16/12/19

Software Engineering

The Evolving Role of software

→ Software takes a dual role. It is both a product and vehicle for delivering a product.

→ Software acts as the basis for the control of the computer (operating systems), the communication of information (networks) and the creation and control of other programs (software tools and environments).

→ The role of computer s/w has undergone significant change over a span of more than 50 years.

→ During 1970's & 1980's changing the perception of computers and s/w and their impact on our culture.

→ Osborne characterized a "new industrial revolution"

→ Toffler called the advent of microelectronics part of the "third wave of change" in human history.

→ Naisbitt predicted the transformation from industrial society to an information society.

→ Feigenbum & Mccorduck suggested that information and knowledge (controlled by computers) would be the focal point for power in the twenty first century.

→ Stoll argued that the electronic community created by n/w's and s/w was the key to

knowledge interchange throughout the world.

→ 1990's began Toffler described a "power shift" in which old power structures (governmental, educational, industrial, economic, & military) disintegrate as computers and software lead to a "democratization knowledge"

→ 2000's progressed, Johnson discussed the power of "emergence" a phenomenon that explains what happens when interconnections among relatively simple entities result in a system that self-organizes to form more intelligent more adaptive behavior.

→ Yourdon revisited the tragic events 9/11 to discuss the continuing impact of global terrorism on the IT community.

→ Questions when modern computer base system are built.

(1) Why does it take so long to get s/w finished?
(2) Why are development costs so high?
(3) Why can't we find all errors before we give s/w to our customer?
(4) Why do we spend so much time and effort maintaining existing program?
(5) Why do we continue to have difficulty in measuring progress as software is being developed and maintained?

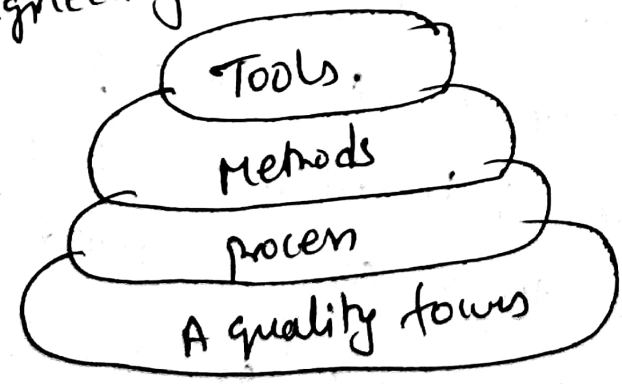
Software Engineering - A layered Technology:-

→ Software Engineering is the establishment and use of sound Engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.

→ The above definition says about the technical aspects of software quality. (It does not address the need for customer satisfaction or timely product delivery).

By IEEE definition.
Software Engineering (1) The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software i,e the application of Engineering to software.

→ Software Engineering is a layered technology.



→ The benchmark of software Engineering is a quality focus.
→ The foundation for software Engineering is the procen layer.
→ procen defines a framework that must be established for effective delivery of software Engineering technology.

→ The SW process forms the basis for management control of SW projects and establishes the context in which technical methods are applied, work products (models, documents, data, reports, forms etc) are produced, milestones are established, quality is ensured and change is properly managed.

→ SW Engineering methods provide the technical "how to's" for building software.

→ Methods encompass a broad array of tasks that include communication, requirements analysis, design modeling, program construction, testing and support.

→ Methods rely on a set of basic principles that govern each area of the technology and include modeling activities and other descriptive techniques.

→ Software Engineering tools provide automated or semiautomated support for the process and the methods

→ When tools are integrated, so that information created by one tools can be used by another, tool.

→ A system for the support of SW development called computer-aided SW Engineering is established.

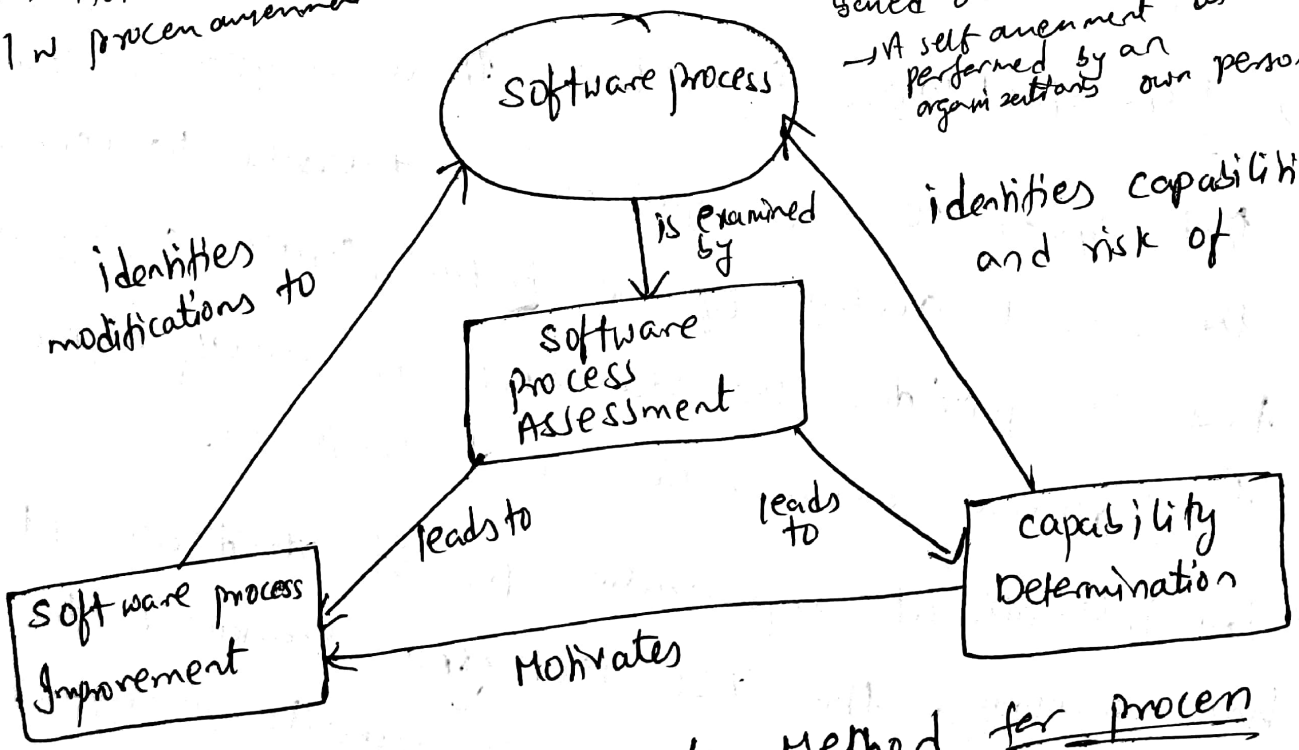
Procen Assessment:- procen is a set of related activities that leads to the production of the s/w.

→ The procen itself should be assessed to ensure that it meets a set of basic process criteria that have been shown to be essential for a successful s/w Engineering. (The activities may involve the development of the s/w from the scratch or modifying an existing system)

→ The relationship b/w the software process and the methods applied for assessment and improvement is as follows.

→ S/w procen assessment is a disciplined examination of s/w processes used by an organization based on a procen model.

→ A self assessment is performed by an organizations own personnel.



① **Standard CMMI Assessment Method for procen improvement (SCAMPI):-** It provides a 5-step procen assessment model that incorporates initiating, diagnosing, establishing, acting and learning. (strengths, weakness)

→ It uses the SEI CMMI as the basis for assessment.

② **CMM-Based Appraisal for internal procen improvement (CBA IPI):-** provides a diagnostic technique for assessing the relative maturity of a s/w organization

using the SEI CMM (a precursor to the CMMI) as the basis for the assessment.

(3) SPICE (ISO/IEC15504) :- standard defines a set of requirements for software process assessment. Software process improvement and capability determination. ~~International organization of standardization.~~

→ The intent of the standard is to assist organizations in developing an objective evaluation of the efficacy of any defined s/w process. (IEC ⇒ International Electrotechnical Commission)

→ ISO 9001:2000 for s/w :- It is a generic standard that applies to any organization that wants to improve the overall quality of the products, systems, or services that it provides. The standard is directly applicable to software organizations and companies.

→ The international organization for standardization (ISO) has developed the ISO 9001:2000 standard to define the requirements for a quality management system that will serve to produce higher quality products and improve customer satisfaction.

→ ISO 9001:2000 stresses the importance for an organization to identify, implement, manage and continually improve the effectiveness of the processes that are necessary for the quality management system, and to manage the interaction of these processes in order to achieve the organization objectives.

ISO 9001:2000 has adopted a "plan-do-check-Act" cycle that is applied to the quality management elements of a SW project.

Plan:- establishes the process objectives, activities and tasks necenary to achieve high quality software and resultant customer satisfaction.

Do:- implements the software process (including both framework and umbrella activities)
↳ RM, SQA, FTR, SCM, RMMM, SPTC

Check:- monitors and measures the process to ensure that all requirements established for quality management have been achieved.

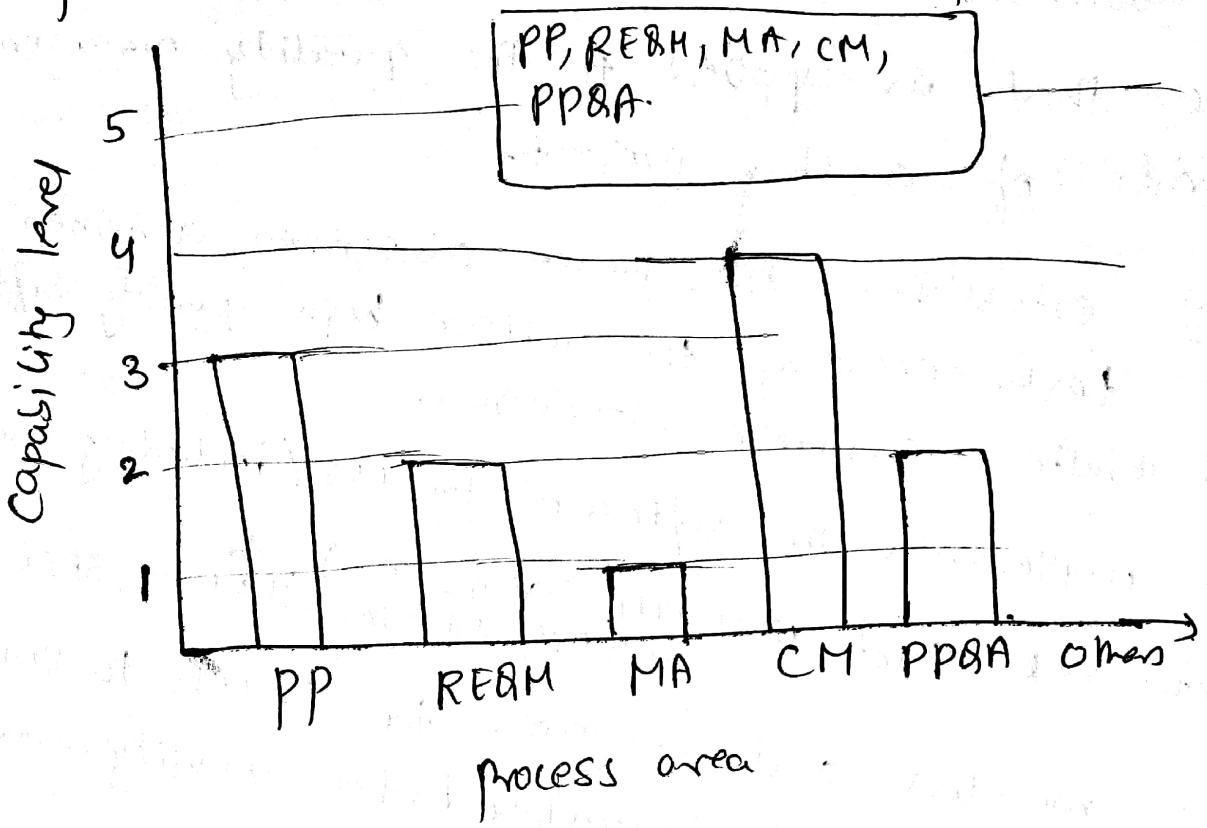
Act:- initiates SW procen improvement activities that continually work to improve the procen.

The Capability Maturity Model Integration (CMMI)

The CMMI represents a procen meta-model in two different ways
① as a continuous model
② as a staged model.

→ The continuous CMMI meta-model describes a procen in two dimensions
→ Each procen area (eg:- project planning or requirements management) is formally assened against specific goals

and practices and is rated as capability levels.



PP :- project planning
REQM :- Requirements Management
MA :- Measurement and analysis
CM :- Configuration management
PPQA :- Procen and product QA.

Level 0 :- Incomplete. The procen area (eg- requirements management) is either not performed or does not achieve all goals and objectives defined by the CMMI for level 1 capability.

Level 1 :- Performed. All of the specific goals of the process area have been satisfied. Work tasks required to produce defined, work products are being conducted.

Level 2:- Managed . All Level 1 criteria have been satisfied. All work associated with the process area conforms to an organizationally defined policy.

→ All people doing the work have access to adequate resources to get the job done.

→ Stakeholders are actively involved in the process area as required.

→ All work tasks and work products are "monitored, controlled and reviewed"; & are evaluated for process description [CMM02].

Level 3:- Defined: All level 2 criteria have been achieved. The process is tailored from the organization's set of standard processes according to the organization tailoring guidelines, and contributes work products, measures and other process improvement information to the organizational process assets [CMM02].

Level 4:- Quantitatively managed:- All level 3 criteria have been achieved. The process area is controlled and improved using measurement and quantitative assessment.

→ Quantitative objectives for quality and process performance are established and used as criteria in managing the process. [CMM02]

Level 5: optimized All capability level 4 criteria have been achieved. The process area is adapted and optimized using quantitative (statistical) means to meet changing customer needs and to continually improve the efficacy of the process area under consideration [CMM02].

→ CMMI defines each process area in terms of specific goals and specific practices required to achieve these goals.

→ CMMI defines set of five generic goals and related practices for each process area.

→ Each of the 5 generic goals corresponds to one of the 5 capability levels.

Process Models: - Process models define a distinct set of activities, actions, tasks, milestones and work products that are required to engineer high quality software.

→ The models provide a road map for s/w engineering work

→ s/w Engineers and their managers adapt the process models i,e decides the process model. according to the customer needs and follows it.

→ People who have requested the s/w have a key role to play for deciding the process model.

→ s/w Engineers have choosen a generic process framework that encompasses the following framework activities:
① communication ② planning ③ modeling ④ construction ⑤ Deployment.

→ perscriptive models became they prescribe a set of process elements framework activities, software engineering actions, tasks, work products, quality assurance and change control mechanisms for each project.

→ Each process model also prescribes a workflow i,e manner in which the process elements are interrelated to one another.

Waterfall Model:-

→ Requirements of a problem are well understood.

→ workflow from communication to deployment are in a linear fashion.

→ It is also called as classic life cycle which suggests a systematic, sequential approach to slw development that begins with customer specification of requirements and progresses through planning, modeling, construction and deployment.

→ The above process is continued until the slw developement is completed.

→ The waterfall model is the oldest Paradigm for slw Engineering.

→ The problems that are sometimes encountered when the waterfall model is applied.

(I) (a) Real projects rarely follow the sequential flow that the model proposes.

(b) The linear model can accommodate iterations, it does so indirectly.

(c) Changes can cause confusion as the project team proceeds.

(II) → It is often difficult for the customer to state all requirements explicitly.

→ But in waterfall model all requirements must be stated at first only.

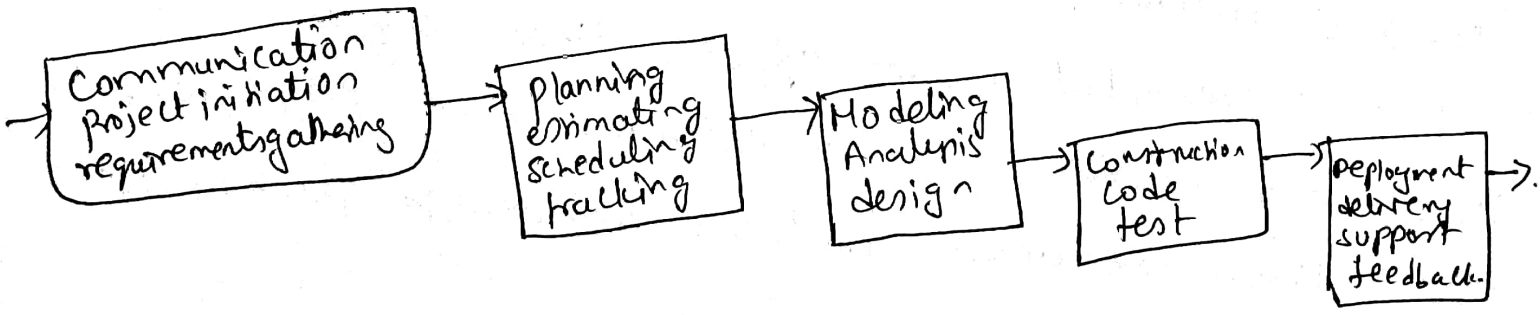
(III) → The customer must have patience.

→ A working version of the programs will not be available until late in the project time span.

→ A major blunder, if undetected until the working program is reviewed, can be disastrous.

→ Waterfall model leads to "blocking states" in which some project team members must wait for other members of the team to complete dependent tasks.

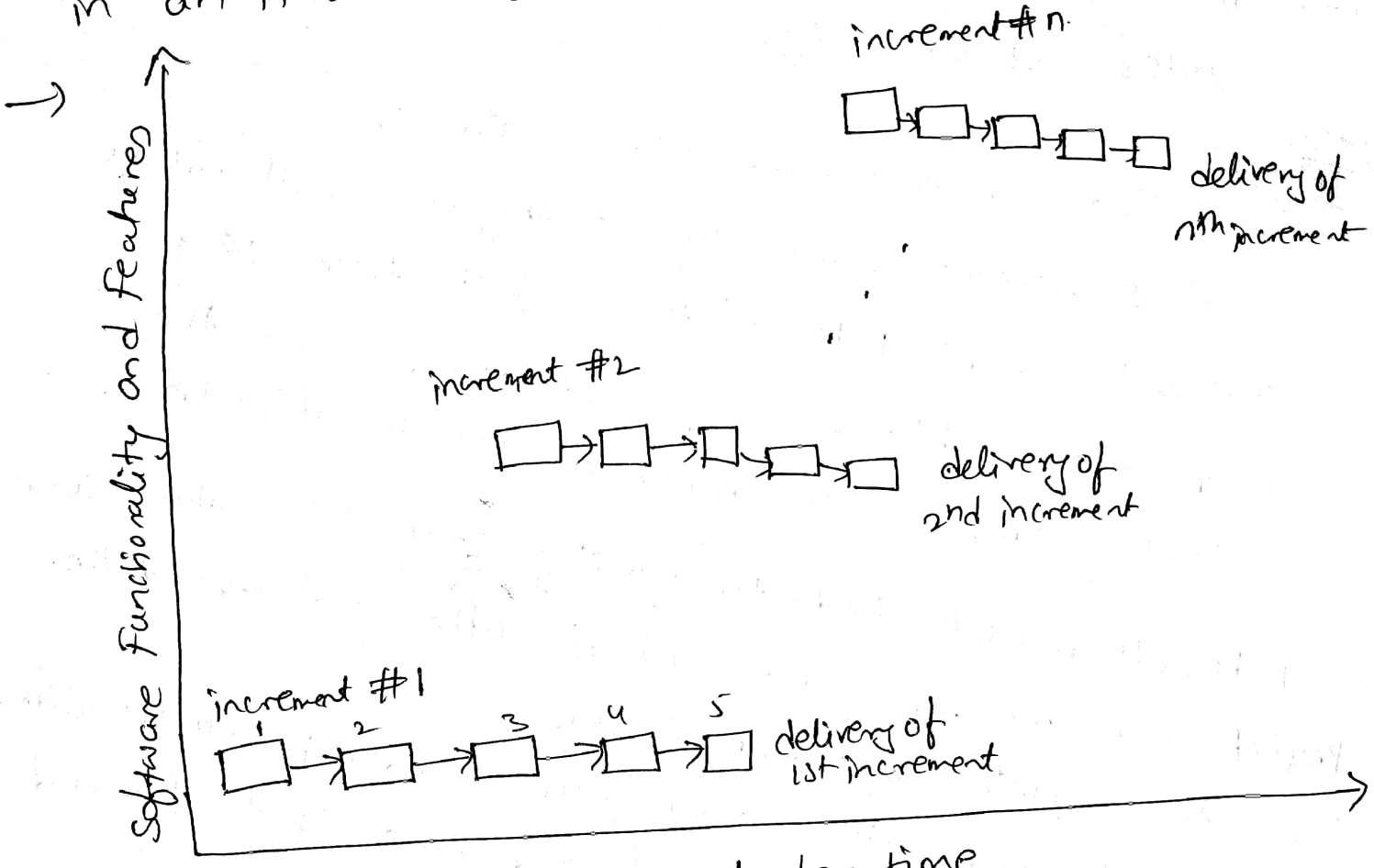
→ Waterfall model is useful process model in situations where requirements are fixed and work is to proceed to completion in a linear manner.



Block diagram of waterfall model.

Incremental process model:-

→ It combines elements of the waterfall model applied in an iterative fashion.



1 → communication, 2 → planning, 3 → Modeling (Analysis, design)
4 → construction (code, test) 5 → Deployment (delivery, feedback)

→ The incremental model applies linear sequences in a staggered fashion as a calender time progresses.

→ Each linear sequence produces deliverable "increments" of the software.

→ Eg:- word-processing software developed using the incremental paradigm might deliver basic file management, editing and document production functions in the first increment,

→ More sophisticated editing, & document production functions

→ capabilities in the second increment, spelling and grammar checking in the third increment.
→ Advanced page layout capability in the fourth increment.
→ When an incremental model is used, the 1st increment is often a "core product"
→ Basic requirements are addressed, but many supplementary features remain undelivered
→ The core product is used by the customer (or undergoes detailed evaluation).
→ 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 and the delivery of additional features and functionality.
→ This process is repeated following the delivery of each increment, until the complete product is produced.
→ The incremental process model, like prototyping and other evolutionary approaches is iterative in nature.
→ The incremental model focuses on the delivery of an operational product with each increment.
→ Incremental development is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been

established for the project.

→ Early increments can be implemented with fewer people.

→ If the core product is well received, additional staff (if required) can be added to implement the next increment. [Advantage:- More flexible, customer can respond to each iteration, Easy to manage risk, Disadvantage:- Total cost is higher needs good plan & design.]

→ Increments can be planned to manage technical risks,

→ A major system might require the availability of new h/w that is under development and whose delivery data is uncertain.

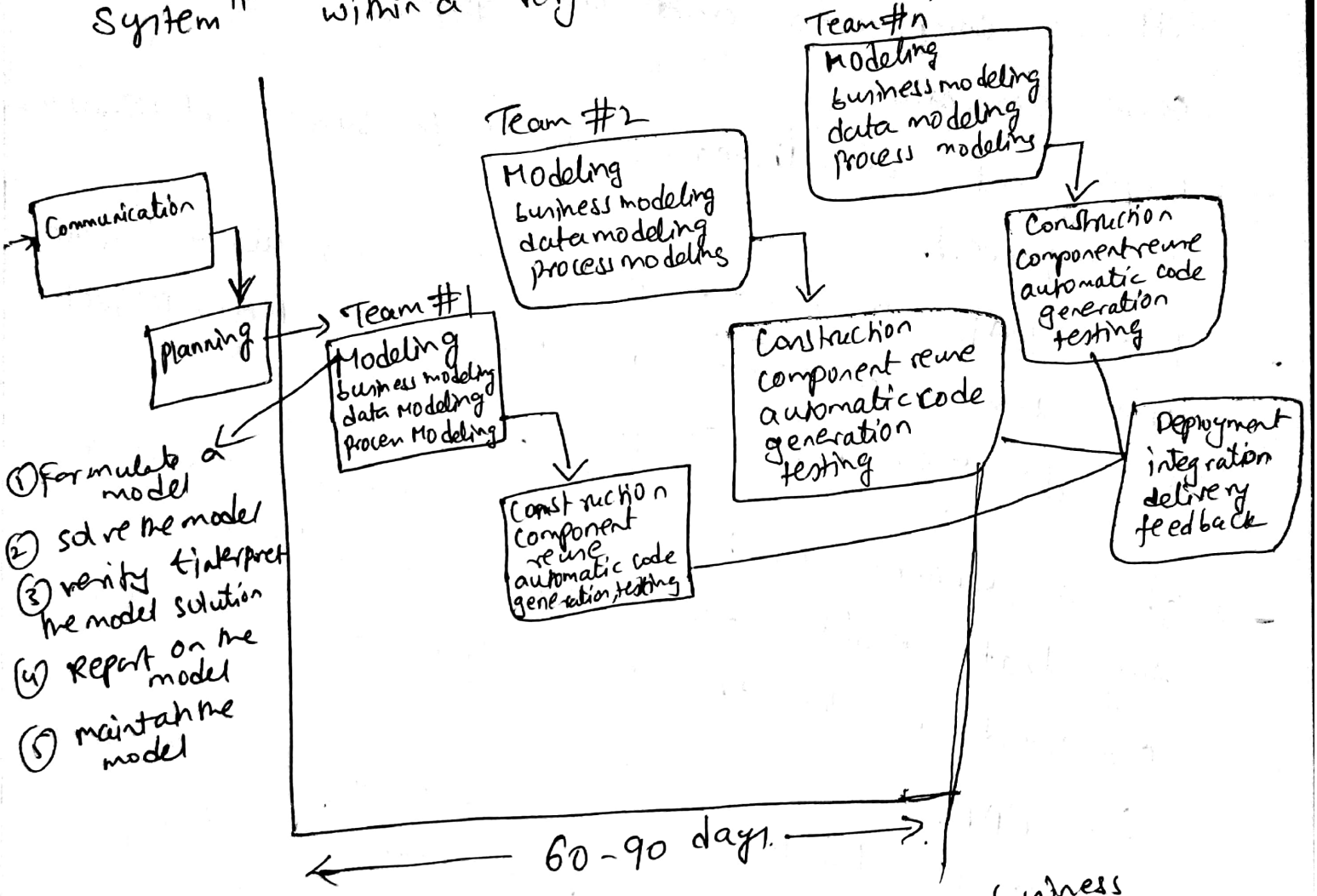
→ It might be possible to plan early increments in a way that avoids the use of this h/w, thereby enabling partial functionality to be delivered to end-users without inordinate delay.

RAD Model:- Rapid application development (RAD) is an incremental s/w process model that emphasizes a short development cycle.

→ RAD model is a "high speed" ~~adaptation~~ adaptation of the waterfall model, in which rapid development is achieved by using a component-based-construction approach.

Business modeling is the graphical representation of a company's business processes or workflows, as a means of identifying potential improvements. This is done through different graphical methods, such as flowchart, data-flow diagram etc. i.e identifying bottlenecks & inefficiencies in the process.

→ If requirements are well understood and project slope is constrained, the RAD process enables a development team to create a "fully functional system" within a very short time period (60 to 90 days)



→ Communication works to understand the business problem and the information characteristics that the s/w must accommodate.

→ Planning is essential because multiple s/w teams work in parallel on different system functions.

→ Modeling encompasses three major phases business modeling, data modeling, process modeling.

data modeling → collection of notations for describing data, data relationships, data semantics, data constraints. → Data structures representation in na table.

process modeling → process modeling is a technique designed to understand & describe the process. It connects & improves the communication b/w the current & future state of a process. i.e improve efficiency, Transparency, process standardization & best practices.

→ It establishes design representations that serve as the basis for RAD construction activity.

→ Construction emphasizes the use of pre-existing software components and the application of automatic code generation.

⇒ Deployment establishes a basis for subsequent iterations, if required.

→ The RAD process model is drawn as above diagram.

→ The time constraints imposed on a RAD project demand "Scalable scope".

→ If a business application can be modularized in a way that enables each major function to be completed in less than 3 months, it is a candidate for RAD.

→ Each major function can be addressed by a separate RAD team and then integrated to form a whole.

→ RAD approache has drawbacks

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

② if developers and customers are not committed to the rapid-fire activities necessary to complete the system in a much abbreviated time frame, RAD projects will fail.

(3) if a system cannot be properly modularized building the components necessary for RAD will be problematic.

(4) If high performance is an issue, the performance is to be achieved through tuning the interfaces to system components, the RAD approach may not work,

(5) RAD may not be appropriate when technical risks are high. (i.e new application makes heavy use of new technology).

Evolutionary Process Models:-

→ Evolutionary models are iterative.

→ They are characterized in a manner that enables software engineers to develop increasingly more complete versions of the software.

Prototyping:- A customer defines a set of general objectives for s/w.

→ It does not identify detailed input, processing, or output requirements.

→ The developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system, or the form that human-machine interaction should take.

→ A prototyping can be used as a standalone process model, it is more commonly used as a technique that can be implemented within the context of any one of the process models.

→ The prototyping paradigm assists the s/w engineer and the customer to better understand what is to be built when requirements are fuzzy.

→ The prototyping paradigm begins with communication.

→ The s/w engineer and customer meet & define the overall objectives for the s/w.

→ Identify whatever requirements are known, & outline areas where further definition is manadatory.

→ A prototyping iteration is planned quickly and modeling (in the form of a quick design) occurs.

→ The quick design focuses on a representation of those aspects of the s/w that will be visible to the customer/end-user. (eg: human interface layout or output display formats).

→ The quick design leads to the construction of a prototype.

→ The prototype is deployed and then evaluated by the customer/user.

→ Feedback is used to refine requirements for the S/W.

→ 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.

→ Prototype serves as a mechanism for identifying S/W requirements.

→ If a working prototype is built, the developer attempts to make use of existing program fragments or applies tools (eg:- report generators, window managers, etc) that enable working programs to be generated quickly.

→ The prototype can serve as the "first system" the one that Brooks recommends we throw way.

→ It is true that both customers and developers like the prototyping paradigm.

→ Users get a feel for the actual system, and the developers get to build something immediately.

→ Prototyping can be problematic for the following reasons.

(1) The customer sees what appears to be a working version of the S/W, unware that the prototype is held together "with chewing gum and baling wire".

→ unware that in the rush to get it working we (h) haven't considered overall software quality or long-term maintainability.

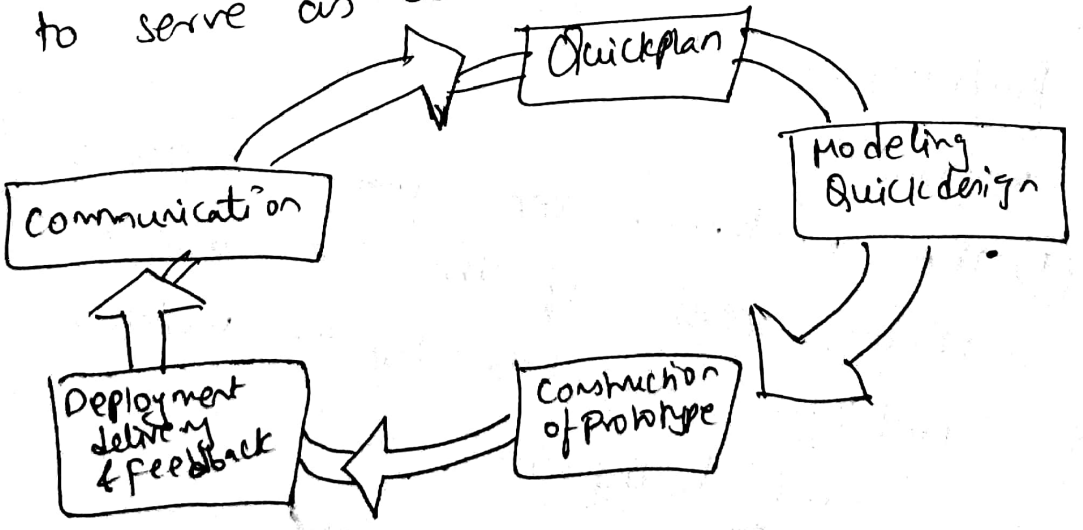
→ When informed that the product must be rebuilt so that high-levels of quality can be maintained, the customer cries foul and demands that a "a few fixes" be applied to make the prototype a working product.

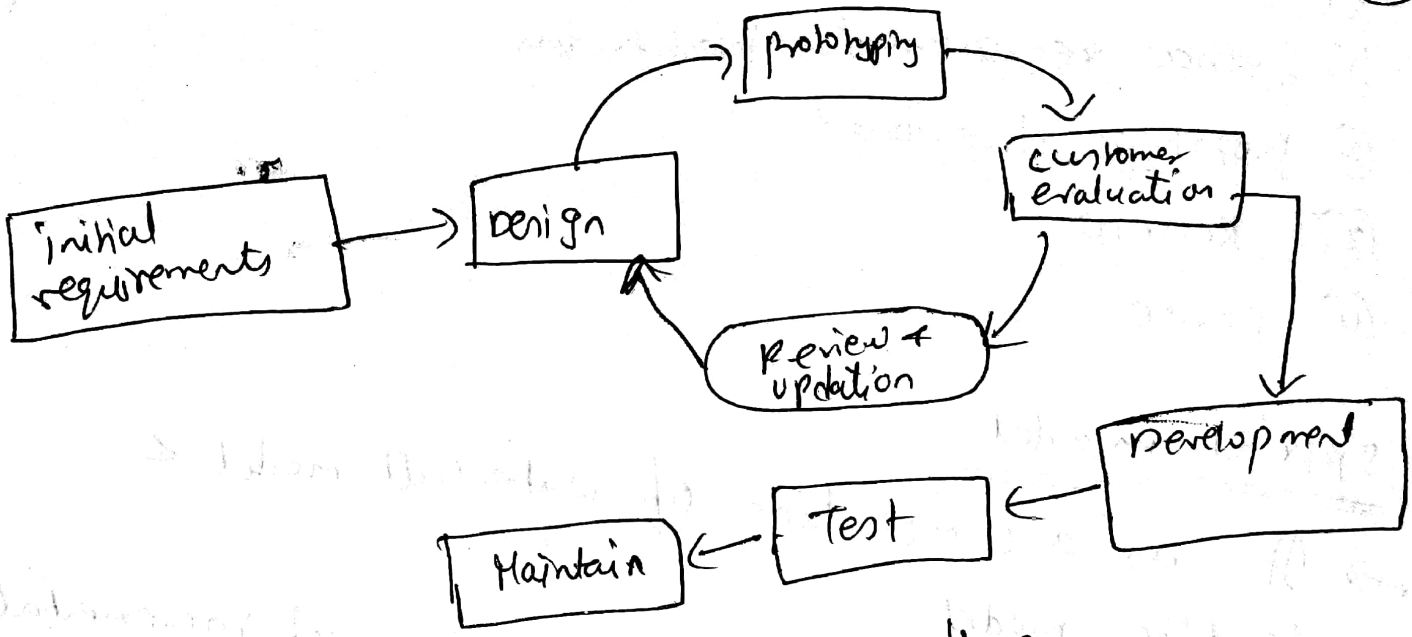
(2) The developer often makes implementation compromises in order to get a prototype working quickly.

→ An inappropriate operating system or programming language may be used simply because it is available and known.

→ An inefficient algorithm may be implemented simply to demonstrate capability.

→ Although problems occur, prototyping can be effective paradigm for SW engineering. The key is to define the rules of the game at the beginning, the customer and developer must both agree that the prototype is built to serve as a mechanism for defining requirements.





Prototype Model Block diagram

Advantages:-

→ customer can see steady progress
→ This is useful when requirements are changing rapidly

Disadvantages

→ It is impossible to know how long it will take.
→ There is no way to know the no. of iterations will be required

→ prototyping helps a lot in getting the feedback from the customer
→ This model is to get the actual requirements more deeply from user.
→ It is a process in which developes create a model of the actual s/w.

Steps:-

1. Initial requirements identification
2. prototype development
3. Review
4. Revise

spiral model

→ It is a combination of waterfall model & iterative model

→ S/w is developed in a series of incremental releases

→ Each phase in spiral model begins with design goal & ends with client reviewing

→ inner circle is concept development

→ second inner circle is system development

→ Third inner circle is system enhancement.

→ fourth is system maintenance.

When to use spiral model

1. when projects is large
2. when releases are required to be frequent
3. risk & cost evaluation is important
4. requirements are unclear & complex
5. when changes may require at any time.
6. For medium to high risk projects.

Advantages	Disadvantages
1. Cost estimation is easy	1. Risk is not meeting the schedule or budget
2. Development is fast & features are added in a systematic way	2. Documentation is more as it has intermediate phases
3. Additional functionality or changes can be done at later stage.	3. It is not advisable for small projects, it might cost
4. There is always space for customer feedback.	

The Spiral Model:-

→ It is an evolutionary slw process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model.

→ It provides the potential for rapid development of increasingly more complete versions of the slw.

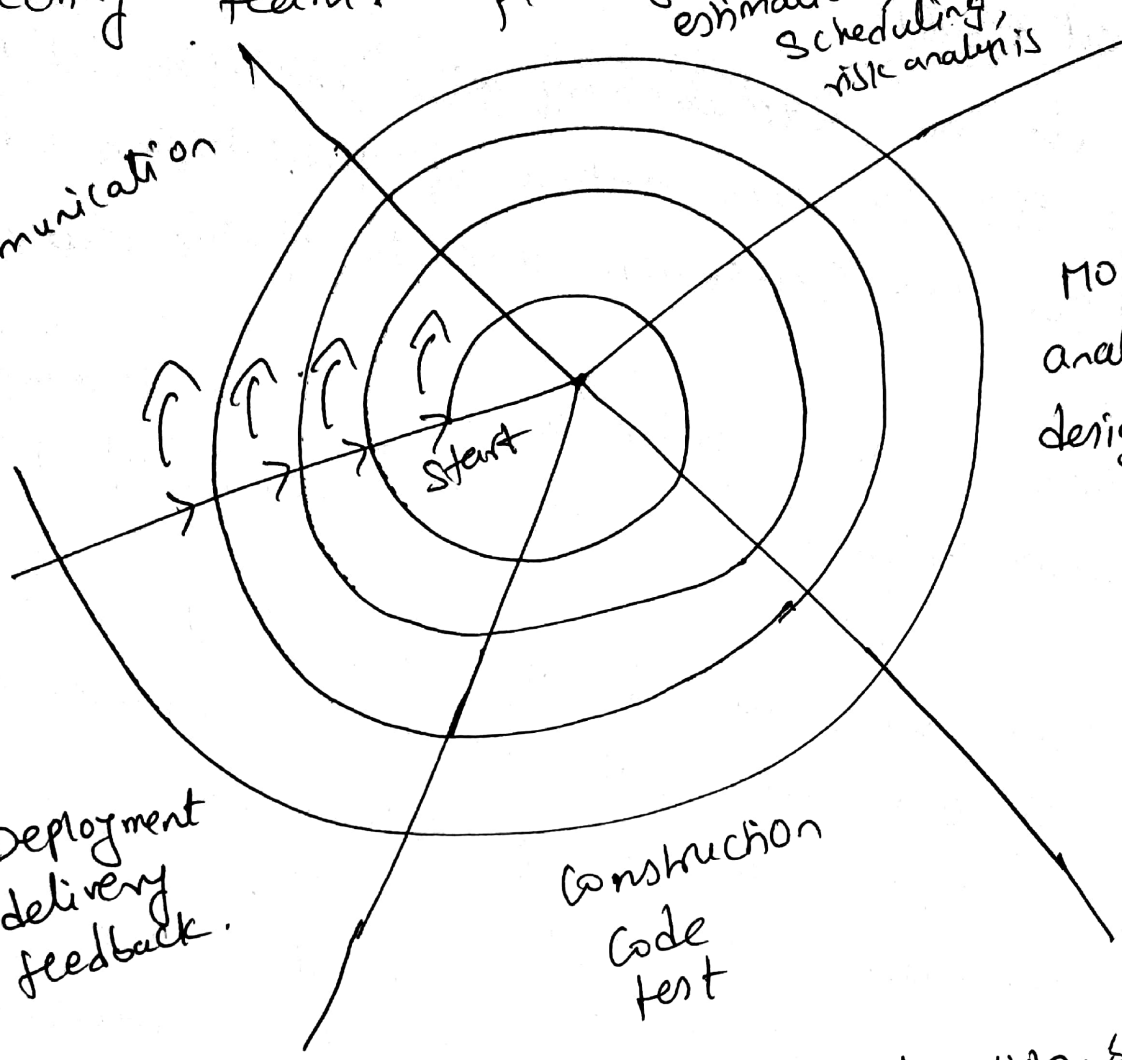
→ It has two main distinguishing features
(i) one is cyclic approach for incrementally growing a system's degree of definition and implementation while decreasing its degree of risk.
(ii) It is a set of anchor point milestones for ensuring stakeholder commitment to feasible and mutually satisfactory system solution.

→ using the spiral model, slw is developed in a series of evolutionary releases.

→ During early iterations, the release might be a paper model or prototype.

→ During later iterations increasingly more complete versions of the engineered system are produced.

→ A spiral model is divided into a set of framework activities defined by the slw



→ This process begins in a clock-wise direction, beginning at the center.

→ Risk is considered as each revolution is made.

→ Anchor points milestones a combination of work products and conditions that are attained along the path of the spiral are noted for each evolutionary pass.

→ The first circuit around the spiral might result in the development of the product specification.

→ Subsequent passes around the spiral might be used to develop a prototype & then progressively more sophisticated versions of the s/w.

→ Each pass through the planning region results in adjustments to the ~~product~~ project plan

→ Cost and schedule are adjusted based on feedback derived from the customer after delivery.

→ Project manager adjusts the planned number of iterations required to complete the s/w.

→ other process models ends when s/w is delievered

→ But spiral model can be adapted to apply throughout the life of the computer s/w.

→ First circuit around the spiral might represent a "concept development project" which starts at the core of the spiral and continues for multiple iterations until concept development is complete.

→ If the concept is to be developed into an actual product. The process proceeds outwards on the spiral and a new product development project commences

→ The new product will evolve through a number of iterations around the spiral.

→ Later circuit around the spiral might be used to represent a "product enhancement project"

→ In spiral when characterized, remains operative until the slw is retired.

→ Whenever a change is initiated, the process starts at the appropriate entry point (Eg:- product enhancement).

→ The spiral model is a realistic approach to the development of large-scale systems and software.

→ Because slw evolves as the process progresses, the developer and customer better understand and react to risks at each evolutionary level.

→ "The spiral model uses prototyping as a risk reduction mechanism"

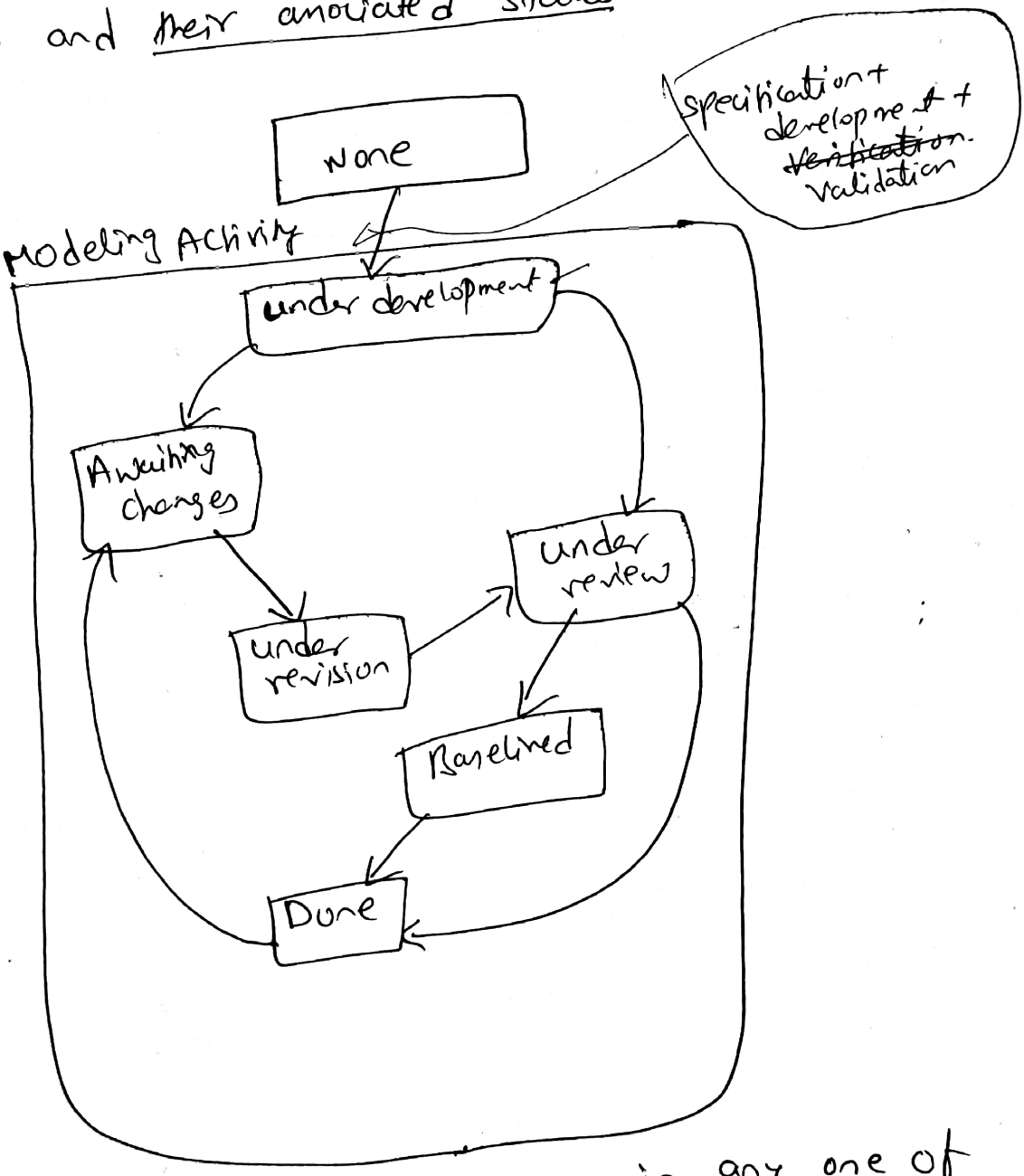
→ It enables the developer to apply the prototyping approach at any stage in the evolution of the product.

→ The spiral model demands a direct consideration of technical risks at all stages of the project, and if properly applied, should reduce risks before they become problematic.

Concurrent Development model

→ It is also called as concurrent engineering, which can be represented schematically as a series of framework activities, s/w Engineering actions & tasks, and their associated states

→



→ The activity - Modeling may be in any one of the states.

→ All ~~exes~~ activities exist concurrently but reside in different states.

→ In a project the communication activity has completed its first iteration and exists in the awaiting changes state.

→ The modeling activity which existed in the none state while initial communication was completed, now makes a transition into the under development state.

→ If customer indicated that changes in requirements must be made, the modeling activity moves from under development state into the awaiting changes state.

→ Concurrent process model defines a series of events that will trigger transitions from state to state for each of the SW engineering activities, actions or tasks.

→ During early stages of design, an inconsistency in the analysis model is uncovered. This generates the event analysis model correction which will trigger the analysis action from the done state into the awaiting changes state

→ The concurrent process model is applicable to all types of SW development and provides an accurate picture of the current state of a project.

→ Rather than containing SW engineering activities, actions & tasks. to a sequence of events.

→ It defines a nlw of activities.

→ Events generated at one point in the process nlw trigger transitions among the states.

Advantages

→ easy to understand
→ Accurate picture of the current state.

Disadvantages

→ Requires more requirements to understand exact state position of the each state.

Unified process models

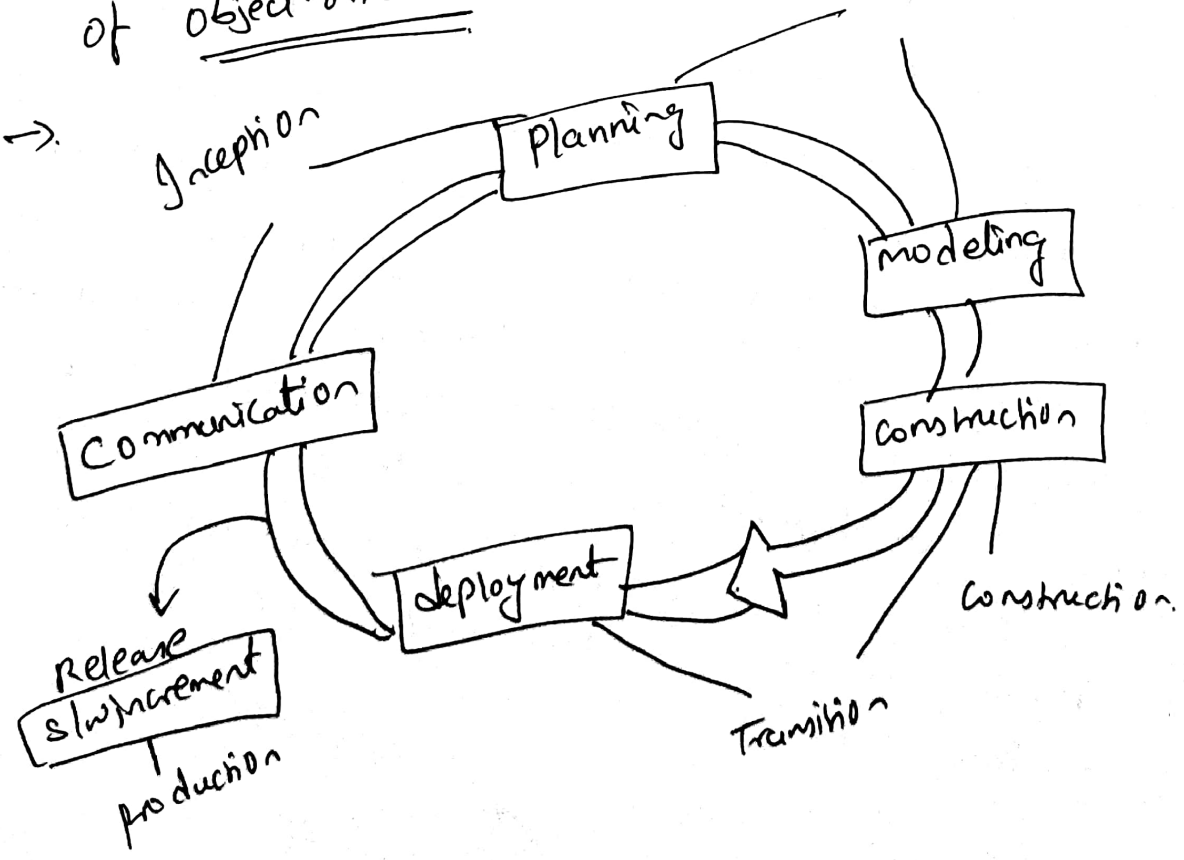
→ It recognizes the importance of customer communication and streamlined methods for describing the customer view of a system. (use-cases).

→ It emphasizes the important role of s/w architecture and "helps" the architect to focus on the right goals, i.e understandability, reliance to future changes, & reuse.

→ It suggests a process flow that is iterative and incremental, providing the evolutionary feel that is essential in modern s/w development.

→ Unified process, a framework for object-oriented s/w engineering using UML [unified modeling language]. UML uses notations for the modeling and development of object-oriented system

→



→ The inception phase of the up (unified process) encompasses both customer communication and planning activities.

→ By collaborating with the customer and end-users, business requirements for the slw are identified, → A rough architecture for the system is proposed and a plan for the iterative, incremental nature of the project is developed.

→ Fundamental business requirements are described through a set of preliminary use-cases that describe what features and functions are desired by each major class of users.

→ use-case describes a sequence of actions that are performed by an actor, as the actor interacts with the slw.

→ use-cases help to identify the scope of the project and provide a basis for project planning.

Elaboration:- Encompasses the customer communication and modeling activities of the generic process model.

→ Elaboration refines & expands the preliminary usecases that were developed as a part of inception phase and expands the architectural representation to include five different views of the slw.

↳ use-case model, ↳ implementation model
↳ Analysis model ↳ deployment model.
↳ Design model

→ Construction phase :- It is identical to the construction activity defined for the generic slw process.

→ It develops slw components that will make each use-case operational for end-users.

→ All necessary and required features & functions of the slw increment are then implemented in source code.

→ As components are being implemented, unit tests are designed & executed for each.

→ Use cases are used to derive a suit of acceptance tests that are executed prior to the initiation of the next united process phase.

(after delivery)

Transition phase

→ Slw is given to end-users for Beta testing and user feedback reports both defects and necessary changes.

→ Slw team creates the necessary support information Eg:- user manuals, trouble shooting guides, installation procedures that is required for the release.

→ At the conclusion of the transition phase, the slw increment becomes a usable slw release.

Production phase

→ Deployment activity of the generic process.

→ On-going use of the slw is monitered, support for the operating environment is provided, defect reports & request changes are submitted and evaluated.

→ All 3 phases (construction, transition, production) are being conducted at the same time.

Specialized process models

Component based development

↳ The process to apply when reuse is a development object

Formal methods

Emphasizes the mathematical specification of requirements.

Aspect oriented s/w development

↳ It provides a process and methodological approach for designing, specifying, defining, & constructing aspects.

Aspects:- multiple system features, functions, information.

Specialized Process models:-

→ They are applied when a narrowly defined SW Engineering approach is chosen.

Component-Based Development:-

↳ Commercial off-the-shelf (COTS) SW components, developed by vendors who offer them as products, can be used when SW is to be built.

→ It incorporates many of the characteristics of the spiral model.

→ Following steps are used in component based development

(1) Available component-based products are researched and evaluated for the application domain in question.
(2) Component integration issues are considered
(3) A SW architecture is designed to accommodate the components
(4) components are integrated into the architecture.
(5) comprehensive testing is conducted to ensure proper functionality.

Formal Methods model

→ It encompasses a set of activities that leads to formal mathematical specifications of computer software.

→ It enables SW engineer to specify, develop, verify a computer based system by applying a rigorous mathematical notations

→ It is a cleanroom s/w Engineering approach.

→ Ambiguity, incompleteness, inconsistency can be discovered and corrected more easily by using the application of mathematical analysis

→ When formal methods are used during design they serve as a basis for program verification and therefore enable the s/w engineer to discover and correct errors that might otherwise go undetected.

→ It promises "A defect-free software" using formal methods model.

↳ The development of formal models is currently quite time consuming and expensive.

↳ Because few s/w developers have the necessary background to apply formal methods, extensive training is required.

↳ It is difficult to use the models as a communication mechanism for technically unsophisticated customers.

(spiral+concurrent model)
Aspect-oriented s/w development (AOSD) is also referred as aspect oriented programming (AOP) is a relatively new s/w engineering paradigm that provides a process and methodological approach for defining specifying designing and constructing aspects mechanisms beyond subroutines and inheritance for localizing the expression of a cross cutting concern"

(9)

Unified process

1. Use-case driven
2. Architecture centric
3. Iterative & incremental
4. Risk focused
5. object oriented s/w development

5 phases

1. Inception
2. Elaboration
3. construction
4. Transition
5. production

Inception:-

- ↳ define the scope of the system
- ↳ A rough architecture is proposed
- ↳ Identifying risks & determining how & when it will be resolved
- ↳ define estimates of cost, schedule, effort

Milestones (Life cycle objectives)

- ↳ stakeholders agrees on the scope of proposed system.
- ↳ A business case for the project is decided.

functionalities of the software system.

4. **Transition**: In this phase, you'd be rolling out the next iterations to the customer and fixing bugs for previous releases. You would also deploy builds of the software to the customer.

(10)

(2) Elaboration:-
↳ capturing the remaining functionalities
↳ Expanding architecture outline into Architecture baseline.

Milestones: (life cycle Architectures)
↳ Most of the functional requirements are captured using use-case models.
↳ Architecture baseline provides solid foundation of the projection.
↳ project plan for next phase construction.

(3) construction:-
↳ Building a system for initial release.
↳ Incremental & iterating approach will be followed.

Milestones:- Initial operational capability

(4) Transition:-
↳ Release fully functional system to customers
↳ creating necenary support information for release

Milestone:- product release.

11

(5) Production:-

↳ ongoing use of SW is monitored

↳ defects reports & request in changes are submitted & evaluated.

Definition

(12)

The **unified process model** (or **UPM**) is an iterative, incremental, architecture-centric, and use-case driven approach to software development. Let's first take a look at the use-case driven approach.

Use-Case Driven Approach

A **use-case** defines the interaction between two or more entities. The list of requirements specified by a customer are converted to functional requirements by a business analyst and generally referred to as use-cases. A use-case describes the operation of a software as interactions between the customer and the system, resulting in a specific output or a measurable return. For example, the online cake shop can be specified in terms of use cases such as 'add cake to cart,' 'change the quantity of added cakes in cart,' 'cake order checkout,' and so on. Each use case represents a significant functionality and could be considered for an iteration.

Architecture-Centric Approach

Now, let's take a closer look at the **architecture-centric approach**. Using this approach, you'd be creating a blueprint of the organization of the software system. It would include taking into account the different technologies, programming languages, operating systems, development and release environments, server capabilities, and other such areas for developing the software.

Iterative and Incremental Approach

And finally, let's take a closer look at the **iterative and incremental approach**.

Using an iterative and incremental approach means treating each iteration as a mini-project. Therefore, you'd develop the software as a number of small mini-projects, working in cycles. You'd develop small working versions of the software at the end of each cycle. Each iteration would add some functionality to the software according to the requirements specified by the customer.

Now that we saw the distinctive characteristics of the unified process model, let's take a look at the process steps involved.

Unified Process Model Phases

We'll go through the four different phases, one at a time, here:

1. **Inception**: The inception phase is similar to the requirements collection and analysis stage of the waterfall model of software development. In this phase, you'd collect requirements from the customer and analyze the project's feasibility, its cost, risks, and profits.

2. **Elaboration**: In this phase, you'd be expanding upon the activities undertaken in the inception phase. The major goals of this phase include creating fully functional requirements (use-cases) and creating a detailed architecture for fulfillment of the requirements. You'd also prepare a business case document for the customer.

3. **Construction**: In this phase, you'd be writing actual code and implementing the features for each iteration. You'd be rolling out the first iteration of the software depending on the key use-cases that make up the core

System requirements:-

→ They are expanded versions of the user requirements that are used by s/w Engineers as the starting point for the system design.

→ System requirements adds details and explain how the user requirements should be provided by the system.

→ They used as a part of contract for the implementation of the system and should be a complete and consistent specification of the whole system.

→ System requirements describes the external behaviour of the system and its operational constraints.

→ Natural language is often used to write system requirements specifications as well as user requirements.

→ System requirements are more detailed than user requirements.

→ Natural language specifications can be confusing and hard to understand

(i) Using of same words for the same concept.
(ii) Reader must find out when requirements are the same & when they are distinct.
(iii) Difficult to find all related requirements.

Notations	Descriptions
(1) Structured natural language.	Depends on defining standard forms of templates to express the requirements specifications
(2) Design description languages	Uses a language like a programming language but with more abstract features to specify the requirements by defining an operational model of the system.
(3) Graphical notations	A graphical language, supplemented by text annotations is used to define the functional requirements of the system. Eg:- Usecases, Sequence diagram, SADT (Structured analysis & design technique)
(4) Mathematical specifications	notations based on mathematical concepts such as finite state machines or sets.

→)

Software requirements document :- Sometimes also called as Software requirements specifications or (SRS).

→ It is an official statement of what the system developes should implement.

→ It should include both user requirements for a system and a detailed specification of the system requirements.

→ The user and system requirements may be integrated into a single description.

→ User requirements are defined in introduction part.

IEEE Standard suggests the following structure for requirements documents.

(1) Introduction.
- 1.1 purpose of requirements document
- 1.2 scope of the product
- 1.3 Definitions, acronyms and abbreviations
- 1.4 References
- 1.5 Overview of the remainder of the document.

(2) General description.
- 2.1 product perspective
- 2.2 Product functions
- 2.3 User characteristics
- 2.4 General constraints
- 2.5 Assumptions and dependencies

(3) Specific requirements covers functional, non-functional and interface requirements. (4)

The requirements may document external interfaces, describe system functionality and performance, specify logical database requirements, design constraints, emergent system properties and quality characteristics.

(4) Appendices [a section at the end of a book or document]

(5) Index.

Users of a requirements document

System customers	Specify the requirements and read them to check that they meet their needs. Customers specify changes to the requirements.
Managers	Use the requirements document to plan a bid for the system and to plan the system development process
System Engineers	Use the requirements to understand what system is to be developed
System test engineers	Use the requirements to develop validation tests for the system
System maintenance engineers	Use the requirements to understand the system and the relationships btw its parts

Software requirements:-

↳ The requirements for a system are the description of the services provided by the system and its operational constraints.

↳ The process of finding out, analysing, documenting and checking these services and constraints is called requirements engineering

↳ Requirements is simply a high level, abstract statement of a service that the system should provide or a constraint on the system.

↳ Requirements are explained by
(1) Functional requirements
(2) Non-functional requirements
(3) User requirements
(4) system requirements.

} described by SRS Software Requirement Specification.

Functional requirements:-

↳ Functionality of the system.

↳ Functional requirements for a system describes "what ~~should~~ system should do".

↳ Requirements totally depend on the type of sw being developed.

↳ FSR describe the system function in detail its inputs and outputs, exceptions and so on.

→ Functional Requirements specification of a system should be both complete and consistent

→ Completeness means that all services required by the user should be defined.

→ Consistency means that requirements should not have contradictionary definitions.

→ Eg:- Functional requirements for a university library system called LIBSYS. used by the students and faculty to order the books. and documents from other libraries.

① user shall able to search all the initial set of databases or set of subsets from it.

② system should provide appropriate views for the user to read documents in the document store.

③ Every order shall be allocated a unique identifier (ID) which the user shall be able to copy to the accounts storage area.

The above user requirements defines a specific facilities provided by the system.

Functional requirements should include the following things

(1) Details of operations conducted in every screen.
(2) Data handling logic should be entered into system.
(3) It should have descriptions of system reports or other outputs.
(4) complete information about the workflows performed by the system.

Benefits of FR

↳ Helps to define the functionality of a system or one of its subsystems.
↳ Clearly define the expected system services and behavior.

Types of FR

↳ Transaction handling
↳ Business rules
↳ certification requirements
↳ Audit tracking
↳ Historical data management
↳ Administrative functions.

Non-functional requirements

↳ These are constraints on the services or functions offered by the system.

↳ They include timing constraints, constraints on the development process and standards.

↳ These are not directly concerned with the specific functions delivered by the system.

↳ They relate to systems Reliability, response time, storage capacity.

↳ capabilities of I/O devices and data representations used in system interfaces.

↳ specify system performance, security, availability, efficiency, stability, portability etc.

∴ Total s/w Qualities is concentrated.

→ If you fail to meet the non-functional requirements then the entire system becomes unusable

→ NFR (Non-functional requirements) arised by user needs.

→ Because of budget constraints, organizational policies, need for interoperability with other s/w & h/w systems.

The types of non-functional requirements are (5)

(1) Product requirements :- These requirements specify product behaviour.

Efficiency
- ↳ performance requirements on how fast the system must execute.
- ↳ space requirements how much memory it requires.

↳ Reliability requirements that set out the acceptable failure rate.

↳ Portability requirements [ability of doing any work necessary to make the computer program run in the new environment]
i.e ability to easily carried out or move the changes for the system.

↳ usability requirements is the degree to which a s/w can be used by specified consumers to achieve quantified objectives with effectiveness, efficiency and satisfaction in a quantified context of use.

i.e how easily the system can be used by an end user.

(2) Organizational requirements:-

↳ Requirements derived from policies and procedures in the customers and developers organisation.

↳ process standards must be used.

↳ Implementation requirements such as programming language or design method used.

↳ Delivery requirements that specifies when the product & its documentation to be delivered.

(3) External requirements

→ Interoperability requirements that define how the system interacts with systems in other organizations.

→ Legislative requirements must be followed to ensure that the system operates within the law

→ Ethical requirements are requirements placed on a system to ensure that it will be acceptable to its users and the general public

Principles

∴ NFR such as safety & security requirements are particularly important for critical systems

User requirements

→ The user requirements for a system should describe the functional and non functional requirements so that they are understandable by system without detailed technical knowledge.

→ They should only specify the external behaviour of the system.

→ When you are writing user requirements, you should not use S/W jargon, structured notations, formal notations,

→ User requirements in simple language with simple tables and forms and intuitive diagrams.

→ Various problems can arise when requirements are written in natural language sentence in a text document.

① Lack of clarity:- It is sometimes difficult to use language in a precise and unambiguous way without making the document wordy and difficult to read.

② Requirements confusion:- Functional & non-functional requirements systems goals and design information may not be clearly distinguished.

(3) Requirements analgamation:- Several different requirements may be expressed together as a single requirement.

User requirements include too much information Constrain the freedom of the system developer to provide innovative solutions to user problems and are difficult to understand.

→ Mainly focus on the key facilities to provide.

Guidelines for writing user requirements to overcome misunderstandings.

(1) Invent a standard format and ensure that all requirements definitions adhere to that format.

(2) Use language consistently Always distinguish b/w mandatory and desirable requirements.

(i) Mandatory requirements are requirements that the system must support and usually written using "shall"

(ii) Desirable requirements are not essential and are written using "should"

(3) Use text highlighting bold, italic or colour to pick out key parts of the requirements.

(4) Try to avoid the use of computer jargon (noun.)
↓
special words, expressions,

Requirements validation:-

(5)

→ Shows that the requirements actually define the system that the customer wants.

→ Finding the problems with requirements.

→ It is important because errors in a requirements document can lead to extensive rework costs when they are discovered during development or after the system is in service.

During requirements validation process, it is necessary to check the requirement document.

(1) validity checks:- (checks when a function is performing correctly)

(2) consistency checks (requirements in the document should not conflict).

(3) completness checks (requirements document should include requirements which define all functions, constraints by the system user)

(4) Realism checks (using knowledge of existing technology)

(5) verifiability (to reduce the potential for dispute b/w customer and contractor)

A no. of requirements validation techniques can be used

(1) Requirements reviews:- Requirements are analysed systematically by a team of reviewers.

(2) Prototyping:- A executable model of the system is demonstrated to end-users & customers to see if it meets their real needs of requirements.

(3) Test-case generation:- Requirements should be testable
→ If a test is difficult or impossible to design, it means that the requirements will be difficult to implement and should be reconsidered.

Requirements Reviews:-
→ It is a manual process that involves people from both client and contractor organizations.
→ They check the requirements document for anomalies and omissions.
→ The review process may be managed in the same way as program inspections.
→ Requirements reviews can be informal or formal
→ Informal reviews simply involve contractors discussing requirements with as many system stakeholders as possible
→ Many problems can be detected simply by talking about the system to stakeholders before making a commitment to a formal reviews.

Formal Reviews:- The development team should 'walk' the client through the system requirements.

→ Explaining the implications of each requirement.

→ The review team should check each requirement for consistency as well as check the requirements as a whole for completness

→ Reviewers may also check for

(1) Verifiability:- is the requirement as stated realistically testable?

(2) Comprehensibility:- Do the procurers or endusers of the system properly understand the requirement.

(3) Traceability:- Is the original requirements are clearly stated?

(4) Adaptability:- Is requirements adaptable? i,e can the requirements be changed without large-scale effects on other system requirements.

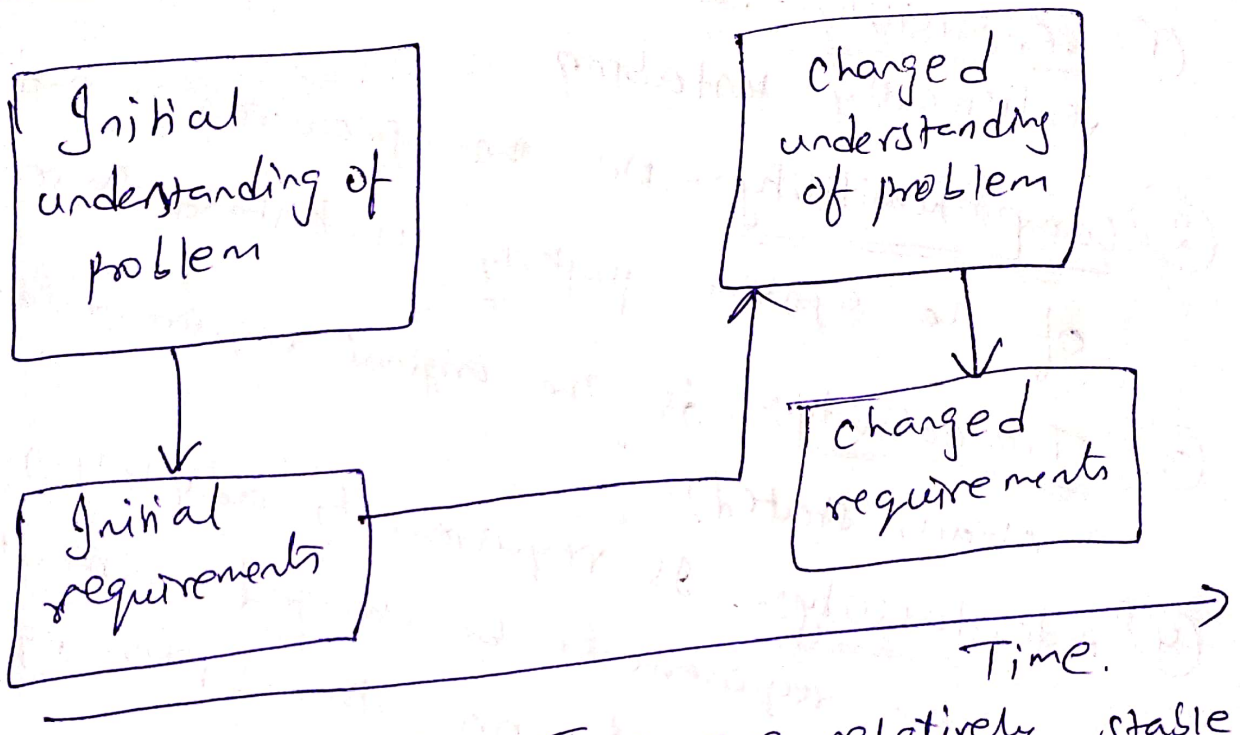
Requirements Management:- It is a process of understanding and controlling changes to system requirements.

→ Track the individual requirements and maintain links blw dependent requirements so that we can ames the impact of requirements changes.

→ A formal methodology procen is required for making changes of proposals and linking to these system requirements

(1) Enduring and volatile requirements
(2) Requirements management planning } Types of Requirement management.
(3) Requirements change management

(1) Enduring and volatile requirements



(1) Enduring requirements:- These are relatively stable requirements that derive from the core activity of the organization and which relate directly to the domain of the system.

Eg:- Hospital
- patients
- doctor
- nurses
- treatments

} Core domain of hospital requirements

(2) volatile requirements:- Requirements likely to change during system development procen or after the system has been become operational (in use) Eg:- Government health care policies changes

(2) Requirements management planning :-

→ It is an essential first stage in the requirement management process.

→ It is very expensive

→ planning stage establishes the level of requirements management detail that is required.

→ During Requirement management planning you have to decide the following things.

(i) Requirements identification :- Each requirement must be uniquely identified so that it can be cross-referenced by other requirements.

(ii) A change management process :- It is a set of activities that assess the impact & cost of changes.

(iii) Traceability policies :- Policies define the relationships b/w requirements & system design.
→ It should be recorded and see how these records should be maintained.

(iv) CASE Tools support :- RM involves the procesing of large amount of information about the requirements
→ Tools may may be used range from specialist RM systems to spreadsheets and simple database system.

There are 3 types of traceability information.

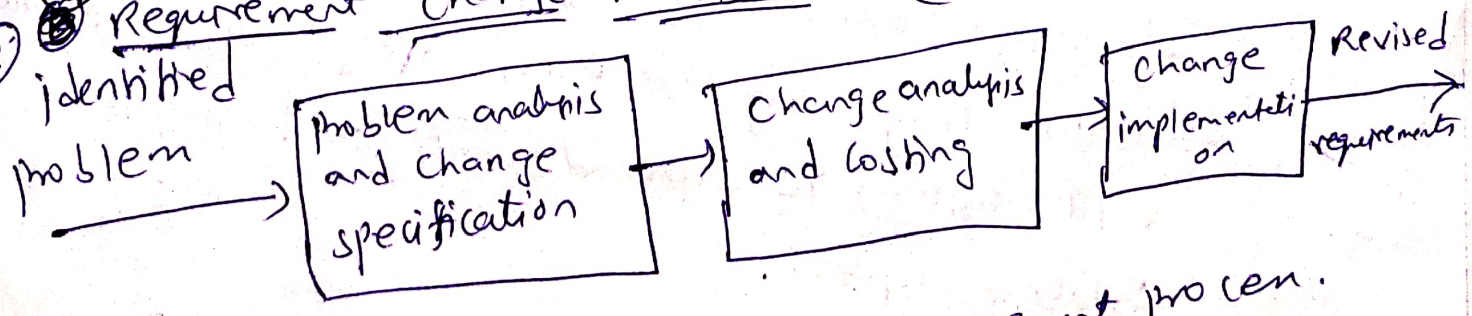
(i) Source traceability information links the requirements to the stakeholders who proposed the requirements and to the rationale for these requirements.

→ When a change is proposed, we use this information to find and consult the stakeholders about the change.

(ii) Requirements traceability information links dependent requirements within the requirements document. i,e how many requirements are likely to be affected by a proposed change and the extent of consequential requirements changes that may be necenary.

(iii) Design traceability information links the requirements to the design modules where these requirements are implemented. i,e impact of proposed requirements changes on the system design and implementation.

(4) Requirement change management (RCM)



Requirements change management process.

→ RCM should be applied to all proposed changes to the requirements. (11)

→ There are 3 principal stages to a change management process.

(i) Problem analysis and change specification :-

→ It starts with an identified requirement problem or sometimes with a specific change proposal.

→ The problem or change proposal is analysed to check whether it is a valid or not.

→ The results of analysis ~~are~~ sends the feedback to the change requester.

(ii) Change analysis and costing :- The effect of proposed change is assessed using traceability information and general knowledge of the system requirements.

→ The cost of making the change is estimated in terms of modifications to the requirements document.

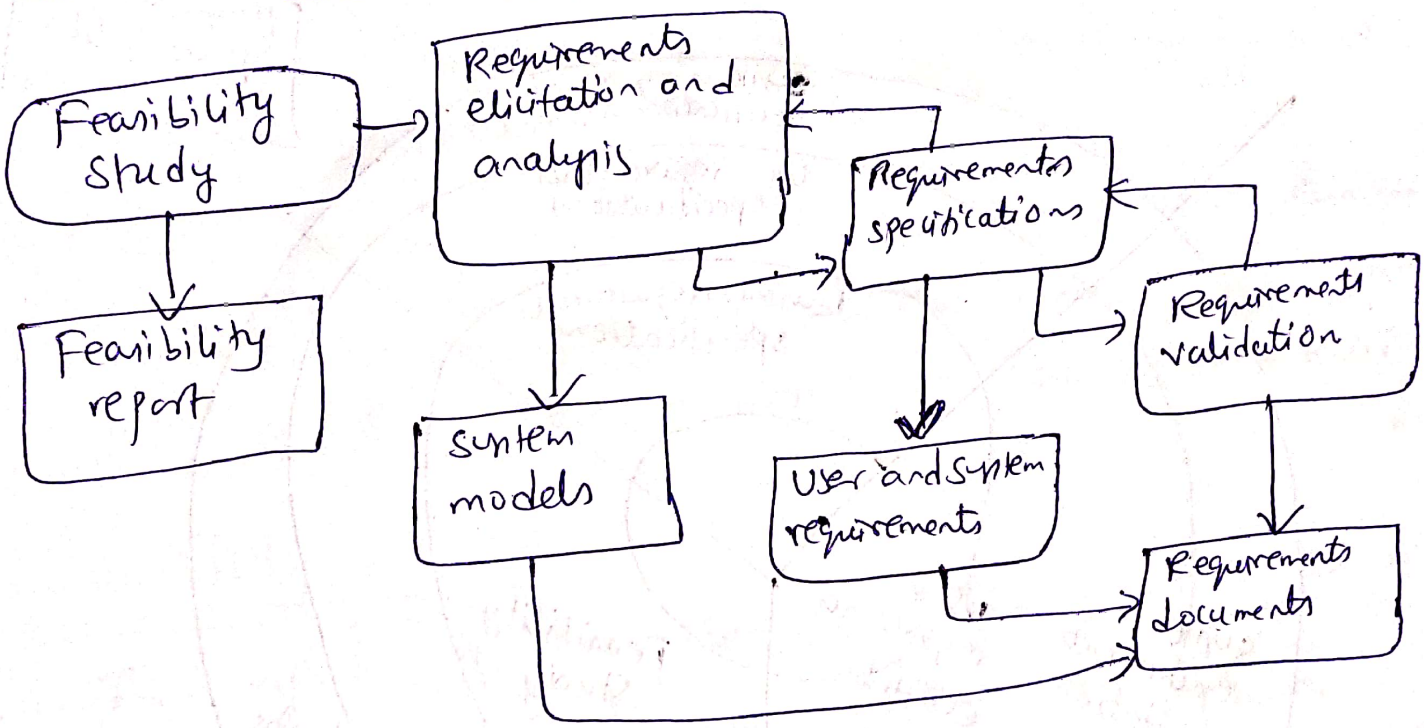
→ once the analysis is completed, a decision is made whether to proceed with the requirement changes.

(iii) Change implementation:- Requirement documents, system design, implementation are modified. (12)

→ Organise the requirements document so that you can make changes to it without extensive rewriting or reorganisation.

→ Individual sections can be changed and replaced without affecting other parts of the document.

Requirement Engineering Process:-



→ The goal of the requirements engineering process is to create and maintain a system requirements document.

→ The overall process includes 4 high-level requirements engineering sub processes.

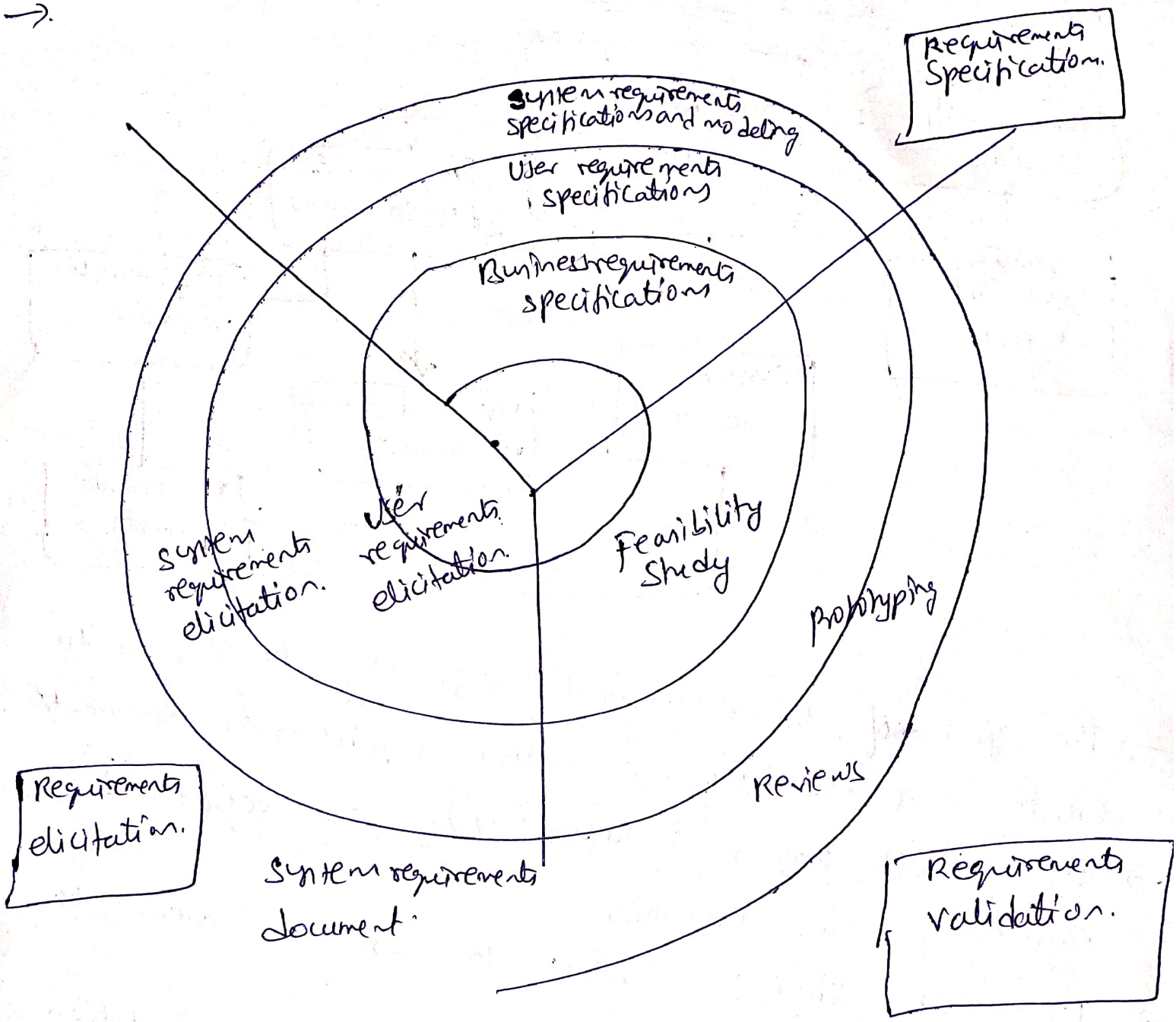
→ Feasibility Study:- Whether the system is useful to the business.

→ Discovering the requirements is Requirement elicitation and Analysis

→ Converting these requirements into some standard form is Requirement Specifications.

→ Checking the requirements actually define the system that the customer wants is Requirement validation

→



Spiral model of requirements engineering processes.

→ The amount of time and effort devoted to each activity in an iteration depends on the stage of the overall process and the type of system being developed.

→ Most effort will be spend on understanding high level business and non-functional requirements and the user requirements.

→ outer rings of spiral, more effort will be devoted to system requirements engineering and system modelling.

→ The no. of iterations around the spiral can vary, so the spiral can be exited after some or all of the user requirements have been elicited.

Feasibility Studies:-

→ The input of the feasibility study is a set of preliminary business requirements, an outline description of the system and how the system is intended to support business processes.

→ A Feasibility study focus on the answers for the following questions.

(i) Does the system contribute to the overall objectives of the organisation?

(ii) can the system be implemented using current technology and within given cost and schedule constraints?

(iii) can the system be integrated with other systems which are already in place?

Feasibility study involves ① Information amenment ② Information collection ③ report writing.

Information assesment :- Identifies the information that is required to answer for the above Questions stated

Information collection :- Information is identified by "talk" with different stakeholders.

(i) How would the organisation scope if this system is not implemented

(ii) What are the problems with current processes and how would a new system help to alleviate these problems.

(iii) what must be supported by the system and what need not be supported.

(iv) can information be transferred to and from other organisational systems.

(v) Does the system require technology that has not previously been used in the organization.

(vi) what direct contribution will the system make to the business objectives and requirements.

Information collection sources such as Managers of the departments where the system will be used, software Engineers who are familier with the type of system is proposed, technology expert and end users of the system.

→ once information collection is over, feasibility study report is done.

→ In the report, propose changes of the scope, budget and schedule of the system, further high-level requirements for the system.

Requirements Elicitation and Analysis

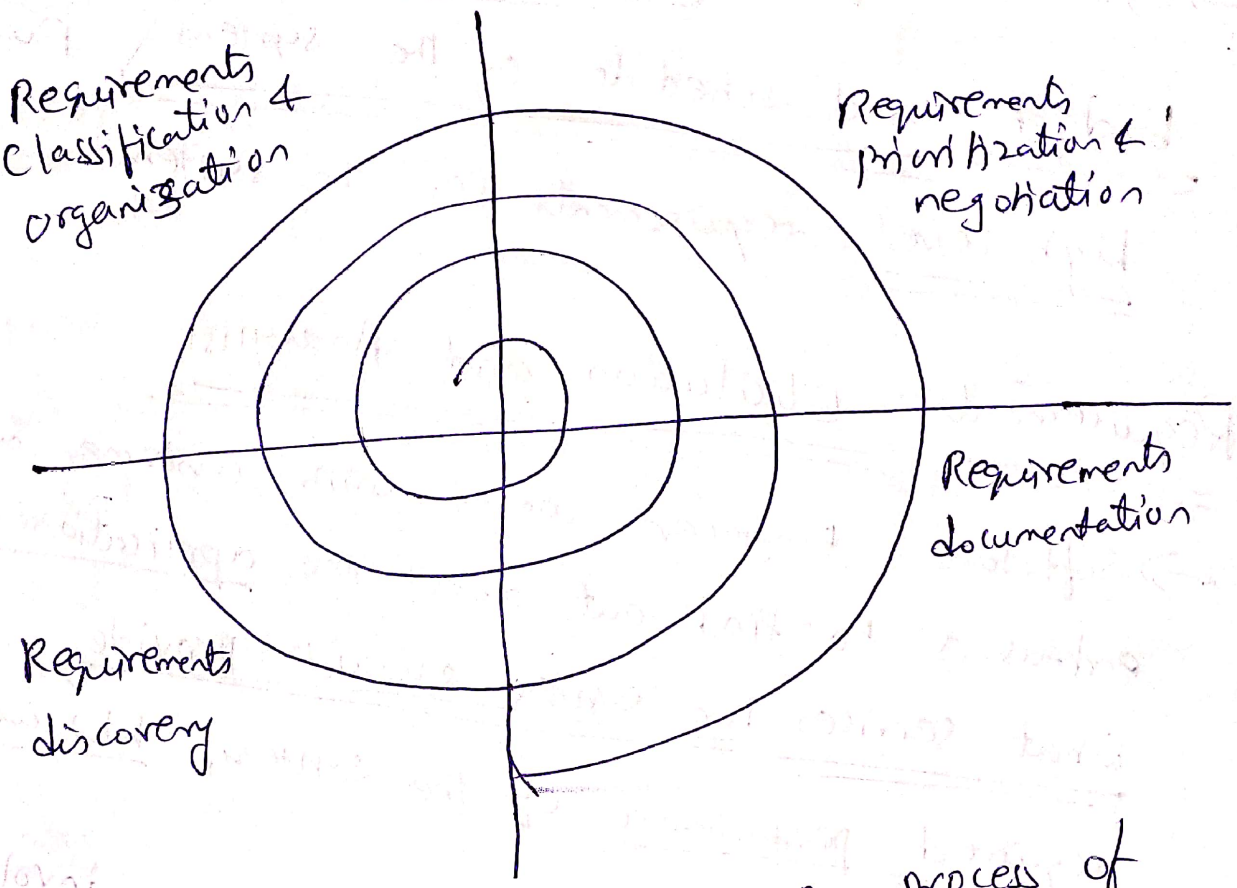
→ Software Engineers works with customer and system end-users to find out about the application domain, what services the system should provide, the required performance of the system, hlw constraints etc.

→ Requirements elicitation and analysis involve a variety of people in an organisation.

→ The term stakeholders is used to refer to any person or group who will be affected by the system, directly or indirectly.

→ Stakeholders include end-users who interact with the system and everyone else in an organization that may be affected by its installation.

→ Other system stakeholders may be engineers who are developing or maintaining related systems, business managers, domain experts and trade union representatives.



① Requirements discovery :- This is the process of interacting with stakeholders in the system to collect their requirements.
→ Domain requirements from stakeholders and documentation are also discovered during this phase.

② Requirements classification and organisation :- This activity takes the unstructured collection of requirements, groups related requirements and organises them into coherent clusters

(3) Requirements prioritisation and negotiation:- where multiple stakeholders are involved, requirements will conflict. This activity is concerned with prioritising requirements, finding and resolving requirements conflicts with negotiation.

(4) Requirements documentation:- The requirements are documented and input into the next round of the spiral.

Requirements discovery:- It is a process of gathering information about the proposed and existing system.

→ It includes interviewing, seenarios (usecases, sequencediagram), viewpoints

(1) viewpoints:- A key strength of viewpoint oriented analysis is that it recognises multiple perspectives and provides a framework for discovering conflicts in the requirements proposed by different stakeholders.

Viewpoints are 3 generic types.

(1) Interacter viewpoints:- represent people or other systems that interact directly with the system eg:- ATM system [bank customer, bank account database

(ii) Indirect viewpoints :- represents stakeholders who do not use the system themselves but who influence the requirements in some way.
Eg:- ATM system [Management of bank
bank security staff

(iii) Domain viewpoints :- represents domain characteristics and constraints that influence the system requirements.
Eg:- ATM system [Inter bank communications
sbi → HDFC bank atm.
Intercommunication.

~~(2) Interacter~~

(2) Interviewing :-

→ Formal or informal interviews with system stakeholders are part of most requirements engineering processes.

→ Interviews may be of two types
(i) Closed interviews :- Where the stakeholders answers a predefined set of questions.
(ii) open interviews :- Where there is no predefined agenda. The Engineering team explores a range of issues with system stakeholders and hence develops a better understanding of their needs.