installed sometime proper stakeholder is required.

(vi) Maintainence –

In maintenance we provide a number of services in order to keep software operational. In general good development will require less maintainence.



# SDLC

Models - The life of a software starts from the movement of concept or idea comes and continue till we maintain software for use. The selection of correct development models plays a very important role cost schedule, quality and overall success of the project. In software model using life cycle model helps us to define each step separately with well define inputs and outputs. It also helps organisation to complete and follow every process maturely.

1. Waterfall model or classical life cycle model = Mathematically it looks like waterfall. It is the oldest SDLC model Therefore it is also known as classical life cycle model.

Advantages :-

(i) It is easy to use.
(ii) Easy to understand.
(iii) Each phase have a well define input and output. So, better clarity.
(iv) It is easy to manage resources as

all the staff on the same time on the same project.

(v) It must be use when requirement are well understood, project are of small size and customer is a friendly to trusted party.

Disadvantages:-

(i) It do not support any itteration or change in the requirement after the first phase.

(ii) High risk is involved as the customer get workable copy in the last phase.

(iii) It should not be used if technology is new, project size is very large or customer is not a trusted party and not friendly.

2. Evolutionary life Cycle model :-

This model accomodates increment development using experience from earlier increment to help define requirement for subsequent increment. This model also ~~helps~~ known as evolutionary prototype. Prototyping/ ~~increaments are~~ Simulation, Rapid Delivery. Increments are developed in a sequential manner rather than parallel

and with in each incremental development cycle. There is a normal progression through analysis, design, coding, test and implementation. Followed operations and maintenance. During the operation and maintenance stage for an increament, feedback from customer ~~make~~ may cause a re-iteration of some steps of development for that increament.

Experience with each finished released is incorporate in requirement for the next development cycle. From the customer point of view, the system will "evolve" as increment that are delivered over time.

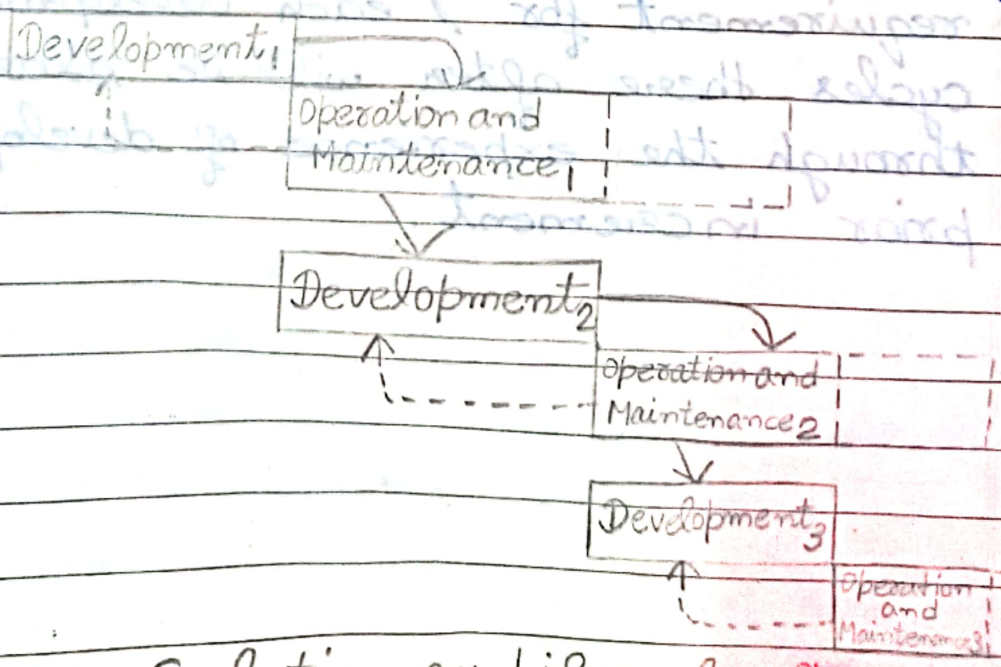
From the developers' point of view, those requirements that are clear at the begining of the project will dictate the initial increment, and the requirement for each development cycles there after will be clarified through the experience of developing prior increment.

## Advantages –

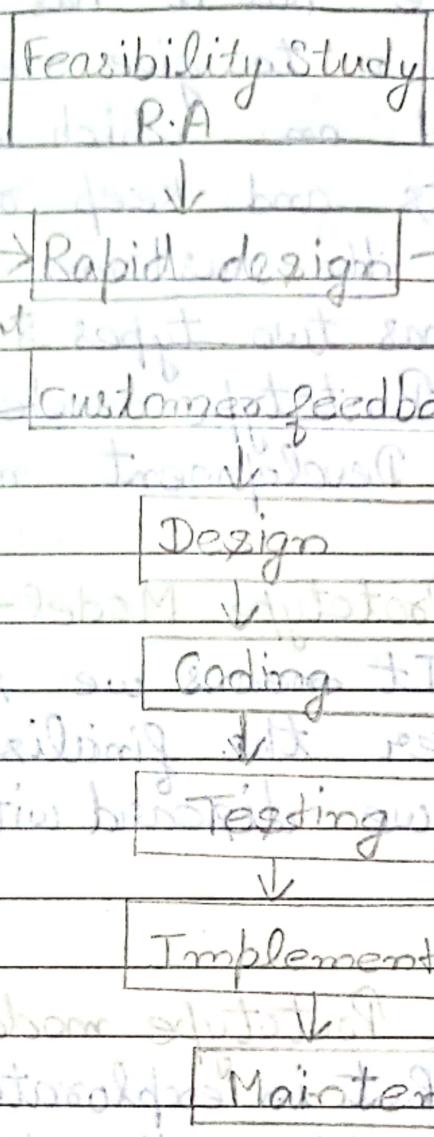
1. Early delivery of portions of the system even though some of the requirements are not yet decided.
2. Use of early releases as tool for requirement elicitation.

## Disadvantage –

1. The overall elapsed time for the project may be longer if the scope and requirements are established before any increments are developed.
2. Difficulty in estimating costs and schedule at the start of the project when scope and requirements haven't been established.
3. Additional time must also be planned for testing and development of project.



3. Prototyping model –




* Major disadvantage of waterfall model :-

Is there is high risk because of requirement but actual problem is nobody comes with the complete set of requirements at the first time. Anti-requirements keep on changing. So, in prototype model, based on requirement analysis we go for a rapid design and then construct a software prototype which looks like a software but it has

minimum functionality with untrusted performance. But it has sufficient to the customer to give necessary feedback based on which we refine our requirements and keep repeating days loop until the customer is satisfy.
Prototyping has two types :-
1. Throwaway Prototype model
2. Exploratory Development model

1. Throwaway Prototype Model-
It means we start from the scratch after the finalising the requirement and we discard with prototype model.

2. Exploratory Prototype model-
It means But in exploratory prototype we keep working in the prototype and keep enhancing till we get the final software.

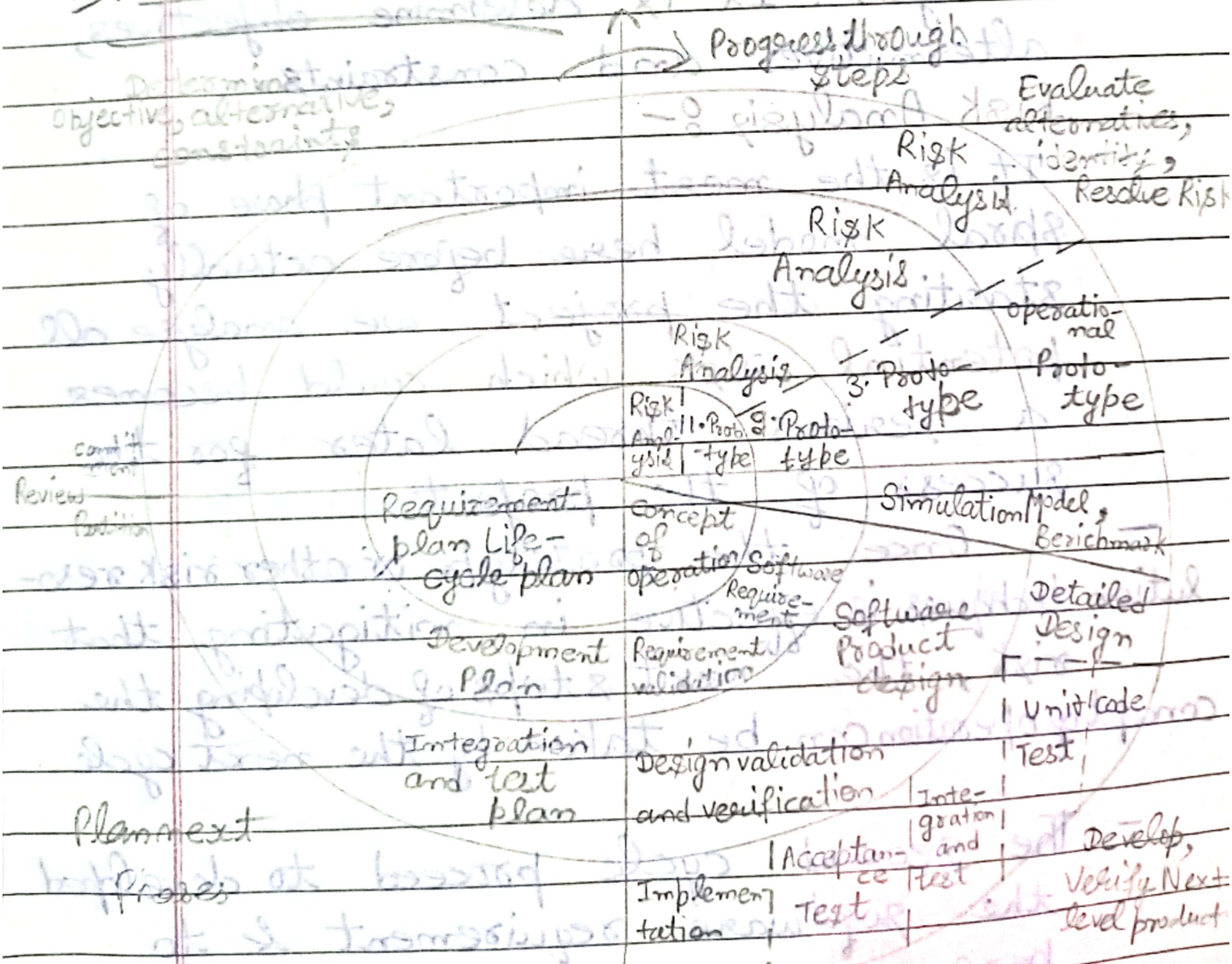
Advantages :-

1. Lesser risk involved as customer has a idea but to aspect.
2. Better customer satisfaction in terms of confidence because customer himself approved the prototype.
3. Better quality software as it suppo changing in requirement only in starting.

Disadvantage :-

1. Loop must be manage properly otherwise we will lead to waste
2. After seeing early prototype customer demand final project
3. Sometime customer or company will use interest or patience in initial one or two loops.

## 4.3. Spiral Model :-



# Spiral Model :-

Spiral model is working on risk management. It is developed by Berry Bohem in 1986. The major consideration of this model is risk management. It is only software development model which is also used for hardware development model. In the first phase, we identify objectives based on the feedback of the customer. It is more or less works like requirement analysis. It is determine objectives, alternatives and constraints.

Risk Analysis :-

It is the most important phase of spiral model here before actually starting the project we analyse all potential risk which could becomes a serious thread later for the success of the project.

- Once the prototype or other risk resolution techiques is effective in mitigating that risk, the next steps of developing the concept of operation can be taken by the next cycle

- The second cycle proceed to developed the software requirement & to perform requirement validation, Again preceeded by planning determination of objectives, alternatives and constraints and risk analysis and mitigation.

-> The third phase, we design, coding and testing. -> In the fourth phase customer feedback for new one. The third and fourth cycle focus on further stage of development as shown in the lower right quadrant of fig. and we preceded by planning through risk mitigation relevant to those stages of development.

Advantage —

1. For the large stage, it provide risk analysis which
2. It actually supports requirement change and it increases customer participation, which ensures better quality with and satisfied customer.
3. The spiral model can be used with a spectuam of methodology whether they are specification oriented, objected oriented, process oriented.

Disadvantage :-

1. It should not be used with small size project.
2. It is difficult to schedule with time & human resource.
3. Total cost of development will be more.
4. Sometimes company plays very

5. The lifecycle steps need further elaboration to ensure that participants in software development have a common understanding.

# Chapter-5

DELTA Pg No. Date / /

* Requirement Engineering

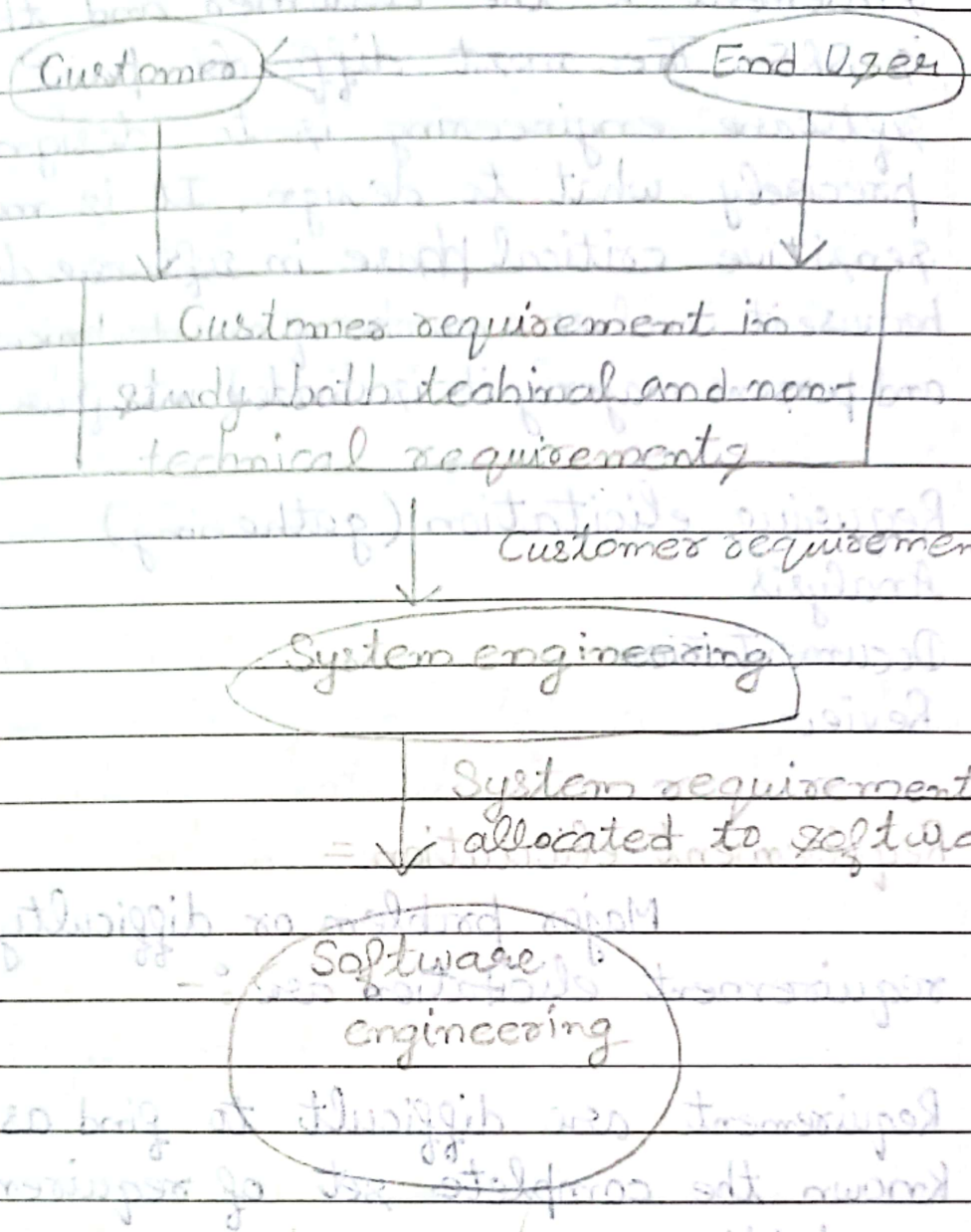
⇒ Requirement engineering is a set of activities that are associated with the elisitation (specification) ^ and management of requirement for a computer based syste ~~system~~. Requirement Engineering can be defined as the systematic process of developing requirement through an interactive co-operative process of analysis a problem documenting the resulting observation in a variety of represen-tation formats and checking the accuracy of the understanding gain.

⇒ Requirement Engineering is the process of founding at the future requirement and associated changes are R.E focuses on What meet to design rather than HOW it can be design.

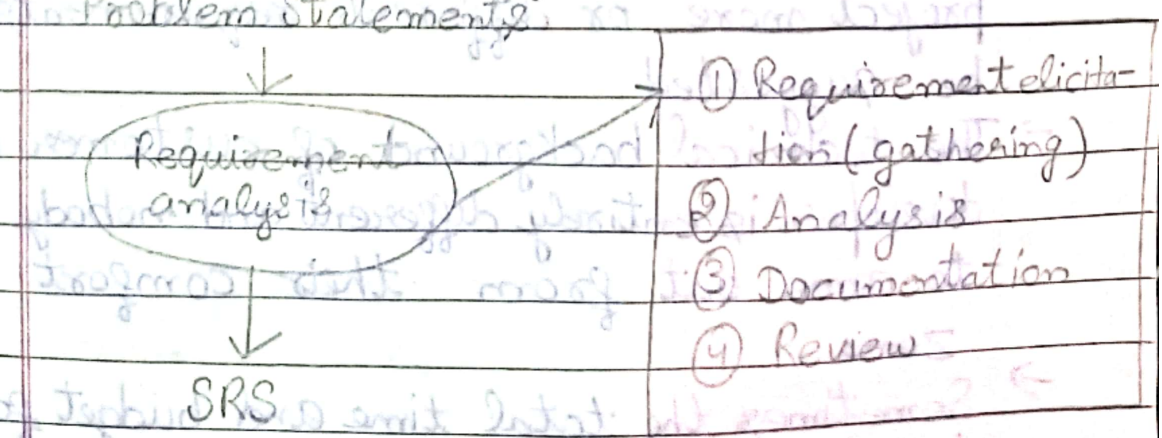
⇒ Requirement Engineering is ~~is~~ not a one time activity but instead should be re-visited at every stage of development process to find out if the requirement have change at all and if not are they being met.

Date / /

## Requirement Analysis :-




## Requirement Analysis :-






Sign.

In requirement analysis input is problem statement & the customer and the output is SRS. The most difficult part of software engineering is to design precisely what to design. It is most sensitive critical phase in software development because it involves a number of non-technical persons and persons majorly it is divided into four step :

1. Requirement elicitation (gathering)
2. Analysis
3. Documentation
4. Review

1. Requirement elicitation =

Major problem or difficulty in requirement elicitation are :-

→ Requirement are difficult to find as nobody known the complete set of requirements in starting.

→ Requirement keep on changing As more the customer understanding the project more or different requirements will be suggested.

→ The technical background of customer and developer is entirely different and nobody wants to come out from their comfort zone.

→ Sometimes the total time and budget for R.E. is so less. Then we cannot find total

set of requirements.

Requirement can be of two types :-
(i) Functional requirements
(ii) Non-functional requirements

Functional requirement - Functional means which are measurable and can go under validation as they are properly documented in SRS.

Non-Functional requirement - Non functional means they are defacto requirements which are not documented but are expected by customer.

Validation - Validation are we making the right product.

Verification - Verification are we building the Product right.

Method of elicitation :-

(i) Interview = Interview is the simplest method of gathering requirement. It is the first step after the problem statement. Here both parties are try to understand each other, requirements and their background for a better fruitful relation in the future.

It could be of two types :—

1. Open ended or unstructured
2. Close ended or structured

1. Open ended or unstructured = No preset question we just try to understand thinking of customer and the subsequent question based the answer of the previous question.

2. Close ended or structured = We have a fixed set of questions and target to have those answers.

Advantage :—

1. Simple
2. Easy to understand
3. Easy to use
4. Good for starting

Dis-advantage—

3. Its not so creative & not so customer friendly.
2. It should be used for small size project

Brainstorming :—

Brainstorming is the type of group discussion which leads to more creative idea very quickly. It is the one of the popular method used in industry now a days where each participants of brainstorming is asked to give his/her ideas about the project but nobody will be criticised for some wrong suggestion.

Advantage —

1. Very Creative
2. Easy to Understand
3. Easy to use.
4. Give better result to comparison to interview in less time.

Disadvantage —

1. Not the professional method. Some time G.D becomes agressive must be properly managed by moderator. People were introward natured may depried ~~in a~~ in brain storming and everybody knows who is saying what so no privacy.

~~3. Del~~

* Delphi method :-

Delphi is like a written brain storming where each participat writes their ideas and views on a piece of paper & this papers are exchanged and then we can edit the received paper by adding something and removing something and this process continue untill we come to conclusion. The proper will become static.

Advantage -

1. More formal.
2. Avoid agressive situation and querrels.
3. Every participants gets equal change to participate with privacy.

Decision

Disadvantage :-

More complex then

* QFD (Quality Funtionality Deployment) :-

In this method scaling is used.
V no. is very important, IV no. is important
III no. is not so important but nice to have,
II not important, I is unrealistic

and require further discussion. In QFD, we ask customer to rate every requirement on a scale of V to I. It give more emphasis on pratical demands and avoid fuzzy discussion.

Advantge —

1. Precise and confined discussion.
2. ON actual practical requirement
3. Should be used with complex and large size project.

Disadvantage

1. Sometime customer may feels pressure & losses creativity.
2. It should not be used for small size project.

* Used case diagram :—

In this approach, it acc. to the discussion of requirement of the user. We make some diagram & charts which narrates the whole story. It helps both party to have a better sense of understanding about the requirement.

Advantage —

1. Graphical & diagramatical representation easy to understand and interpret by human mind.

Disadvantage —

1. It is little complex
2. Time Consuming
3. To manage the activity properly.

Fast (Facilitated application specification technique) :—

In large size projects, everyone works on every requirements we makes small team which works on a specific set of requirements. Here,

1. We understand what is the enviroment which surrounds the system.
2. What is the input and output of system.
3. What are the major functionality required by the system.
4. We divide into small team and in last we merge their reports.

Advantage —

1. It should be used in large company and large size project.



Sign.

Disadvantage -

1. It should not be used with small size project.

2. Requirement Analyst -

In requirement analysis, we try to find mistakes, bugs that are migrated into requirements. It is very important to rectify these problem now only because if these problem subsequent phase then ~~case~~ cost of their correction will be many time. Following are the major tools used for analysis.

1. DFD
2. Control flow diagram
3. Decision tree or table
4. E-R diagrams
5. Data dictionary
6. Class digram
7. Activity chart.

3. Requirement Documentation -

Here we follow internationally accepted standard primary given by IEEE. To write and compile requirement properly. So that they are easily understandable and can be maintained in ~~in~~ iteration version.

4. Requirement Review—

It is very critical to have before we actually designing to find some remaining inconsistencies or mistake ~~and~~. Even review can be of ma types—

1. Formal review
2. Informal review
3. Walk through review
4. Inspection etc.

Requirement management :—

The purpose of requirement management is to establish the common understanding between the custome and the software project of the customer requir That will be address by the software project. Requirement management involves establishing, maintaining on agreement with customer on the requirement for the software project.

SRS Specification :—

The SRS is a blue print for completing the project with as little cost growth as possible. The SRS is also referred as parent document because all subsequent project manageme document such as design specification, design of work, software architecture

Date / /

specification, testing, validation plan and documentation plan are related to it.

Goals of SRS document :-

1. Feedback to customer = SRS document provide feedback to the customer. It is a customer assurance that the development organization understands the issues or problem to be solved and the software behaviour necessary to understand those problem. Therefore, SRS should be written in natural language, in an ambiguous manner that may also include chart, decision table and so on.
2. Problem decomposition :- SRS document decomposes the problem into requirement and well subsequent or component parts. The simple act of writing software design format organises information places borders around the problem break down the problem into its component parts in an orderly fashion.

3. Input to design specification :-

SRS document serves as a input to the design specification. The SRS is also serves as the parent subsequent document such as software designing specification document and statement of work. Therefore the SRS must contains sufficient details in the functional system requirements


Sign.

so that the design solution can be devised.

4. Production validation check :—
SRS document serves as a product validation check. The SRS also serves as the parent document for testing and validation strategy that will be applied to the requirements for verification.

Benefits / uses / characteristics / Quality and Components of SRS.

- SRS document serves as an input to the design specification.
- The SRS also serves as the parent document to subsequent documents, such as the software design specification and statement of work.
- Therefore, the SRS must contain sufficient detail in the functional system requirements so that a design solution can be devised.

*Production Validation Check*

- SRS document serves as a product validation check.
- The SRS also serves as the parent document for testing and validation strategies that will be applied to the requirements for verification.

### 5.7.3 Qualities/Characteristics of a Good SRS Document

The following are the quality characteristics of a good SRS document:

*Complete*

- SRS should be complete.
- SRS defines precisely all the live situations that will be encountered and the system's capability to successfully address them.

*Consistent*

- SRS should be consistent.
- SRS capability functions and performance levels are compatible, and the required quality features (security, reliability, etc.) do not negate those capability functions.

*Accurate*

- SRS precisely defines the system's capability in a real-world environment, as well as how it interfaces and interacts with it.
- This aspect of requirements is a significant problem area for many SRS's.

*Modifiable*

- The logical, hierarchical structure of the SRS should facilitate any necessary modifications and that too with a greater ease.

*Ranked*

- Individual requirements of an SRS are hierarchically arranged according to stability, security, perceived ease/difficulty of implementation, or other parameter that helps in the design of that and subsequent documents.

*Testable*

- SRS must be stated in such a manner that unambiguous assessment criteria can be derived from the SRS itself.

*Traceable*

- Each requirement in SRS must be uniquely identified to a source (e.g. use case, government requirement, industry standard, etc.)

*Unambiguous*

- SRS must contain requirements statements that can be interpreted in one way only i.e., it should be unambiguous.
- This is another area that creates significant problems for SRS development because of the use of natural language.

*Valid*

- A valid SRS is one in which all parties and project participants can understand, analyze, accept, or approve it.
- This is one of the main reasons SRSs are written using natural language.

*Verifiable*

- A verifiable SRS is consistent from one level of abstraction to another.
- Most attributes of a specification are subjective and a conclusive assessment of quality requires a technical review by domain experts.
- Using indicators of strength and weakness provide some evidence that preferred attributes are or are not present.

### 5.7.4 Who Writes SRS Document?

Usually, systems architects and programmers write SRSs with little, if any, help from the technical communications organization. And when that assistance is provided, it's often limited to an edit of the final draft just prior to going out the door.

In short, a requirements-gathering team consisting solely of programmers, product marketers, systems analysts/architects, and a project manager runs the risk of creating a SRS document that may be too heavily loaded with technology-focused or marketing-focused issues.

With the increased complexity of the SRS, it is better to involve the technical writers in the process of writing SRS document. The presence of a technical writer on the team helps place at the core of the project those user or customer requirements that provide more of an overall balance to the design of the SRS, product, and documentation.

### 5.7.5 Benefits of Involving Technical Writers in SRS Writing

Having technical writers involved throughout the entire SRS development process can offer several benefits:

- *Technical writers are skilled information gatherers, ideal for eliciting and articulating customer requirements.* The presence of a technical writer on the requirements-gathering team helps balance the type and amount of information extracted from customers, which can help improve the SRS.
- *Technical writers can better assess and plan documentation projects and better meet customer document needs.* Working on SRSs provides technical writers with an opportunity for learning about customer needs firsthand—early in the product development process.
- *Technical writers know how to determine the questions that are of concern to the user or customer regarding ease of use and usability.* Technical writers can then take that knowledge and apply it not only to the specification and documentation development, but also to user interface development.
- *Technical writers involved early and often in the process, can become an information resource throughout the process, rather than an information gatherer at the end of the process.*

### 5.7.6 Uses of SRS Document

The following are few major uses of SRS documents:

- *Project managers base their plans and estimates of schedule, effort and resources on it.*
- *Development team needs it to develop product.*
- *The testing group needs it to generate test plans based on the described external behaviour.*
- *The maintenance and product support staff need it to understand what the software product is supposed to do.*
- *The publications group writes documentation, manuals, etc. from it.*
- *Customers rely on it to know what product they can expect.*
- *Training personnel can use it to help develop educational material for software product.*

### 5.7.7 Components of SRS Document

SRS development will be a collaborative effort for a particular project and ultimately results into a SRS Document. Several standards organizations including IEEE have identified the following nine components that must be addressed when designing and writing SRS:

1. Interfaces
2. Functional Capabilities
3. Performance Levels
4. Data Structures/Elements
5. Safety
6. Reliability
7. Security/Privacy
8. Quality
9. Constraints and Limitations