Introduction to Software Engineering (SE).

→ SE — composed of 2 words — software & Engineering.

Program is an executable code which serves some computational purpose,

Slw → is collectn of executable programming code, associated libraries & documentations.

Slw product :- slw, when made for a specific requirement is called "slw product".

Engineering : is all about developing products, using well-defined, scientific principles & methods.

So, SE is an engineering branch associated with the development of slw product using well-defined scientific principles, methods & procedures.

→ The outcome of slw engineering is an efficient & reliable slw product.

IEEE defn :- defines SE as:
The applⁿ of a systematic, disciplined, quantifiable approach to the development, operation & maintenance of slw.

→ A small prog can be written without using slw engineering principles. But if any one wants to develope a large slw product then SE principles are absolutely necessary to achieve a good quality slw & cost effective slw.

→ Without using SE principles, it would be difficult to develop large programs.

→ In Industry, in order to handle multiple functions, it is necessary to develop large programs.

→ In such cases, the problem with developing large commercial programs is that complexity & difficulty

levels of the progams increase exponentially with their sizes. Hence, SE helps to reduce this complexity.

→ For this, SE principles used 2 imp techniques to reduce ~~com~~problem complexity namely:

ⓐ Abstraction & ⓑ Decomposition (modularity or modularization)

"Abstraction" → This principle implies that a problem can be simplified by omitting irrelevant details: i.e, consider only relevant aspects & suppress the other aspects of the problem.

→ It is the powerful way of reducing the complexity of the problem.

ⓑ Decomposition :- This is another technique to solve problem complexity.

→ In this technique, complex problem is divided into several smaller problems & then the smaller problems are solved one by one.

→ In this technique, random decomposition of a problem will not help. So the problem has to be decomposed such that each component of the decomposed problem can be solved independently & then the soln of the different components can be combined to get the full soln.

→ A good decomposition of a problem should minimize interactions among various components because, if the diff subcomponents are interrelated, then the different components cannot be solved separately & the desired reduction in complexity will not be realized/achieved.

Need of SE :-

* The need of SE arises because of higher rates of changes in "user requirements" & "environment" on which the slw is working.

→ **Large slw** :- It is easier to build a wall than to a house or building. To build a wall we don't require much any principles of complenity of process but to build a house we require a proper plan, certain principles etc. ~~Likewise, as the~~ Likewise, as the size of the slw become large, engineering has to step to give it a scientific process.

→ **Scalability** :- If the slw process were not based on scientific & engineering concepts, it would be easier to re-create new slw than to ~~create~~ scale an existing one.

→ **Cost** :- As Hlw industry has shown its skills & huge manufacturing has ~~done~~ lowered down the price of comp & electronic Hlw. But the cost of Slw remains high if proper process is not adapted.

→ **Dynamic nature** :-

The always growing & adapting nature of slw hugely depends upon the environment in which the user works. If the nature of the slw is always changing new enhancements need to be done in the existing one. This is where SE plays a good role.

→ **Quality mgmt** :- Better process of slw development provides better & quality slw product.

Chars of good slw :-

→ A slw product can be judged by what it offers & how well it can be used. This slw must ~~satisfy~~ be judged on the following grounds :-

① Operational ② Transitional ③ Maintenance.

Diff b/w words Warranty & Guarantee → service → replacement

sw ← reliable, defect free, quality

Well-engineered & crafted sw is expected to have the fol chars:—

Operational: This tells us how well sw works in operations. It can be measured on:
ⓐ Budget ⓑ Usability ⓒ Efficiency ⓓ Correctness ⓔ Functionality ⓕ Dependability ⓖ security & ⓗ safety

Transitional:— This aspect is important when the sw is moved from one platform to another:
ⓐ Portability ⓑ Interoperability ⓒ Reusability ⓓ Adaptability.

Maintenance:— This aspect briefs ~~be~~ about how well a sw has the capabilities to maintain itself in the ever-changing environment:
ⓐ Modularity ⓑ Maintainability ⓒ flexibility ⓓ Scalability.

⟹ SE is a branch of comp sc, which uses well-defined engineering concepts reqd to produce efficient, durable, scalable, in-budget & on-time sw products.

Unit - I

SE: It is an engineering branch associated with Slw system development.

→ **Diff b/w Computer prog & Slw:-**

* A **computer prog** is a piece of programming code/executable which performs a well-defined task.
* **Computer Slw:-** includes programming code, its documentation & user guide.

→ When you know programming, what is the need to learn Slw engineering concepts?

A) A person who knows how to build a wall may not be good at building an entire house.
Likewise, a person who can write progs may not have knowledge of other concepts of SE.

⇒ Slw ~~engg~~ concepts guide programmers on how to assess requirements of end user, design the algo by actual coding skills, create progs by coding, testing the code & its documentation.

⇒ What is Slw process or SDLC?

SDLC is the systematic development of Slw by following every stage in the development process namely, requirement Gathering, System Analysis, Design, coding Testing, Maintenance & Documentation in that order.

⇒ Several SDLC models are:

1. Waterfall
2. Iterative
3. Spiral
4. V-model
5. Incremental &
6. Iterative model etc.

Unit-1

→ **SE**: It is an engineering branch associated with the development of slw system.

"SE is the technological & managerial discipline process concerned with systematic production & maintenance of slw products that are developed & maintained (ie, modified on time & within the cost estimates)".

⇒ SE involves = Development (of slw products) + their maintenance with in time (specified) & cost (estimated).

Normally, SE involves 3 phases: They are:—

① Definition → non-technical Activity.
② Development
③ Maintenance
→ Technical activities

Definition (of problem stmt) →
- Information Engineering or system engi
- Slw planning
- requirements gathering

Studying / understanding → Diagnosing the problem

Development 40% →
- Analysis & Design:
  - ↳ preparation of blue print.
- Coding.
- Testing → Debugging:
  - → Debugger removes the bugs.
  - ↳ identify the bugs or errors.

Maintenance (giving tech support to the customer) 60% →
- Corrective maintenance (Bug removal)
- Adaptive maintenance
- Enhancement/perfective/preventive maintenance or re-engineering
  - means: It includes additional changes to be done.

(I) Total effort devoted to slw :-

Year	Hlw cost	slw cost.
1960	80%	20%
1980	20%	80%
1990	10%	90%

[slw cost increased boos of slw maintenance & personal costs].

→ In 1990, it is observed as Demand of slw engineering is much greater than the supply of engineers.

1990 → 7,50,000 lacs to 20 lacs.

→ Goal of SE :- is to increase the productivity of SE. i.e, to design tools & techniques to increase the productivity of SE.

(II) Distribution of efforts in the slw life cycle :-

i> slw development

40

later on, 30%. ↓ decreasing.

10%

slw maintenance

60%.

70%, → Ptdead to categorizing our activities in maintenance

90%.

[maintenance

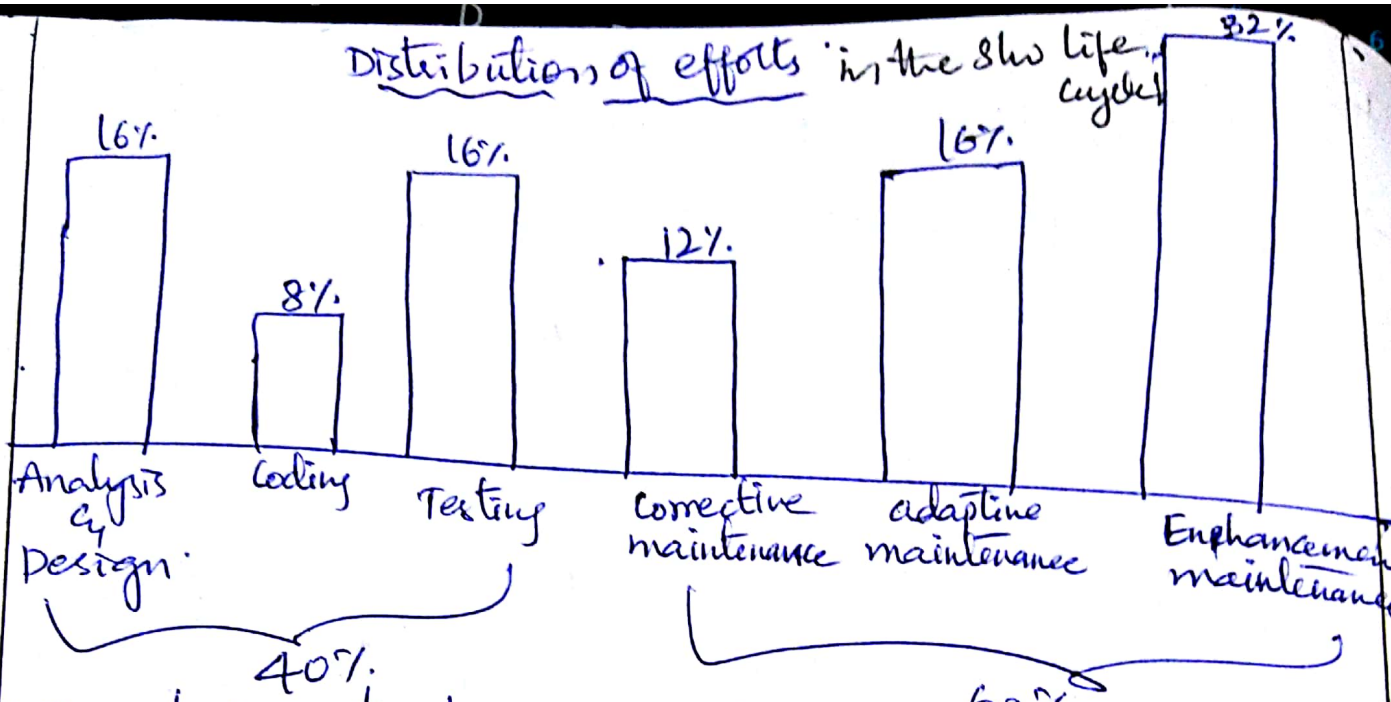
↓ 60% Enhancement of slw

↓ 20% Adaptability of slw.

20% ↓ corrective (correcting bugs)

These are the activities carried out during the slw maintenance. The major activity of maintenance phase is enhancement of the slw.

Distribution of efforts in the Sw life cycle



Sw. Development phase. Sw. maintenance phase.

→ corrective maintenance deals with Bug removal & it is of 2 types. ① Static/proactive bugs removal.
② Dynamic / reactive "

① Removing the known/predictable bugs is known as static/proactive bug removal.

② Removing the unknown bugs (ie, bugs which occur dynamically) is known as Dynamic/reactive bug removal.

* During the maintenance phase it is not possible to remove them completely bcoz they occur then & there dynamically.

Notion of a slw :-

→ slw takes a dual role :
* It is a product.
* It is a vehicle for delivering a product.

→ slw acts as a basis for
① control of the ~~sys~~ computer (ie, OS).
② comm^n of information (N/ws).
③ creation & control of other progs (slw tools & environment).

Evolution of slw :-

Early systems : The first era.
- In 1950–1960,
- Batch orientation
- Limited Distribution.
→ custom slw.
(ie, these slws are custom built).

The 2nd era :- Mid 1960 – late 1970
→ multi-user systems
- Realtime
→ Database
→ product slw.
→ personalized nature of many progs made them virtually un-maintainable

The third era :- 1973–1988.
- Distributed systems.
→ Embedded Intelligence
→ hw cost ↓↓
- Consumer impact on slw product.
- microprocessor generation.

The fourth era :-
- powerful Desktop systems
- obj oriented technologies.
- Expert systems.
- Artificial neural nws.
- ||el computing.
- nw computers.
- Decentralized c/s (concurrent Distributed environments.

Slw characteristics :-

→ Slw is a logical system elt rather than a physical element.

① Slw is developed or engineered, not manufactured in the classical sense. Manufacturing phase for Hlw can produce quality problems.
→ Non-existent (or can be easily corrected) for Slw.

② Software doesn't wear out (failure) but becomes outdated.

* → Hlw exhibits relatively high failure rates early in its life. (Defects may be because of design or manufacturing defects). Hlw begins to wear out after sometime.

* → Undelected defects will cause a high failure rates early in the life of a program. However, once they are corrected the ~~curve flattens~~ failure rate decreases.

* → Slw doesn't wear out, but it does Deteriorate (out date) (reduce in quality).

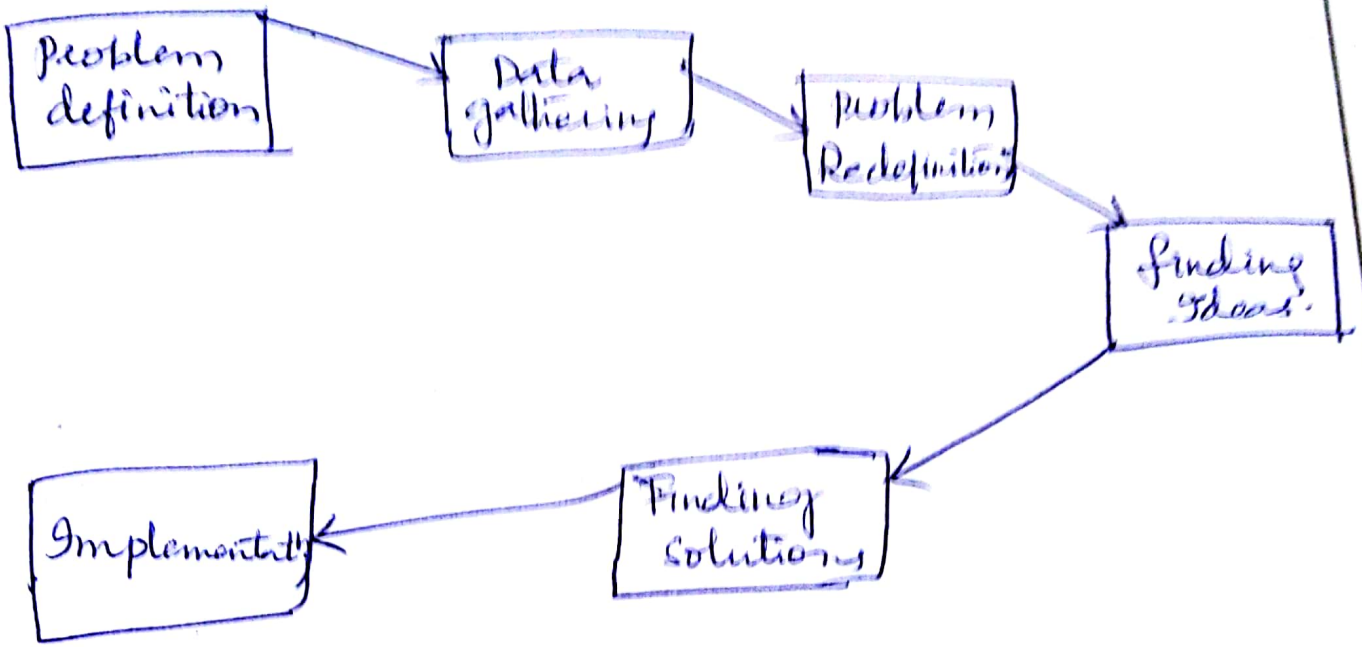
③ Most slw is ~~custom~~ custom built rather than being assembled from existing components.

→ catalogs are available for assembling the Hlw components. No catalogs of Slw components.

Diff bln bug & Defect.

Bug.	Defect.
→ It is an error & is passive which doesn't harm.	It is an ~~Active~~ error which harms the process/product if it is not corrected. ⇒ If it is corrected it may lead to failure of the system or Slw product.

General problem solving model (adapted from Hicks, 1991):



Project life cycles / ~~life cycle models of project~~ / development :

→ Subdividing the process of sw development produces what is known as a life cycle model.

→ Various project life cycles models can be applied to computerised / Information system / sw development.

→ Infact 2 activities precede the info system's development project, strategic info system's planning & business modelling.

→ The successful completion of these activities should ensure that the info system that is developed is appropriate to the organization.

→ Diff b/n sw development & systems development projects—
"sw development project" is, by defn focused solely on producing a sw system that will satisfy the user requirements where as a "systems development project" has a wider scope & may not even include sw as a part of the solution.

Strategic Information systems planning :—

→ Info systems work within the context of an organisation & must satisfy the current requirements as well as providing a basis from which future needs can be addressed.

→ In order to do this, strategic plans are developed for the organisation as a whole & within their content a strategic view of information systems needs can be formed.

Eg :- Agate case study a strategic decision may be made to target multinational companies for international advertising campaign. This has consequences for campaign mgmt & its supporting info systems.

Business modelling:-

→ In order to determine ① how an info system can support a particular business activity, it is imp to understand ② how the activity is performed ③ how it contributes to the objectives of the organization.

→ Campaign mgmt is an important business fn for Agate & it should be modelled in order to determine how it is carried out, thus providing some of the parameters for subsequent information systems development.

~~General~~

Life cycle models :- SDLC (S/w Development life cycle models)

→ A s/w life cycle model (also called process model) is a descriptive & diagramatic representation of "s/w Development life cycle"

→ A "life cycle model" represents all the activities required to make a s/w product & all the stages that passes through its during its development.

→ It also captures the order in which these activities performed on a s/w product from its inception to retirement.

→ Different life cycle models map the basic development activities to phases in different ways.

→ Thus, no matter which life cycle model is followed, the basic activities are included in all life cycle models although the activities may be carried out in diff orders in different life cycle models. During any life cycle phase, more than one activity may also be carried out.

The need for a s/w development Life cycle models :- SDLC models :-

SDLC (Software Development life cycle):

→ SDLC is a process used by sw industry to design, develop & test high quality sws. SDLC also called as "sw development process".

→ Different types of projects have different requirements. Therefore it may be required to choose the SDLC phase (life cycle models) according to the specific needs of the project. Types of sw developing life cycles:

1. Waterfall model
2. Prototyping "
3. RAD "
4. Incremental "
5. Spiral "
6. V-shaped "
7. Evolutionary prototyping model
8. Iterative & incremental "
9. Extreme programming (Agile Development)

S/w Developement life cycle models :- (Life cycle models are also known as process models / Project of s/w ~~life cycle~~ models.

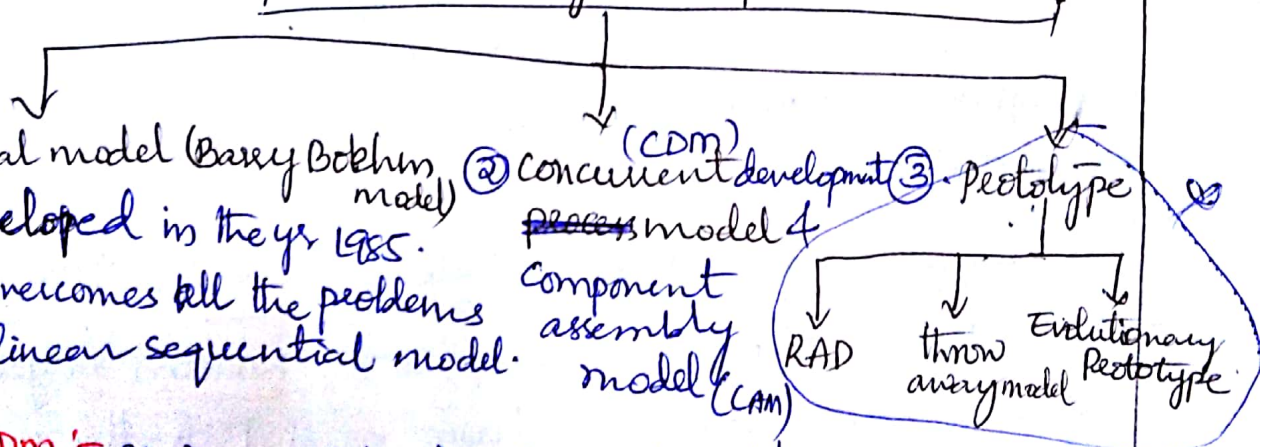
→ A process model for SE is chosen based on the nature of the project & appln, the methods & tools to be used, & the controls & deliverables that are required.

The various life cycle models are :-

1. Linear sequential / classical (Tradilional) life cycle / waterfall model.
2. Prototype model.
3. RAD model.
4. Incremental models.

In addition to these models, there are some models ~~evolv~~ known as evolutionary s/w process models.

→ The various evolutionary s/w process models are-



(2) ~~R~~ DM :- It is used to develope client/server systems s/ws. used for simultaneous developments.

There are 2 types of concurrent client/server systems.

concurrent C/S systems.

- loosely coupled systems (or) Distributed systems (or) Decentralized systems. (i.e, independent systems).
- tightly coupled systems / ||el systems / centralized (i.e, master slave concept) (Resource sharing takesplace)

RAD (Rapid Appln development) :- used only for appln development & takes time only 1-3 months.

Throw away model :- is used by Infosys. In this, one prototype is discarded & another new prototype is used.

Component based model (OO (object oriented) Technology) :-

→ It facilitates reusability.

① object oriented tech :-

→ These are 3 types of components used.

* OSC — Off shelf components.
* Fully experienced components.
* Partially " "

② Formal methods : (Transformational methods) :-

It is a spec driven approach. In this verification is easier, and defect free.

③ RUP (Rational Unified process) Approach / model :-

It follows diff phases / models like "inception", "elaboration", "construction" & Transition.

Linear sequential / waterfall model :- (1970).

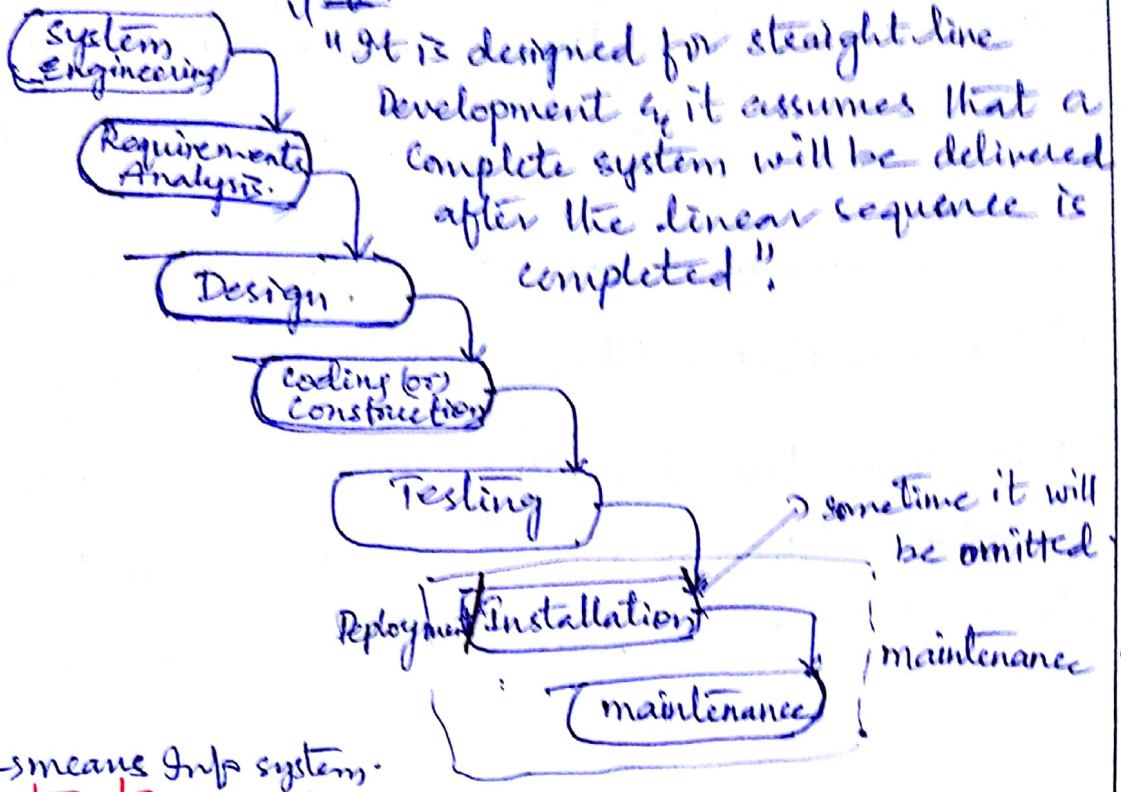
→ It is the first process model developed by "winston Royce". So, it is also called as Royce model ~~in the year 1970~~

→ It is also known as "waterfall model" / "linear sequential model" or "classical (Traditional) life cycle model" or "Royce model" or "Document driven approach".

→ It suggests a systematic, sequential approach to sw development that begins at the system ① level & progresses through analysis ②, design ③, coding ④, testing ⑤ and maintenance ⑥ (6 phases).

The diag to show diff Phases in Waterfall model:

Def":
" It is designed for straight line Development & it assumes that a complete system will be delivered after the linear sequence is completed ".



① System/Information Engineering and modelling :-

→ System Engineering ~~encompasses~~ includes requirements gathering at the system level with a small amount of top-level analysis & design.

→ Information Engineering encompasses (includes) requirements gathering at the strategic business level & at the business area level.

② Slw Requirements Analysis :-

→ To understand the nature of the programs to be built, the slw engineer must understand the information domain for the slw as well as the required function, behavior, performance & Interfacing.

→ Requirements for both the system & the slw are documented & reviewed with the customer.

Eg:- Abstract of my project ~~title~~

③ Design :-

→ S/w design is actually a multi-step process.

→ Focuses on the following 4 distinct attributes of a program.

① Data Structure ; ② Software Architecture

③ Interface Representations ④ Procedural (Algorithmic) detail

→ This phase translates the requirements into a representation of the s/w, which can be assessed for quality before code generation begins.

④ Coding / code generation / construction phase :-

→ In this phase, design must be translated into a machine readable form.

→ If design is performed in a detailed manner, code generation can be accomplished mechanically (e

⑤ Testing :-

→ Testing process focuses on the logical internals of the s/w, assuring that all statements have been tested & on the functional externals i.e. for conducting tests to uncover (find out) (check / discover) errors and ensure that defined i/p will produce actual results that agree with required results.

⑥ Maintenance :-

S/w will undoubtedly undergo change after it is delivered to the customer. ~~S/w~~

→ S/w maintenance reapplies each of the preceding phases to an existing program rather than a new one.

· Problems :-

→ Linear Sequential model is the oldest & the most widely used paradigm for SE. Some of the problems encountered when the linear sequential model is applied are :-

① Real projects rarely follow the sequential flow that the model proposes.

② It is often difficult for the customer to state all requirements explicitly at the beginning of the project because of natural uncertainties.

③ Customer must have patience. A major blunder, if undetected until the working program is reviewed, can be disastrous.

④ Developers are often delayed unnecessarily. In some projects team members must wait for other members to complete dependent tasks. The time spent waiting can exceed the time spent on productive work.

Advantages

→ Each of these problems is real. However, classic life cycle paradigm has a definite & important place in SE.

→ It provides a template into which methods for analysis, design, coding, testing & maintenance can be placed.

Drawbacks :-

→ No versions
→ No iteration
→ Requirements in prior.
→ Invisible performance & Interface.
↳ bcoz at each & every stages, its related activities are well-defined.

Disadvs :-

→ Assume that all requirement can be flagged no further development.
→ very difficult to go back at any stage after it is finished.
→ costly & required more time

Advs :-

→ Easy to explain to the users.
→ Stages & activities are well-defined.
→ Verification at each stage ensure early detection of errors.

→ "It is designed to assist the customer (or developer) in understanding requirements. In general, it is not designed to deliver a product (Production system)

Prototyping model :- (produces trial version, dummy products)
↳ it is used with systems which needs more user interaction

→ Often, a customer defines a set of general objectives (vague idea) for slw but does not identify detailed i/p, processing or o/p requirements.

→ In other cases, the developer may be unsure of the efficiency of an alg, the adaptability of an OS or the form that human-machine interaction should take.

→ In these & many other situations, a prototyping paradigm may offer the best approach.

① The prototyping paradigm begins with requirements gathering. Developer & customer meet & define the overall objectives for the slw, identify whatever requirements are known & outline areas where further definition is mandatory.

② A quick design then occurs. Quick design focuses on representation of those aspects of the slw that will be visible to the customer/user (input approaches, o/p formats).

③ Quick design leads to the construction of a prototype. The prototype is evaluated by the customer/user & is used to refine requirements for the slw to be developed.

④ Iteration occurs as the prototype is tuned to satisfy the needs of the customer, at the same time enabling the developer to better understand what needs to be done

⇒ The prototype serves as a mechanism for identifying sw requirements.

⇒ If a working prototype is built the developer attempts to make use of existing program fragments or applies tools (Example – Report generator, window managers, etc.) that enable working programs to be generated quickly.

⇒ It is true that customers & developers like the prototyping paradigm.

⇒ Users get the feel of the actual system & developers get to build something immediately.

Problems :-

⇒ Yet prototyping can also be problematic for the following reasons.

① Customers sees what appears to be a working version of the sw, haven't considered overall sw quality or long-term maintainability and

→ when informed to rebuild to maintain high levels of quality, the customer cries foul.

② Developer often makes implementation compromises in order to get a prototype working quickly.

→ The less-than-ideal choice has now become an integral part of the system.

→ Although problems can occur, prototyping can be effective paradigm for SE. Key is to define the rules of the game at the beginning; i.e, the customer & developer must both agree that the prototype is built to serve as a mechanism for defining requirements.

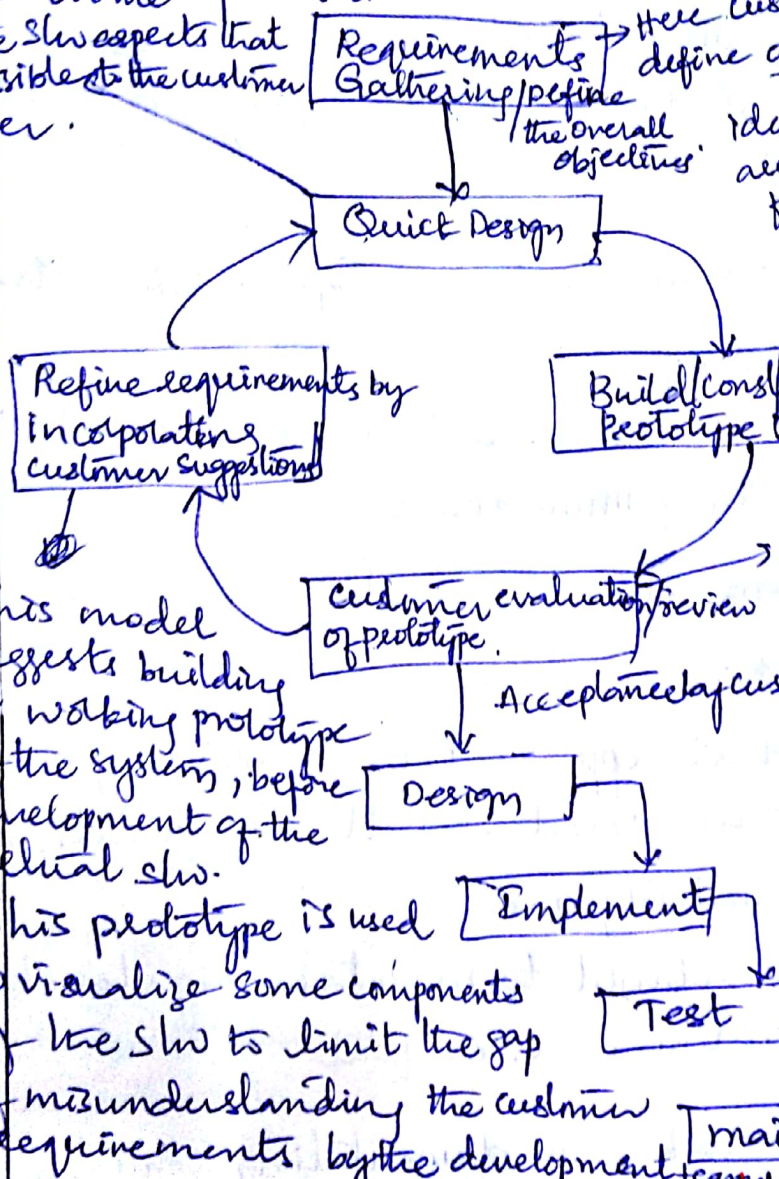
→ It is then discarded (at least in part), & the actual

Slw is engineered with an eye towards quality & maintainability.

→ It serves as a mechanism for identifying the slw requirements.

Diagram to show prototyping model :-

Focuses on the representation of the slw aspects that are visible to the customer or user.



→ This model suggests building a working prototype of the system, before development of the actual slw.

→ This prototype is used to visualize some components of the slw to limit the gap of misunderstanding the customer requirements by the development team.

→ When the final prototype is developed the requirement is considered to be frozen.

Types of prototyping model :-

1. Throw away model
2. Evolutionary prototyping model
3. Incremental " "
4. Extreme prototyping " "

→

Advs :- Improves or increases user involvement

Disadvs :-

→ User confuses between prototype & a finished system.

→ Excessive development of the prototype

→ Implementation of prototyping is expensive.

RAD model :-

→ Rapid Appln Development (RAD) is a linear sequential sw development process model that emphasizes on extremely short development cycle.

→ RAD is a "high speed" adaptation of the linear sequential model in which rapid development is achieved by using a component-based construction Approach.

→ If requirements are well understood & project scope is constrained (conditioned), the RAD process enables a development team to create a "Fully Functional system" within a very short time periods.

→ Used primarily for Information systems applns, RAD approach encompasses (includes) the following phases:

① **Business modeling:-**

Information flow among business fns is modeled in a way that answers the following questions:

ⓐ what information drives the business process?.
ⓑ what " is generated?.
ⓒ who generates it?.
ⓓ where does the information go?.
ⓔ who process it?, etc...

② **Data Modeling:-**

Information flow defined as a part of the business modeling is refined into a set of data objects that are needed to support the business.

→ The characteristics of each object are identified & the relnships bw these objs are defined.

③ Process modeling :-

→ The data objects defined in the data modeling phase are transformed to achieve the information flow necessary to implement a business function.

→ Processing descriptions for adding, modifying, deleting or retrieving a data object is created.

④ Application Generation :- Rather than creating s/w using conventional 3rd generation programming langs.

→ RAD process works to reuse existing program components (when possible) or create reusable components.

→ In all cases, automated tools are used to facilitate construction of the s/w.

⑤ Testing & Turnover :- Since RAD process emphasizes reuse, many of the program components have already been tested. This reduces overall testing time

Drawbacks of RAD :-

→ Like all process models, RAD approach has drawbacks.

① For large, but scalable projects, RAD requires sufficient human resources to create the right number of RAD teams.

② RAD requires developers & customers who are committed to the rapid-fire activities necessary to complete a system in a much abbreviated time frame.

→ If commitment is lacking from them, RAD projects will fail.

Note :-
Not all types of applns are appropriate for RAD. RAD is not appropriate when technical risks are high.

Incremental model :- (Pilot Approach).
follows linear sequential model + prototyping models.

→ It combines the elts of the linear sequential model with the iterative philosophy of prototyping.
→ When the incremental model is used, the first increment is often a "Core product".
→ Basic requirements are addressed, but many supplementary (additional) features remain undelivered. The core product is used by the customer (or undergoes detailed review).
→ As a result of use and/or evaluation, a plan is developed for the next increment.
→ The plan addresses the modification of the core product to better meet the needs of the customer & the delivery of additional features & functionality.
→ This process is repeated following the delivery of each increment, until the complete product is produced.
Advantages: Difference b/w Incremental & Prototyping:
→ ① Incremental process model, like prototyping is iterative in nature. But unlike prototyping, the incremental model focuses on the delivery of a operational product with each increment.

Advantages :-
→ ① Incremental model is particularly useful when staffing is unavailable for a complete implementation within the business deadline.
② Early increments can be implemented with fewer people.
③ If core product is well received, then additional staff

can be planned to manage technical risks.

→ High level interaction is reqd bw customer & developer

Evolutionary Software process models :-

→ Evolutionary nature of the slw is not considered in linear sequential ~~prototyping~~ prototyping) the classic slw engineering paradigms.

→ Evolutionary models "are iterative". They are characterized in a manner that enables slw engineers to develop increasingly more complete versions of the slw.

→ Incremental model is also comes under evolutionary process model ~~as it is~~ because of its iterative in nature

Spiral model :- iterative nature of prototyping + linear sequential model
↳ W·F+PM.

→ It is an evolutionary Sw process model that couples the iterative nature of prototyping with the controlled & systematic aspects of the linear sequential model.
→ Sw is developed in a series of incremental releases.
→ During early versions, the incremental release might be a paper model or prototype.
→ During later versions, increasingly more complete versions of the engineered system are produced.

→ Spiral model is divided into a number of framework activities also called "task regions". Typically there are between 3-6 ~~so~~ task regions.

~~Spiral model :-~~
⇒ It includes systematic approach of Waterfall model
& iterative approach of prototyping model (PM) i.e, WF+PM.

Diag.

Adv: Imp issues are discovered early.
→ Early involvement of developers.
→ manages risks & develops the system in phase.

→ Spiral model is implemented in the case of large projects.

→ The control in the spiral model is in a clockwise direction

→ This model starts with the "core spiral" where the requirements are gathered & the last spiral is the production system.

→ The various task region are described as follows :—

(a) customer communication : Tasks required establishing effective comm" b/n customer & developer.

(b) Planning :- Tasks required for defining resources, timelines & other project related informations.

(c) Risk Analysis :- Tasks required to assess both technical & management risks.

(d) Engineering :- Tasks required for building one or more representations of the applⁿ.

(e) Construction & release :—
Tasks required constructing, testing, installing & providing user support (documentation & training).

(f) Customer evaluation :- Tasks required for obtaining customer feedback based on evaluation of the SW representations created during the engineering stage & implemented during the installation stage.

→ Each of these regions is populated by a series of work tasks. For small projects, the number of work tasks & their formality is low.

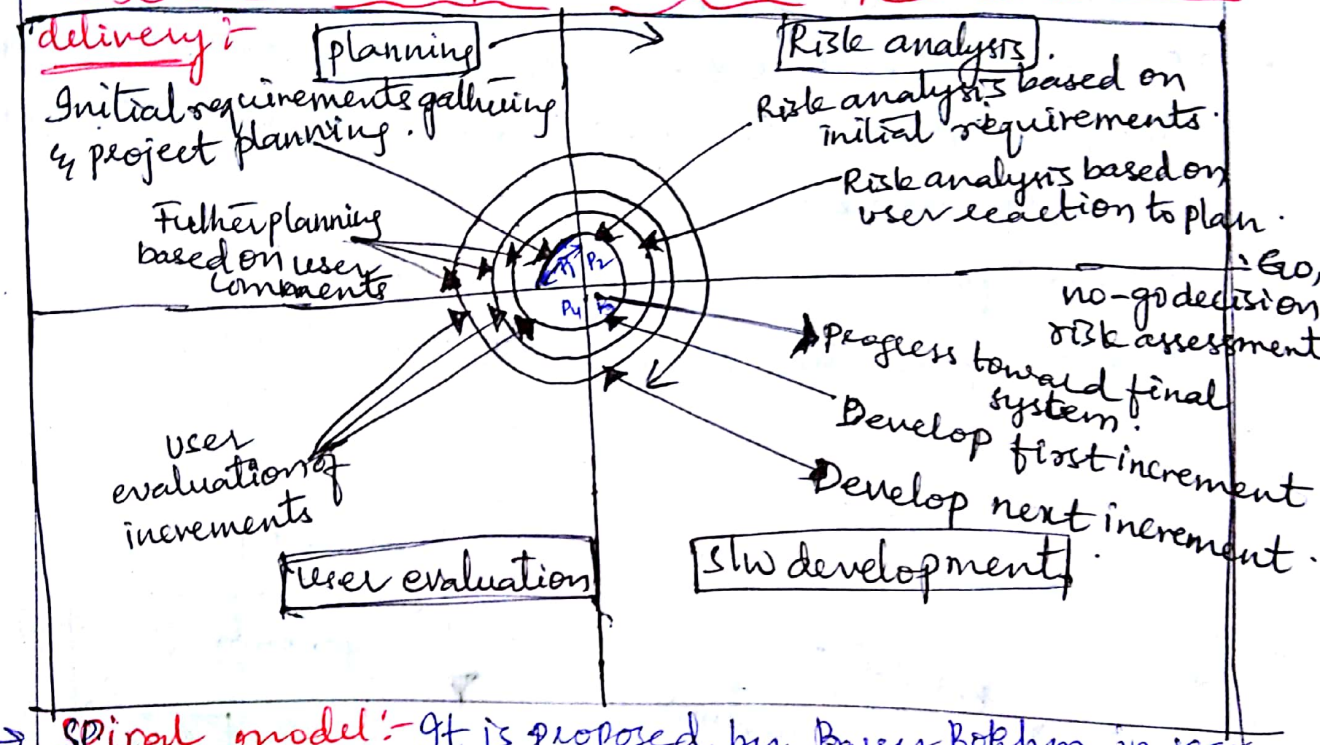
→ For larger, more critical projects, each task region contains more work tasks that are defined to achieve a higher level of formality.

Drawbacks / Disadvs :-

→ It requires expertise to carryon the risk analysis.

→ It is not frequently used process model because of the risk associated in absence of expertise.

Diag to show the spiral model to support Incremental delivery :-



→ Spiral model :- It is proposed by Barry Boehm in 1985.

→ It's a risk-driven Sw process framework.

→ It is represented as a spiral rather than a sequence of activities with some backtracking from one activity to another.

→ Each loop in the spiral represents a phase of the Sw process.

→ Thus, the innermost loop might be concerned with system feasibility, the next loop with requirements definition, the next loop with system design & so on.

→ It combines change avoidance with change tolerance.

→ Changes are a result of project risks to reduce the risks.

→ It includes risk management activities to reduce the these risks.

→ Advs :- * early detection of error or issues.
* early involvement of developers.
* Manages the risks & develops a phased system ~~Note phases~~

Component Assembly model (CAM):-

→ CAM is described based on object oriented technologies.

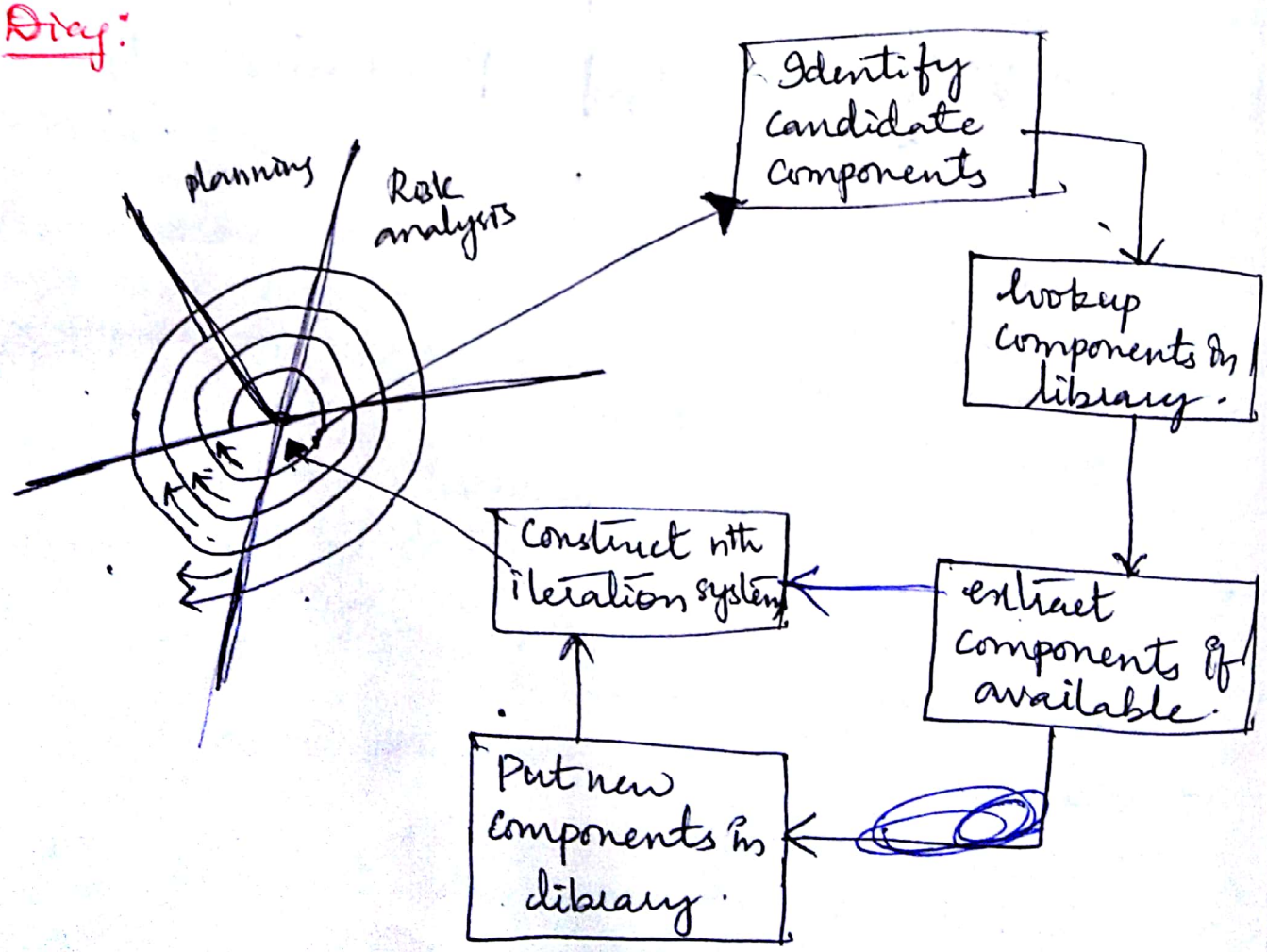
→ The obj oriented paradigm includes essential features of binding data & its member functions (relative functions) together (i.e, encapsulation).

→ CAM composes applns from free packed s/w components (library components).

→ Obj oriented classes are reusable across diff applns & system Architectures. Engineering in this process model is very simple.

* Identifying the candidate components.
* lookup for the components in Library.

Diag:



→ If the components are existing in the library extract the components & move ahead to construct the iteration.

→ If candidate components are not available then ~~the~~ components are built & added to the components in the library & then pass it to the construction iteration system.

→ Collect all the components & forward them to engineering region where they are installed & tested (user support is also provided).

Advs :-

→ CAM provides s/w reuse.

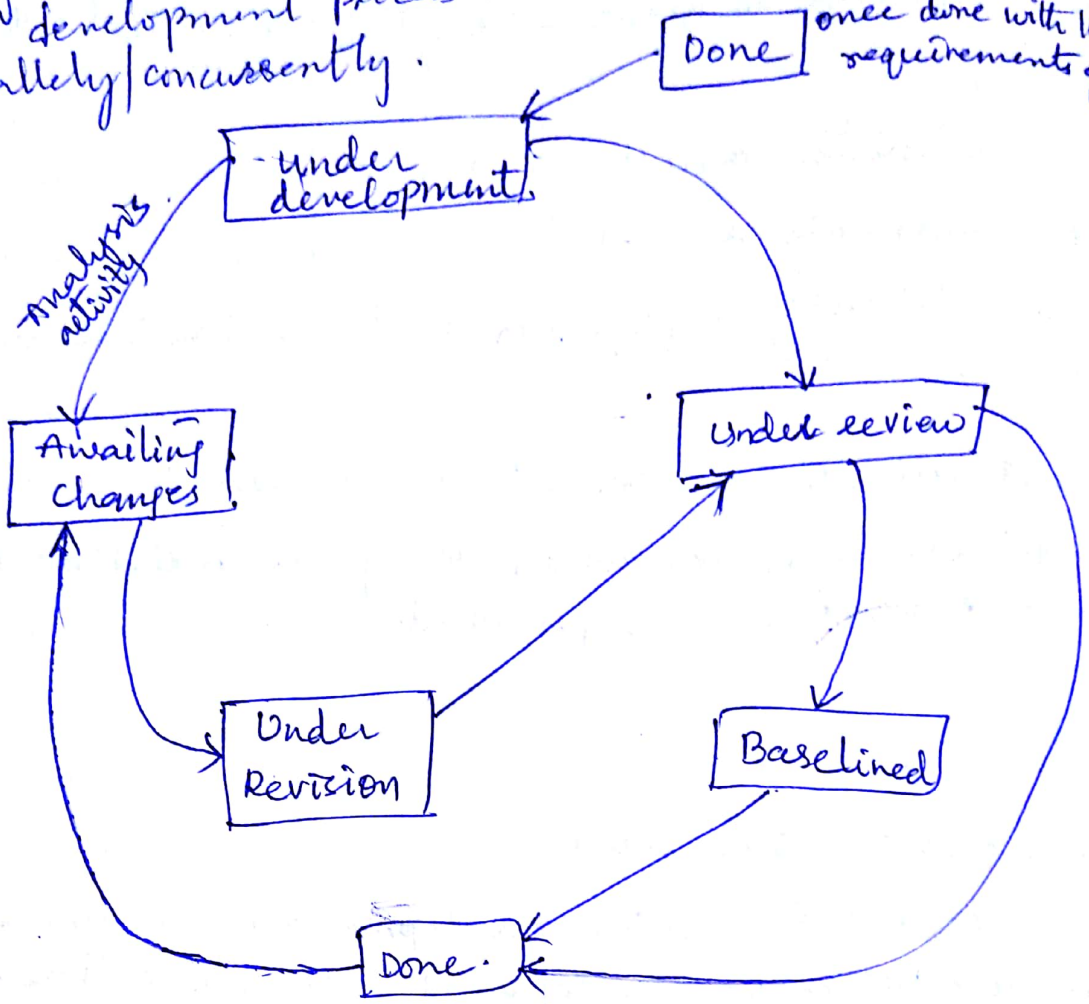
→ 70% reduction in the development cycle time ie, it reduces 70% of the development cycle/phase time.

→ 84% reduction in project cost. (ie, ~~It~~ reduces 84% of the project cost).

→ 26.2% productivity index & industry norm 16.9%. (ie, it increases the productivity index up to 26.2% & reaches 16.9% of industry norms).

Concurrent development model (CDM) :-

Slw development process is carried on Parallely/concurrently.



→ From the above fig, the analysis activity can be in any of the states described as in the above fig.

→ The concurrent process model defines a series of activities or series that will trigger transitions from state to state for each of the slw engineering activities.

** Initially the activity begins at analysis phase & ends at maintenance phase.

* The current activity is always related to the o/p of the previous stage/phase.

** Mostly applicable in client/server environment.

** In reality concurrent process model is applicable to all types of s/w development & provides an "accurate picture" of the current state of the project.

** Rather than confining s/w engineering activities to a sequence of events, it defines a n/w of activities.

Process models (slw developing life cycles).

- Linear sequential model (or) Classical life cycle model (or) waterfall model (or) Royce model, 1970.
  - (Its Document Driven model).
- Incremental Process model (pilot model)
  - α – version
  - β – version.
  - γ – version.
- Evolutionary Process model.
  - Spiral model (proposed by Barry Boehm model)
    - (Its a risk Driven Approach) used for slw & Hw design.
  - Concurrent model
    - used to develop cls. systems slws.
  - Prototype.
    - RAD
    - throwaway model
      - Prototypes that are eventually discarded rather than becoming a part of the finally delivered slw.
    - evolutionary Prototyping model.
      - Extreme Prototype model
- Component based model (OO Technology)
  - facilitates reusability
  - CAM ~~& DAA~~.
- Formal methods
- RUP (Rational Unified (Transitional process) model)
  - follows Verification diff
  - It is easier Phases
  - Defect free like
  - slw @ inception
  - (Its a Spec") @ elaboration
  - Driven @ Construction
  - approach) @ Transition

* Evolutionary prototyping model: prototypes that evolve into final systems through an iterative incorporation of user feedback
* Extreme prototyping: used for developing web applns mainly. Basically, it breaks down web development into 3 phases, each one based on the preceding one.
  - → the first phase is the static prototype that consists mainly Html pages. In the 2nd phase, screens are programmed & in the 3rd phase, the services are implemented.

Planning organisation structure

SDLC (Sftw development life cycle) : It has 6 phases.
from Analysis
to maintenance.

→ In planning organisation structure we discuss on the various Tasks that are related to each phase & the members involved in performing the tasks.

→ Planning an organisational structure can be classified into 3 categories. ① Project Format.
② Functional Format.
③ matrix format.

Project format :-

→ It involves a gp of programmers right from the beginning to the end of the sftw development. Each of the programmer works for 1 to 3 years on each project.

→ All the programmers involved do not excel in all the phases of sftw development life cycle. ~~phases~~.

→ The tasks are assigned to the programmers depending on the area of their interest.

→ After the completion of their part, a new project is assigned to the programmer.

Advs :-

→ The project is restricted only to the teams of programmers.

→ The programmers have the leverage (chance/scope) of working only in areas of their interest.

Disadvs :-

→ They do not focus on the other areas of sftw development except the area of their interest

Phases

→ until they unless completion of their part in the current project, programmers can't be assigned to any other project.

Functional format :- It includes a gp of programmers working in different phases as teams.

→ Team is managed by project manager/header.

→ It rules out (removes/reduces) over specialization i.e. It fairly states, ap programmer can be involved in more than one phase/module.

Eg:- Suppose there are 5 programmers say P_1, P_2, P_3, P_4, P_5.

Analysis → P_1, P_2, P_5.

coding → P_2, P_3.

maintenance → P_1, P_4, P_5.

Disadvs :-

→ ① High level of comm is required in the sw development. (If there is no interaction/comm b/n them they develop similar modules/overlapping components. So, it is waste of time, we don't complete the task in time. So, it is required b/n the teams involved)

② The project manager should be an expertise in all the areas of sw development.

③ Maintaining documents of the activities performed becomes compulsory.

Note :-

over specialization means → The programmer is assigned to phases which are not of much interest. Project manager assigns it.

Matrix format :-

→ It includes more than one programmer & each programmer is assigned to more than one project performing/carrying out diff activities in each project ruling out over-specialization.

-ve

→ Each project manager / leader who should be an expert.

DBacks:-

→ Documentation is necessary bcoz project spread over many programmers who interns are assigned to more than one project.

→ As the no of projects, the programmer is involved increases then the performance decreases.

[i.e. no of projects (work load) $\propto$ (inversely proportional) $\frac{1}{\text{Performance}}$].

Drawbacks if we do not use process models:-

→ Quality can't be achieved.

→ Efforts can't be estimated

→ Continuous Improvement can't be achieved.

→ No control over the activities of the process.

Slw concepts :- Reliable, Defect free, Quality. These are the concepts/terms we come across from time to time. so, know their meaning.

① Reliable (reliability) :- less amt of failure of the system.

↳ It is concerned about to measure failure of any system.

Eg: If a system can't work for 20min in a year then calculate the failure % of a system (reliability)

Sol: failure % = $\frac{20\text{min}}{365 \times 24 \times 60\text{min}} = 0.000038\%$

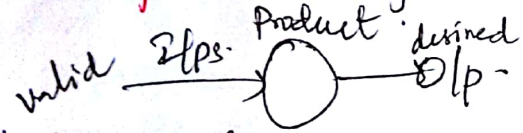
Realiability / (how much reliable the system) = Tot reliability − failure.
↳ by default it will be 100%.
= 100% − 0.000038% = 99.99% (approx).

→ "It is the ability of a prog to perform a required function under stated conds for a stated period of time.

→ The degree of reliability required for a particular product can be expressed in terms of the cost of product failure"

② Defect free :- No error in the product

③ Quality :- reaching the levels of user satisfaction / meeting the needs of the user.



for required i/ps we have to get desired o/ps.

Types of Slws :-

① Application slw. (Organic mode)
② Utility slw. (semidelached mode)
③ system slw. (Embedded mode).

Barry Boehm's COCOMO (constructive cost model) model :- classification of project slws based on their size. This model classified the projects into diff categories based on their size. The~~ory~~

category of s/ws.	Requirements (to develop the s/ws)	Skills (skilled persons required)	Team	Duration.
① Organic mode: It doesn't include database not concerned about DB. Eg: simple compilers, OS (Laptop, desktops), Scientific s/w, Business s/w & Simple inventory systems.	simple in nature	HLL or 1 package	2-50 members	< 2 yrs.
② Semidetached mode. Eg: utility s/ws. Eg: Simple command control systems, OS (medium level systems), Transaction processing system (Desktops systems).	simple + complex in nature.	mixed mode	4-100	1 yr-5 yrs
③ Embedded mode. Eg: sophisticated systems, OS (for high end systems) → complex (command & control systems) Eg satellites → very large Transaction processing systems.	complex	needs high experienced skilled persons	1000	10 years.

They are:

size	type of project
① < 2KLOC	small projects.
② 2 to < 8KLOC	medium "
③ 8 to 32 KLOC	large projects
④ > 32 KLOC	very large "

EM : SM : O.M (in the 3 modes)

1 : 3 : 9.

i.e, 1 line written in EM is equivalent to 3 in SM & 9 lines in O.M.

Eg: C = A+B ##

ALP — C-lang — calculator

logical/ideal: 20 lines, 5 lines, 1 line.

physical code: 20 lines, 60 lines, 180 lines.

1 : 3 : 9.

main()
{ int a,b,c;
c=a+b;
}

This has to be converted to m/c

Size categories for sw products | Project size categories (6 types) | Project size is a major factor that determines the level of mgmt control & the types of tools & techniques required on a sw project.

	category	No of programmers	Duration	product size	Subroutine	Description (nothing but end product) Examples :-
1.	Trivial projects	1	1-4 weeks working part time few days few weeks.	500 source lines	10-20.	Personal sw (sws used for personal which are developed use) for the exclusive use of the programmer & usually discarded after a few months.
(2)	~~Trivial~~ small projects.	1	1-6 (full time)	1000-2000 lines	25-50.	scientific appln written by Engineers to solve numerical problems.

Notes:- Little interaction bln programmers & customers, No formal description techniques, standarized techniques, & notations - documents & systematic project reviews are required (no documentation is required) eg: WT projects we need to develop front end ~~tools~~ programs like web pages.

(3)	Medium size	2-5	1-2 years	10000 5000 10K-50K lines	250-1000	Assemblers, small compilers, inventory systems, process control Appln.

* It requires interaction bln programmers & customers, certain degree of formality. It also required planning, documentation & project reviews.

(4)	very large projects	100-1000	4-5 years	1 million ~~50K-100K~~ 10,00,000K lines	several major subsystems	Real time processing, telecomm, multitasking, large operating systems & large database systems.

* Needs interaction bln customer & developer
* Needs " " developer & developer.

(5) ~~Large proj~~

(5) large projects	5–20	2–3 years	50K–100K	several subsystems	large compilers, time sharing systems (Appln/sws for), database packages, real-time control systems.

* Significant interaction with other programs & sw systems.
* Require more than one programming team (3-5 persons) often involves more than one level of mgmt.
* It includes systematic process standardized documents & formal reviews are essential throughout the large project.
* level of formality is very high ie standardized formality is required in planning, documents & project reviews.

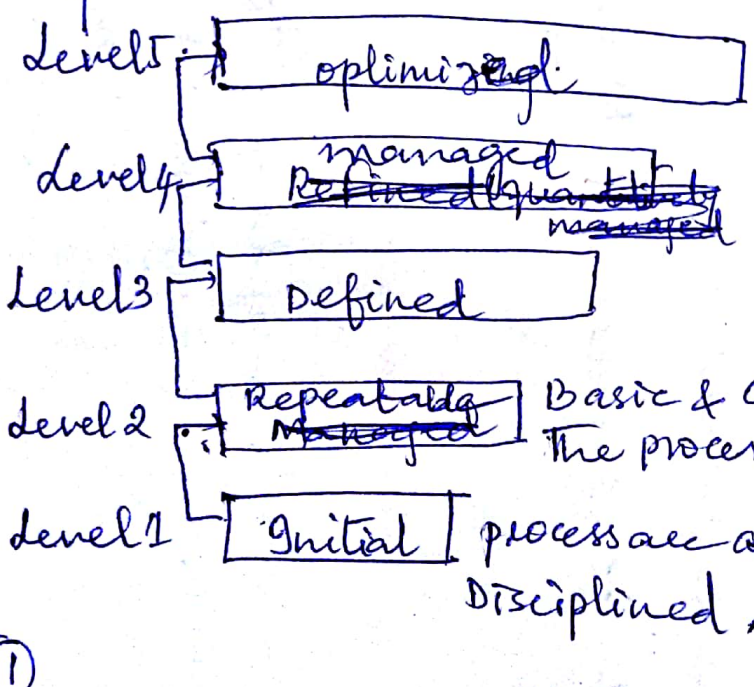
(6) Extremely large projects	2000–5000	5–10 years	1 million to 10 millions.	very large subsystems	eg: Real time processing, distributed processing, air traffic control, military command control system & telecomm

CMM levels :- (Capability Maturity model) Levels :

The CMM for slw is a model for judging the maturity of the slw processes of an organizations & for identifying the key practices that are required to increase the maturity of these processes. i.e, CMM is model that is used for judging the maturity of processes used for slw products developed by an organization & identifies the key practices that increase the maturity of these products.

There are 5 levels :

highest level & dream companies will be assigned this level.



Basic & consisted processes are established
The processes are repeated for like projects

process are adhoc, inconsistent. Concerned with Disciplined, background work of project.

Slw Attributes · CUPRIMDA

C – Capability –
U – Usability
P – Performance.
R – Reliability.
I – Instalability
M – Maintainability
D – Documentation
A – Availability.

Requirement Engineering :- tasks

It can be performed as follows :

1. Identify requirements ⇒ It is also called eliciting requirements.
2. Analyse "
3. Specify "
4. ~~Evaluate~~ Elaborate "
5. manage "
6. Negotiation & then prepare SRS.

These are the requirement engineering tasks.
& ~~prepare~~ SRS.

⇒ **Inception :-**

→ Most of projects begin when a business need is identified or a potential new market (or) service is discovered.

Eliciting / Identifying requirements :-

→ the task of communicating with customers & users to determine what their requirements are. This is sometimes also called as "requirement Gathering".

Analysing requirements :-

→ Determining whether the stated requirements are unclear, incomplete, ambiguous or contradictory & then resolving the issues.

Recording / specifying requirements :- (specification. ⇒ Eg :- Abstract.

→ Requirements might be documented in various forms such as natural language documents, use cases, user-stories or process specifications.

→ Analyst (person who prepare SRS/document) can employ several techniques to elicit (gather) the requirements from the customer.

* Interviews
* Scenarios (eg :- UML)
* View pts.
* Ethanography.
* Prototyping & usecases
* Focus groups & requirements list creating.

Elaboration :- The info obtained from the customer during Inception & Elicitation is expanded & refined during elaboration.

→ It develops a refined requirements model.

Negotiation :-

customers, users & other stakeholders are asked to rank requirements.

→ Based on these ranking (priority) the case requirements are eliminated.

Requirement management :-

→ It is a set of activities that help the project team identify, control, & track requirements as the project proceeds.

SRS stands for { Sw requirement Spec'n. (or) System (or) " " Spec'n Document. (or) Requirement " .

SRS :-

SRS, briefs about "system description".

→ It is a complete description of the behaviour of the system to be developed.

→ i.e., It is the document which clearly & precisely describes each of the "essential requirements" of the Sw & "external interfaces". like domain requirement which involve.

① Information domain
② functional " } altogether forms an analysis model.
③ Behavior "

Requirements are 2 types { Functional requirements (FR) & Non-functional requirement (NFR) } together referred as "Domain requirements".

Functional requirements → operational constraints.

Services (things taken by the product)

Goals of SRS :-

→ Describing the scope of work.
→ Providing a reference to Sw designers.

Properties of good SRS document :-

* Concise
* Structured
* Black-Box view (functionality of an appln).
* Conceptual integrity.
* Response to undesired events.
* Verifable.

Concise :- Giving a lot of info clearly in a few words.

Structured :- It should be arranged correctly according to a plan.

Black Box view :- Black-box testing is a method of Sw testing that examines the functionality of an appln.

conceptual integrity :-

→ It shows "lack of unity" in conceptual design.

Verifoable :-

→ It checks for the accuracy & justification.

Problems without a SRS document :-

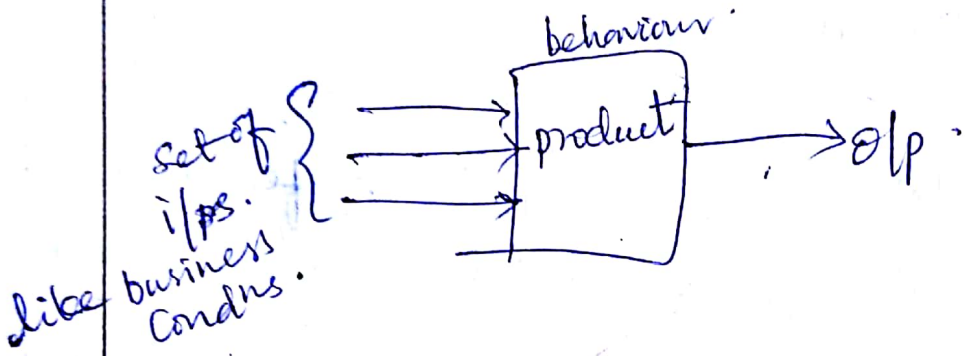
→ System would not be implemented according to the customer needs.
→ Developers would not know what they are developing is exactly required by customer.
→ Difficult to understand system functionality & maintenance of the system.

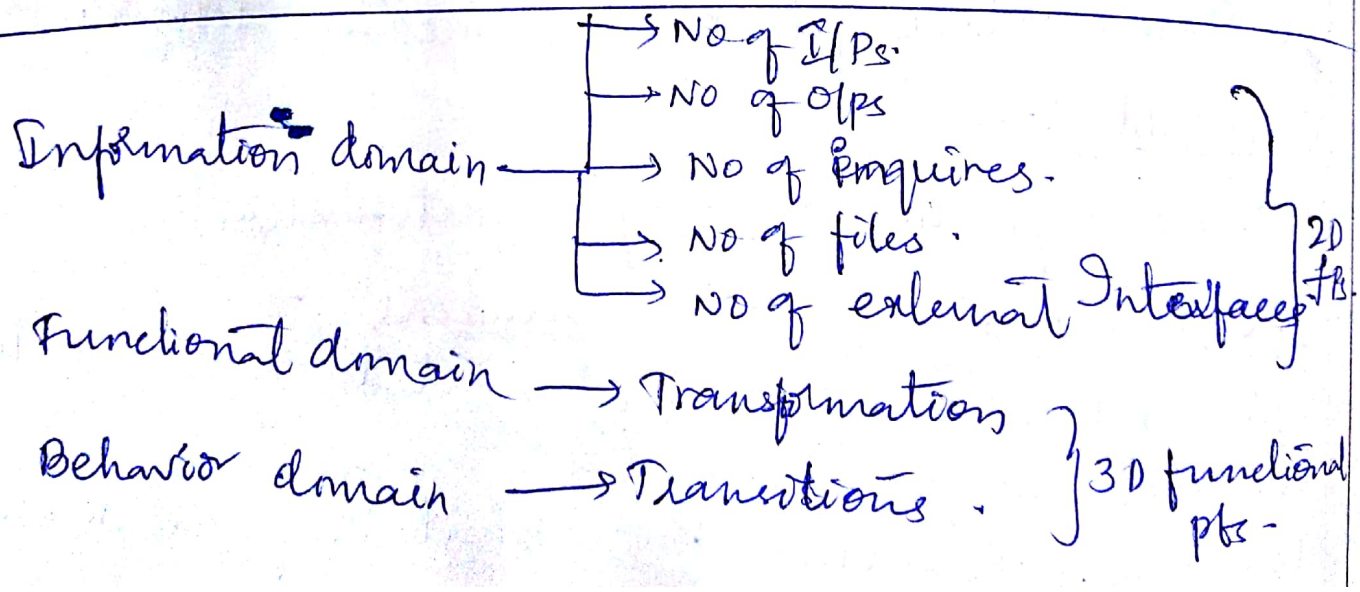
Functional Requirements :- (FR)

→ A FR requirement document defines the functionality of a system or one of its subsystems.

→ A function is a set of I/ps, behavior & O/p.

→ It may be high-level statements of what the system requirements should also describe clearly about the system services in detail.

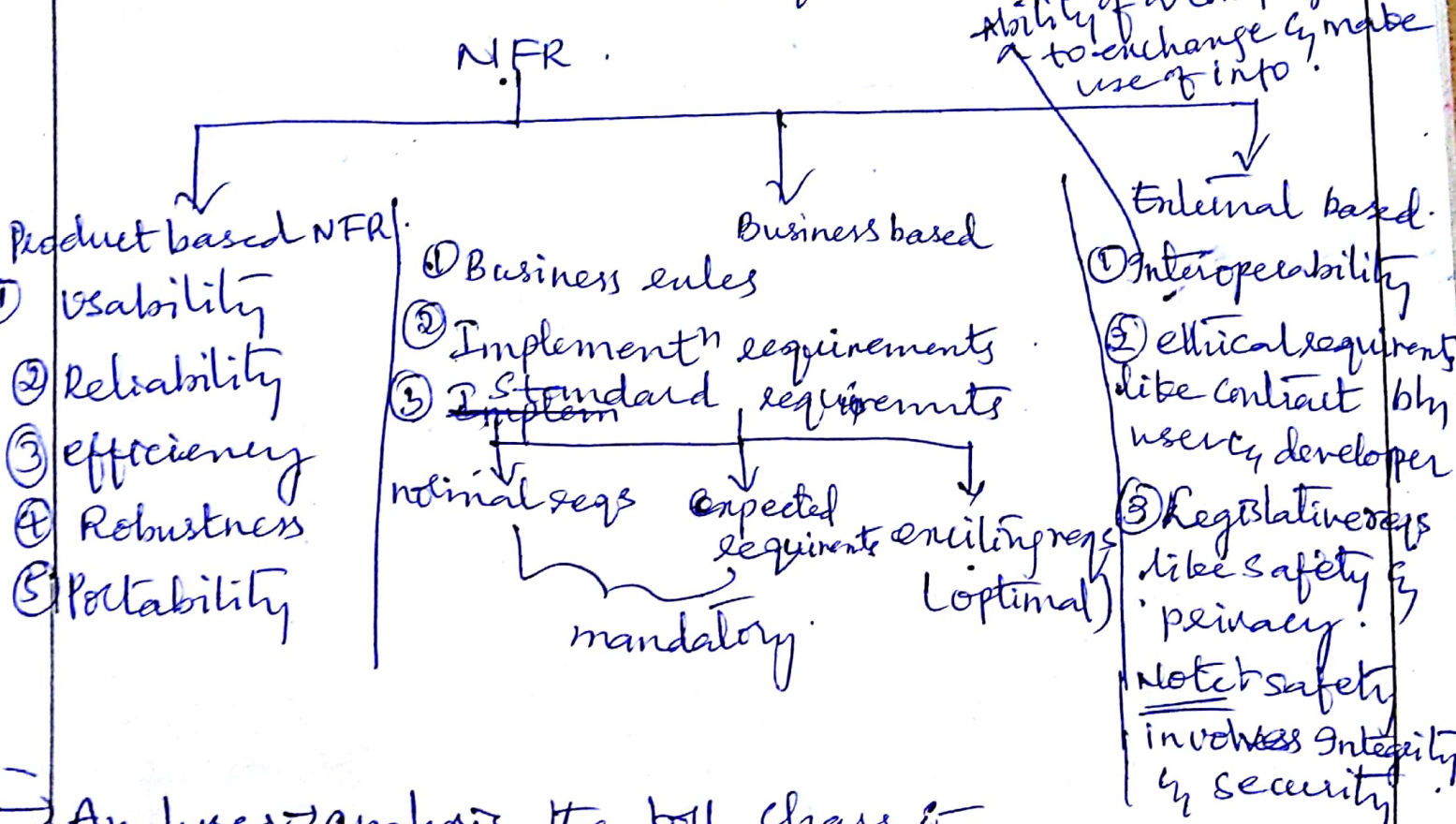




Non-Functional Requirements: (NFR).

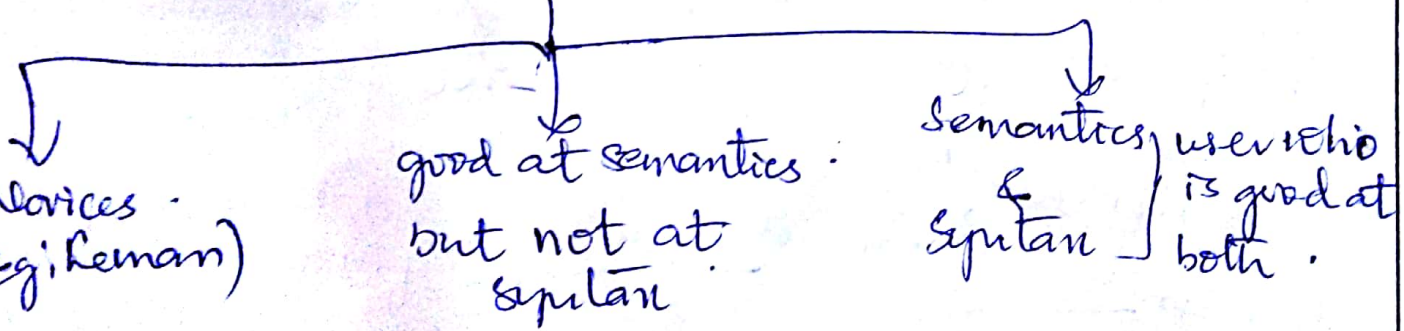
→ It is a requirement that can be used to "judge the operations" of a system rather than a functionality (behaviour).

→ It can be described as a quality attribute, performance attribute, security attribute, reliability attributes as well as the maintainability of the system.

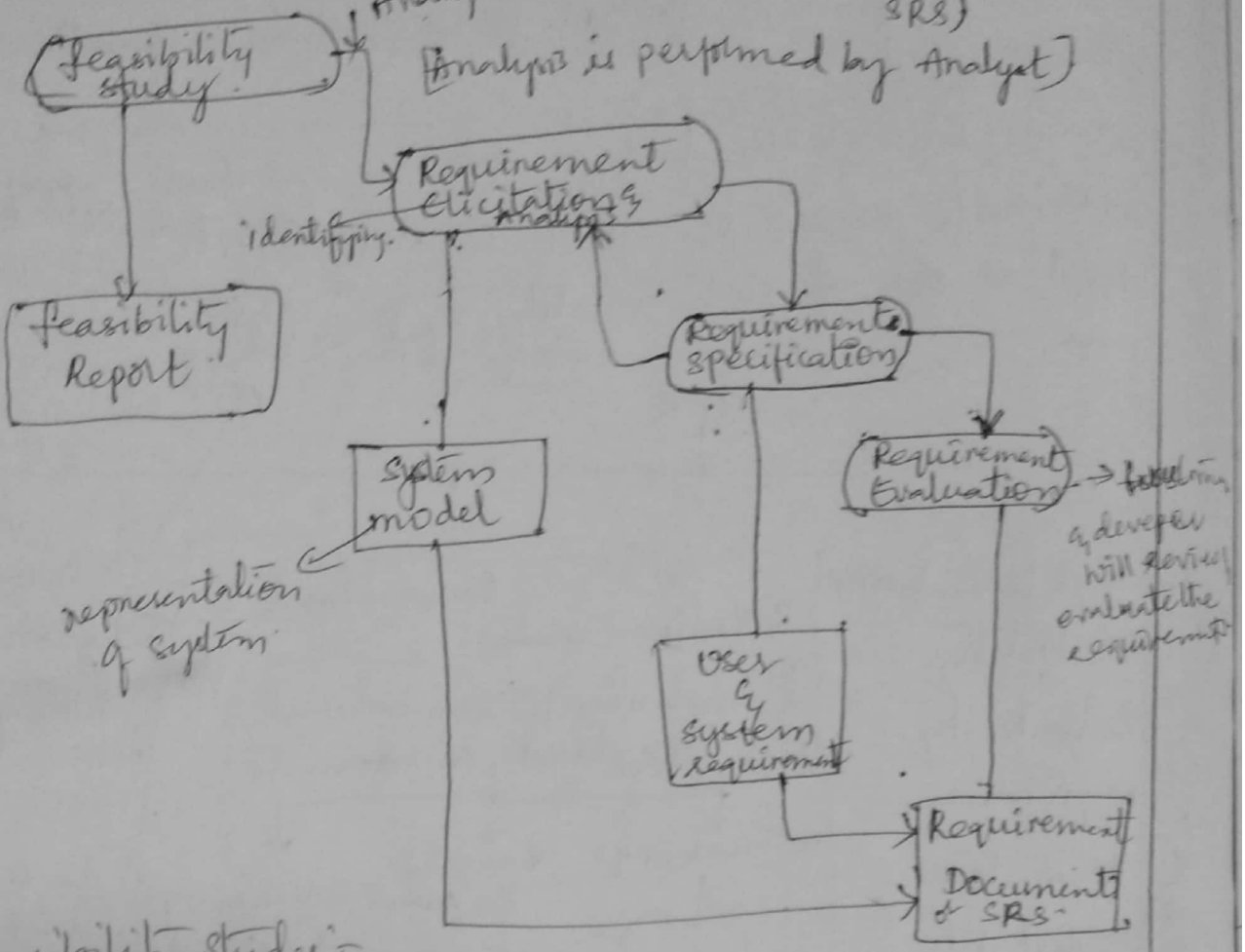


→ Analyser → analysis the following chars :-

User characteristics.



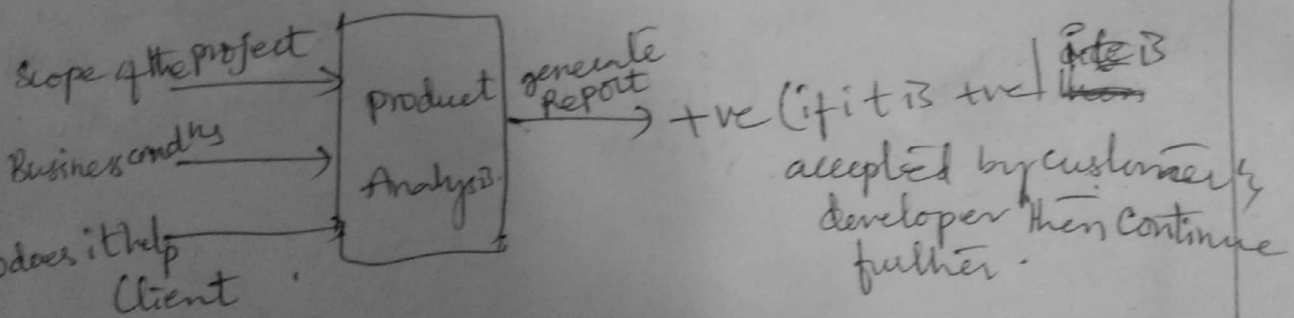
Requirements Engineering process :- Result of this process is a SRS document.



Feasibility study :-

→ Feasibility study is performed for every new system which takes the i/p as :

① Scope of the problem.
② Business constraints of developers & customers.
③ operational support to client.



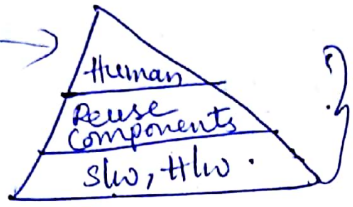
→ This feasibility study provides info based on which decisions are taken whether the project is taken up by the organizations or rejected.

So, in feasibility study 3 aspects are used for evaluation.

① Technical Feasibility :— It involves resource evaluation.

② Economical " : Profit evaluation.

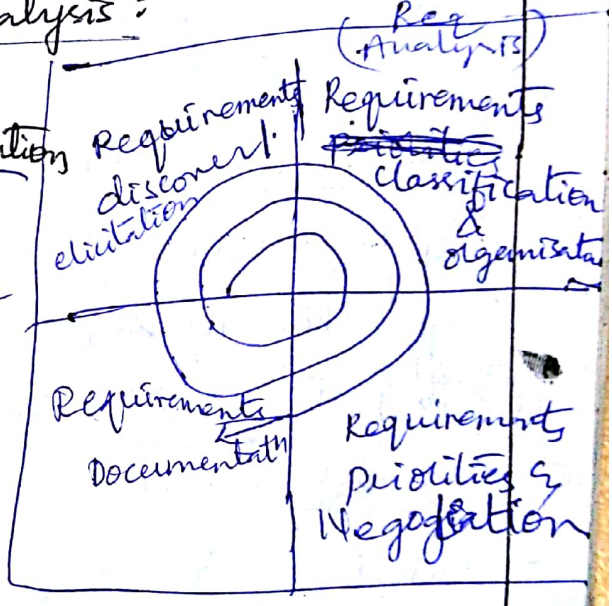
③ operational " : How does the product support client organisations.



We've to consider all these things.

If all these things are true, then we've to approve & call the analyst who will in turn prepares a analysis report.

Requirements Elicitation & Analysis :-



Requirements discovery or Information gathering :—

→ Requirements / Information can be gathered by the analyst using the fol techniques :

① view points
② scenarios (eg : UML)
③ Interviews
④ Ethanography.

View pts :— → Indirect View pts.
→ Direct / Interactive "
→ Domain " "

→ It is a frame work used by analyst which is used to resolve ambiguity b/w the requirements provided by customer. They are categorized into 3 grps.

① Direct / Interactive : The requirement gathered from customer are placed under this.

Indirect view pts :- These requirements are concerned with those objects or human being who do not interact with product but they'll have impact on the product.

Domain view pt :-

→ They are the conditions / constraints that reflects the characteristics of the product that is under development.

Scenario based Approach :-

→ It's a structured Spec'n Lang where some uniformity, standards are maintained in gathering the requirements. These things are only understood by the developers but not ~~by~~ the customers.

→ Interviews :- There are of 2 types.

Open Interview :- which is a unstructured approach of gathering requirements. It is used only for organizations (Appl'ns) projects of small scale.

Closed Interviews :- (Structured Interviews)

→ In most of the projects, ~~in~~ which requirements are gathered using this approach even though it is time consuming & expensive. Generally interviews are not organised in client loc'n / developers organisation. Developers use one approach which is refered as FAST (Facilitate Appl'n specification Technique).

~~Ethanography~~ :-

Ethanography :- This is one of the effective way of gathering requirements where analysts will gather the requirements based on customer's work environment rather than following process definitions or any guidlines.

IEEE Format of SRS plans :-

→ nothing but report/record of project work done.

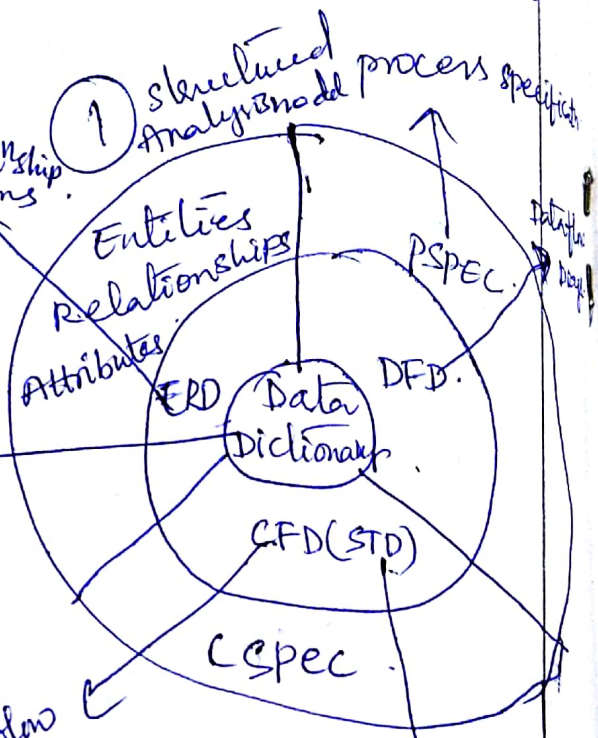
1. General Description or Introduction
   - 1.1 Purpose of Requirement document.
   - 1.2 Scope of problem.
   - 1.3 Defn, Acronyms, synonyms.
   - 1.4 References.
   - 1.5 Remainder part of Requirement document.
2. General constraints.
   - 2.1 Process Spec^n.
   - 2.2. Process function
   - 2.3 User characteristics.
   - 2.4 Assumptions & Dependencies.
   - 2.5 Constraints.
3. Specific Requirements.
   - 3.1 Functional Requirements
   - 3.2 Non-functional "
   - 3.3. Domain Requirements.
4. Appendices.
5. Index.

→ If all

→ If all these activities are performed then we say that defn phase is completed.

Analysis model :-

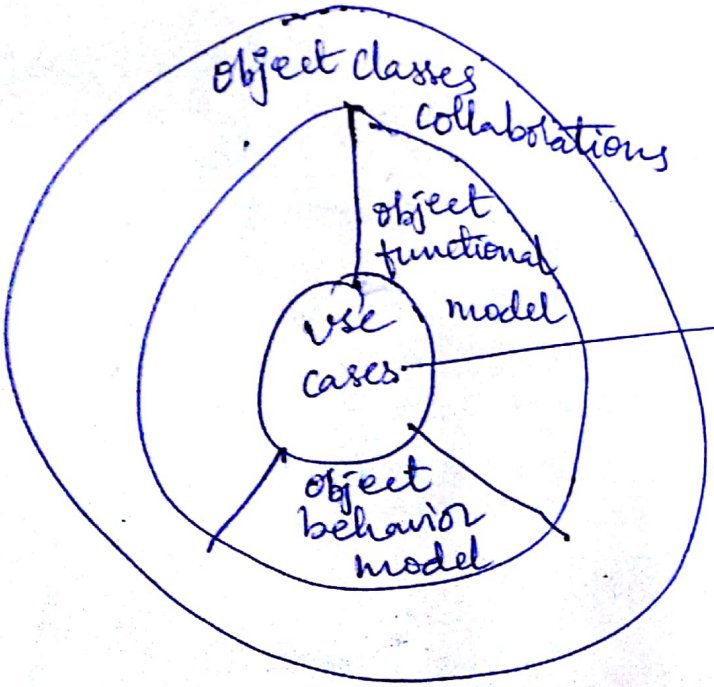
1. Structured Analysis model
2. Object oriented Analysis model.



→ It contains system descriptions i.e, whatever we've gathered from the customer we place here in Data Dictionary.

E/R diags :- It is the only way to represent the things in the real world.

2.



→ provides scope of the problem.

Requirement Engineering :-

⇒ Requirements engineering provides the appropriate mechanism for understanding what the customer wants, analyzing need, assessing feasibility, negotiating a reasonable solutions, specifying the soln unambiguously, validating the spec, managing the requirements as they are transformed into an operational system.

⇒ The requirements engineering process can be described in 5 distinct steps.

1. requirements elicitation
2. requirements analysis & negotiation.
3. requirements specifn.
4. system modeling.
5. requirements validation
6. requirements management.

Requirements Elicitation :-

It includes asking the customer, the users & others what the objectives for the system or product are, what is to be accomplished, how the system or product fits into the needs of the business, & finally how the system or product is to be used on a day-to-day basis.

→ This ~~steps~~ also involves identifying the number of problems such as
   (a) Problems of scope
   (b) Problems of understanding.
   (c) Problems of volatility.

Hence these problems makes requirements elicitation phase 1st step very difficult.

Problems of scope:- The boundary of the system is ill-defined or the customers / users specify unnecessary technical detail that may confuse, rather than clarify, overall system objectives.

Problems of understanding:-

The customers / users are not completely sure of what is needed, have a poor understanding of the capabilities & limitations of their computing environment, don't have a full understanding of the problem domain, have trouble communicating needs to the system engineer, omit information that is believed to be "obvious", specify requirements that conflict with the needs of other customers / users, or specify requirements that are ambiguous or untestable.

Problems of volatility :- The requirements change over time.

→ To overcome these problems, system engineer must approach the requirements gathering activity in an organized manner.

→ ~~Summarized steps of~~ A set of detailed guidelines for requirements elicitation are summarized in the following steps.

① Assess the business & technical feasibility for the proposed system.

② Identify the people who will help specify requirements & understand their organizational bias (problem).

→ Define the technical environment (eg; computing architecture, operating system, telecommunication needs) into which the system or product will be placed.

→ Identify "domain constraints" (i.e, characteristics of the business environment specific to the appln domain) that limit the functionality or performance of the system or product to be built.

→ Define one or more requirements elicitation methods (eg; interviews, focus groups, team meetings)

→ Solicit participation from many people so that requirements are defined from diff points of view; be sure to identify the rationale for each requirement that is recorded.

→ Identify ambiguous requirements as candidates for prototyping.

→ Create usage scenarios to help customers/users better identify key requirements.

→ The result of this activity is called as a work product which include :-

(a) A statement of need & feasibility

(b) A bounded statement of scope for the system or (we say meeting & limits) product.

(c) A list of customers, users, & other stakeholders who participated in the requirements elicitation activity

(d) A description of the system's technical environment.

(e) A list of requirements (preferably organized by function) & the domain constraints that apply to each.

(f) A set of usage scenarios that provide insight into the use of the system or product under different operating conditions.

(g) Any prototypes developed to better define requirements.

→ Each of these work products is reviewed by all people who have participated in the requirements elicitation.

Requirements Analysis & Negotiation :-

Once requirements have been gathered, the work products ~~gathered~~ noted in the earlier phase form the basis for 'requirements analysis'.

⇒ Analysis categorizes requirements & organizes them into related subsets; explores each requirement in relationship to others; examines requirements for consistency, omissions, & ambiguity; & ranks requirements based on the needs of customers/users.

→ As a part of requirements analysis activity, the fol questions are asked & answered :—

(a) Is each requirement ~~&~~ consistent with the overall objective for the system/product ?.

reconcile, settle, resolve.

② Have all requirements been specified at the proper level of abstraction? That is, do some requirements provide a level of technical detail that is inappropriate at this stage?

③ Is the requirement really necessary or does it represent an add-on feature that may not be essential to the objective of the system?

④ Is each requirement bounded & unambiguous?

⑤ Does each requirement have attribution? i.e., is a source (generally, a specific individual) noted for each requirement?

⑥ Do any requirements conflict with other requirements?

⑦ Is each requirement achievable in the technical environment that will built the system/product?

⑧ Is each requirement testable, once implemented?

→ It is usual for customers & users to ask for more than can be achieved, given limited business resources.

→ It is also common for different customers/users to propose conflicting requirements, arguing that their version is "essential for our special needs".

→ The system engineer must reconcile (resolve/settle) the conflicts if any, through a process of negotiation.

Customers, users & stakeholders are asked to rank requirements & then discuss conflicts in priority.

Risks associated with each requirement are identified & analyzed.

Rough guesstimates (estimation based on guesswork) of development effort are made & used to assess the impact of each requirement on project cost & delivery time.

Using an iterative approach, requirements are eliminated, combined and/or modified so that each party achieves some measure of satisfaction.

Requirements specification :-

Specification → can be written document, a graphical model, a formal mathematical model, a collection of usage scenario (use cases), a prototype or any combination of these.

System Specification → It is the final work product produced by the system & requirements engineer.

It serves as the foundation for hw engineering, software engineering, database engineering & human engineering (eg: teaching the individuals).

It describes the function & performance of a computer-based system (product) & the constraints that will govern its development.

→ The specification bounds each allocated system elt.

→ The system specification also describes the info (data & control) that is i/p to & o/p from the system.

System Modeling :-

→ In order to fully specify what is to be built, it necessary to build a meaningful model for the system (product) to be developed.

→ From the model, it would be relatively easy to assess the efficiency of work flow (a.

→ It is important to evaluate the systems components in relationship to one another, to determine how requirements fit into this picture, and to assess the "aesthetics" (beauty) of the system as it has been conceived.

Requirements Validation :-

→ The work products produced as a consequence of requirements engineering (a system spec & related info) are assessed for quality during a validation step.

→ "Requirements validation examines the spec to ensure that all system requirements have been stated unambiguously; that inconsistencies, omissions, & errors have been detected & corrected; & that the

work products conform to the standards established for the process, the project & the product."

→ A key concern during this step is consistency across the system model to ensure that requirements have been consistently stated.

→ The primary requirements validation mechanism is the formal technical review.

→ The review team includes system engineers, customers, users, & other stakeholders who examine the system specification looking for errors in content or interpretation, areas where clarification may be required, missing information