SOFTWARE ENGINEERING

Introduction ⇒ • Software Engineering is the product of two words

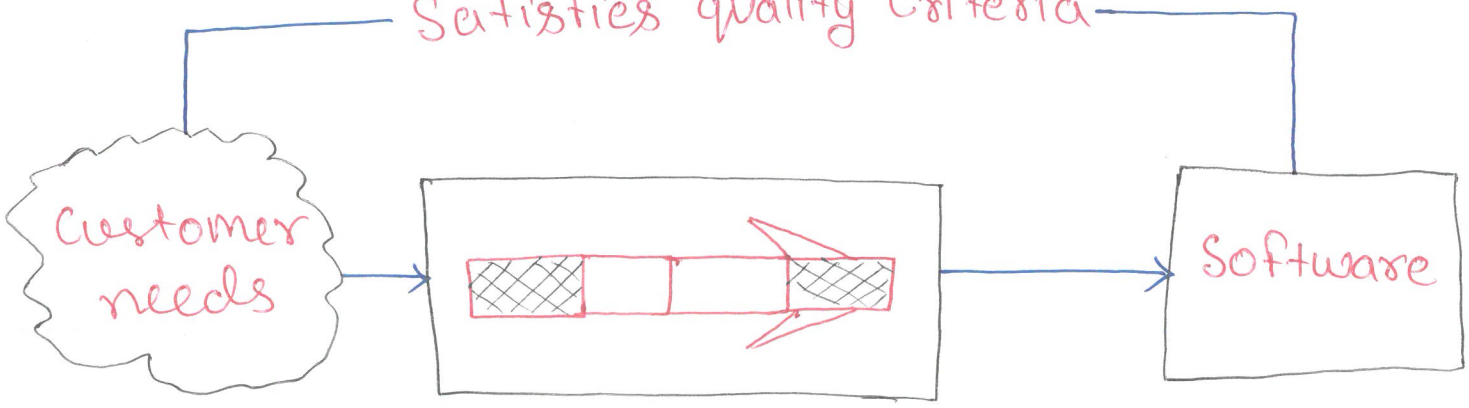
Software + Engineering = Software Engineering

Software ⇒ Software is a program or set of programs containing instructions that provide desired functionality.

Engineering ⇒ Engineering is the process of designing and building something that serves a particular purpose and find a cost-effective solution to problems.

- Software Engineering is the branch of Computer Science that deals with the design, development, testing and maintenance of Software application.
- Software Engineering is a Systematic and disciplined approach to Software development

that aims to Create high-quality, reliable and maintainable software.



Software approach

- Software Engineering includes a variety of techniques, tools and methodologies, including requirement Analysis, design, testing, and maintenance.
- The main goal of Software Engineering is to develop Software applications for improving quality, budget, and time efficiency.
- Software Engineering ensure that the software that has to be built should be Consistent, Correct, also on budget, on time, and within the required requirements.

Key Principles of Software Engineering ⇒ 3

1) Modularity: ⇒ Breaking the software into smaller, reusable components that Can be developed and tested independently

2) Abstraction: ⇒ Hiding the implementation details of a component and Exposing only the necessary functionality to other parts of the software.

3) Encapsulation: ⇒ Wrapping up the data and functions of an object into a single unit, and protecting the internal State of an object from External modifications.

4) Reusability: ⇒ Creating Components that Can be used in multiple projects, which Can save time and resources.

5) Maintenance: ⇒ Regularly Updating and improving the software to fix bugs, add new features, and address security vulnerabilities.

6) Testing: ⇒ Verifying that the software meets its requirements and is free of bugs.

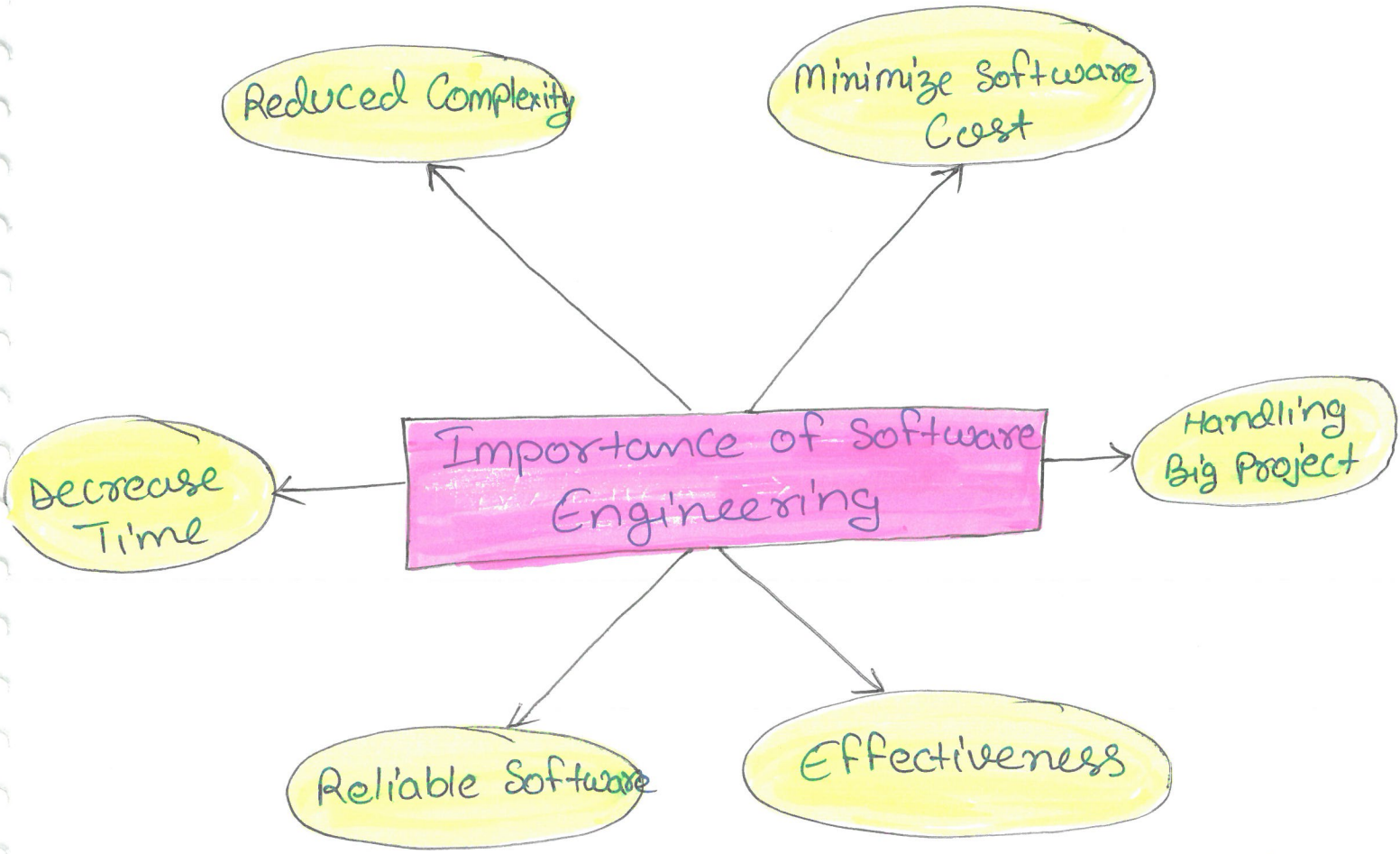
7) Design Patterns: ⇒ Solving recurring problems in software design by providing templates for solving them.

8) Agile Methodologies: ⇒ Using iterative and incremental development processes that focus on customer satisfaction, rapid delivery, and flexibility.

9) DRY principle (Don't Repeat Yourself): ⇒ Avoiding duplication of code and data in the software. This makes the software more understandable, maintainable and less error-prone.

10) KISS principle (Keep it Simple, Stupid): ⇒ Keeping the software design and implementation as simple as possible. This makes the sw more understandable, testable, and maintainable

Importance of Software Engineering



1 ⇒ Reduces Complexity ⇒ • Big Software is always complicated and challenging to progress.

• Software Engineering divides big problems into various small issues. And then start solving each small issue one by one. All these small problems are solved independently to each other.

2 ⇒ To minimize Software Cost ⇒ Software engineers are

are paid highly as software needs a lot of hardwork and workforce development.

- These are developed with the help of a large number of codes. But programmers in software engineering project all things and reduce the things which are not needed.
- As a result of the production of S/w, costs become less and more affordable for S/w that does not use this method.

3 ⇒ To Decrease Time ⇒ • Anything that is not made according to the project always waste time.

- If you are making great software, then you may need to run many codes to get the definitive running code.
- This is a very time-consuming procedure, and if it not well handled, then this can tak a lot of time.
- if you are making your S/w according to the S/w engineering method, then it will decrease a lot of time.

4 ⇒ Handling big projects ⇒ Big projects need lots of patience, planning and management, which you never get from any company.

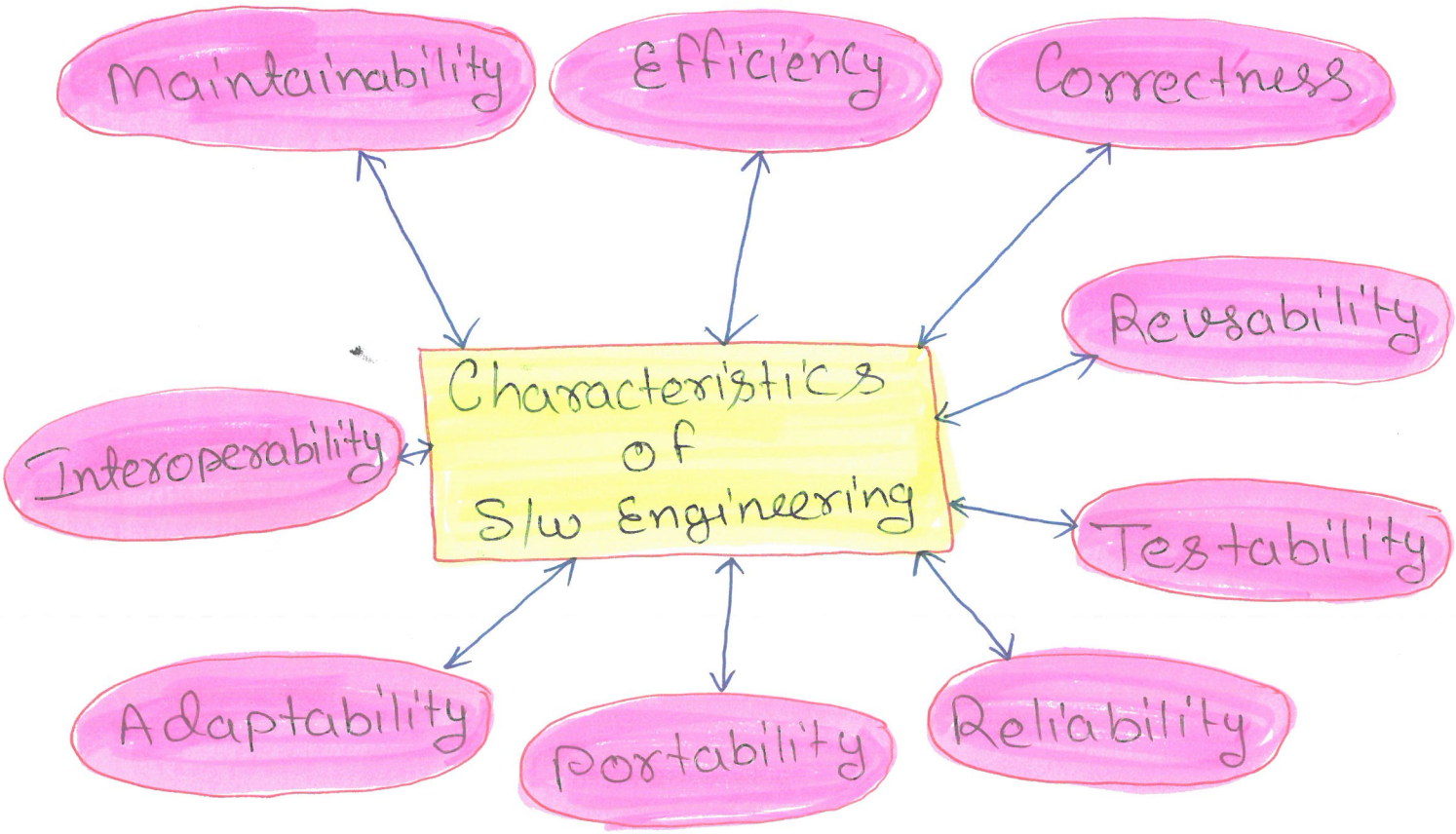
- The company will invest its resources, therefore, it should be completed within the deadline.
- It is only possible if the company uses S/w Engineering to deal with big projects without problems.

5 ⇒ Reliable Software ⇒ The Software will be reliable if S/w Engineering, testing and maintenance are given. As a S/w developer, you must ensure that the S/w is secure and will work for the period or subscription you have agreed upon.

6 ⇒ Effectiveness ⇒ Making Standards decides the effectiveness of things.

Therefore, a company always targets the S/w Standard to make it more effective. and S/w becomes more effective only with the help of S/w Engineering.

Characteristics / objective of S/w Engineering



1 ⇒ Maintainability ⇒ It should be feasible for the S/w to Evolve to meet changing requirements.

2 ⇒ Efficiency ⇒ The software should not make wasteful use of Computing devices such as memory, processor cycles, etc.

3 ⇒ Correctness ⇒ A Software product is Correct if the different requirements Specified in the SRS document has been Correctly implemented.

4⇒ Reusability ⇒ A software product has good reusability if the different modules of the product can easily be reused to develop new products.

5⇒ Testability ⇒ Here S/w facilitates both the establishment of test Criteria and the Evaluation of the S/w with respect to those Criteria.

6⇒ Reliability ⇒ It is an attribute of S/w quality. The extent to which a program can be expected to perform its desired function, over an arbitrary time period.

7⇒ Portability ⇒ the S/w can be transferred from one Computer S/m or Environment to another.

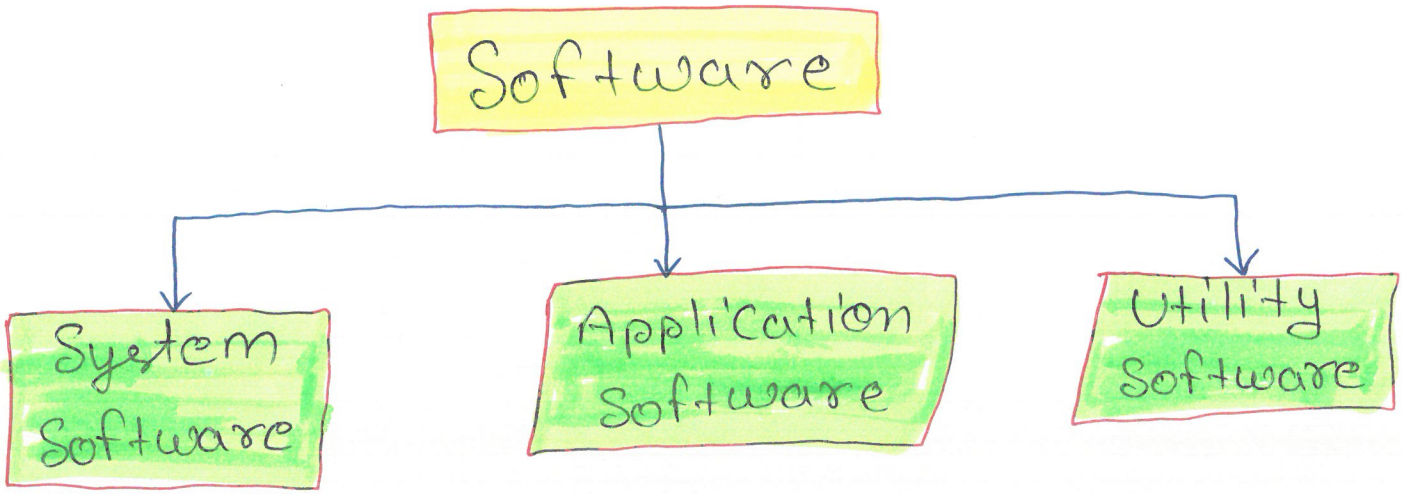
8⇒ Adaptability ⇒ The S/w allows differing S/m constraints and the user needs to be satisfied by making changes to the S/w.

9⇒ Interoperability ⇒ Capability of 2 or more functional units to process data cooperatively.

Introduction to Software

- Software is a set of instructions, data or programs used to operate computers and execute specific tasks.

- Software is a collection of instruction, data or computer programs that are used to run machines and carry out particular activities.
- Software is a set of programs, which is designed to perform a well-defined function.
- Software is a Collection of instructions that enable the user to interact with a Computer.



System Software ⇒

- System Software Control a Computer's internal functioning, and also controls such peripherals as monitors, printers, and storage devices.
- System Software is a set of programs that control and manage the operations of Computer hardware.
- System Software makes the operation of a Computer more fast, effective, and secure.
- OS (operating System) is the best example of System Software. it manages all the other Computer programs.
- Other Example of System Software include the firmware, Computer Language translators and System Utilities.

Features of System Software ⇒

- System Software is Closer to the System.
- Generally written in a Low-Level Language.

- The System software is difficult to design and understand.
- fast in speed.
- Less interactive.
- Smaller in size.
- Hard to manipulate.

Application Software ⇒

- Application Software is a type of Computer program that performs a specific personal, educational, and business function.
- Each application is designed to assist end-users in accomplishing a variety of tasks, which may be related to productivity, creativity or Communication.
- Application Software, or app, is software that performs specific tasks for an end-user.

Features of Application Software

- Perform more specialized tasks like word processing, spreadsheets, email, photo editing, etc.
- It needs more storage space as it is bigger in size.
- Easy to design and more interactive for the user.
- Generally written in a high-Level Language.

Types of Application Software ⇒

⇒ Word processors
⇒ Graphics software
⇒ Database software
⇒ Spreadsheet software
⇒ Presentation software
⇒ Web browsers
⇒ Multimedia software.

Utility Software ⇒ • Utility S/w is S/m Software designed to help analyze, configure, optimize or maintain a Computer.

- Utility Software usually focus on how the Computer infrastructure (including the Computer h/w, operating system, Software and data storage) operates.
- These Software may come along with OS like windows defender, windows disk Cleanup tool, Antivirus, backup Software, file manager, disk Compressior tool all are utility software.

Features of Utility Software ⇒

⇒ Data Compression

⇒ Data Synchronization

⇒ file Synchronization

⇒ Disk Compression

Types of Utility Software ⇒

⇒ System Utilities

⇒ File management Utilities

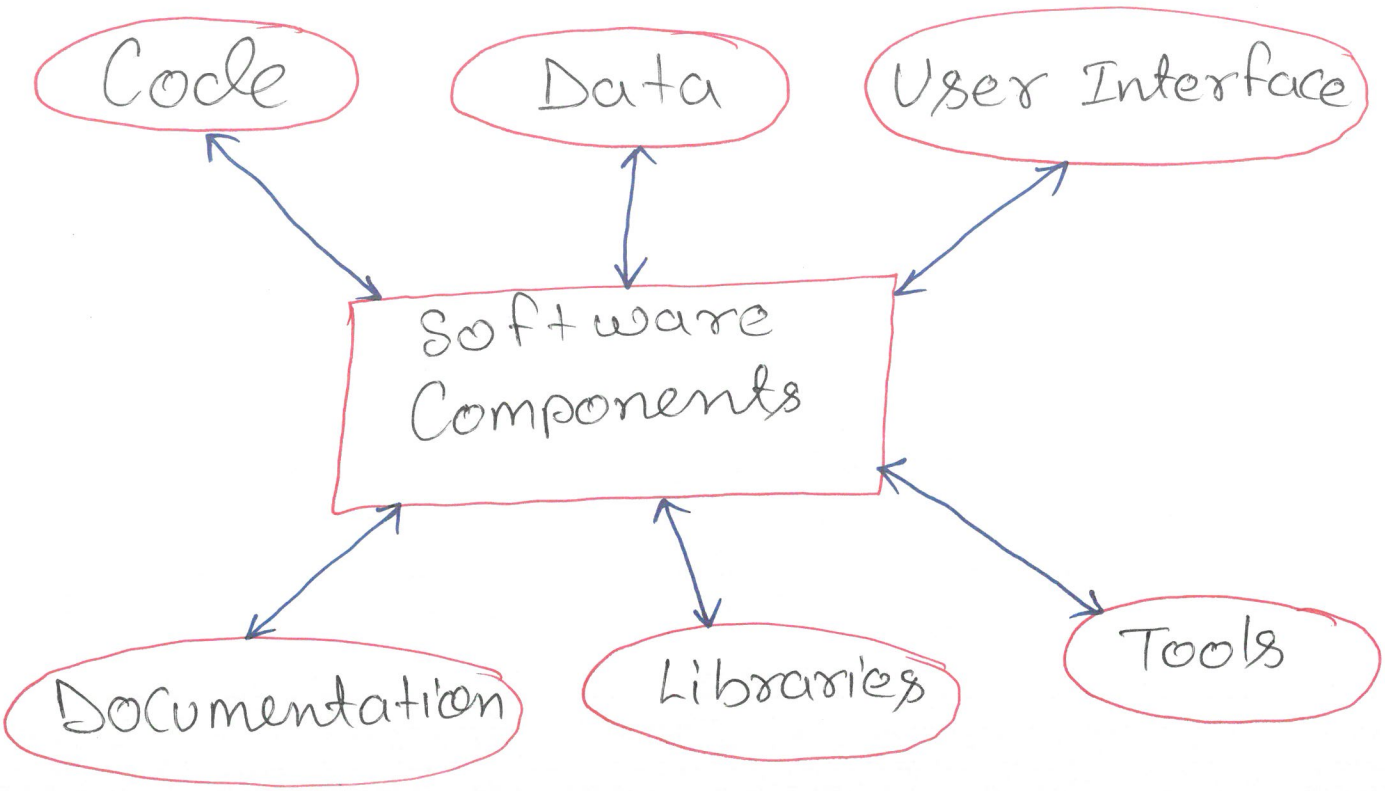
⇒ Storage Device management Utilities.

⇒ Backup Utilities etc.

Software Components

- A Component is a self-contained system that has one or more input channels and one or more output channels. without input it has nothing to do. without output it serves no purpose.
- Computer Slw consists of several basic Components that work together to provide functionality and perform tasks.

The main Components of Computer Slware:



1) Code: ⇒ It is a set of instructions that tell the Computer what to do

and how to do it.

- The code is written in a specific programming Language.

2) Data ⇒ • Software requires data to perform its functions.

- Data can be input from the user or from external source, and the software must process and manipulate the data to produce output.

3) User Interface ⇒ • The user interface is the component that enables users to interact with the Slw.

- It can be a graphical interface, a Command-line interface or a combination of both.

4) Documentation ⇒ • It provides information about the Slw features, functions, and usage, as well as troubleshooting and support information.

5) Libraries ⇒ Libraries are collection

of pre-written code that can be used to perform specific function.

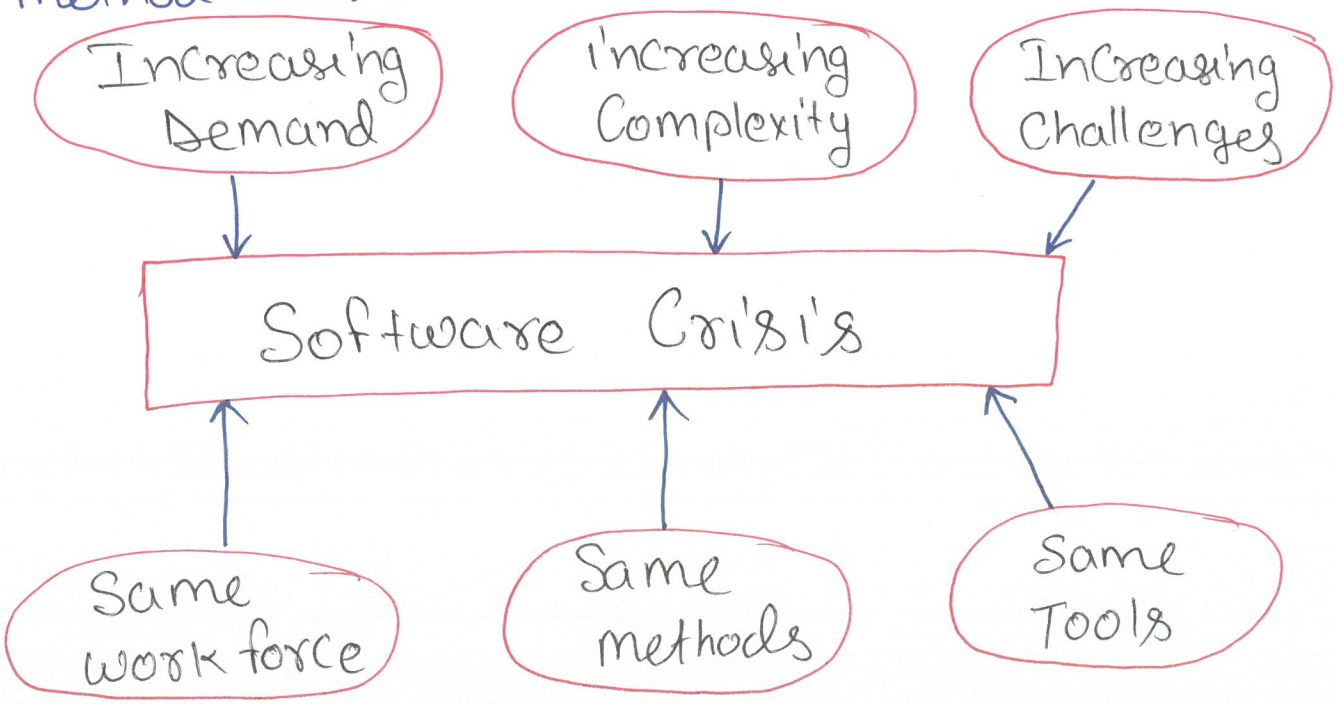
6) Tools ⇒ Software development tools are programs that help developers to create, test, and debug software.

Example of SW development tools include text editors, Compilers and debuggers.

Overall, these components work together to create software that performs specific functions and provides value to users.

Software Crisis & myths

- Software Crisis is a term used in the early days of Computer science for the difficulty of writing useful and efficient Computer programs in the required time.
- The Software Crisis was due to the rapid increases in Computer power and the Complexity of the problems that Could not be tackled.
- The Software Crisis was due to using the Same workforce, same methods and same tools even through rapidly increasing S/w demand, the complexity of S/w and S/w challenges.
- With the increase in software Complexity, many S/w problems arise because existing method were insufficient.



- If we use the same workforce, same methods, and same tools after the fast increase in (20) s/w demand, s/w complexity, and s/w challenges, then there arise some issue like Software budget, software efficiency problem, s/w quality problems, s/w management and delivery problem etc. This condition is called a Software Crisis.

Causes of Software Crisis :⇒

⇒ The cost of owning and maintaining Software was as Expensive as developing the Software.

⇒ At that time Projects were running overtime.

⇒ At that time Software was very inefficient.

⇒ The quality of the Software was Low quality.

⇒ Software often did not meet user requirement

⇒ At the time Software was never delivered.

⇒ Non-optimal resource utilization.

⇒ challenging to alter, debug and enhance.

⇒ The Software Complexity is harder to change.

factors Contributing in S/w Crisis :⇒

- Poor Project management.
- Lack of adequate training in S/w Engineering.
- Less skilled project members.
- Low productivity improvements.

Solution of Software Crisis :⇒

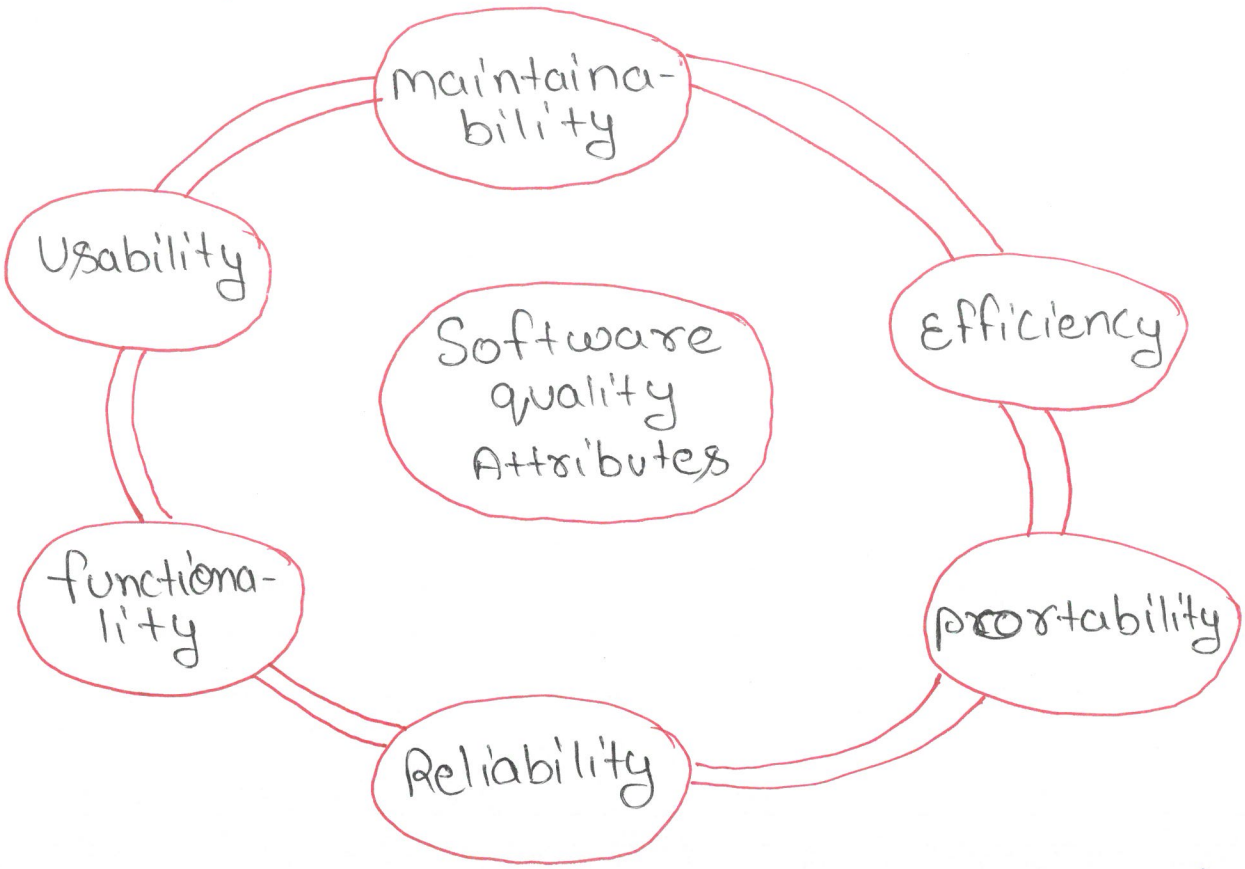
There is no single solution to the Crisis. one possible solution to a Software Crisis is Software Engineering because Software Engineering is a systematic, disciplined, and quantifiable approach.

- For preventing S/w crisis, there are some guidelines.

⇒ Reduction in Software over budget.
⇒ The quality of the software must be high.
⇒ Less time is needed for a S/w project.
⇒ Experienced and skilled people working on the software project.
⇒ Software must be delivered.
⇒ Software must meet user requirements.

Software Quality Attributes

- Software Quality shows how good and reliable a product is.
- Software Quality can be defined as
  - ⇒ Degree of Excellence
  - ⇒ fitness for purpose
  - ⇒ Best for the customer's use and selling price.
  - ⇒ The totality of characteristics of an entity that bears on its ability to satisfy stated or implied needs.



- Quality may be defined from different perspectives. The quality of the product Contributes to improved user requirements

& satisfaction, Clearer software design, and ultimately greater end product quality.

#Factors of Software Quality ⇒ (Attributes)

1) Reliability ⇒ • Software is more reliable if it has fewer failures.

- Since Slw Engineers do not deliberately plan for their software to fail, reliability depends on the number and type of mistakes they make.
- Designers can improve reliability by ensuring the software is easy to implement and change, by testing it thoroughly, and also by ensuring that if failures occur, the system can handle them or can recover easily.

2) Maintainability ⇒ A Software is reparable, if errors may be simply corrected as and once they show up, new functions may be simply added to the mechandise, and therefore the functionalities of the merchandise may be simply changed, etc

3) Usability ⇒ • This attribute refers to the quality of the end user's Experience while interacting with the application or service.

- Usability is concerned with effectiveness, efficiency, and overall user satisfaction.
- This attribute helps to measure the ease of use of any s/w application or service.

4) Portability ⇒ • It refers to the extent to which a system or its Components can be migrated (transported) to other environments consisting of different hardware and/or different software (operating system).

- Sub attributes for portability are adaptability, Replaceability, Installability, and Compatibility.

5) Correctness ⇒ This refers to the ability of software products or services to perform tasks (e.g. Calculations,

Sign-ups, navigations) correctly as specified by the predefined requirements when used under specified conditions.

- It is a measure of whether a S/w S/m is correct or not.

6) Efficiency ⇒ • The more efficient S/w is, the less it uses of CPU-time, memory, disk space, network bandwith and other resources.

- This is important to customers in order to reduce their costs of running the S/w, although with today's powerful computers, CPU time, memory and disk usage are less of a concern than in years gone by.
- Sub-attributes for efficiency are Time Behavior, Resource Utilization, Capacity etc.

7) Security ⇒ • It consist of authorization and authentication techniques, Safeguarding against network attacks, Sending encrypted data wherever necessary, prevent SQL injection.

- Sub-attributes for security are Confidentiality, Authorization and Authentication Integrity and Accountability.

8) Reusability ⇒ A software has smart reusability if completely different modules of the merchandise will simply be reused to develop new merchandise.

Comparison Between S/w Engineering and Computer Science

Software Engineering	Computer Science
1⇒ Software Engineering is the study of how S/w S/m are built.	1⇒ Computer Science is the study of how computers performs theoretical and mathematical tasks.
2⇒ It involves the study and application of S/w only.	2⇒ It involves the study and application of S/w and h/w both.
3⇒ It is structural process of checking, verifying, finding the errors and bugs according to the need of S/w and then provide a solution for removing that bug.	3⇒ It is not a structural process as everything is to be done in a process and requires proper study before Executing.
4⇒ It involves some area of study which are S/w development, S/w testing and quality assurance.	4⇒ It involves area of study which are networking, Artificial intelligence, database system etc.
5⇒ Software Engineering majorly defines architectural and structural properties.	5⇒ Computer Science involves the study of both principals and the use of Computer.

Product and Process

Product ⇒ • In the context of software Engineering, Product includes any Software manufactured based on the customer's request.

- This can be a problem solving software or Computer based system. It can also be said that this is the result of a project.

Process ⇒ • Process is a set of sequence steps that have to be followed to Create a project.

- The main purpose of a process is to improve the quality of the project.

Difference between Product and Process :-

Product	Process
1⇒ Product is the final production of the project.	1⇒ the process is a set of sequence steps that have to be followed to Create a project.
2⇒ product focuses on the final result.	2⇒ The process is focused on completing each step being developed.

Product	Process
3⇒ Product tends to be a short-term aspect	3⇒ Process tends to be a Long-term aspect.
4⇒ Product follows the firm guidelines.	4⇒ Process follows the guidelines consistently.
5⇒ The goal of a product is to complete the work successfully.	5⇒ The goal of a process is to improve the quality of the project.
6⇒ Product is created based on the needs and Expectations of the Customers.	6⇒ A process serves as a model for producing various goods in a similar way.
7⇒ Its main goal is to Complete the work Effectively.	7⇒ Its main goal is to make the better quality of project.

Deliverables and Milestone

Deliverables ⇒ • A deliverable is a measurable and tangible outcome of the project.

- They are developed by project team members in alignment with the goals of the project.
- It simply means result or s/w product, designed document, or asset of project plan that can be submitted to customers, clients or end-users.
- A deliverable should be completed in all aspects.
- It is an element of output within scope of project or processes in the project.

What is a project deliverable?.

- must be tangible (i.e. a product or service)
- Signals the completion of a project phase
- An important sign for the client.
- A point for the client to sign-off on project status.

- **Milestones** ⇒ • Milestones are checkpoints throughout the life of the project.
- They identify when one or multiple groups of activities have been completed thus implying that a notable point has been reached in the project.
- Milestone Can be defined as recognizable endpoint of software project activity. At each milestone, report must be generated.
- It is used as signal post for project start and end date, need for external review or input and for checking budget, submission of the deliverable, etc.

What is project milestone ?.

- Can be Conceptual or tangible.
- Signals the reaching of a key stage in the project.
- An important sign for the team.
- A point for project management to check project goal alignment.

Software Measurement

- Software measurement is a quantified attribute of a characteristics of a S/w product of the S/w process.
- A measurement is a manifestation of the size, quantity, amount, or dimension of a particular attribute of a product or process.
- Software measurment is a titrate impute of a characteristic of a S/w product or the S/w process.
- The S/w measurement process is defined and governed by ISO standard.

Software Measurement Principles :⇒

- The S/w measurement process can be characterized by five activities :-

1⇒ formulation ⇒ The derivation of S/w measures and metrics appropriate for the representation of the S/w that is being considered.

2) Collection ⇒ The mechanism used to accumulate data required to derive the formulated metrics.

3) Analysis ⇒ The computation of metrics and the application of mathematical tools.

4) Interpretation ⇒ The evaluation of metrics results in insight into the quality of the representation.

5) Feedback ⇒ Recommendation derived from the interpretation of product metrics transmitted to the software team.

Need for Software Measurement ⇒

- Create the quality of the current product or process
- Enhance the quality of a product or process.
- Regulate the state of the project concerning budget and schedule.
- Enable data-driven decision-making in project planning and control.

- Ensure that industry standards and regulations are followed.
- Give S/w products and processes a quantitative basis for Evalution.
- Enable the ongoing improvement of S/w development practices.

Classification of Software Measurement:

There are two types of S/w measurement

1⇒ Direct measurement ⇒ In direct measurement the product, process or thing is measured directly using a standard scale.

2⇒ Indirect measurement ⇒ In indirect measurement, the quantity or quality to be measured is measured using related parameters i.e. by use of reference.

Software Metrics

- A metric is a measurement of the level at which any impute belongs to a system product or process.
- It is a quantitative measure of the degree to which a system, component or process possesses a given attribute.
- S/w metrics is s/w Engineering are the Standards for estimating the quality, progress, and health of s/w development activity.
- A s/w metric is a quantifiable or Countable assessment of the qualities of software.

There are 4 functions related to Software metrics:

1 => Planning

2 => Organizing

3 => Controlling

4 => Improving

Characteristics of Software Metrics: ⇒

- **Quantitative**: ⇒ Metrics must be quantitative in order to be helpful. It means that metrics can be stated numerically.
- **Understandable**: ⇒ Metrics computation should be easily understood, and the method of computing metrics should be clearly defined.
- **Applicability**: ⇒ Metrics should be applicable in the initial phases of the development of the software.
- **Repeatable**: ⇒ When measured repeatedly, the metric values should be the same and consistent in nature.
- **Economical**: ⇒ The computation of metrics should be economical.
- **Language Independent** ⇒ Metrics should not depend on any programming language.

Classification of Software Metrics:

1⇒ Product Metrics ⇒ • Product metrics are used to evaluate the state of the product, tracking risk and undercover prospective problem areas.

- The ability of the team to control quality is Evaluated.
- Examples include Lines of code, Cyclomatic Complexity, code coverage, defect density, and code maintainability index.

2⇒ Process Metrics ⇒ • Process metrics pay particular attention to enhancing the Long-term process of the team or organization.

- These metrics are used to optimize the development process and maintenance activity of Slw.
- Examples include effort variance, schedule variance, defact injection rate, and Lead time.

3 ⇒ Project Metrics ⇒ • The project metrics describes the characteristics and execution of a project.

• Example include effort estimation, accuracy, Schedule deviation, Cost variance, and productivity. Usually measures :-

⇒ Number of Software developer.

⇒ Staffing patterns over the life cycle of Slw.

⇒ Cost and Schedule.

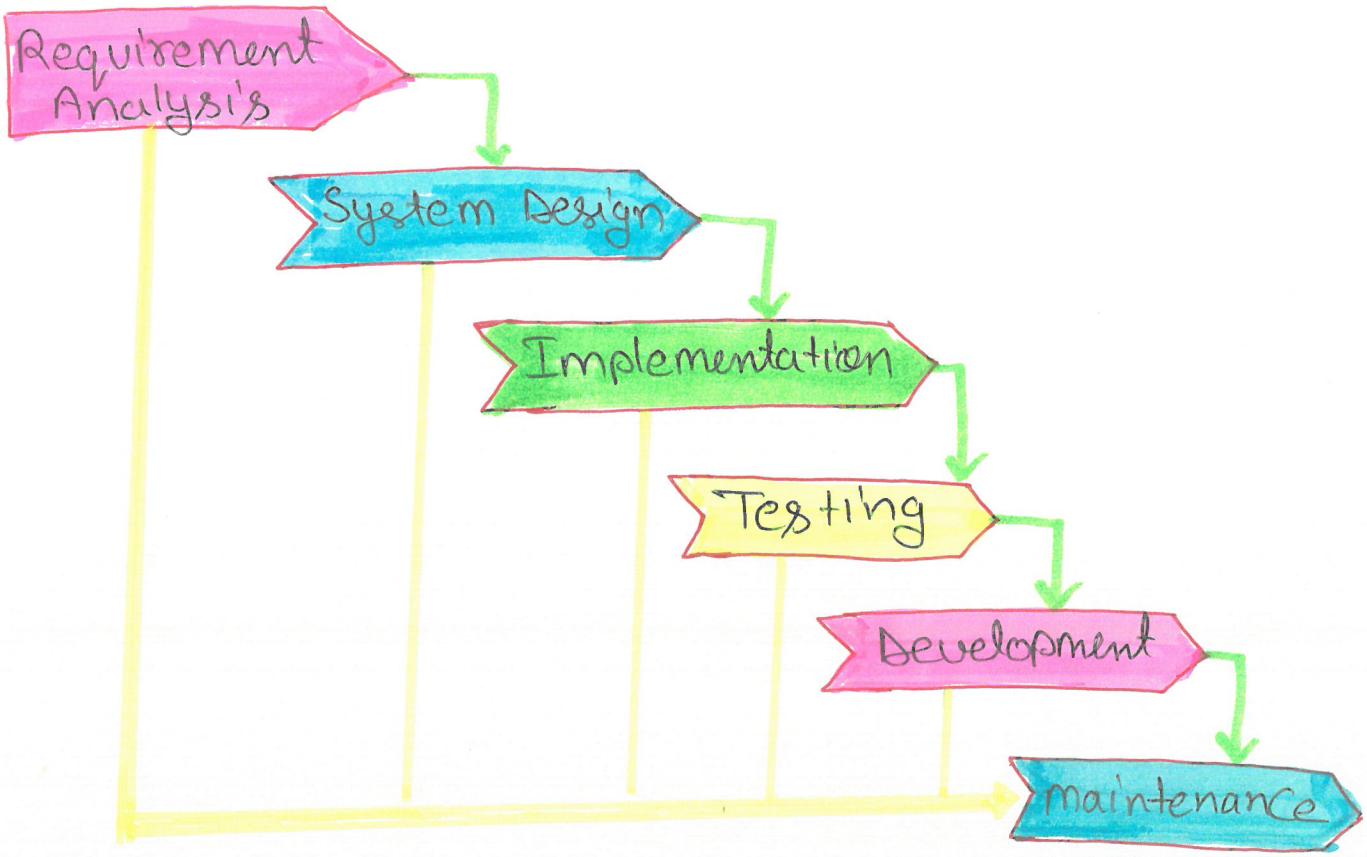
⇒ productivity.

Advantages of Software Metrics :⇒

1. Reduction in Cost or budget.
2. It helps to identify the particular area for improvising.
3. It helps to increase the product quality.
4. Managing the workloads and teams.
5. Reduction in overall time to produce the product.
6. It helps to determine the Complexity of the Code and to test the Code with resources.
7. It helps to providing effective planning, controlling and managing of the entire product.

Software Development Life Cycle (SDLC)

- Software Development life cycle (SDLC) is a structured process that is used to design, develop, and test good-quality s/w.
- SDLC is a systematic process for building s/w that ensures the quality and correctness of the s/w built.
- SDLC process aims to produce high-quality s/w that meets customer expectations.
- SDLC is a methodology that defines the entire procedure of software development step-by-step.



Stage-1: Panning and Requirement Analysis

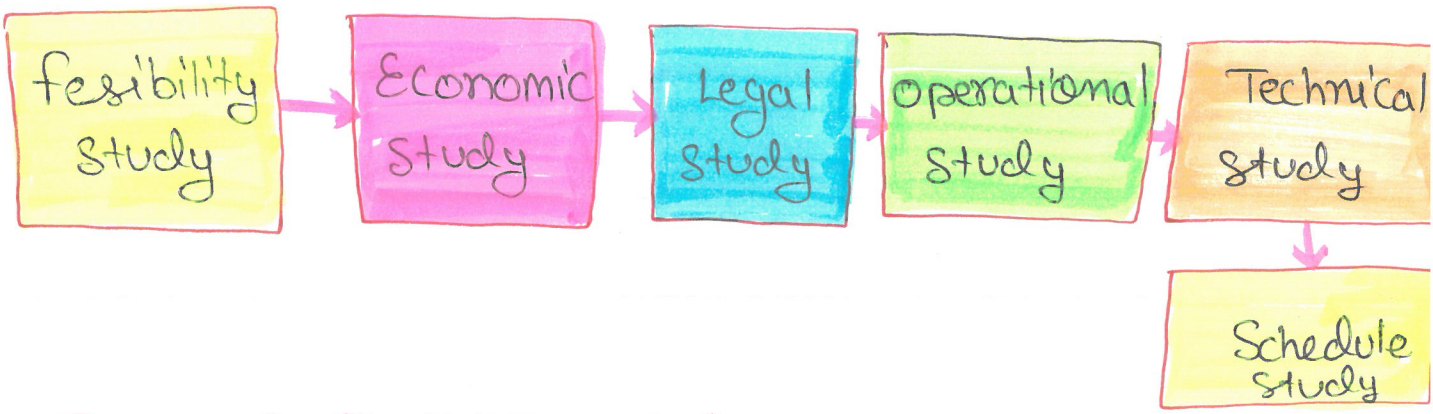
- The requirement is the first stage in the SDLC process.
- It is conducted by the senior team members with inputs from all the Stakeholders and domain Expert in the industry.
- planning for the quality assurance requirement and recognization of the risks involved is also at this stage.
- The information from this analysis forms the building blocks of a basic project.
- The quality of the project is a result of planning. Thus, in this stage, the basic project is designed will all the available information.

Stage-2: Feasibility study

- Once the requirement analysis phase is completed the next SDLC step is to define and document software needs.

- This process conducted with the help of "Software Requirement Specification" document also known as "SRS" document.
- It includes everything which should be design and developed during the project life cycle

There are mainly five types of feasibilities:



1 ⇒ Economic fesibility study ⇒ Can we complete the project within the budget or not?.

2 ⇒ Legal fesibility study ⇒ Can we handle this project as cyber Law and other regulatory framework/ compliances.

3 ⇒ Operation feasibility ⇒ Can We Create operations which is expected by the client?

4 ⇒ Technical feasibility ⇒ Need to check whether the current Computer Sim can support the software.

5 ⇒ Schedule feasibility ⇒ Decide that the project can be completed within the given Schedule or not.

Stage-3: Design :⇒

- In this phase, the system and sw design documents are prepared as per the requirem specification document. This helps define overall system architecture.
- This design phase serves as input for the next phase of the model.

High-Level Design (HLD) ⇒

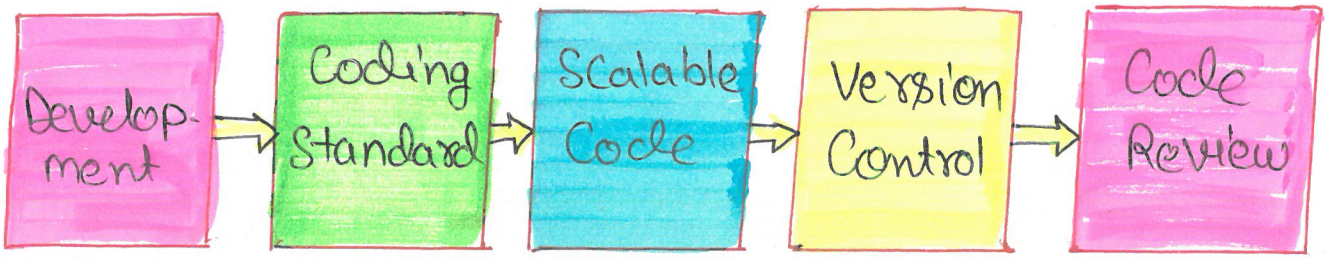
- Brief description and name of each module
- An outline about the functionality of every module.
- Interface relationship and dependencies between modules.
- Database tables identified along with their key elements.
- Complete architecture diagrams along with technology details.

Low-Level Design (LLD) ⇒

- functional Logic of the modules.
- Database tables, which include type and size.
- Complete detail of the interface
- Addresses all types of dependency issues
- Listing of error messages
- Complete input and outputs for every module.

Stage-4 Coding ⇒

- Once the System design phase is over, The next phase is coding.
- In this phase, developers start build the entire system by writing code using the Chosen programming Language.

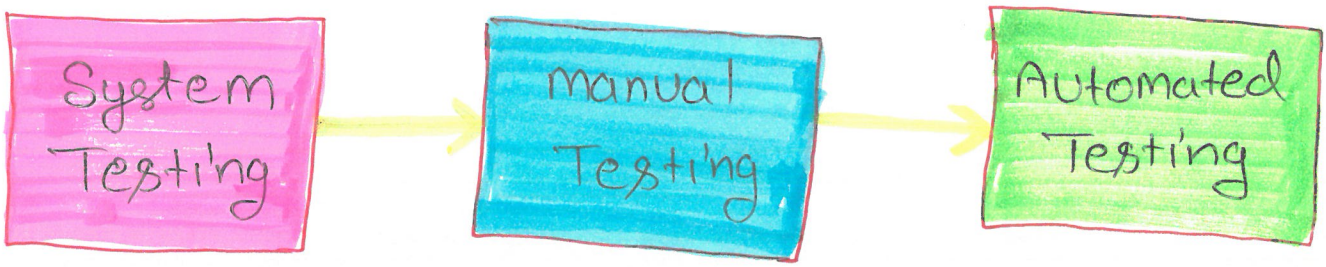


- In the Coding phase, tasks are divided into units or modules and assigned to the various developers.

- It is the Longest phase of the Software Development Life Cycle process.
- In this phase, Developer needs to follow certain predefined coding guidelines. They also need to use programming tools like Compiler, interpreters, debugger to generate and implement the code.

Stage-5: Testing ⇒

- Once the software is complete, and it is deployed in the testing environment.
- The testing team starts testing the functionality of the entire System. This is done to verify that the entire application works according to the customer requirement.



- During this phase, QA and testing team may find some bugs/defects which they Communicate to developers.
- The Developement team fixes the bug and Send back to QA for a re-test.

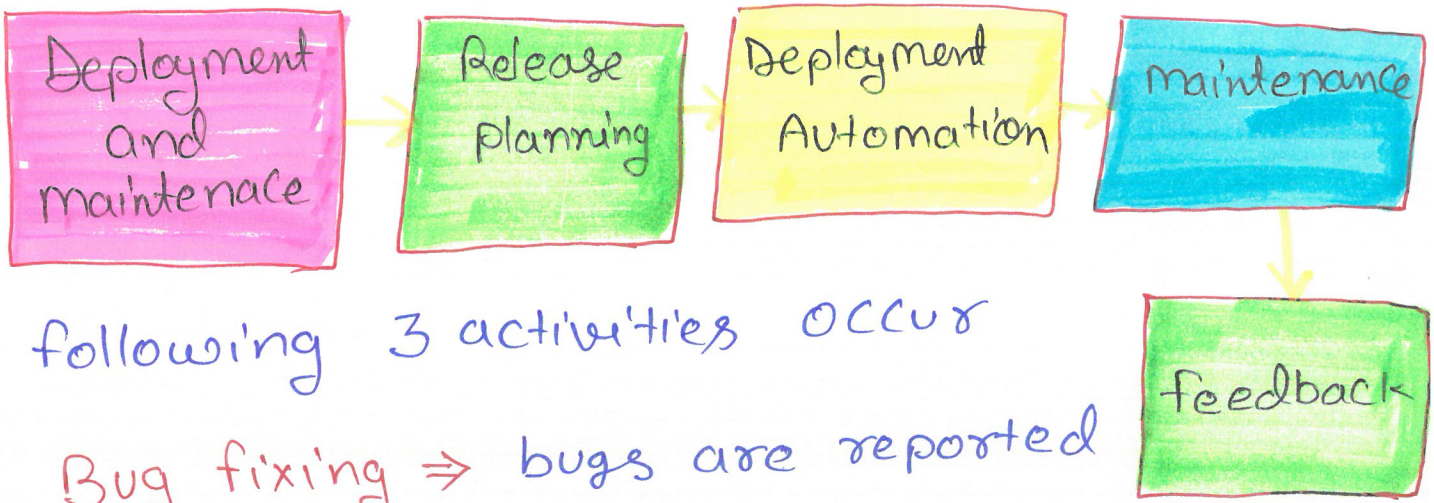
- This process continues until the Slw is bug-free, stable, and working according to the business needs of that system.

Stage 6: Installation/ Deployment ⇒

- Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts.
- Based on the feedback given by the project manager, the final Slw is released and checked for deployment issues if any.

Stage 7: Maintenance ⇒

- Once the System is deployed, and Customers start using the developed system,



- following 3 activities occur

⇒ Bug fixing ⇒ bugs are reported because of some scenarios

which are not tested at all.

⇒ Upgrade ⇒ Upgrading the application to the newer versions of the software.

⇒ Enhancement ⇒ Adding some new features into the existing software.

The main focus of this SDLC phase is to ensure that needs continue to be met and that the system continues to perform as per the specification mentioned in the first phase.

Water fall model

- The waterfall model is a Software developme model used in the context of large, complex projects, typically in the field of information technology.
- It is characterized by a structured, sequential approach to project management and Software development.
- The waterfall model is useful in situations where the project requirements are well-defined and the project goals are clear.
- It is often used for Large-scale projects with long timelines, where there is little room of error and the project stakeholders need to have a high level of confidence in the outcome.

Features of waterfall model ⇒

1⇒ Sequential Approach ⇒ The waterfall model involves a sequential Approach to Software development, where each phase of the project is completed before moving on the next one.

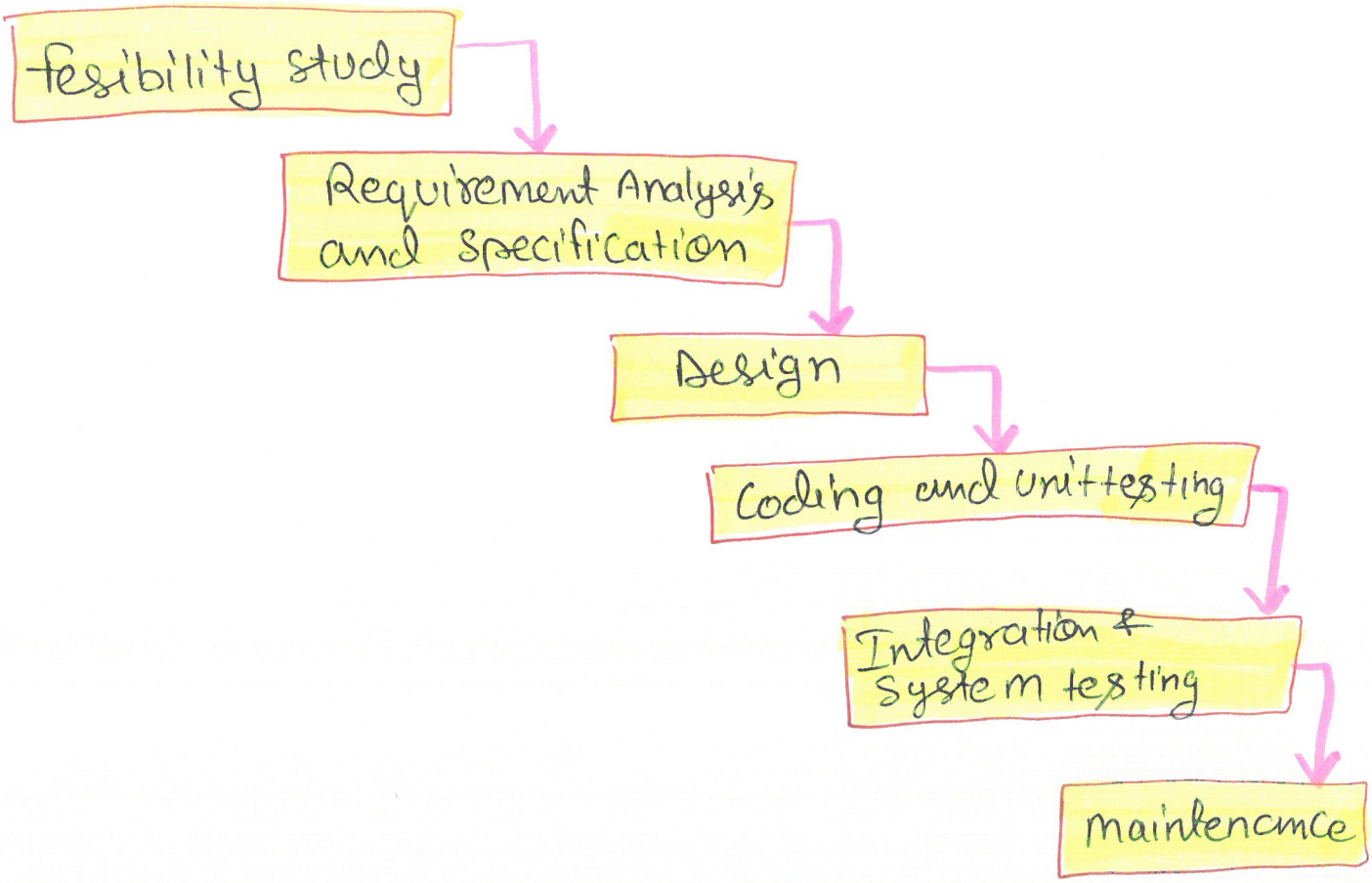
2⇒ Document-Driven ⇒ The waterfall model relies heavily on documentation

To ensure that the project is well-defined and the project team is working towards a clear set of goals.

3 ⇒ Quality Control :⇒ The waterfall model places a high emphasis on quality Control and testing at each phase of the project to ensure that the final product meets the requirements and expectations of the stakeholders

4⇒ Rigorous Planning : ⇒ The waterfall model involves a rigorous planning process, where the project scope, timelines, and deliverables are carefully defined and monitored throughout the project Lifecycle.

Phases of Waterfall Model ⇒



1 ⇒ Feasibility Study ⇒ The main goal of this phase is to determine whether it would be financially and technically feasible to develop the software.

- Feasibility study involves understanding the problem and then determining the various possible strategies to solve the problem.
- Feasibility study is carried out based on many purposes to analyze whether sIw product will be right in terms of development, implantation, contribution of project to the organization etc.

5 types of feasibility study ⇒

⇒ Technical feasibility
⇒ operational feasibility
⇒ Economical feasibility
⇒ Legal feasibility
⇒ Schedule feasibility.

2 ⇒ Requirement Analysis and Specification ⇒

- The aim of the requirement analysis and Specification phase is to understand the Exact requirements of the Customer and document them properly.

This phase consists of two different activities.

⇒ Requirement gathering and analysis :⇒ the goal of the analysis part is to remove incompleteness and inconsistencies.

⇒ Requirement specification :⇒ • These analyzed requirements are documented in a SW requirement specification (SRS) document.

- SRS document serves as a contract between the development team and customers. Any future dispute b/w the customers and the developers can be settled by examining the SRS document.

3⇒ Design ⇒ • The goal of this phase is to covert the requirements acquired in the SRS into a format that can be coded in a programming language.

- It includes high-level and detailed design as well as the overall SW architecture.
- A Software Design Document is used to document all of this effort (SDD).

4⇒ Coding and Unit Testing ⇒ In the coding phase S/w design is translated into Source code using any Suitable programming Language.

- Thus each designed module is coded.
- The aim of the unit testing phase is to check whether each module is working properly or not.

5⇒ Integration and System testing ⇒

- Integration of different modules is undertaken soon after they have been coded and unit tested.
- Integration of various modules is carried out incrementally over a number of steps.
- During each integration step, previously planned modules are added to the partially integrated System and the resultant system is tested.
- finally, after all the modules have been successfully integrated and tested, the full working system is obtained and System testing is carried out on this.

System testing consists of three different kinds of testing activities.

1) Alpha testing ⇒ It is the system testing performed by the development team.

2) Beta testing ⇒ It is the s/m testing performed by a friendly set of customers.

3) Acceptance testing ⇒ After the s/w has been delivered, the Customer performed acceptance testing to determine whether to accept the delivered s/w or reject it.

6 ⇒ Maintenance ⇒ Maintenance is the most important phase of a s/w Life cycle. The effort spent on maintenance is 60% of the total effort spent to develop a full s/w.

There are basically three types of maintenance.

- Corrective maintenance :⇒ This type of maintenance is carried out to correct errors that were not

discovered during the product development phase.

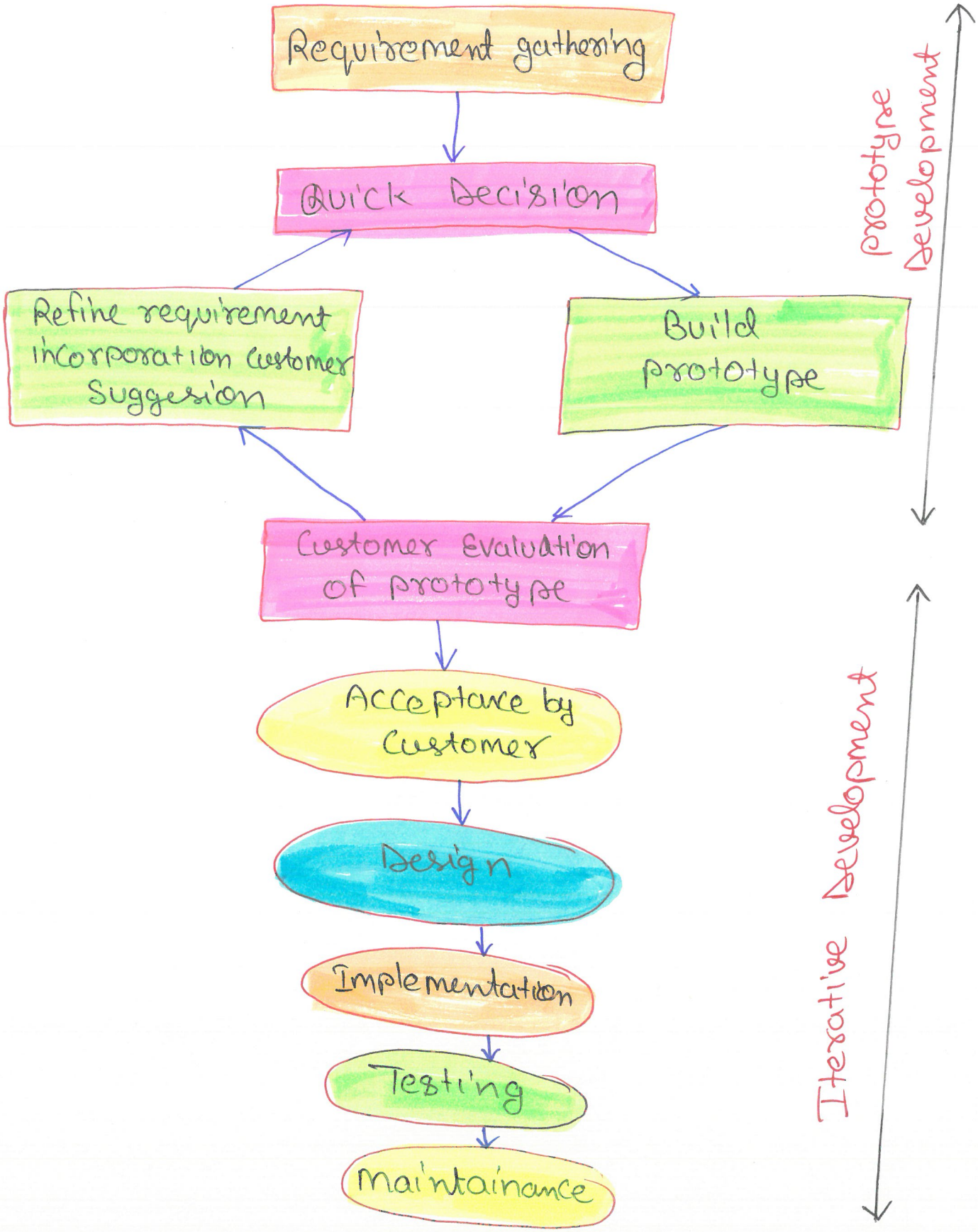
- Perfective Maintenance :⇒ This type of maintenance is carried out to enhance the functionalities of the system based on the customer's request.

- Adaptive Maintenance :⇒ Adaptive maintenance is usually required for porting the S/w to work in a new environment such as working on a new computer platform or with a new operating system.

Prototyping Model

- The prototyping model is SDM development method in which a prototype is built, tested and then reworked as necessary until an acceptable outcome is achieved from which the complete system or product can be developed.
- This model work best in scenarios where not all the project requirements are known in detail ahead of time.
- It also creates base to produce the final system or software.
- The prototype model requires that before carrying out the development of actual software, a working prototype of the system should be built.
- A prototype is a toy implementation of the system.
- This model is used when the customers do not know the exact project requirements beforehand.
- In this model, a prototype of the end product

first developed, tested and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product.



Step 1: Requirements gathering and Analysis ⇒

- A prototyping model starts with requirement analysis.
- In this phase, the requirements of the system are defined in detail.
- During the process, the users of the system are interviewed to know what is their Expectation from the system.

Step 2: Quick Design ⇒

- The second phase is a preliminary design or a quick design.
- In this stage, a simple design of system is Created.
- However, it is not a Complete design. It gives a brief idea of the system to the user.
- The quick design helps in developing the prototype.

Step 3: Build a Prototype ⇒

- In this phase, an actual prototype is designed based on the information gathered from quick design.
- It is a small working model of the required System.

Step 4: Initial User Evaluation :⇒

- In this stage, the proposed system is presen to the client for an initial evaluation.
- It helps to find out the strength and weakness of the working model.
- Comment and suggestion are collected from the customer and provided to the developer.

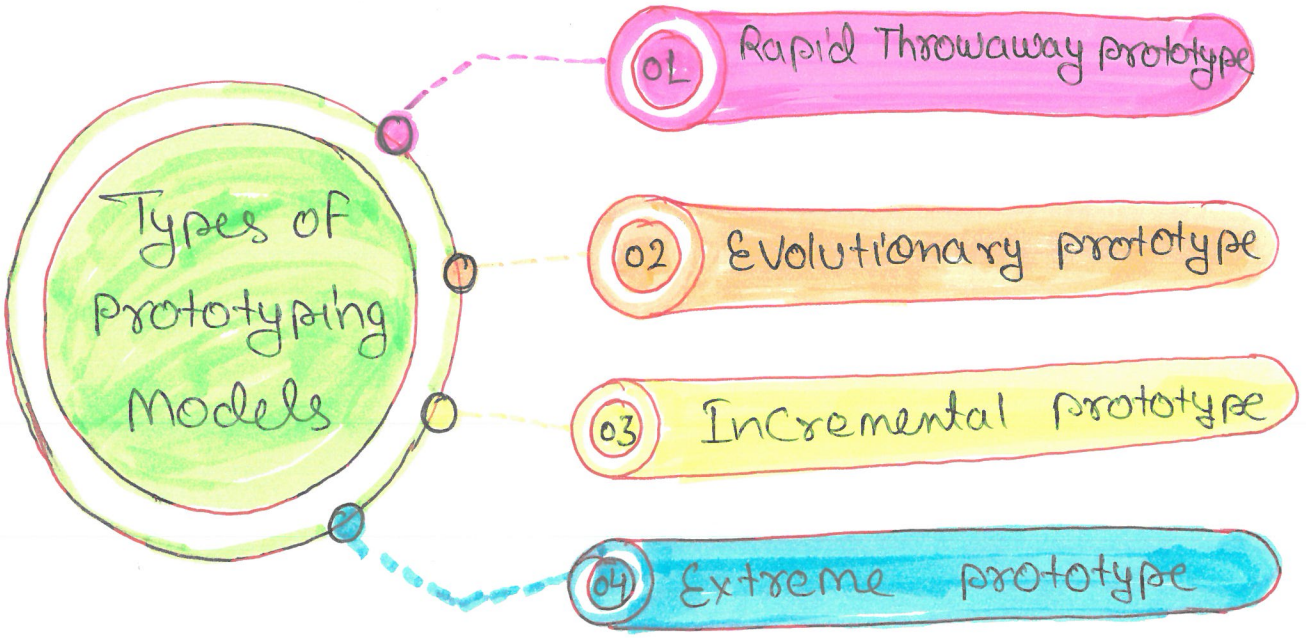
Step 5: Refining prototype :⇒

- If the user is not happy with the current prototype, you need to refine the prototype according to the user's feedback & suggestions.
- This phase will not over until all the requirements specified by the user are met.
- Once the user is satisfied with the developed prototype, a final SIm is developed based on the approved final prototype.

Step 6: Implement Product and maintain :⇒

- Once the final SIm is developed based on the final prototype, it is throughly tested and developed to production.
- The System routine maintenance for minimizing downtime and prevent Large-Scale failures.

Types of Prototyping Models



01 ⇒ Rapid Throwaway Prototype ⇒

- Rapid throwaway is based on the preliminary requirement.
- It is quickly developed to show how the requirement will look visually.
- The customer's feedback helps drives changes to the requirement, and the prototype is again created until the requirement is baselined.
- This technique is useful for exploring ideas and getting instant feedback for customer requirements.

02 ⇒ Evolutionary Prototyping ⇒

- This approach uses a continuous, working prototype that is refined after each iteration of customer feedback.
- This method saves time and effort because each prototype is not started from scratch.

03 ⇒ Incremental prototyping ⇒

- This technique breaks the concept for the final product into smaller pieces and prototypes are created for each one.
- In the end, these prototypes are merged into the final product.

04 ⇒ Extreme prototyping ⇒

- This prototype model is used specifically for web development.
- All web prototypes are built in an HTML format with a services layer and are then integrated into the final product.

Advantages of Prototyping Model ⇒

1 ⇒ Reduce the risk of incorrect user requirement.

2 ⇒ Good where requirement are changing / uncommitted.

3 ⇒ Regular visible process aids management.

4 ⇒ Support early product marketing.

5 ⇒ Reduce maintenance cost.

6 ⇒ Error can be detected much earlier as the system is made side by side.

Disadvantages of Prototyping Model ⇒

1 ⇒ Costly with respect to time as well as money.

2 ⇒ There may be too much variation in requirements each time the prototype is evaluated by the customer.

3 ⇒ Poor Documentation due to continuously changing customer requirements.

4 ⇒ It is very difficult for developers to accommodate all the changes demanded by the customer.

5 ⇒ Difficult to know how long the project will last.

6 ⇒ It is a time-consuming process.

RAD (Rapid Application Development) Model

- Rapid Application Development model is a Software development process based on prototyping without any specific planning.
- The RAD model was first proposed by IBM in the 1980s.
- The RAD model is a type of incremental process model in which there is an Extremely short Development Cycle.
- When the requirements are fully understood and the component-based construction approach is adopted then the RAD model is used.
- The RAD model enables a development team to create a fully functional system with a concise time period.

RAD (Rapid Application Development) is a concept that products can be developed faster and of higher quality through:

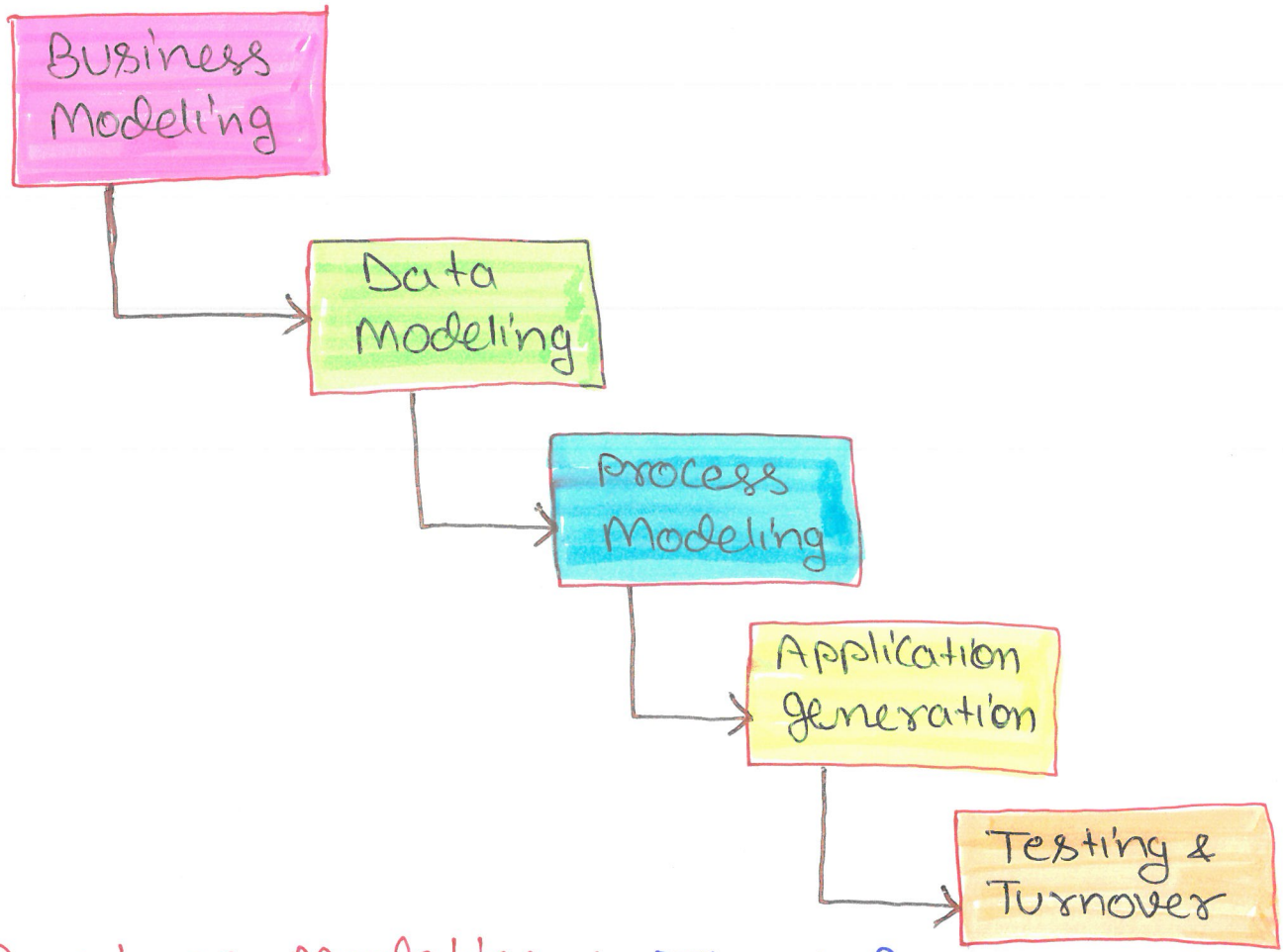
⇒ Gathering requirements using workshops or focus groups.

⇒ prototyping and early, reiterative user testing of design.

⇒ The re-use of software components.

⇒ A rigidly paced schedule that refers design improvements to the next product version.

⇒ Less formality in reviews and other team communication.



1⇒ Business Modeling ⇒ The information flow among business functions is defined by answering questions like what data drives the business process, what data is generated, who generate it, where does the information go, who process it and so on.

2 ⇒ Data Modeling ⇒ • The data collected from business modeling is refined into a set of data objects (entities) that are needed to support the business.

- The attributes (character of each entity) are identified, and the relation blw these data objects (entities) is defined.

3 ⇒ Process Modeling ⇒ • The information object defined in the data modeling phase are transformed to achieve the data flow necessary to implement a business function.

- processing descriptions are created for adding, modifying, deleting, or retrieving a data object.

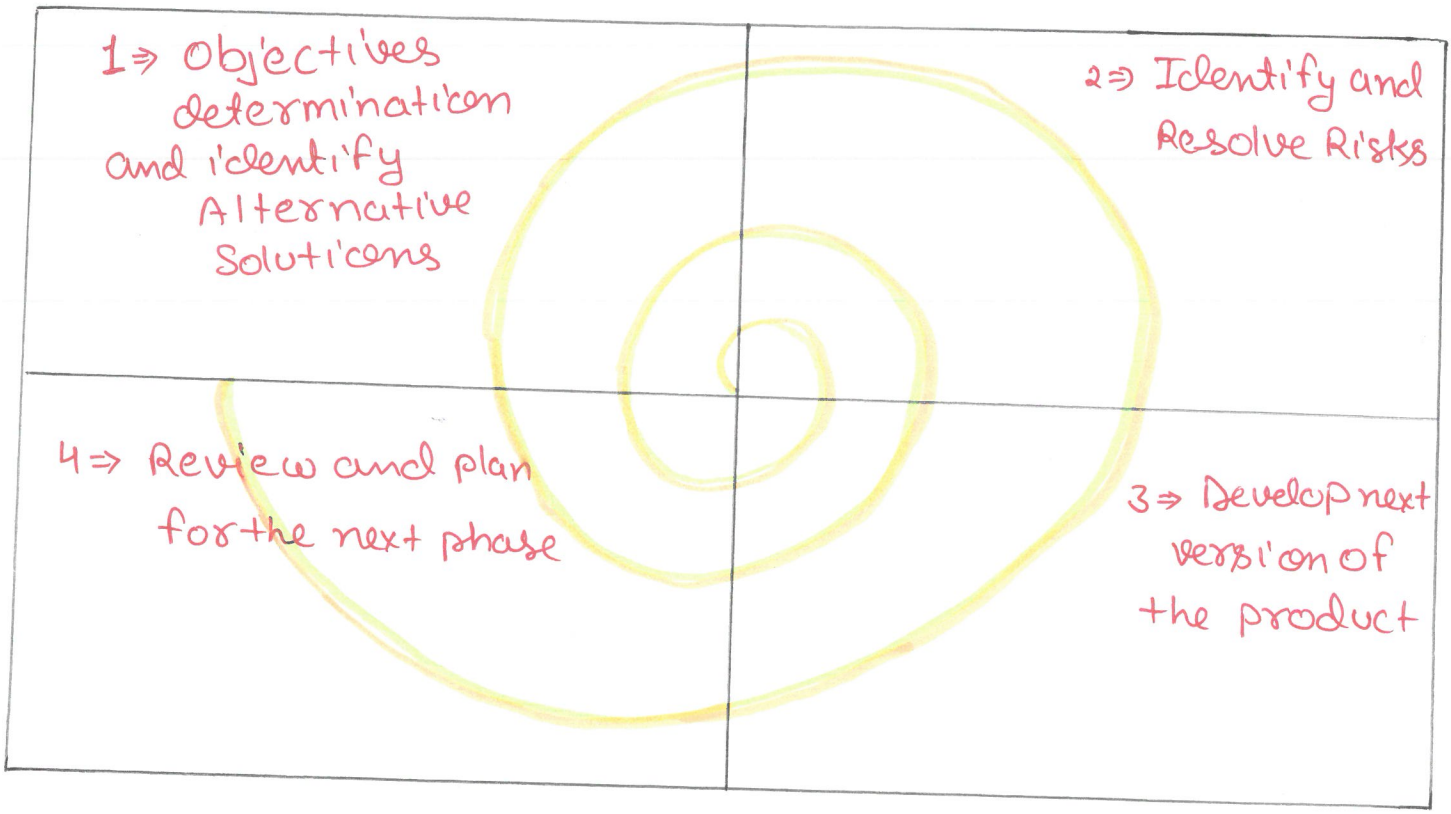
4 ⇒ Application Generation ⇒ Automated tools are used to facilitate Construction of the software, Even they use the 4th GL Techniques.

5 ⇒ Testing & Turnover ⇒ • many of the programming components have already been tested since RAD emphasis reuse.

- This reduces the overall testing time. But the new part must be tested, and all interface must be fully exercised.

Spiral Model

- The Spiral model is a Systems development Lifecycle (SDLC) method used for risk management that combines the iterative development process model with elements of the Waterfall model.
- The Spiral model is used by S/w Engineers and is favored for Large, Expensive and Complicated projects.



- In its diagrammatic representation, Looks like a Spiral with many loops.
- The Exact no. of loops of the Spiral is unknown and can vary from project to project.
- Each loop of the Spiral is called a phase of the Software development process.

- As the project manager dynamically determines the no. of phases, the project manager has an important role in developing a product using the Spiral model.
- Spiral model is based on the idea of a Spiral, with each iteration of the spiral representing a complete SLW development cycle, from requirement gathering and analysis to design, implementation, testing, and maintenance.

Phases of Spiral Model

⇒ The Spiral model is a risk-driven model, means that the focus is on managing risk through multiple iterations of the SLW development process. It consist of phases:

1 ⇒ planning ⇒ The first phase of the Spiral model is the planning phase, where The scope of the project is determined and a plan is created for the next iteration of the Spiral.

2 ⇒ Risk Analysis ⇒ In the risk analysis phase, the risks associated with the project are identified and Evaluated.

3 ⇒ Engineering ⇒ In this phase, the Sw is developed based on the requirements gathered in the previous iteration.

4 ⇒ Evaluation ⇒ In this phase, the Sw is Evaluated to determine if it meets the Customer's requirements and if it is of high quality.

5 ⇒ Planning ⇒ The next iteration of the Spiral begins with a new planning phase, based on the results of the Evaluation.

Each phase of the Spiral Model is divided into four quadrants as :-

1 ⇒ Objectives determination and identify alternative Solutions ⇒

- Requirements are gathered from the Customers and the objectives are identified, elaborated, and Analyzed at the start of every phase.
- Then alternative solutions possible for the phase are proposed in this quadrant.

2 ⇒ Identify and resolve Risks : ⇒

- During the second quadrant, all the possible Solutions are Evaluated to Select the best possible solution.
- Then the risks associated with that Solution are identified and the risks are resolved using the best possible strategy.
- At the end of this quadrant, the prototype is built for the best possible solution.

3 ⇒ Develop the next version of the Product ⇒

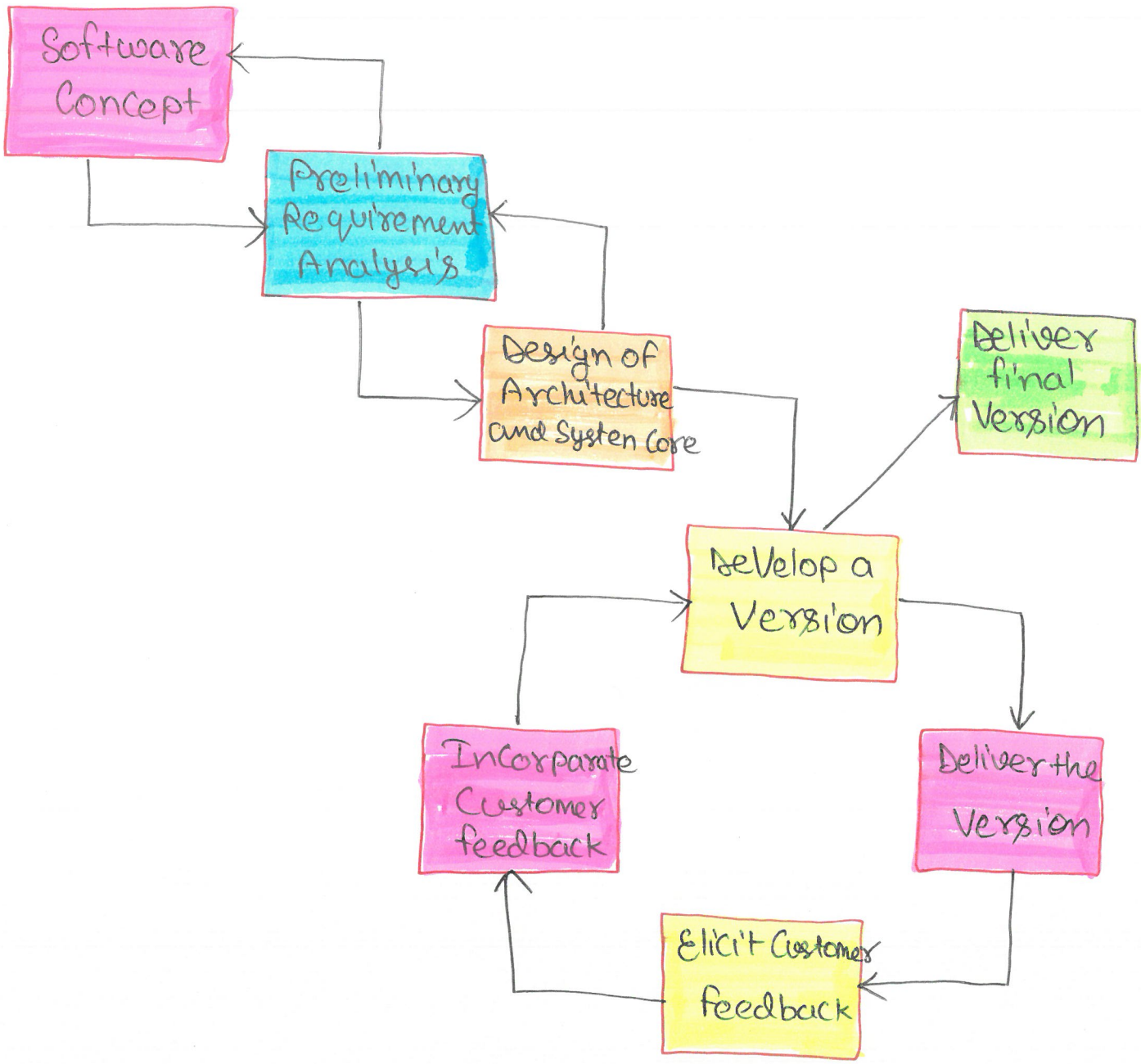
- During the third quadrant, the identified features are developed and verified through testing.
- At the End of the third quadrant, the next version of the Slw is available.

4 ⇒ Review and Plan for the next Phase ⇒

- In the fourth quadrant, the Customers Evaluate the So-far developed version of the sofware.
- In the end, planning for the next phase is started.

Evolutionary Development Models

- The evolutionary Development models is a Combination of the Iterative and Incremental models of the software development life cycle.
- The Evolutionary Development model divides the development cycle into smaller, incremental waterfall models in which users can get access to the product at the end of each cycle.



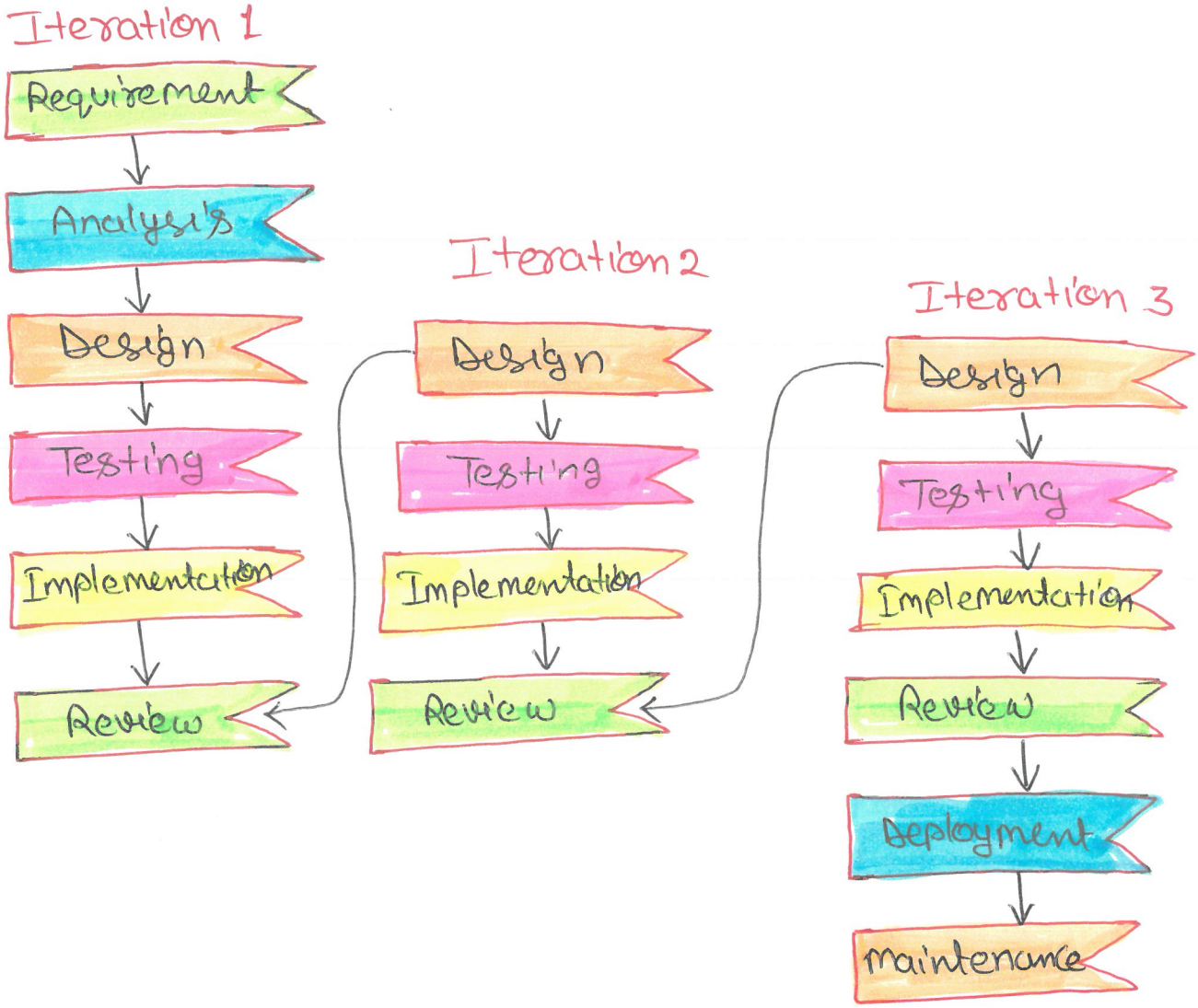
* feedback is provided by the users on the product for the planning stage of the next cycle and the development team responds, often by changing the product, plan or process.
* Therfore, the software product evolves with time.
* All the models have the disadvantage that the duration of time from the start of the project to the delivery time of a solution is very high.
* The Evolutionary model solves this problem with a Different approach.
* The Evolutionary model Suggests breaking down work into Smaller chunks, prioritizing them, and then delivering those chunks to the Customer one by one.
* The number of chunks is huge and is the number of deliveries made to the Customer.
* The main advantage is that the Customer's Confidence increases as he Constantly get quantifiable goods or services from the beginning of the project to verify and Validate his requirements.
* The model allows for changing requirements as well as all work is broken down into maintainable work chunks.

Iterative Enhancement Model

- Software Development uses a dynamic and adaptive method called the Iterative enhancement model.
- The iterative enhancement model encourages a S/w products ongoing evolution and improvement.
- This methodology is noticeable due to its Concentration on adaptability, flexibility and change responsiveness.
- It makes it easier for a product to evolve because it gives developers the freedom to progressively enhance the S/w, making sure that it complies with evolving specifications, user demands, and market demands.
- This helps products evolve more easily.
- In this model, you can start with some of the Software specifications and develop the first version of the software.
- After the first version if there is a need to change the software, then a new version of the S/w is created with a new iteration.
- Every release of the Iterative model finishes in an exact and fixed period that is called iteration.

- The Iterative model allows the accessing earlier phases, in which the variations made respectively.

The final output of the project renewed at the end of the Software Development Life Cycle (SDLC).



The various phases of Iterative model are as follows: ⇒

1 ⇒ Requirement gathering & analysis ⇒ In this phase, requirements are gathered from customers and check by an analyst whether requirements will fulfil or not. Analyst checks that need will achieve within buget or not. After all of this, the Slw team skips to the

next phase.

2 ⇒ Design ⇒ In the Design phase, team design the Slw by the different diagrams like Data flow diagram, activity diagram, class diagram, state transition diagram, etc.

3 ⇒ Implementation ⇒ In the implementation, requirements are written in the Coding Language and transformed into Computer programmes which are called Slw.

4 ⇒ Testing ⇒ After Completing the Coding phase, Slw testing Starts using different test methods.
There are many test method, but the most common are white box, black box, and grey box test methods.

5 ⇒ Deployment ⇒ After Completing all the phases, software is deployed to its work Environment.

6 ⇒ Review ⇒ In this phase, after the product deployed, review phase is performed to check the behaviour

and validity of the developed product. And if there are any Error found then the process starts again from the requirement gathering.

7 ⇒ Maintenance ⇒ In the maintenance phase, after deployment of the Slw in the working Environment there may be some bugs, some Errors or new Updates are required.
Maintenance involves bedugging and new addition options.

Unit = 02

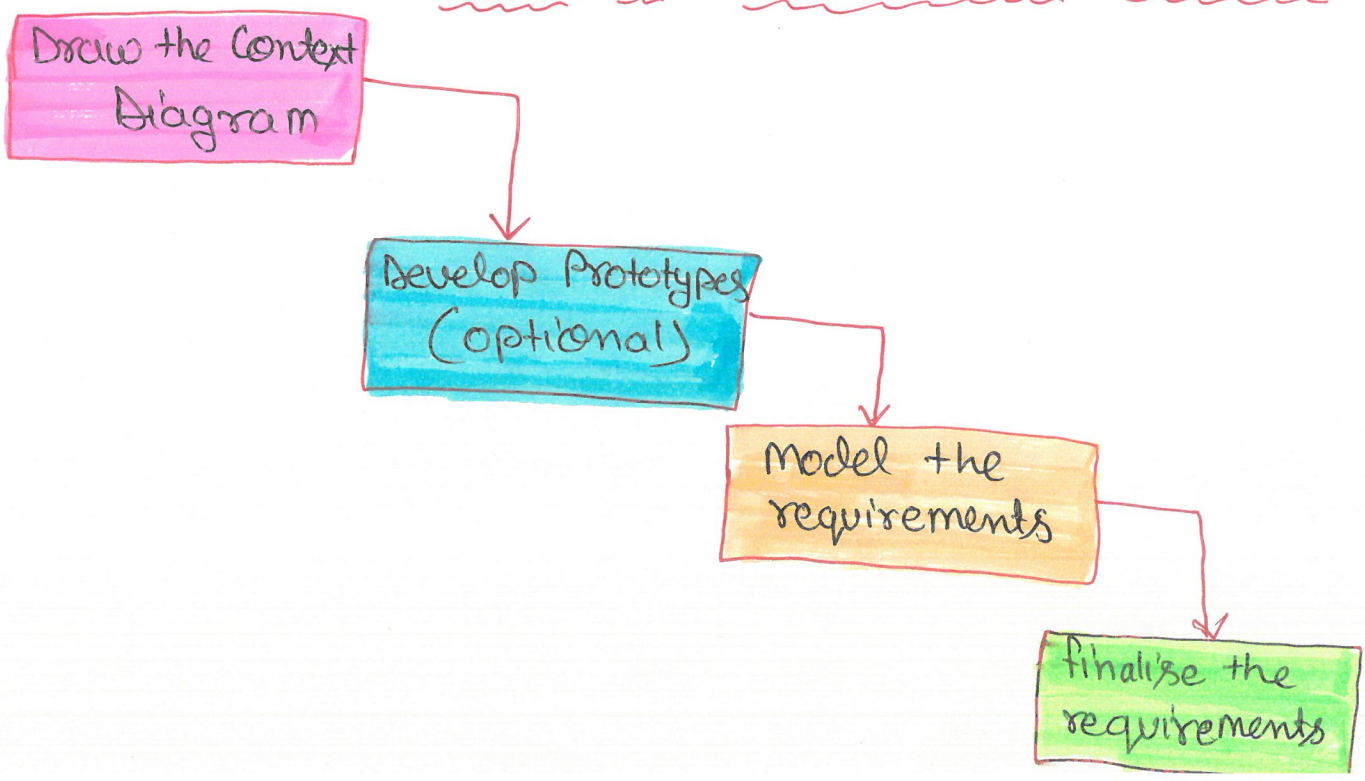
Software Requirement Analysis

* Requirements Analysis ⇒

Requirement Analysis is significant and essential activity after elicitation we Analyze, refine, and scrutinize the gathered requirements to make consistent and unambiguous requirements.

This activity review all requirements and may provide a graphical view of the entire system. After the completion of the analysis, it is expected that the understandability of the project may improve significantly.

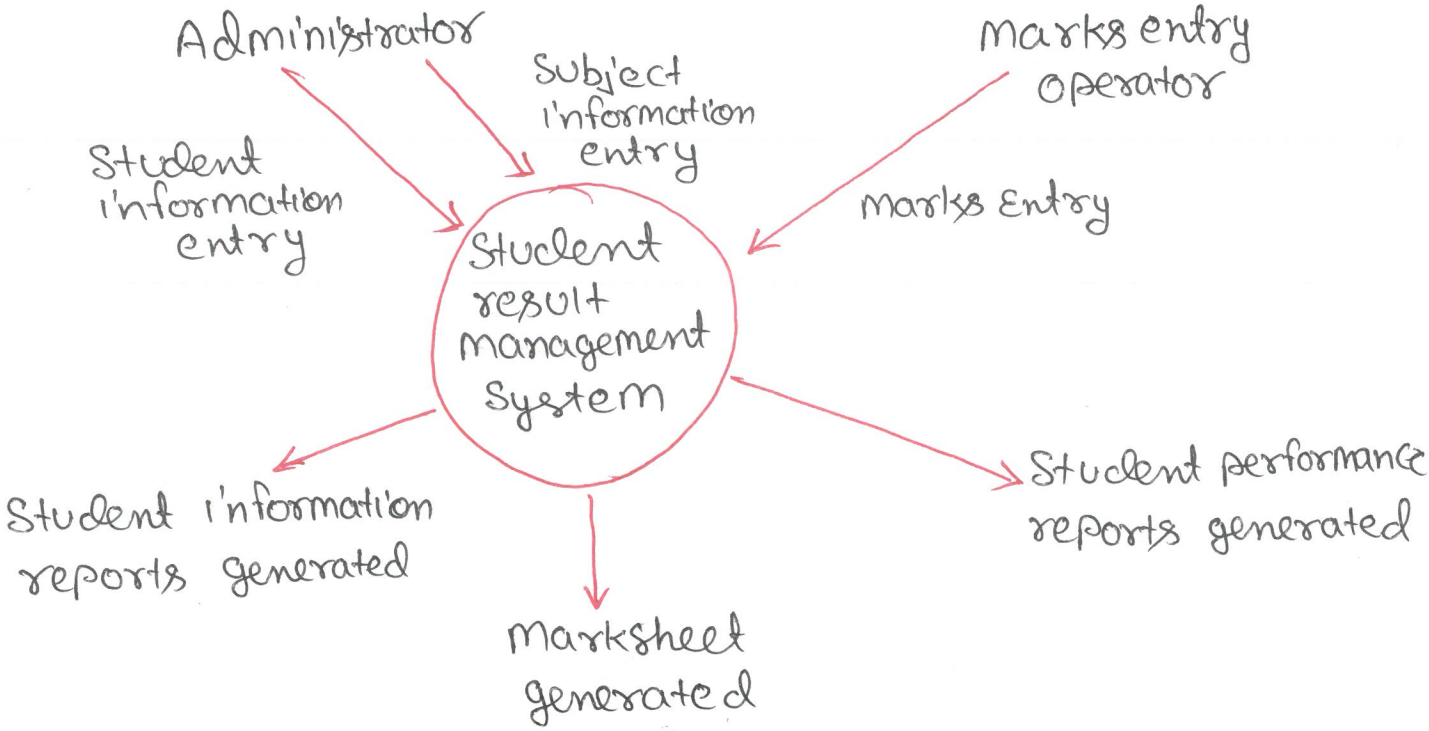
Steps of Requirement Analysis



1) Draw the Context Diagram ⇒

- The Context Diagram is a simple model that defines the boundaries and interfaces of the proposed systems with the external world.
- It identifies the entities outside the proposed system that interact with the system.

The Context Diagram of student result management system is as:-



2) Development of a Prototype (optional) ⇒

- One effective way to find out what the customer wants is to construct a prototype, something that looks and preferably acts as part of the system
- they say they want.

We can use their feedback to modify the prototype until the customer is satisfied continously.

- The prototype should be built quickly and at a relatively low cost. Hence it will always have limitations and would not be acceptable in the final system. This is an optional activity.

3) Model the requirements :⇒

- This process usually consists of various graphical representations of the functions, data entities, external entities, and the relationships b/w them.
- The graphical view may help to find incorrect, inconsistent, missing, and superfluous requirements. Such models include the Data flow Diagram, Entity-Relationship Diagram, Data Dictionaries, State-transition diagram etc.

4) finalise the requirements! ⇒

- After modeling the requirements, we will have a better understanding of the system behavior.
- The inconsistencies and ambiguities have been identified and corrected.
- The flow of data amongst various modules has been analyzed.

End

Structured Analysis

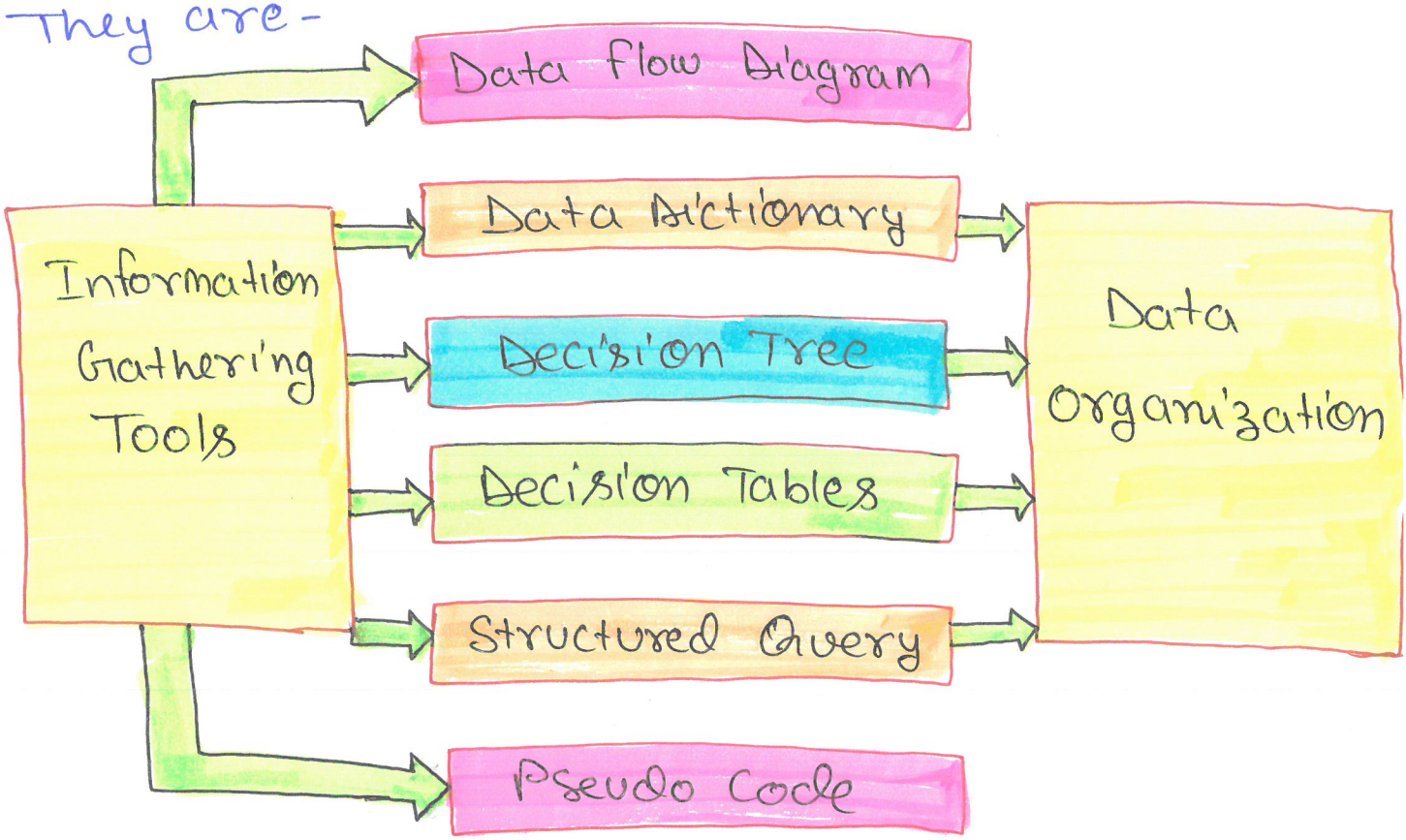
- Structured Analysis is a development method that allows the analyst to understand the system and its activities in a Logical way.
- It is a systematic approach, which uses graphical tools that analyze and refine the objective of an existing system and develop a new system specification which can be easily understandable by user.

It has following attributes –

- It is graphic which specifies the presentation of application.
- It divides the processes so that it gives a clear picture of system flow.
- It is Logical rather than physical i.e, the elements of system do not depend on vendor or hardware.
- It is an approach that works from high-Level overviews to Low-Level details.

Structured Analysis Tools ⇒

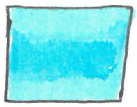
During Structured Analysis, various tools and techniques are used for system development They are-



Data Flow Diagrams (DFD) or Bubble Chart ⇒

- A data flow diagram (DFD) is a graphical or visual representation using a standardized set of symbols and notations to describe a business's operations through data movement.
- DFD is easy to understand and quite effective when the required design is not clear and the user wants a notation Language for Communication

Basic Elements of DFD

Symbol Name	Symbol	Meaning
Square		Source or Destination of Data
Arrow		Data Flow
Circle		Process Transforming data flow
Open Rectangle		Data Store

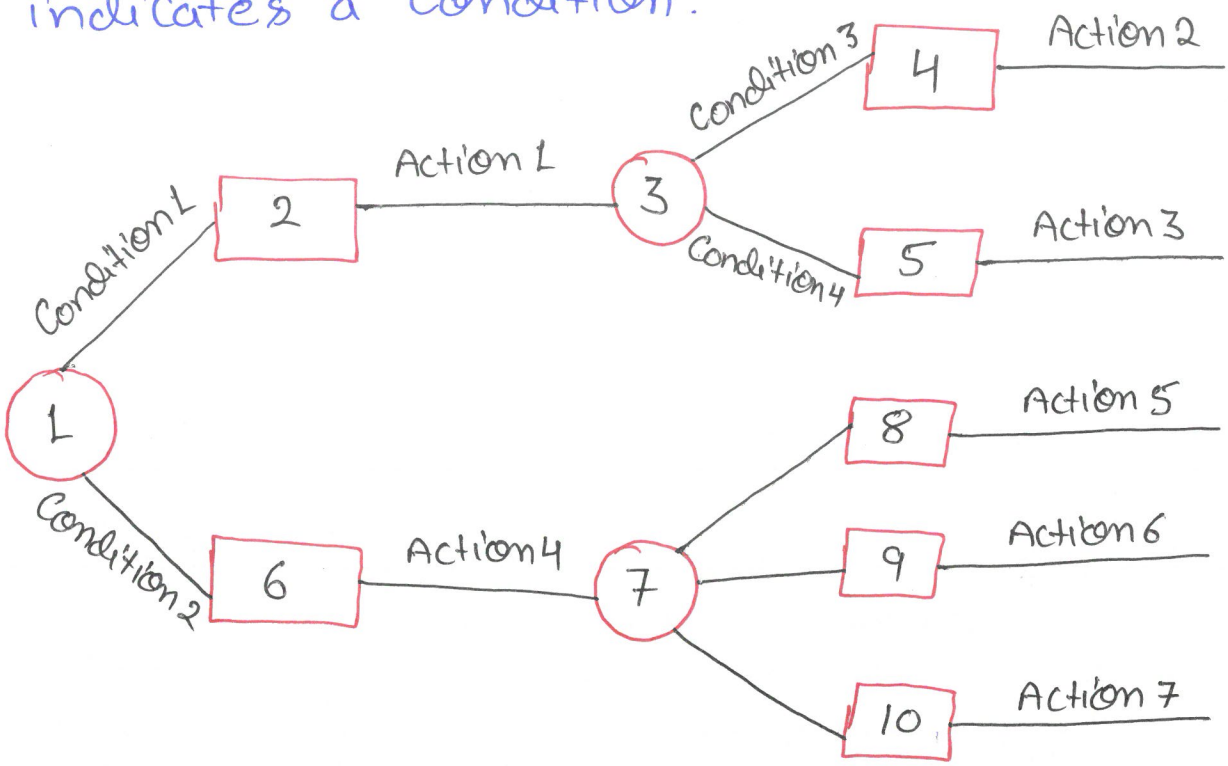
Data Dictionary ⇒

- A data Dictionary is a structured repository of data elements in the system.
- It stores the descriptions of all DFD data elements that is, details and definitions of data flows, data stores, data stored in data stores, and the processes.
- A data Dictionary improves the communication b/w the analyst and the user. It plays an important role in building a database.

Sr. No.	Data Name	Description	No. of Characters
1	ISBN	ISBN Number	10
2	TITLE	title	60
3	SUB	Book Subjects	80
4	ANAME	Author Name	15

Decision Trees ⇒

- Decision trees are a method for defining complex relationships by describing decisions and avoiding the problems in communication.
- A decision tree is a diagram that shows alternative actions and conditions within horizontal tree framework. Thus, it depicts which conditions to consider first, second, and so on.
- Decision trees depict the relationship of each condition and their permissible actions.
- A square node indicates an action and a circle indicates a condition.



Decision Tables ⇒

- Decision tables are a method of describing the Complex Logical relationship in a precise manner which is easily understandable.
- It is useful in situations where the resulting actions depend on the occurrence of one or several combinations of independent conditions.
- It is a matrix containing row or columns for defining a problem and the actions.

Components of a Decision Table ⇒

- **Condition Stub** ⇒ It is the upper left quadrant which lists all the Condition to be checked.
- **Action Stub** ⇒ It is the Lower Left quadrant which outlines all the action to be carried out to meet such condition.
- **Condition Entry** ⇒ It is in upper right quadrant which provides answers to questions asked in condition stub quadrant.
- **Action Entry** ⇒ It is in Lower right quadrant which indicates the appropriate action resulting from the answers to be conditions in the condition entry quadrant.

The entries in decision table are given by Decision Rules which define the relationships b/w combination of conditions and courses of action. (82)

In rules section,

- Y shows the existence of a condition.
- N represents the condition, which is not satisfied.
- A blank - against action states it is to be ignored.
- X (or a check mark will do) against action states it is to be carried out.

For example,

Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Advance payment made	Y	N	N	N
Purchase amount = Rs 10,000/-	—	Y	Y	N
Regular Customer	—	Y	N	—

Structured Query / Structure English ⇒

- Structure English is derived from structured programming Language which gives more understandable and precise description of process.
- It is based on procedural Logic that uses Construction and imperative sentences designed

to perform operation for action.

- It is best used when sequences and Loops in a program must be considered and the problem needs sequences of actions with decisions.
- It does not have strict Syntax rule. It Expresses all Logic in terms of sequential decision structures and iterations.

for Example ⇒

```
if Customer pays advance
    then
        Give 5% Discount
    else
        if purchase amount >= 10,000
          Then
          if the Custome is a regular Customer
              then Give 5% Discount
          Else No Discount
        end if
    Else No Discount
  End if
End if
```

Pseudo Code ⇒

- Psedo code is written more close to programming Language. It may be considered as augmented programming Language, full of comments and descriptions.
- Psedo Code avoids variable declaration but they are written using some actual programming Language's Constructs, like C, Fortran, Pascal etc.
- Psedo Code contains more programming details than structured English. It provides a method to perform the task, as if a computer is Executing the code.

Ex ⇒ WAP to print the given no. Even or odd in Python.

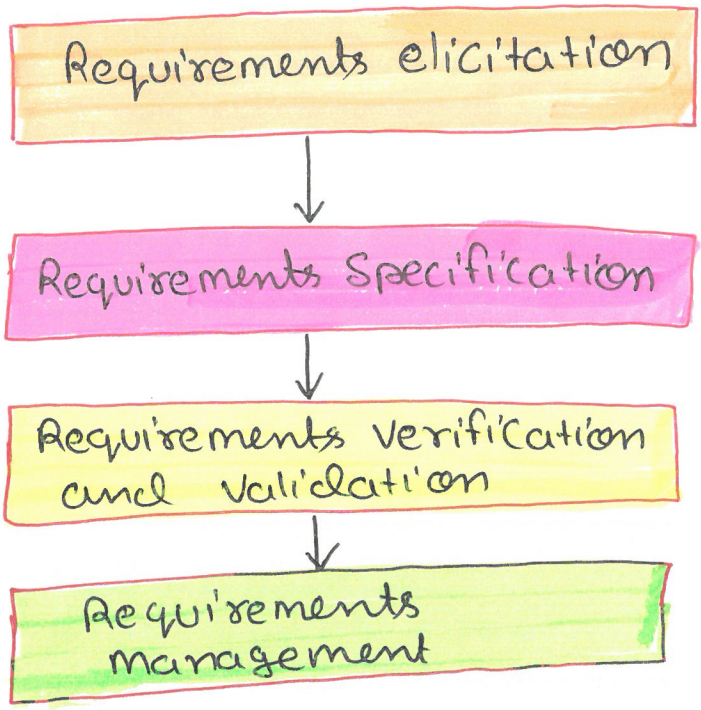
```
num = int(input("Enter any number"))
if num%2==0:
        print("Even no.")
Else:
        print("odd no.")
```

— * —

Requirements Engineering Process in Software Engineering

- Requirements engineering is the process of identifying, eliciting, analyzing, specifying, validating, and managing the needs and expectation of stakeholders for a software system.
- A systematic and strict approach to the definition, creation and varification of requirements for a s/w system is known as requirements engineering.

Steps in Requirements Engineering Process ⇒



Step 01 ⇒ Requirements elicitation is the process of gathering information about the needs and Expectations of stakeholders for a s/w system. This is the first step in the requirements engineering

- process and it is critical to the success of the Software Development project.
- The goal of this step is to understand the problem that the Slw System is intended to solve, and the needs and Expectations of the Stakeholders who will use the System.

There are several techniques that can be used to elicit requirements, including:

- Interviews ⇒ These are one-on-one Conversations with Stakeholders to gather information about their needs and Expactations.
- Surveys ⇒ These are questionnaires that are distributed to Stakeholders to gather information about their needs and Expectations.
- Focus Groups ⇒ These are small groups of Stakeholders who are brought together to discuss their needs and expectations for the Slw Sm.
- Observation ⇒ This technique involves observing the Stakeholders in their work environment to gether information about their needs & Expectations.
- Prototyping ⇒ This technique involves Creating a working model of the Slw Sm, which can be used to gather feedback from Stakeholders and to Validate requirements.

Step 2 ⇒ Requirement Specification ⇒

- Requirements specification is the process of documenting the requirements identified in the analysis step in a clear, consistent, and unambiguous manner.
- This step also involves prioritizing and grouping the requirements into manageable chunks.
- The goal of this step is to create a clear and comprehensive document that describes the requirements for the S/w S/m.
- This document should be understandable by both the development team and the stakeholders.

There are several types of requirements:-

1⇒ Functional Requirements ⇒ These describe what the System should do.

They specify the functionality that the system must provide, such as input validation, data storage, and user interface.

2⇒ Non-Functional Requirements ⇒ These describe how well the S/w System should do it.

They Specify the quality attributes of the System such as performance, reliability, usability and security

3⇒ Constraints ⇒ These describe any limitations or restrictions that must be considered when developing the software system.

4⇒ Acceptance Criteria ⇒ These describe the conditions that must be met for the S/w S/m to be considered complete and ready for release.

Step-03: Requirements Verification and validation ⇒

Verification ⇒ • It refers to the set of tasks that ensures that the S/w correctly implements a specific function.

- Verification is the process of checking that the requirements are complete, consistent, and accurate.
- It involves reviewing the requirements to ensure that they are clear, testable, and free of errors and inconsistencies.
- This can include reviewing the requirements document, models and diagrams, and holding meeting and walkthroughs with stakeholders.

Validation ⇒ • Validation is the process of checking that the requirements meet the needs and Expection of the Stakeholders.

- Validation refers to a different set of tasks that ensures that the S/w that has been built is traceable to Customer requirements.
- It involves testing the requirements to ensure that they are valid and that the S/w S/m being developed will meet the needs of the stakeholders.
- This can include testing the S/w S/m through simulation, testing with prototypes, and testing with the final version of the S/w.

Verification & Validation (V&V) ⇒

- V&V is an iterative process that occurs throughout the S/w development Life cycle.
- It is important to involve stakeholders and the development team in the V&V process to ensure that the requirements are throughly reviewed and tested.

Step 04 :- Requirements Management ⇒

- Requirements management is the process of managing the requirements throughout the S/w development Life cycle, including tracking and Controlling changes, and ensuring that the requirements are still valid and relevant.
- The goal of requirements management is to ensure

that the S/w S/m being developed meets the needs and expectations of the stakeholders and that it is developed on time, within budget and to the required quality.

90

There are several key activities that are involved in requirements management :⇒

1⇒ Tracking and Controlling Changes: ⇒
This involves monitoring and controlling changes to the requirements throughout the development process, including identifying the source of the change, assessing the impact of the change, and approving or rejecting the change.

2⇒ Version Control ⇒ This involves keeping track of different versions of the requirements document and other related artifacts.

3⇒ Traceability ⇒ This involves linking the requirements to other elements of the development process, such as design, testing and validation.

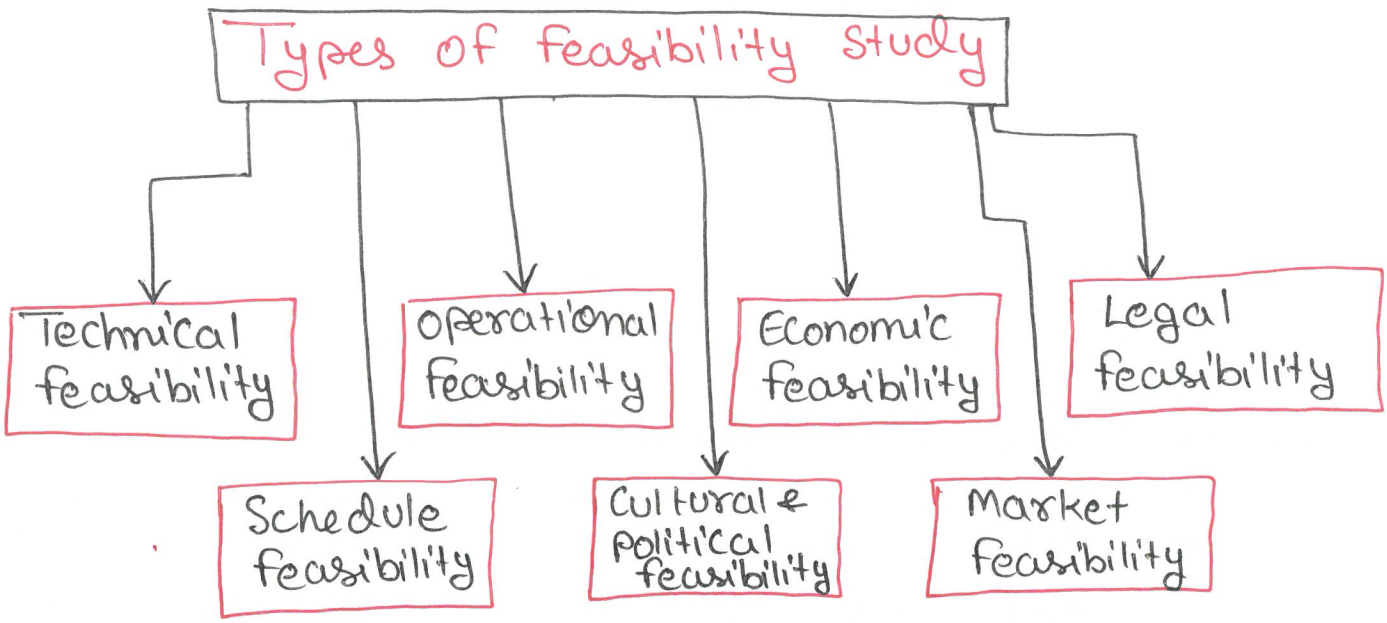
4⇒ Communication ⇒ This involves ensuring that the requirements are communicated effectively to all stakeholders and that any changes or issues are addressed in a timely manner.

5⇒ Monitoring and reporting ⇒ This involves monitoring the progress of the development process and reporting on the status of the requirements.

—— * ——

Feasibility Study ⇒

- A feasibility study is part of the initial design stage of any proposed project/plan.
- It is carried out to evaluate the feasibility of proposed project or an existing s/w used by the business.
- It can assist in identifying and assessing the opportunities and threats present in the natural environment, the resources needed for the project, and the chances of success.
- In Software engineering, the feasibility study is a project done to determine s/w technical and commercial viability before its development.



1) Technical feasibility ⇒ In Technical feasibility current resources both hardware s/w along with required technology are analyzed/accessed to develop project.

- This technical feasibility study gives report whether there exists correct required resources and technologies which will be used for project development.
- Feasibility study also analyzes technical skills and capabilities of technical team, existing technology can be used or not, maintenance and up-gradation is easy or not for chosen technology etc.

2⇒ Operational feasibility ⇒ In operational feasibility degree of providing service to requirements is analyzed along with how much easy product

- will be to operate and maintenance after development
- Along with this other operational scopes are determining usability of product, Determining Suggested solution by Slw development team is acceptable or not etc.

3⇒ Economic feasibility ⇒ In Economic feasibility study Cost and benefit of the project is analyzed.

- In this feasibility study a detail analysis is carried out what will be Cost of the project for development which includes all required cost for final development like hardware and Slw resoure required.

4 ⇒ Schedule feasibility ⇒ • A schedule feasibility study mainly analyzes the proposed project deadlines/deadlines, including the time it will take the team to complete the final project.

- This has a significant impact on the organization as the project's purpose may fail if it is not completed on time.

5 ⇒ Legal feasibility ⇒ • In Legal feasibility study project is analyzed in Legality point of view.

- This includes analyzing barriers of Legal implementation of project, data protection acts or social media Laws, project certificate, License, Copyright etc.
- Overall it can be said that Legal feasibility study is study to know if proposed project conform Legal and ethical requirements.

6 ⇒ Cultural and Political feasibility ⇒ • This section assesses how the slw project will affect the political environment and organizational Culture.

- This analysis takes into account the organization's Culture and how the suggested changes could be received there, as well as any potential political

obstacles or internal opposition to the project.

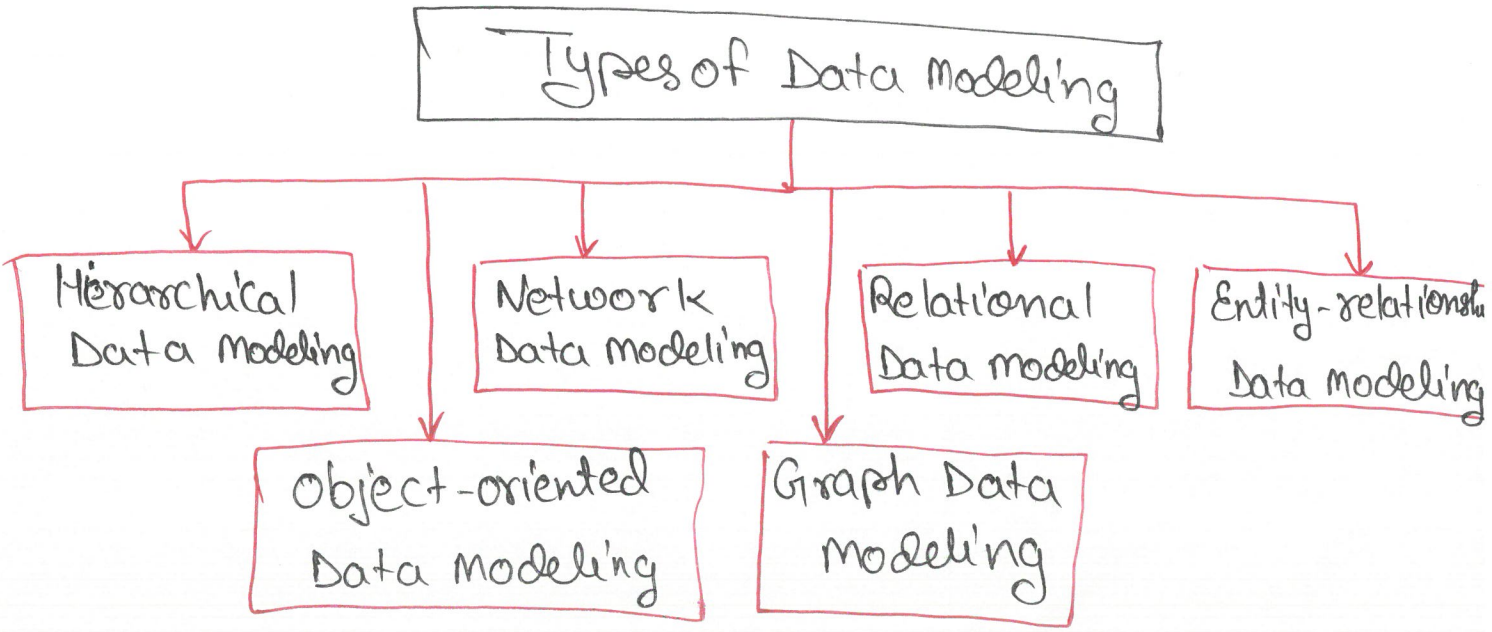
- It is essential that cultural and political factors be taken into account in order to execute project successfully.

7 ⇒ Market Feasibility ⇒ • This refers to evaluating the market's willingness and ability to accept the suggested s/w s/m.

- Analyzing the target market, understanding consumer wants and assessing possible rivals are all part of this study.
- It assists in identifying whether the project is in line with market expactations and whether there is a feasible market for the good or service being offered.

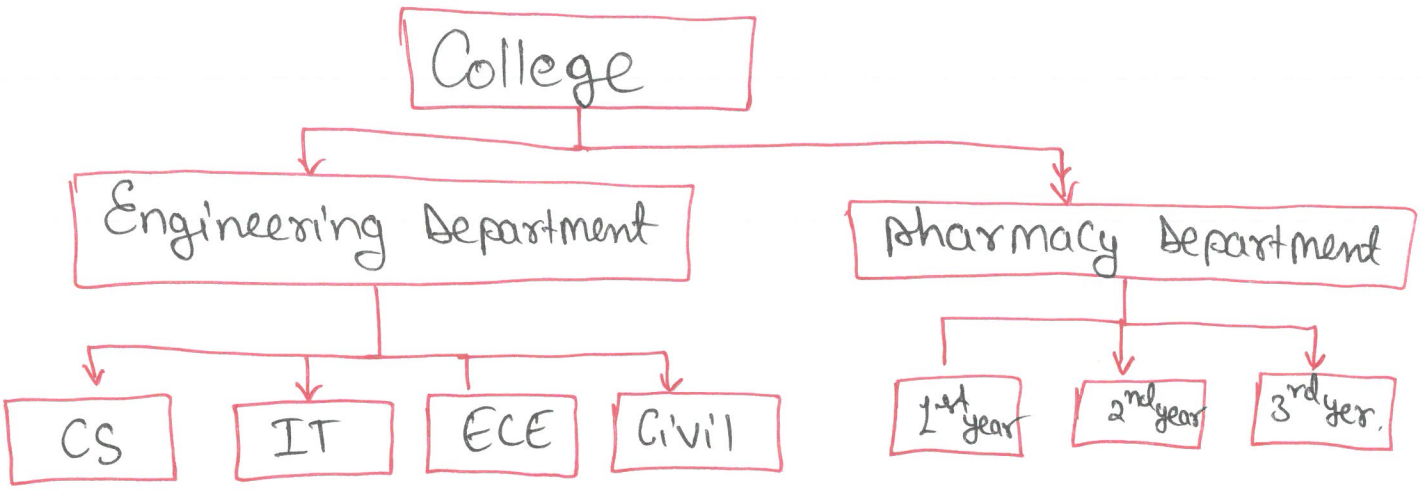
Information Modeling / Data Modeling

- An information modeling is a Conceptual framework that defines how data is Structured, managed, and processed within a System.
- It serves as a blueprint for organizing information ensuring Consistency and Clarity across platforms.
- An information model is used by software engineers and website designers to build an effective platform that is easy to use and navigate.
- Data modeling is the process of Creating a Simplified diagram of a S/w Sm and the data elements it Contains, using text and symbols to represent the data and how it flows.
- Data modeling helps an organization use its data effectively to meet business needs for information.



1⇒ Hierarchical Data modeling ⇒

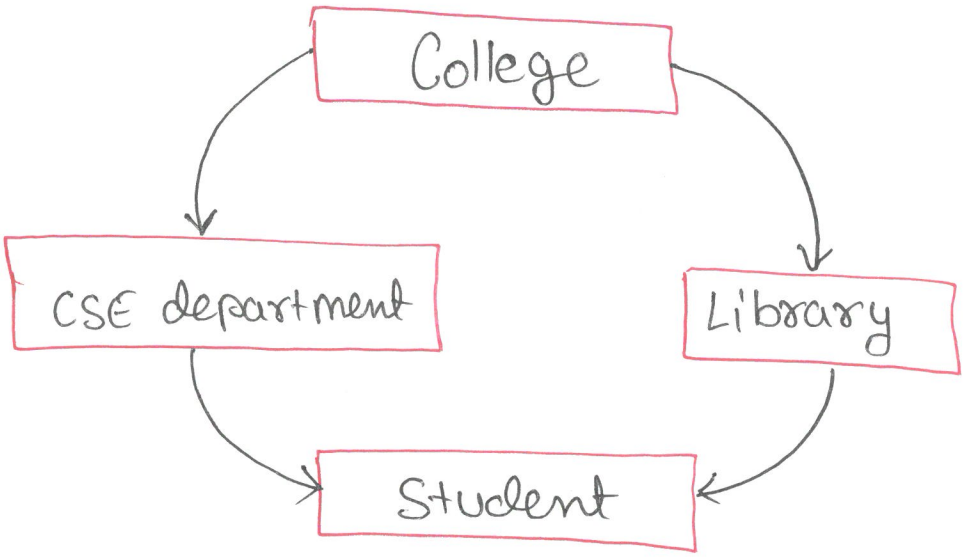
- Hierarchical Data models organize data in a tree like arrangement of parent and child records.
- A child record can have only one parent, making this a one-to-many modeling method.
- It is one of the oldest modeling developed by IBM, in the 1950s.
- A Similar hierarchical method is also used tody in XML, formally known as Extensible markup Language.



2⇒ Network Data modeling ⇒

- Network Data model expanded on hierarchical ones by allowing child records to be connected to multiple parent records.
- The Network Data model enables may-to-many relationship among the connected nodes.
- The data is arranged in a graph-like structure, and here 'child' nodes can have multiple 'parent' nodes.

- The parent nodes are known as owners, and the child nodes are called members.



Network model

3 ⇒ Relational Data Modeling ⇒

- This is a popular Data model example arranges the data into tables.
- The tables have columns and rows, each cataloging an attribute present in the entity.
- It makes relationships between data points easy to identify.

Attributes / field

Name	Rollno.	Branch	Address
Aman	01	IT	Haridwar
Ankit	02	CSE	Dehradun
Kamal	03	ECE	Delhi
Kailash	04	IT	Rishikesh

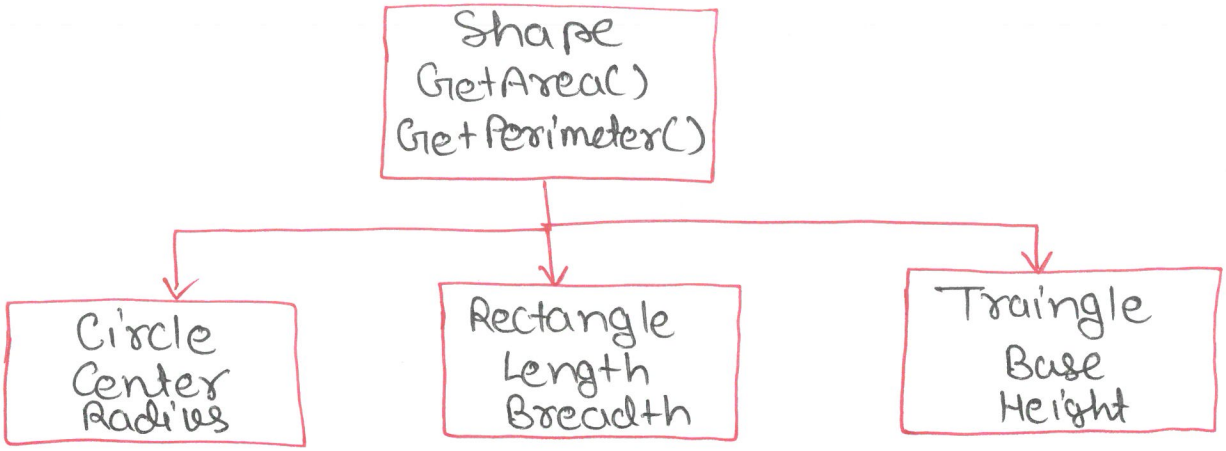
tuple

Name Domain — Rollno Domain — Branch Domain — Address Domain

4 ⇒ Object Oriented Data Modeling ⇒

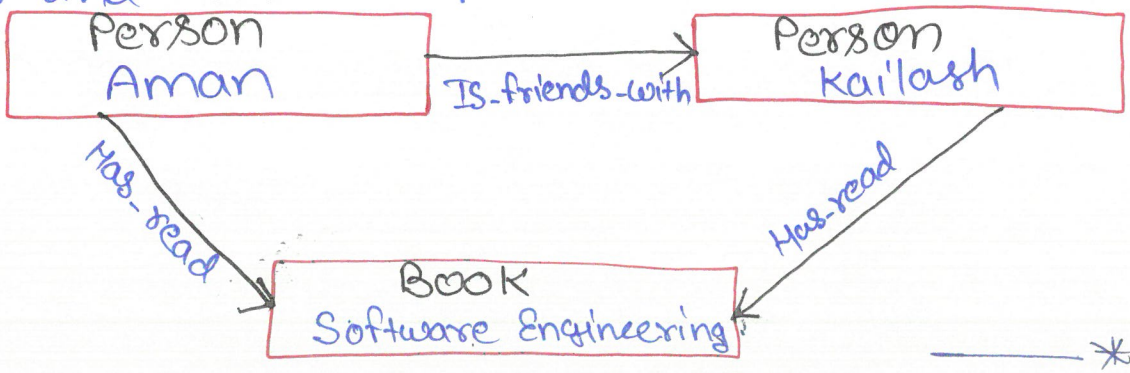
- This data model defines a database as an object collection, or recyclable software components, with related methods and features.
- In the object-oriented Data model, Data and their relationships are contained in a single structure which is referred to as an object in this data model.
- In this, real-world problems are represented as objects with different attributes. All the objects have multiple relationships between them.

5 ⇒



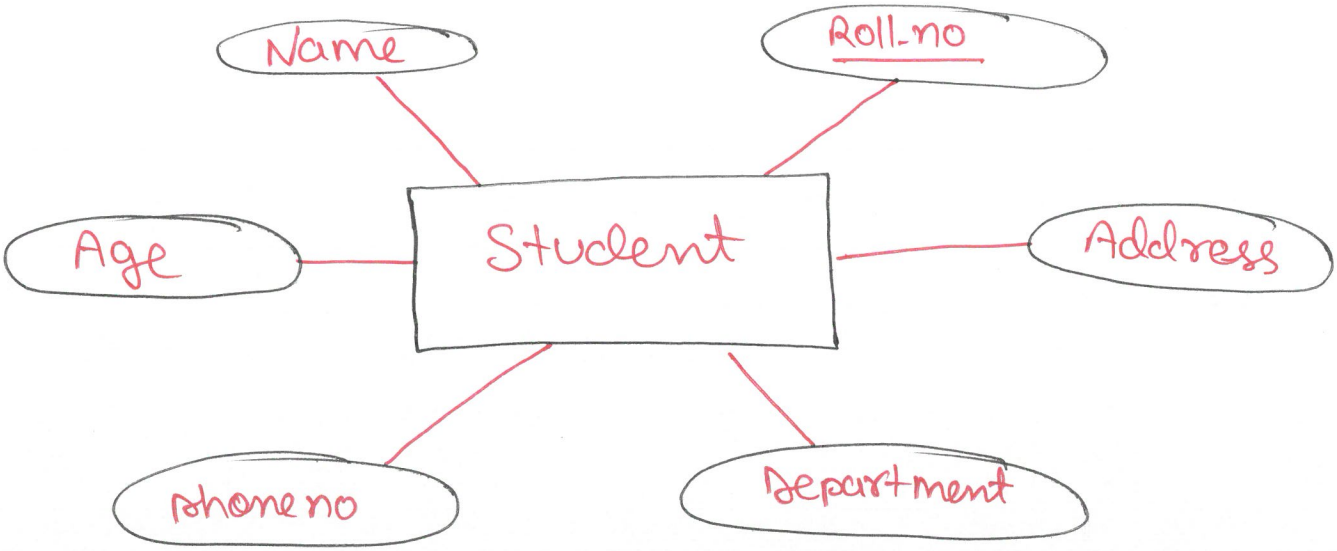
5 ⇒ Graph Data modeling ⇒

- Graph Data modeling is the process in which a user describes an arbitrary domain as a connected graph of nodes and relationships with properties and labels.



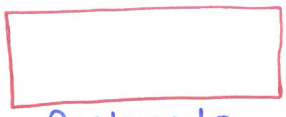
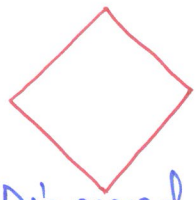
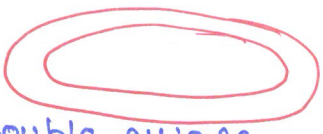
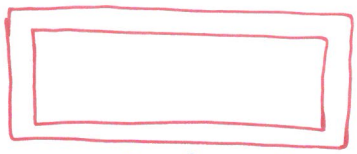
Entity Relationship Diagrams

- The Entity Relationship Diagram is identifying entities to be represented in the database and ~~relation~~ representation of how those entities are related.
- ER Diagram are used to represent the E-R model in a database, which makes them easy to be Converted into relations.
- ER Diagram provide the purpose of real-world modeling of objects which makes them intently useful
- ER Diagram require no technical knowledge and no hardware support.
- It gives a Standard Solution for visualizing the data Logically.

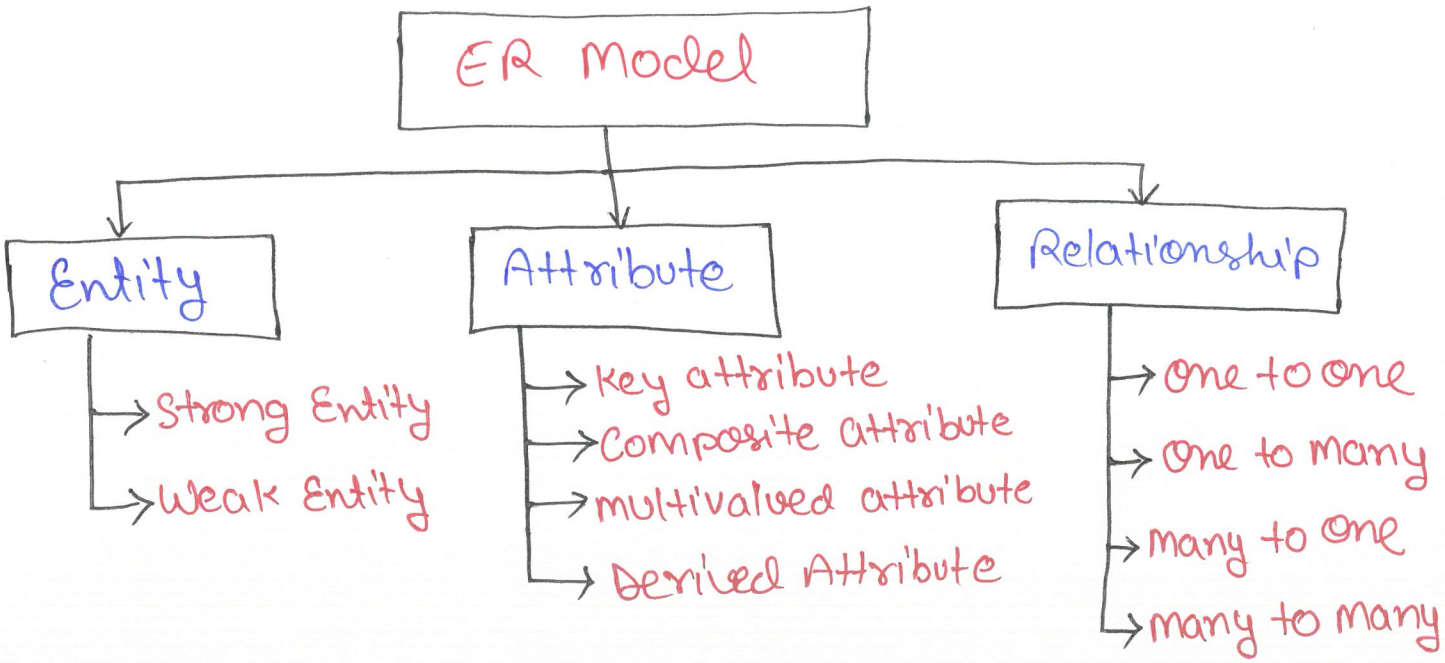


Symbols Used in ER Diagram ⇒

100

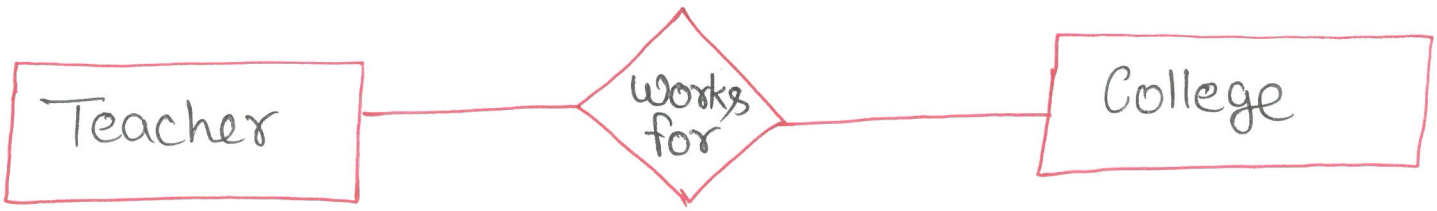
Symbols	Represents
 Rectangle	Entities in ER model
 Ellipse	Attributes in ER model
 Diamond	Relationships among Entities
Line	Attributes to Entities and Entity sets with other Relationship types
 Double Ellipse	multi-valued attributes
 Double Rectangle	Weak Entity

Components of ER Diagram



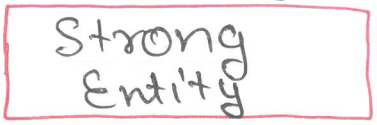
1 ⇒ Entity ⇒ An entity may be any object, class, person or place. (101)

In the ER Diagram, an entity can be represented as rectangles.



a) Strong Entity ⇒ A strong Entity is a type of Entity that has a key attribute.

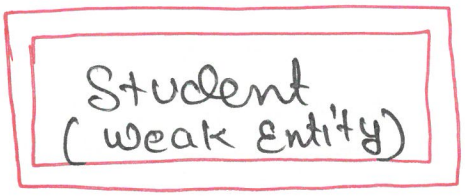
Strong Entity does not depend on other Entity in the Schema. It is denoted by rectangle.



b) Weak Entity ⇒ An Entity that depends on another entity called a weak entity.

The weak entity doesn't contain any key attribute of its own.

The weak entity is represented by a double rectangle

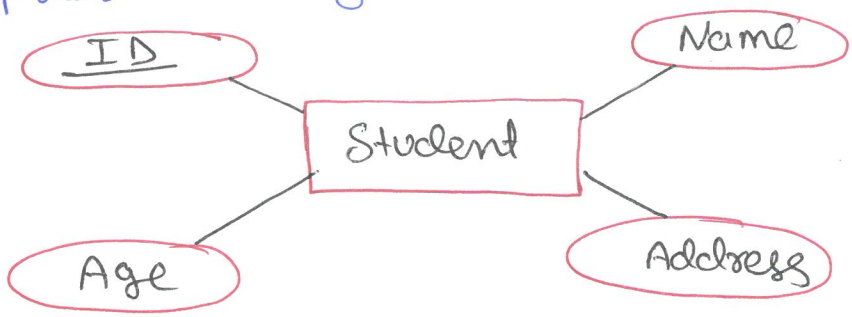


2 ⇒ Attribute ⇒ The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute

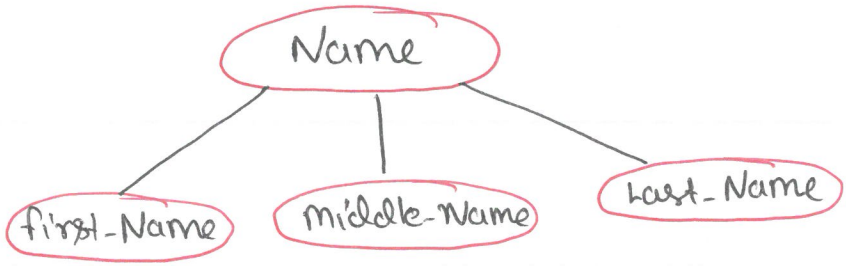
a) key attribute ⇒ The key attribute is used to represent the main characteristics of an entity.

It represent a primary key. The key attribute

is represented by an ellipse with the text underlined.



b) Composite Attribute ⇒ An Attribute that Composed of many other attributes is known as Composite attribute. The composite attribute is represented by an ellipse, and those ellipses are Comected with an ellipse.



c) multivalued Attribute ⇒ An attribute can have more than one value. These attributes are known as a multivalued attribute. The double ellipse is used to represent multivalued attribute.

d) Derived Attribute ⇒ An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by dashed ellipse.

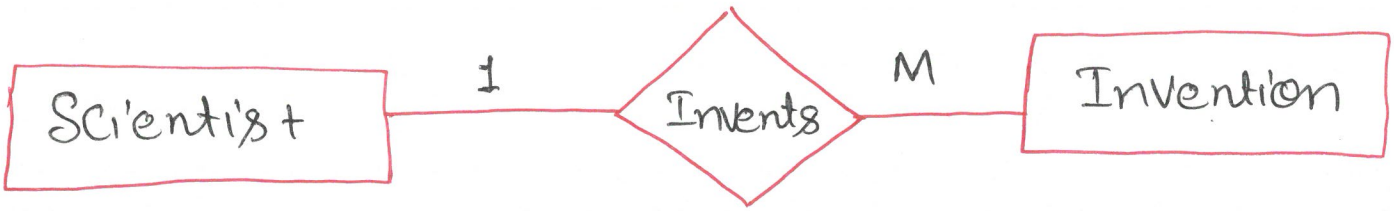
3) Relationship ⇒ A relationship is used to describe the relation between entities.

Diamond or rhombus is used to represent the relationship

⇒ Types of Relationship

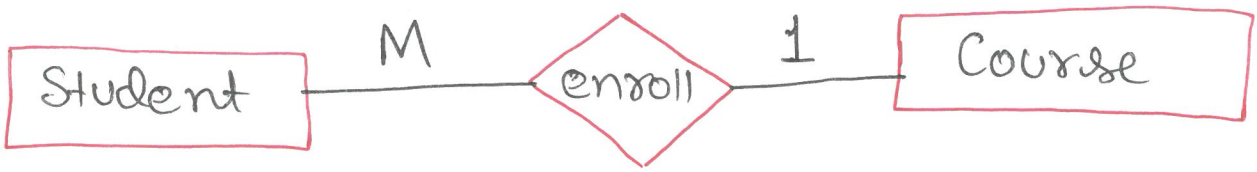
a) One-to-one Relationship ⇒ when only one instance of an Entity is associated with the relationship, then it is known as one to one relationship.

b) One-to-many Relationship ⇒ when only one instance of the entity on the Left, and more then one instance of an entity on the right associated with the relationship then this is known as a one-to-many relationship.

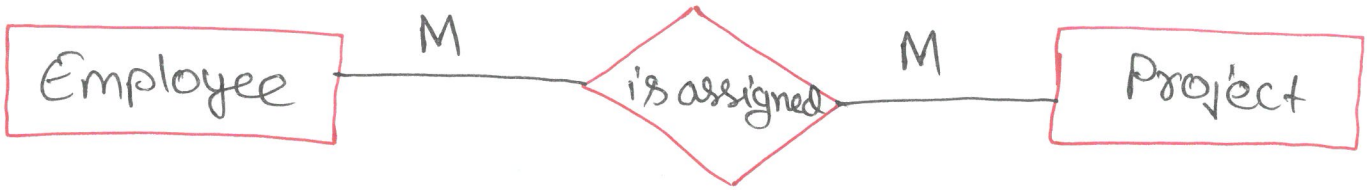


c) Many-to-one Relationship ⇒ When more than one instance of the entity on the Left, and only one instance of an

Entity on the right associated with the relationship then it is known as many-to-one relationship. (104)



d) Many-to-Many relationship ⇒ When more than one instance of an entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

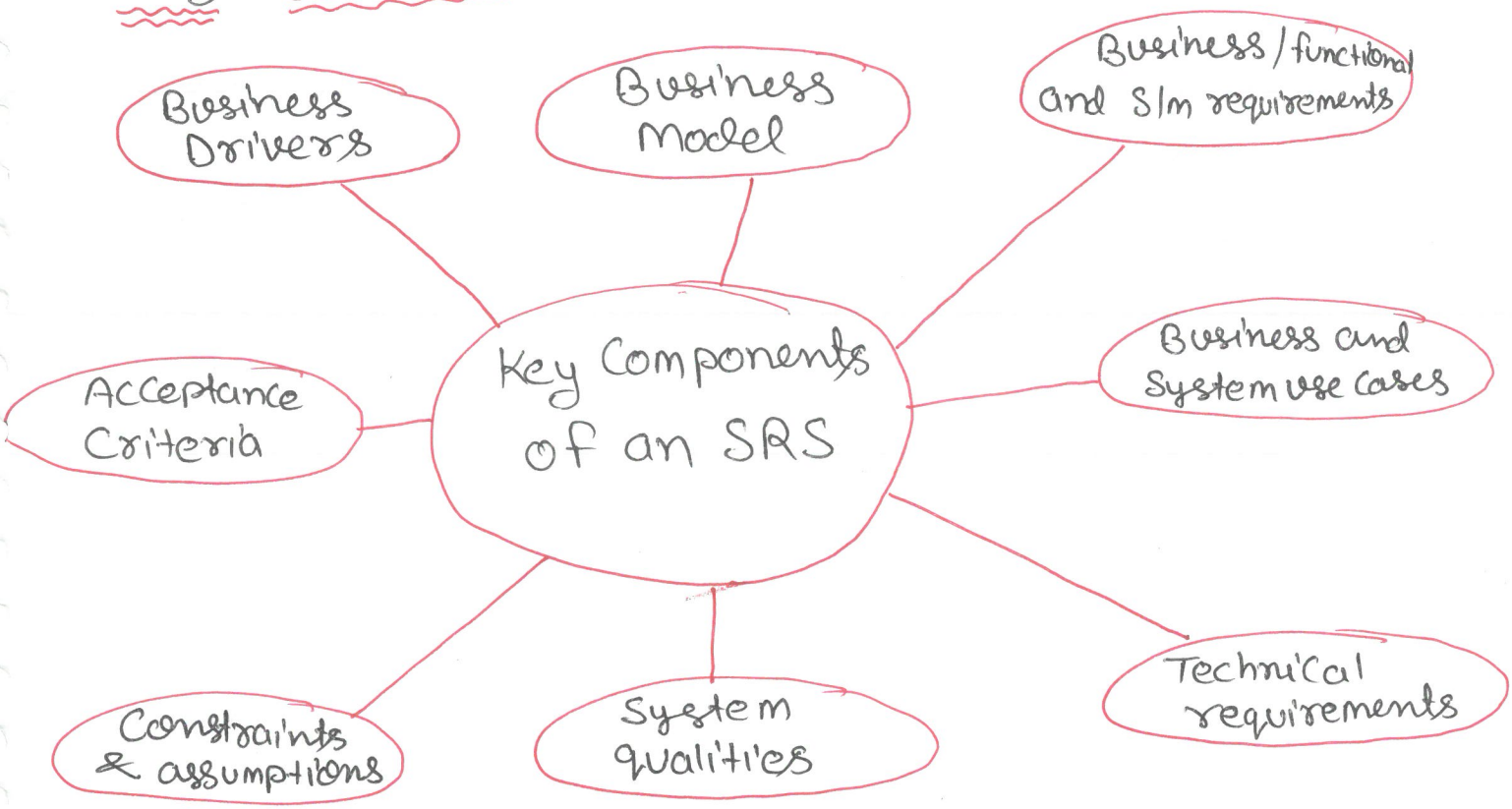


SRS Document

- A Software requirements Specification (SRS) is a document that describes what the S/w will do and how it will be expected to perform.
- It also describes the functionality the product needs to fulfill the needs of all Stakeholders.
- A SRS is a document that describes the nature of a project, software or application.
- SRS document is a manual of a project provided it is prepared before you kick-start a project/application.
- SRS document is also known by the names SRS report, or software document.
- A Software document is primarily prepared for a project, Software, or any kind of application.
- An SRS minimizes the time and effort required by developers to achieve desired goals and also minimizes the development Cost.

- A good SRS defines how an application will interact with system hardware, other programs and human users in a wide variety of real-world situations.

Key Components of an SRS ⇒



- **Business Drivers** ⇒ This section describes the reasons the customer is looking to build the system, including problems with the currently system and opportunities the new S/m will provide.
- **Business model** ⇒ This section describe the business model of the customer that the S/m has to support, including

organizational, business Context, main business functions and process flow diagrams.

- **Business/ functional and system requirements** ⇒ This section typically consists of requirements that are organized in a hierarchical structure.
- **Business and System use cases** ⇒ this section consists of a unified Modeling Language (UML) use case diagram depicting the key External Entities that will be interacting with the SIm and the different use cases that they will have to perform.
- **Technical requirements** ⇒ This section Lists the non-functional requirements that make up the technical environment where SIw needs to operate and the technical restrictions under which it needs to operate.
- **System qualities** ⇒ This section is used to describe the non-functional requirements that define the quality attributes of the System, such as reliability, serviceability, security, Scalability, availability and maintainability

- Constraints and assumptions ⇒ this section includes any constraints that the customer has imposed on the system design. It also includes the requirements engineering team's assumptions about what is expected to happen during the project.

- Acceptance Criteria ⇒ this section details the conditions that must be met for the customer to accept the final system.

features of an SRS ⇒ Characteristics of SRS

- Correctness ⇒ User review is used to provide the accuracy of requirements stated in the SRS SRS is said to be perfect if it covers all the needs that are truly expected from the system.

- **Completeness** ⇒ should contain all the features requested by a client.
- **Consistency** ⇒ Some abbreviation and conventions must be followed throughout the document
- **Unambiguous** ⇒ should not be any confusion regarding interpretation of the requirements
- **Ranked for importance and/or stability** ⇒ Every requirement is important. But Some are urgent and must be fulfilled before other requirements and some could be delayed. It's better to classify each requirement according to its importance and stability.
- **Verifiability** ⇒ an SRS is verifiable only if every stated requirement can be verified.
- **Modifiability** ⇒ an SRS must clearly identify each and every requirement in a systematic manner. If there are any changes, the specific requirements and the dependent ones can be modified accordingly without impact the others.
- **Traceability** ⇒ an SRS is traceable if the origin of each of its requirements is clear and if it makes it easy to reference each requirement in future development.

—— * ——

IEEE Standard for SRS

- IEEE defines S/w requirements specification as, a document that clearly and precisely describes each of the essential requirements (functions, performance, design constraints and quality attributes of the S/w and the external interfaces.
- Each requirement is defined in such a way that its achievement can be objectively verified by a prescribed method for example, inspection, demonstration, analysis or test.
- IEEE standards documents are developed within the IEEE societies and the standards coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board.
- This is a recommended practice for writing S/w requirements specifications.
- It describes the content and qualities of a good S/w requirements specification (SRS) and presents several sample SRS outlines.
- This recommended practice is aimed at specifying requirements of S/w to be develop but also can be applied to assist in the selection of in-house and Commercial S/w product.

The key terms as they are used in this recommended practice are :-

(111)

1 ⇒ Contract ⇒ • A Legally binding document agreed upon by the customer and supplier.

- This includes the technical and organizational requirements, cost and schedule for a product.
- A Contract may also contain informal but useful information such as the commitments or expectations of the parties involved.

2 ⇒ Customer ⇒ • The person, or persons, who pay for the product and usually decide the requirement.

- In the context of this recommended practice the customer and the supplier may be members of the same organization.

3 ⇒ Supplier ⇒ The person, or persons, who produce a product for a customer. In the context of this recommended practice, the customer and the supplier may be members of the same organization.

4 ⇒ User ⇒ The person, or persons, who operate or interact directly with the product. the user and the customer are often not the same person.

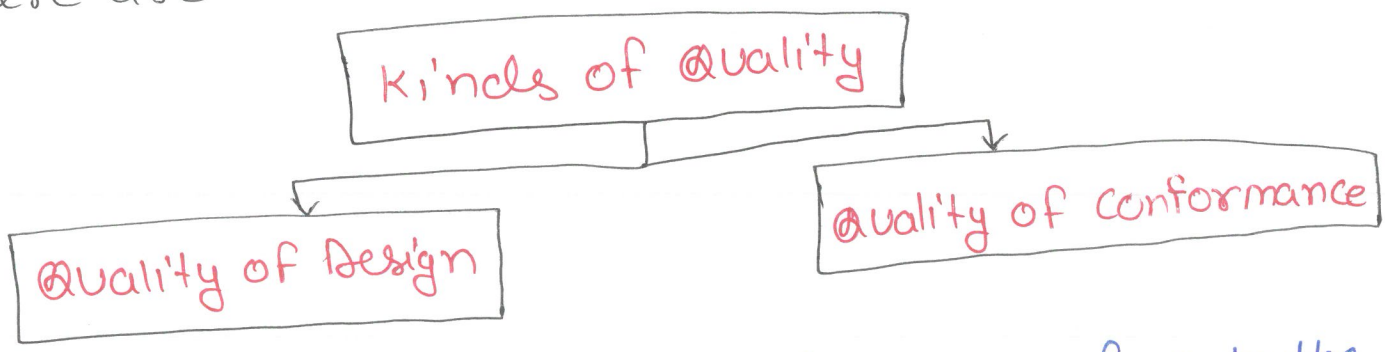
— * —

Software Quality Assurance

What is Quality ?.

- Quality defines to any measurable characteristics such as Correctness, maintainability, portability, testability, usability, reliability, efficiency, integrity, reusability and interoperability.

There are two kind of Quality



Quality of Design ⇒ Quality of Design refers to the characteristics that designers specify for an item. The grade of materials, tolerances, and performance specifications that all contribute to the quality of design.

Quality of Conformance ⇒ Quality of conformance is the degree to which the design specifications are followed during manufacturing Greater the degree of conformance, the higher is the Level of quality of conformance.

Software Quality ⇒ Software quality is defined as a field of study and practice that describes the desirable attribute of Slw product.

- Slw quality is defined as the conformance to Explicitly state functional and performance requirements, explicitly documented development standards, and inherent characteristics that are Expected of all professionally developed software.

There are two main approaches to software quality :-

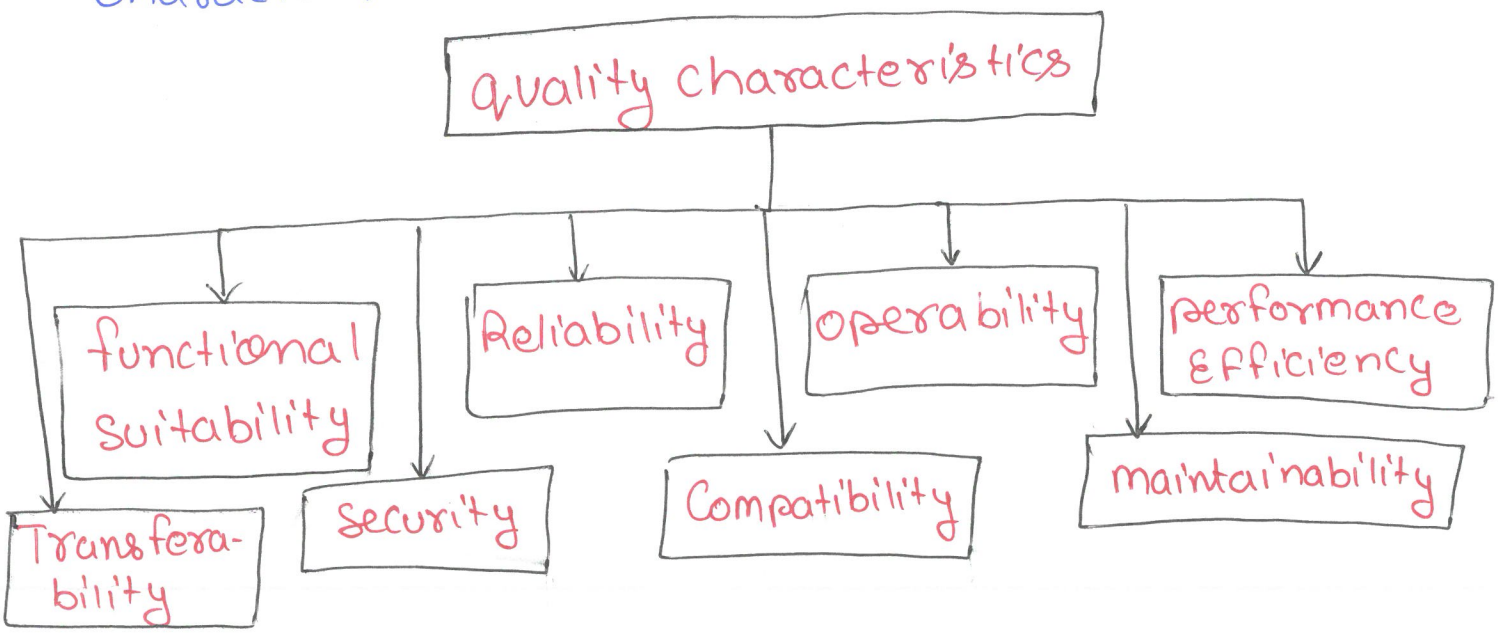
1⇒ Software Quality Defect management Approach ⇒

- A Software defect can be regarded as any failure to address end-user requirements.
- Common defects included missed or misunderstood requirements and error in design, functional Logic, data relationships, process timing, validity checking and coding errors.
- The Software defect management approach is based on counting and managing defects.

2⇒ Software Quality Attributes Approach ⇒

This approach to Slw quality is best exemplified by fixed quality models, sun as ISO/IEC 25010: 2011

This standard describes a hierarchy of eight quality characteristics, each composed of sub-characteristics.

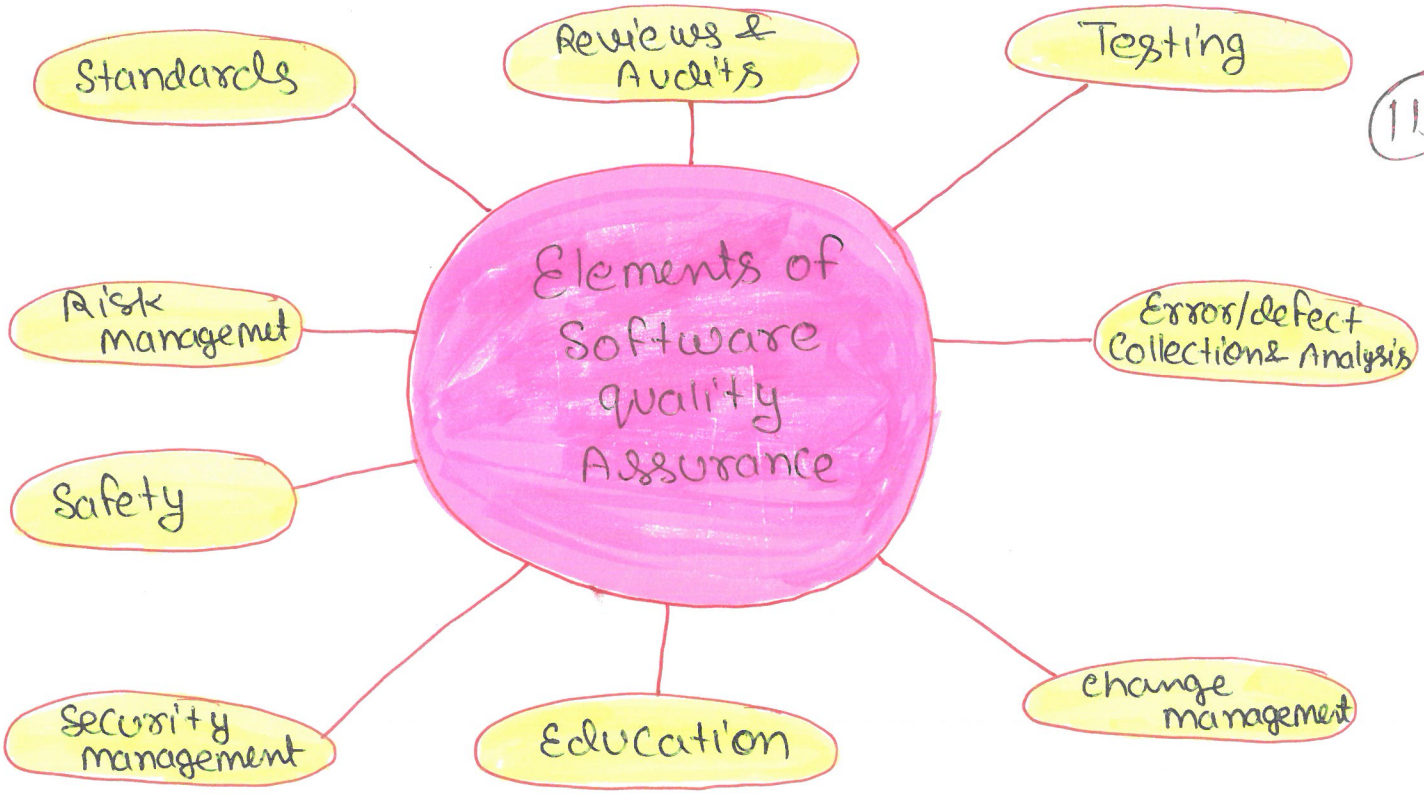


Software Quality Assurance ⇒

- (SQA) Software Quality Assurance is simply a way to assure quality in the software.
- It is the set of activities which ensure processes, procedures as well as standards are suitable for the project and implemented correctly.
- SQA is a process which works parallel to development of SW.
- It focuses on improving the process of development of SW so that problems can be prevented before they become a major issue.
- SQA is a kind of Umbrella activity that is applied throughout the SW process.

—— * ——

Elements of Software Quality Assurance:



1⇒ **Standards** ⇒ The IEEE, ISO, and other standards organizations have produced a broad array of S/w engineering standards and related documents.
The Job of SQA is to ensure that standards that have been adopted are followed, and all work products conform to them.

2⇒ **Reviews and audits** ⇒ Technical reviews are a quality control activity performed by S/w engineers for S/w Engineering. Their intent is to uncover Errors, Audits are a type of review performed by SQA personnel with the intent of ensuring that quality guidelines are being followed

for S/w engineering work.

3⇒ Testing ⇒ Software testing is a quality control function that has one primary goal - to find errors.

The job of SQA is to ensure that testing is properly planned and efficiently conducted for primary goal of S/w.

4⇒ Error / defect collection and analysis ⇒

SQA collects and analyzes error and defect data to better understand how errors are introduced and what S/w engineering activities are best suited to eliminating them.

5⇒ Change management ⇒ SQA ensures that adequate change management practices have been instituted.

6⇒ Education ⇒ Every S/w organization wants to improve its S/w engineering practices.

A key contributor to improvement is education of S/w engineers, their managers, and other stakeholders.

The SQA organization takes the Lead in S/w process improvement which is key proponent and sponsor of education programs.

7 ⇒ Security Management ⇒ SQA ensures that appropriate process and technology are used to achieve S/w security.

8 ⇒ Safety ⇒ SQA may be responsible for assessing the impact of S/w failure and for initiating those steps required to reduce risk.

9 ⇒ Risk Management ⇒ The SQA organization ensures that risk management activities are properly conducted and the risk-related Contingency plans have been established.

Software quality assurance focuses on:

- Software's portability
- Software's usability
- Software's reusability
- Software's correctness
- Software's maintainability
- Software's error control

Benefits of Software Quality Assurance (SQA):

1 ⇒ SQA produces high quality software.

2 ⇒ High quality application saves time and cost.

3 ⇒ SQA is beneficial for better reliability.

4 → SQA is beneficial in the condition of no maintenance for a long time.

5 → High quality commercial s/w increase market share of company.

6 → Improving the process of creating software.

7 → Improves the quality of the s/w.

Software Quality Assurance (SQA) Plan

- A Software Quality Assurance plan resdves arour making Sure that the product or service teaches the market trouble and bug-free.
- It should also meet the requirements defined in the SRS (Software requirement Specification).
- The purpose of an SQA plan is tree-fold. It Comprises the following.
  - Establishing the QA responsibilities of the team in question.
  - Listing areas of concern that need to be reviewed, audited and looked at.
  - Identifies the SQA work products.
- An SQA plan will work alongside the Standard development, prototyping, design, production, and release cycle for a S/w product or service.
- for easy documentation and referencing, an SQA plan will have different section like purpose, references, Configuration and management, problem reporting and Corrective action, tools, code controls, testing methodology and more.

SQA Activities ⇒

1 ⇒ Creating and plan ⇒ It consists of the engineering activities to be carried out, and ensuring the right skill mix in the team.
It also Lays out the specifics of the actions to be taken in different situations as well as the tools and procedures specific to the plan.

2 ⇒ Checkpoint lists ⇒ Evaluation of quality of activities at each project stage.
This means that there are regularly scheduled inspections and adherence to the schedule.

3 ⇒ Executing formal technical reviews ⇒ FTRs are used to evaluate the design and quality of the product prototype. Quality requirements and design quality for the prototype are discussed in meetings with the technical staff.

4 ⇒ Multi-testing Strategy ⇒ Adopting multiple testing approaches to cover all bases and ensure the best possible quality.

5 ⇒ Process adherence ⇒ Designers, developers and other technical staff must conform to established processes and employ defined procedures.

6 ⇒ Control Changes ⇒ Manual and automated Control tools are used for validating change requests, Evaluting the nature of change, Controlling and if required, arresting the change effect.

7 ⇒ Measuring Change Impact ⇒ If defects are found, the concerned department issues a fix.
The QA team then determines the change and the extent of change brought by the fix.
The change should be stable and should be compatible with the whole project.

8 ⇒ SQA Audits ⇒ The audit inspects the entire SDLC process to be established procedure laid out in the SQA plan.
Non-Compliance and missed faults can be unearthed and fixed due to this.

9 ⇒ Record and report keeping ⇒ Keeping SQA documentation and information with associated stakeholders. this includes audit reports, test results, reviews, change request documentation, and other documentation generated during the entire SDLC.

10 ⇒ Relationship management ⇒ managing and interfacing between the QA and the development team, thus keeping roles in check and responsibilities ahead of the individual.

———— * ————

Software Quality framework

- Software Quality framework is a model for Software quality that ensures quality by Connecting and integrating the different views of Slw quality.
- Slw quality framework connects the Customer View with the developer's view of Software quality and it treats software as a product.

1⇒ The Software product view describes the Characteristics of a product that bear on its ability to satisfy stated and implied needs.

2⇒ This is a framework that describes all the different concepts relating to quality in a common way measured by a qualitative scale that can be understood and interpreted commonly.

Software Quality framework can be define with three view ⇒

1⇒ Developers Views ⇒ Validation and verification are two independent methods used together to check that a software product meets the requirements and that it fulfills its intended purpose.

Validation checks that the product design satisfies the purposeful usage and verification checks for errors in the software. (124)

1⇒ The primary concern for developers is in the design and engineering processes involved in producing Slw.

2⇒ Quality can be measured by the degree of conformance to predetermined requirements and standards, and deviations from these standards can lead to poor quality and low reliability.

3⇒ While validation and verification are used by the developers to improve the slw, the two methods don't represent a quantifiable quality measurement.

4⇒ The developer's view of slw quality and the customer's view of slw quality are both different things.

2⇒ Users View ⇒ When the user acquires software, he/she always expect a high-quality software. when end users develop their slw then quality is different.

End-user programming, a phrase popularized by

which is programming to achieve the result of a program primarily for personal, rather than public use.

1⇒ The important distinction here is that S/w itself is not primarily intended for use by many users with varying needs. (125)

2⇒ For example, a teacher may write a spreadsheets to track student's test scores.

3⇒ In these end-user programming situations, the program is a means to an end that could be used to accomplish a goal.

4⇒ In contradiction to end-user programming, professional programming has the goal of producing S/w for others to use.

5⇒ For example, the moment a novice web developer moves from designing a web page for himself to designing a web page for others, the nature of this activity has changed.

6⇒ Users find S/w quality as a fit b/w their goals and S/w functionality.

7⇒ The better the quality, the more likely the user will be satisfied with the soft-ware.

8⇒ When the quality is bad, developers must meet user needs or face a diminishing demand for their S/w.

3⇒ Product View ⇒ • The product view describes quality as correlated to inherent characteristics of the product.

• Product quality is defined as the set of characteristics and features of a product that gives contribution to its ability to fullfill given requirements.

1⇒ Product quality can be measured by the value-based view which sees the quality as dependent on the amount a customer is willing to pay for it.

2⇒ According to the users, a high-quality product is one that satisfies their expectations and preferences while meeting their requirement.

3⇒ Satisfaction of end users of the product represents craft to learn, use, upgrade the product and when asked to participate in rating the product, a positive rating is given.

— * —

ISO 9000 Models

- ISO (International Standards Organization) is a group or Consortium of 63 Countries established to plan and fosters standardization.
- Iso declared its 9000 series of standards in 1987.
- It serves as a reference for the contract blw independent parties.
- The ISO 9000 standard determines the guidelines for maintaining a quality System.
- The ISO standard mainly addresses operational methods and organizational methods such as responsibilities, reporting etc.
- ISO 9000 defines a set of guidelines for the production process and is not directly concerned about the product itself.

Types of ISO 9000 Quality standards

1 ⇒ ISO 9001: ⇒ This standard applies to the organizations engaged in design, development, production, and servicing of goods.
This is the standard that applies to most s/w development organizations.

2⇒ ISO 9002:⇒ This standard applies to those organizations which do not design products but are only involved in the production. (128)

- Examples of these category industries contain Stell and car manufacturing industries that buy the product and plants designs from external sources and are engaged in only manufacturing those products.
- ISO 9002 does not apply for software development organizations.

3⇒ ISO 9003:⇒ This standard applied to organizations that are involved only in the installation and testing of the products. for example, Gas Companies.

ISO 9000 Certification ⇒

1 ⇒ Application ⇒ Once an organization decided to go for ISO certification, it applies to the registrar for registration. (129)

2 ⇒ Pre-Assessment ⇒ During this stage, the registrar makes a rough assessment of the organization.

3 ⇒ Document review and Adequacy of Audit ⇒ During this stage, the registrar reviews the document submitted by the organization and suggest an improvement.

4 ⇒ Compliance Audit ⇒ During this stage, the registrar checks whether the organization has compiled the suggestion made by it during the review or not.

5 ⇒ Registration ⇒ The Registrar awards the ISO certification after the successful completion of all the phases.

6 ⇒ Continued Inspection ⇒ The registrar continued to monitor the organization time by time.

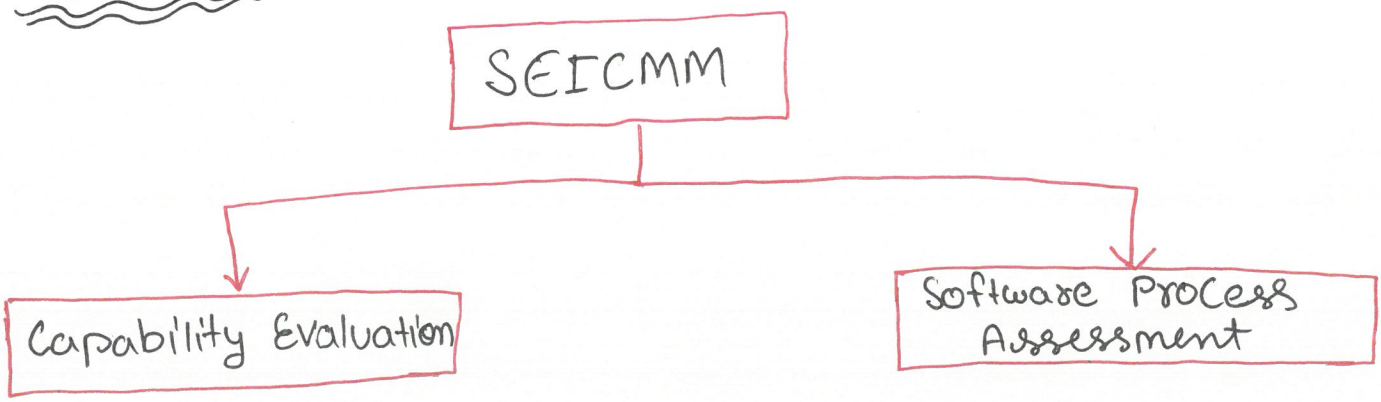
—— * ——

SEI-CMM Model

Software Engineering Institute Capability Maturity Model (SEICMM)

- The Capability Maturity Model (CMM) is a procedure used to develop and refine an organization's Software development process.
- Capability Maturity model (CMM) was developed by the Software Engineering Institute (SEI) at Carnegie Mellon University in 1987.
- CMM is not a Software process model. It is a framework that is used to analyze the approach and techniques followed by any organization to develop S/w products.
- Capability maturity Model is usedas a benchmark to measure the maturity of an organization's Software process.

Methods of SEICMM

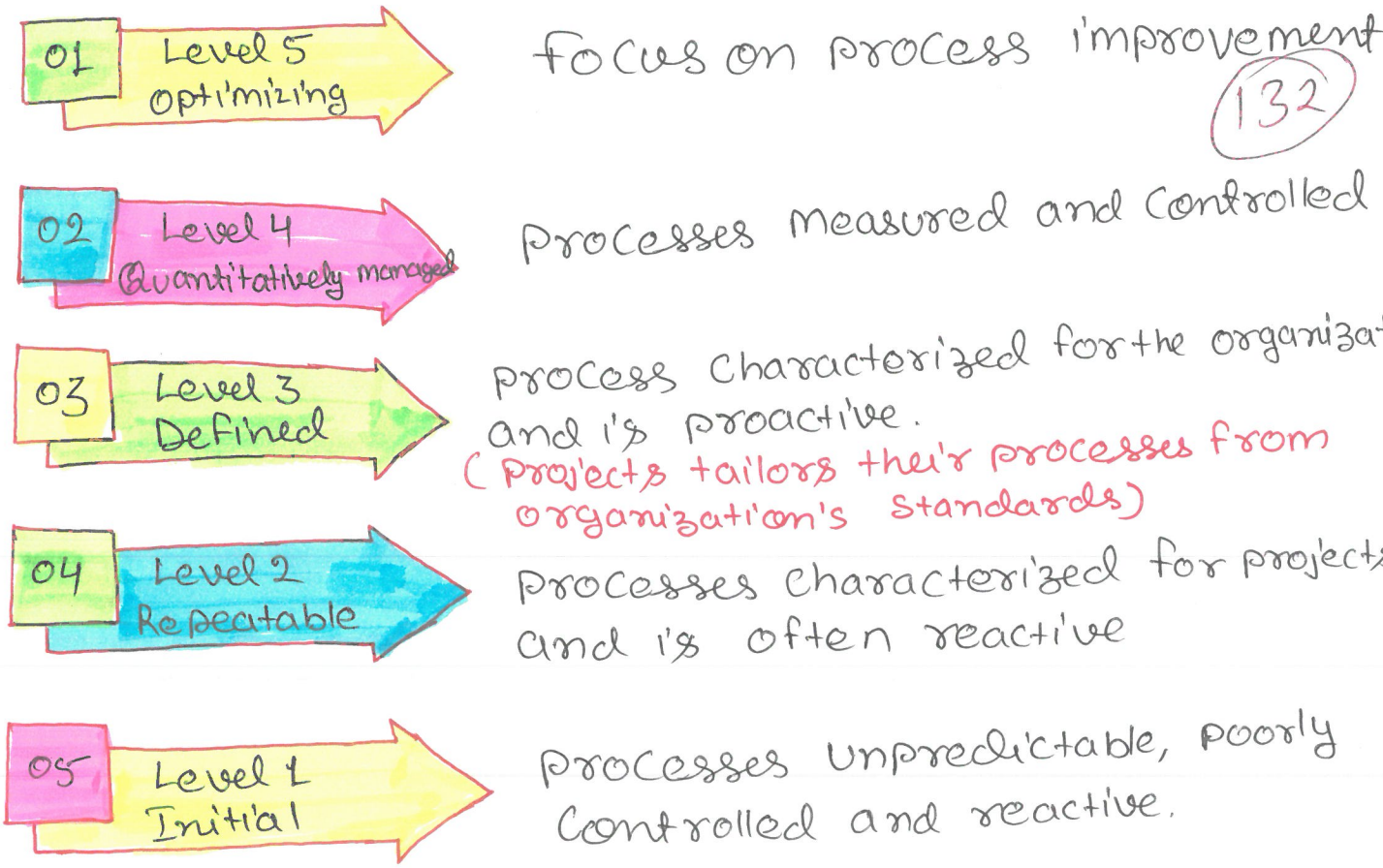


Capability Evaluation ⇒ Capability Evaluation provides a way to assess the slw process Capability of an organization. (131) The result of Capability Evaluation indicate the likely Contractor performance if the Contractor is awarded a work.
Therefore, the results of the slw process Capability assessment can be used to select a Contractor.

Software Process Assessment ⇒ Slw process assessment is used by an organization to improve its process Capability. Thus, this type of Evaluation is for purely internal use.

Levels of SEICMM ⇒

The various Levels of SEICMM have been designed so that it is Easy for an organization to build its quality System starting from Scratch slowly.



Level 1: Initial ⇒

- Processes followed are Adhoc and immature and are not well defined.
- Unstable Environment for Software Development.
- No basis for predicting product quality, time for completion etc.
- Limited project management capabilities, such as no systematic tracking of schedules, budgets or progress.
- We have limited communication and coordination among team members and stakeholders.
- No formal training or orientation for new team members.

- Highly dependent on individual skills and knowledge rather than standardized processes.
- High risk of project failure or delays due to a lack of process control and stability.

Level 2: Repeatable ⇒

- Focuses on establishing basic project management policies.
- Experience with earlier projects is used for managing new similar-natured projects.
- At this Level, the fundamental project management practices like tracking cost and schedule are established.

Size and Cost estimation methods, like function point analysis, COCOMO, etc. are used.

Level 3: Defined ⇒

- At this Level, documentation of the standard guidelines and procedures takes place.
- It is a well-defined integrated set of project-specific Software engineering and management processes.
- At this Level, the methods for both management and development activities are defined and

documented.

- There is a common organization-wide understanding of operations, roles, and responsibilities. The ways through defined, the process and product qualities are not measured.
- ISO 9000 goals at achieving this level.

Level 4: Managed ⇒

- At this stage, quantitative quality goals are set for the organization for s/w products as well as s/w processes.
- The measurements made help the organization to predict the product & process quality within some limits defined quantitatively.
- At this Level, the focus is on software product metrics & Process metrics.

Level 5: optimizing ⇒

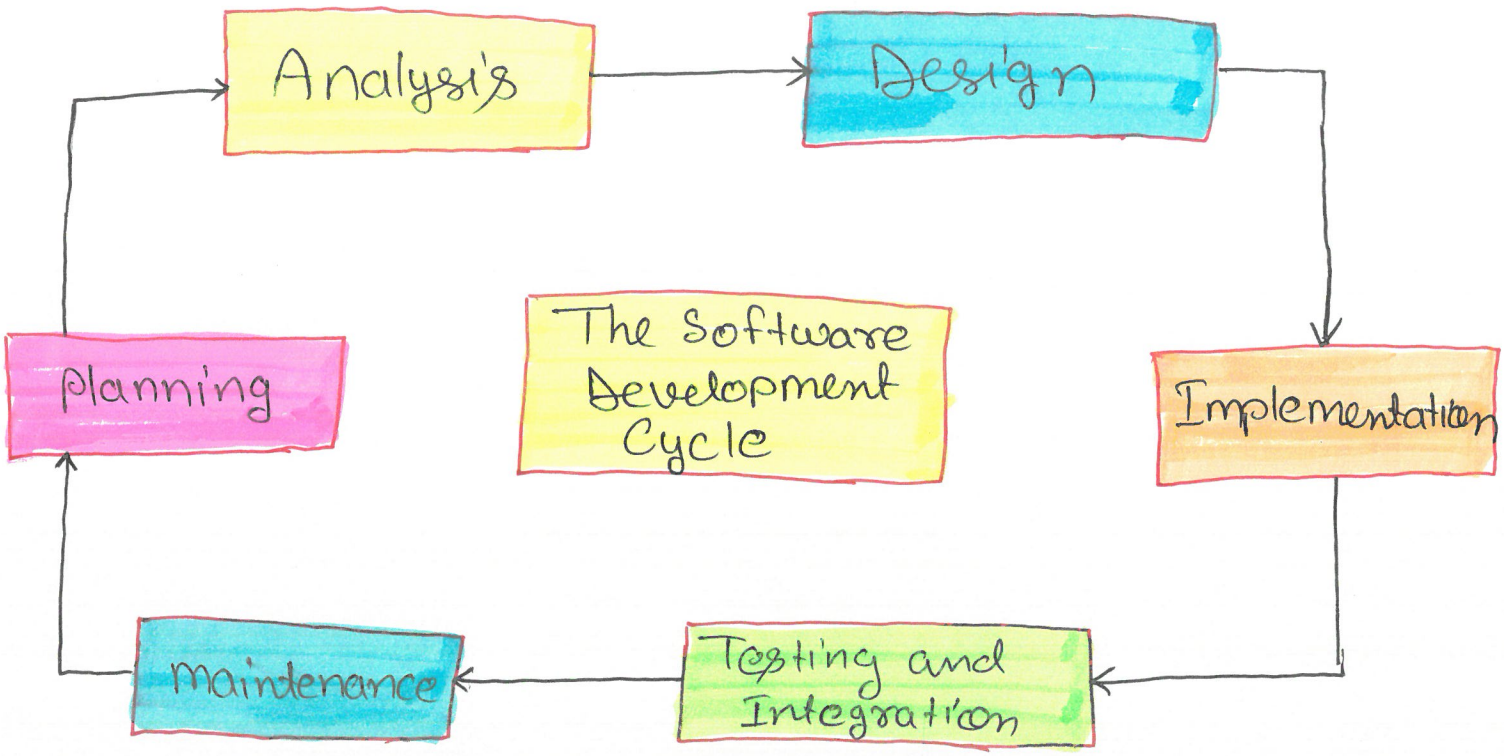
- This is the highest Level of process maturity in CMM and focuses on continuous process improvement in the organization using quantitative feedback
- At this phase, process and product metrics are collected.
- Process and product measurement data are Evaluated for continuous process improvement.

—*—

Unit - 03
Software Design

What is Software Design ?.

- Slw design is a method that converts user requirements into a suitable form for the programmer to employ in slw coding & implementation
- It is concerned with converting the client's requirements as defined in the SRS (Software Requirement Specification) document into a form that can be easily implemented using a programming language.
- A good slw designer needs to have knowledge of what slw Engineering is.



The Software design phase is the first step in the SDLC (Software Development Life cycle) that shifts the focus from the problem domain to the Solution domain.

In Software design, the system is viewed as a Collection of Components or modules with Clearly defined behaviors and bounds.

Objectives of Software Design ⇒

Correctness	Completenes	Efficiency
Flexibility	Consistency	Maintainability

- Correctness ⇒ A good design should be Correct, which means that it should Correctly implement all of the system's features.

- Efficiency ⇒ A good software design should Consider resource, time and cost optimization parameters.

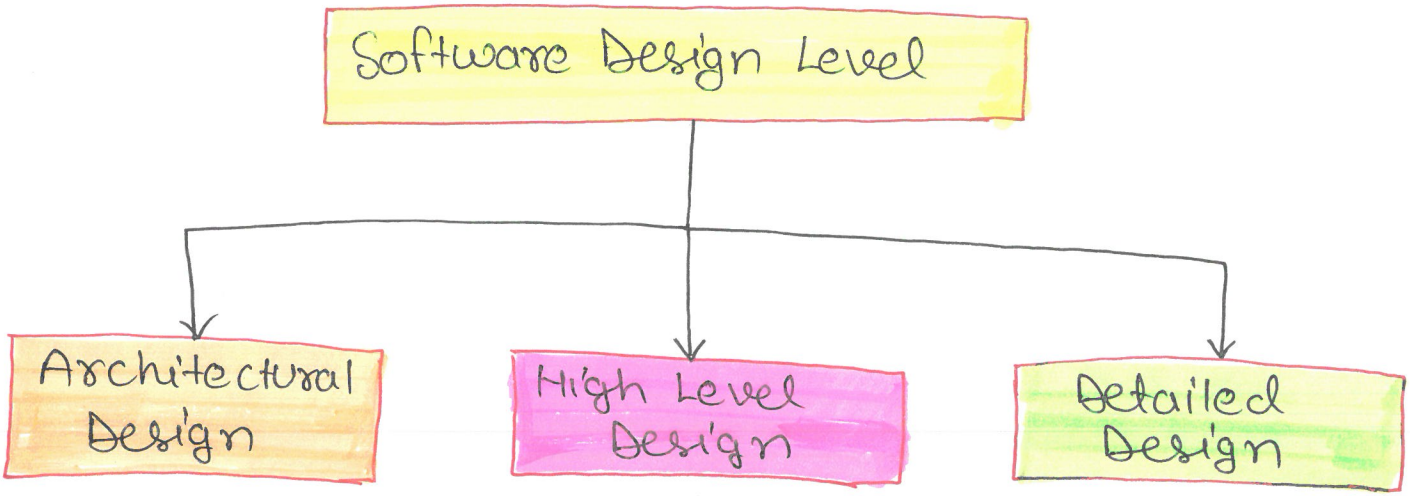
- Understandability ⇒ A good design should be easy to grasp, which is why it should be modular, with all parts organized in Layers.

- Completeness ⇒ The design should include all components, such as data structures modules, and External interfaces, among others.

- Maintainability ⇒ A good software design should be flexible when the client issues a modification request.

—— * ——

Levels of Software Design



- **Architectural Design ⇒**
  - A System's architecture can be defined as the System's overall structure and how that structure offers conceptual integrity to the System.
- The architectural design characterizes the Software as a System with numerous interconnected components.
- The designers acquire an overview of the proposed solution domain at this Level.

- **High-Level Design ⇒**
  - The high-Level Design deconstructs the architectural design's 'single entity-multiple Component' notion into a Less abstract

perspective of subsystems and modules, depicting their interaction with one another.

- High-Level design is concerned with how the System and its components can be implemented as modules.
- It recognizes the modular structure of each subsystem, as well as their relationship and interaction with one another.

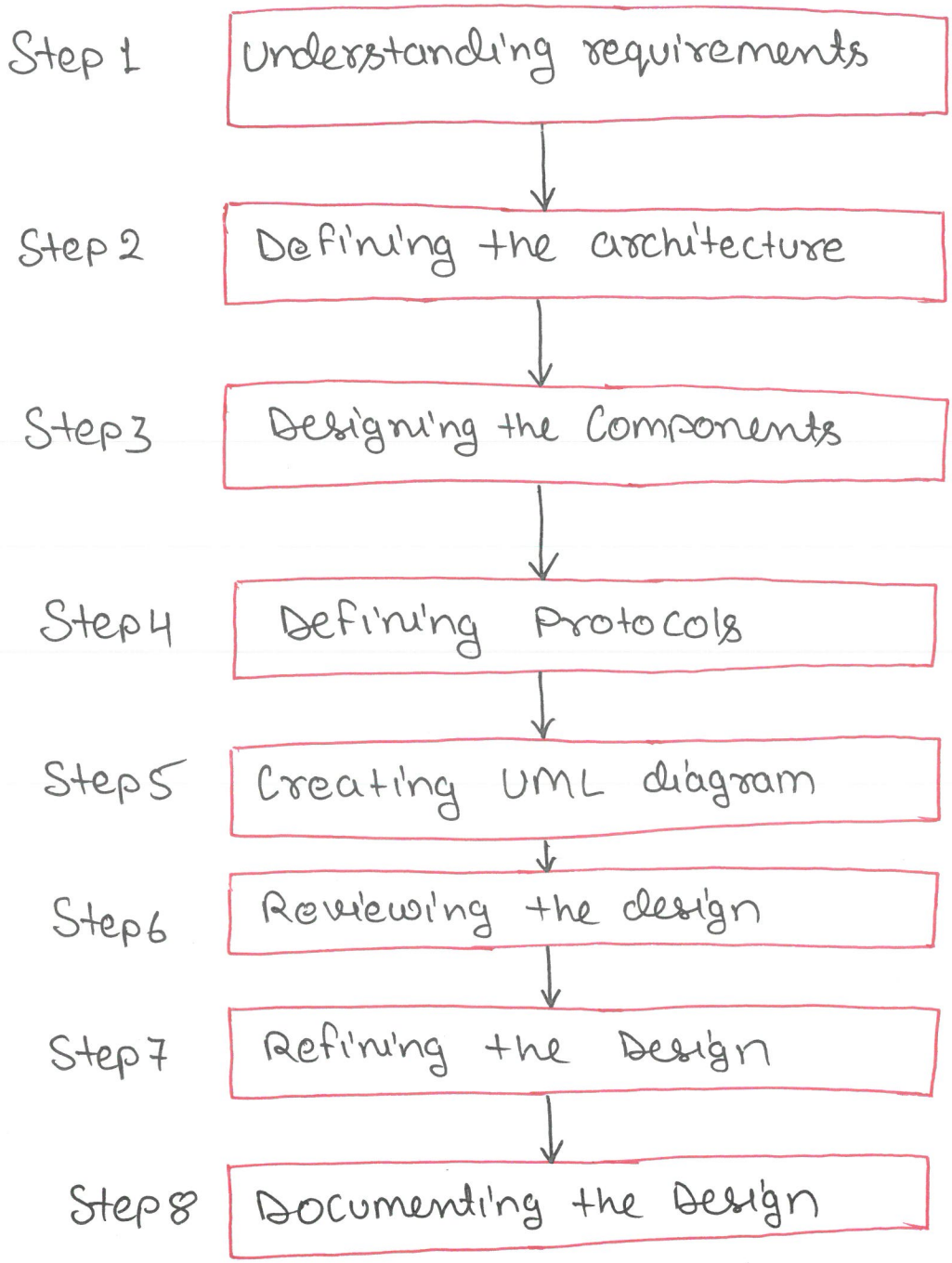
- **Detailed Design** ⇒
  - After the high-Level design is completed, the detailed design begins.
- Each module is extensively investigated at this level of software design to establish the data structures and algorithms is to be used.
- finally, a module specification document is used to document the stage's outcome.
- It defines the Logical structure of each module as well as its interfaces with other modules.

———— * ————

Low-Level Design

- LLD, or Low-Level Design, is a phase in the Software development process where detailed System Components and their interactions are specified.
- Low-Level design refers to the process of specifying and defining the detailed design of a software s/m.
- This type of design focuses on the implementation details of a system and is concerned with how the system will be built and how it will function at a detailed level.
- It provides the foundation for high-level design, which defines a system's overall architecture & design.
- LLD is an important phase of the s/w development process, because in this phase, the design is translated into concrete implementation details.
- This phase of the design process focuses on defining the data structures, algorithms and interaction btw the components that will be used to implement the system.
- Low-Level design is often performed by s/w architects and engineers, who use various techniques such as object-oriented design, design patterns and UML (unified modeling language) diagrams to define the detailed design of the s/m.

Steps involved in the Low-Level design ⇒



Step 1: Understanding Requirements ⇒

- The first step in Low-Level design is understanding the System's requirements.
- This includes reviewing the requirements document, taking a stackeholders, and gathering additional information to better understand what the system should do.

Step2:⇒ Defining the architecture:⇒

- Once the requirements are understood, the next step is to define the system's architecture.
- This involves identifying the various components that make up the system and defining how they will interact with each other.

Step3:⇒ Designing the components:⇒

- After defining the architecture, the next step is to design each component in detail.
- This includes specifying the data structures and algorithms to be used, and defining the interfaces b/w components.

Step4:⇒ Defining protocols:⇒

- In this step, the protocols that will be used for communication between components are defined.
- This includes specifying the data structures and algorithms to be used and defining the interfaces b/w components.

Step 5 ⇒ Creating UML Diagrams ⇒

- To help communicate the design, UML (Unified modeling Language) diagrams may be created to illustrate the architecture and the interactions blw components.

Step 6 ⇒ Reviewing the Design ⇒

- Once the design is completed, it must be reviewed to ensure that it meets the requirements and is technically sound.
- This may include obtaining feedback from stakeholders and conducting peer reviews.

Step 7 ⇒ Refining the Design ⇒

- If necessary, the design can be refined based on feedback received during the review process.
- This may involve changing the data structure, algorithms, or protocols used in the design.

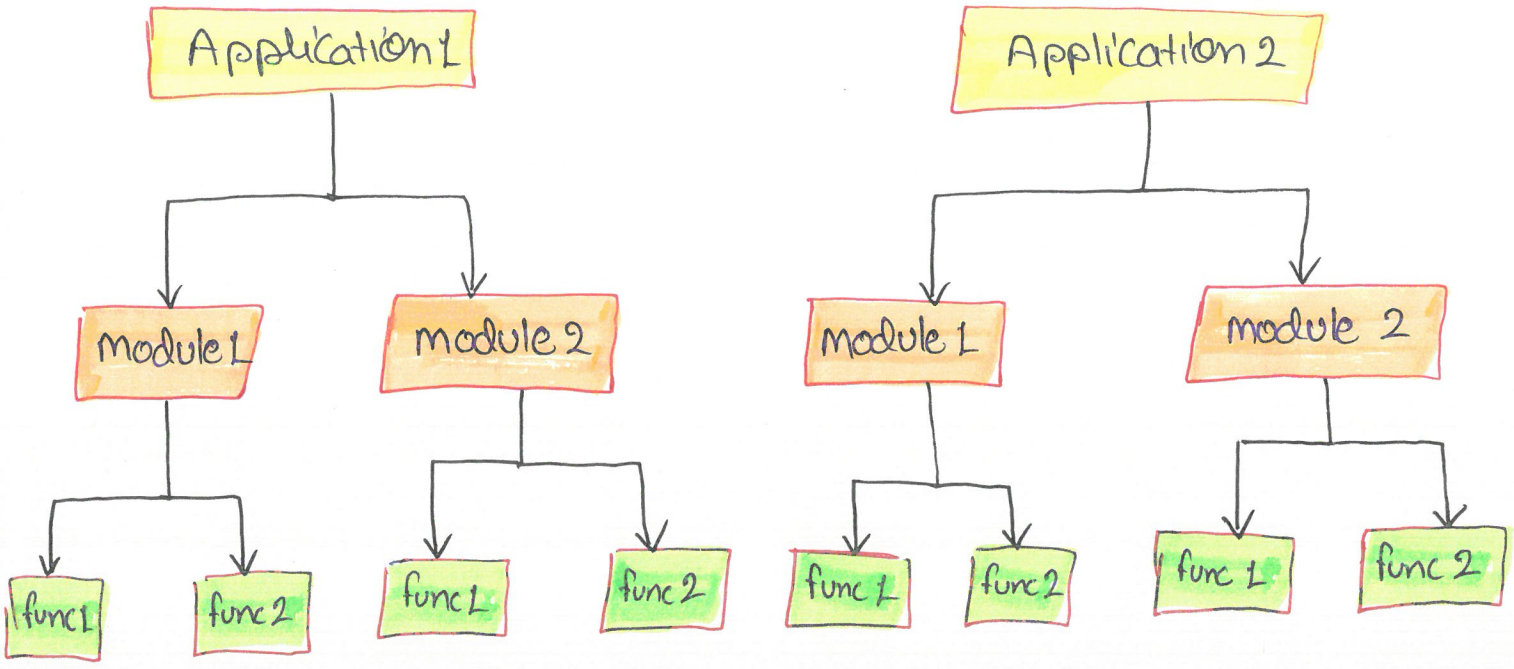
Step 8 ⇒ Documenting the design ⇒

- Once the designed has been finalized, it is important to document it in a way that is clear, concise and easy to understand.
- This may involve creating detailed design specifications or creating diagrams & flowchart to illustrate the design.

—— * ——

Modularization

- Modularization is a technique to divide a SW system into multiple discrete and independent modules, which are expected to be capable of carrying out task(s) independently.
- These modules may work as basic constructs for the entire software. Designers tend to design modules such that they can be executed and/or compiled separately and independently.
- Modularization is the process of separating the functionality of a program into independent, interchangeable modules, such that each contains everything necessary to execute only one aspect of the desired functionality.



- With modularization, we can easily work on adding separate and smaller modules to a program without being hindered by the complexity of its other functions.

- In a S/w Engineering team, we could easily work independently on each module without affecting others' work.
- Modular design unintentionally follows the rules of 'divide and conquer' problem-solving strategy this is because there are many other benefits attached with the modular design of a software.

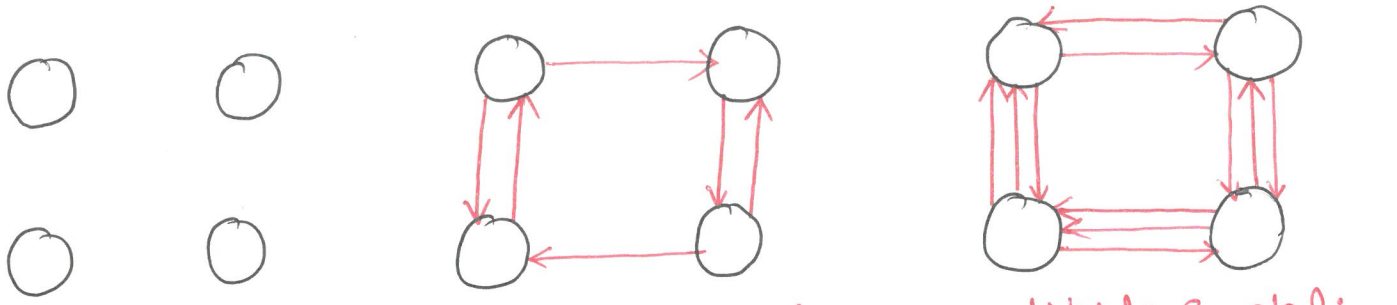
Advantages of modularization :⇒

- Smaller components are easier to maintain.
- Program can be divided based on functional aspects.
- Desired level of abstraction can be brought in the program.
- Components with high cohesion can be re-used again.
- Concurrent execution can be made possible
- Desired from security aspect.

—— * ——

Coupling and Cohesion Measures

- **Coupling** ⇒ • Coupling in Software engineering refers to the degree of interdependence between two or more module in a System.
- It measures how closely the modules are connected & how much they rely on each other.
- Low Coupling in Software Engineering promotes independence, modularity, and flexibility of the code.
- The Lower the degree of coupling, better the quality of the software.



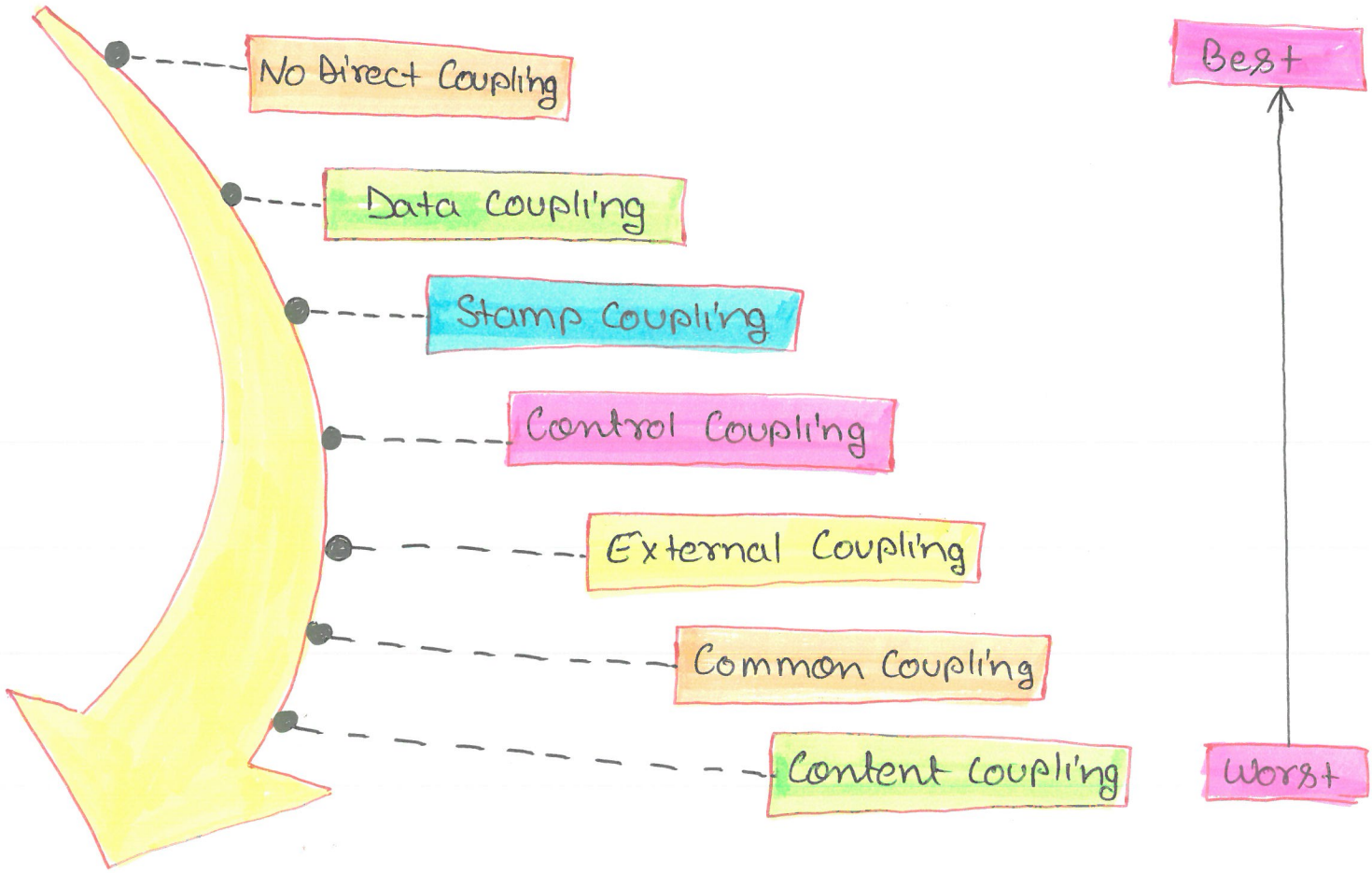
UnCoupled: no dependencies
(a)

Loosely Coupled: Some dependencies
(b)

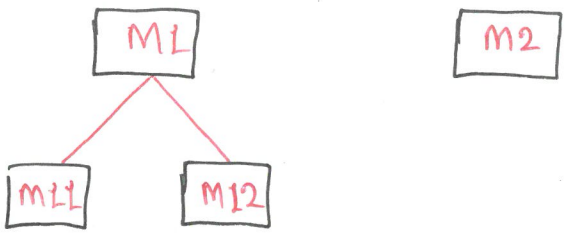
Highly Coupled: Many Dependencies
(c)

- Two modules that are tightly coupled are strongly dependent on each other.
- Two modules that are Loosely Coupled are not dependent on each other.
- Uncoupled modules have no interdependence at all within them.

Types of Coupling

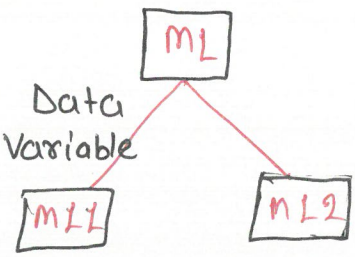


1 ⇒ No Direct Coupling ⇒ There is no direct coupling between M1 and M2.



In this case, modules are subordinates to different modules. Therefore, no direct coupling.

2 ⇒ Data Coupling ⇒ When data of one module is passed to another module, this is called data coupling.



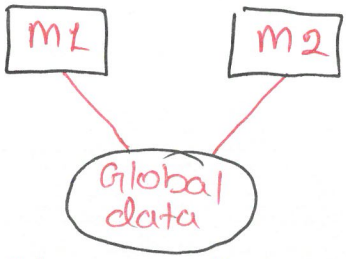
3 ⇒ Stamp Coupling ⇒ Two modules are stamp coupled if they communicate using Composite data items such as structure, objects, etc. When the module passes non-global data structure or entire structure to another module, they are said to be stamp coupled.

Ex ⇒ passing structure variable in C or object in C++ Language to a module.

4 ⇒ Control Coupling ⇒ Control Coupling exists among two modules if data from one module is used to direct the structure of instruction Execution in another.

5 ⇒ External Coupling ⇒ External Coupling arises when two modules share an externally imposed data format, Communication protocols, or device interface. This is related to communication to external tools and devices.

6 ⇒ Common Coupling ⇒ Two modulers are common Coupled if they share information through some global data items.

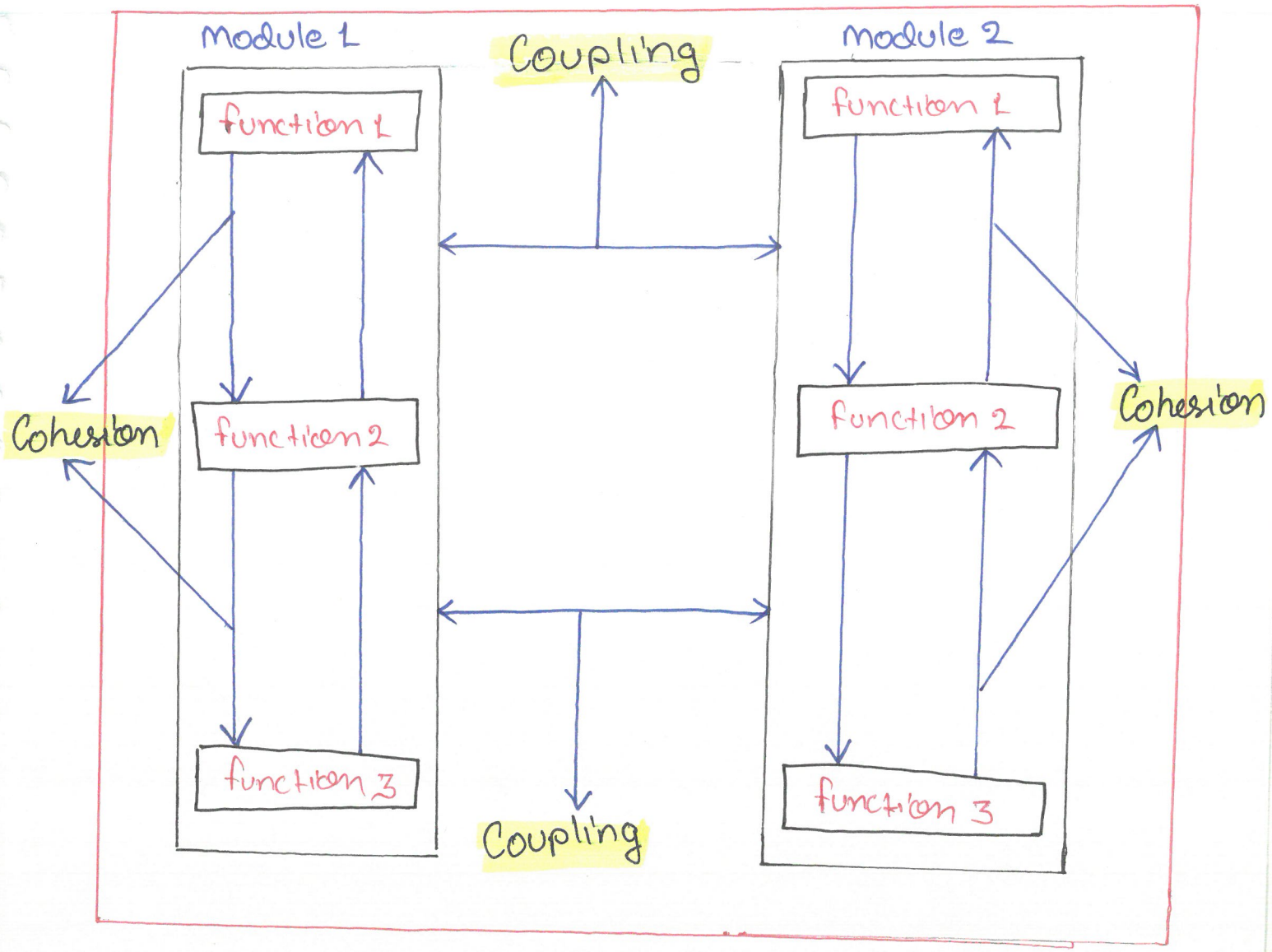


7 ⇒ Content Coupling ⇒ Content Coupling exists among two modules if they share code

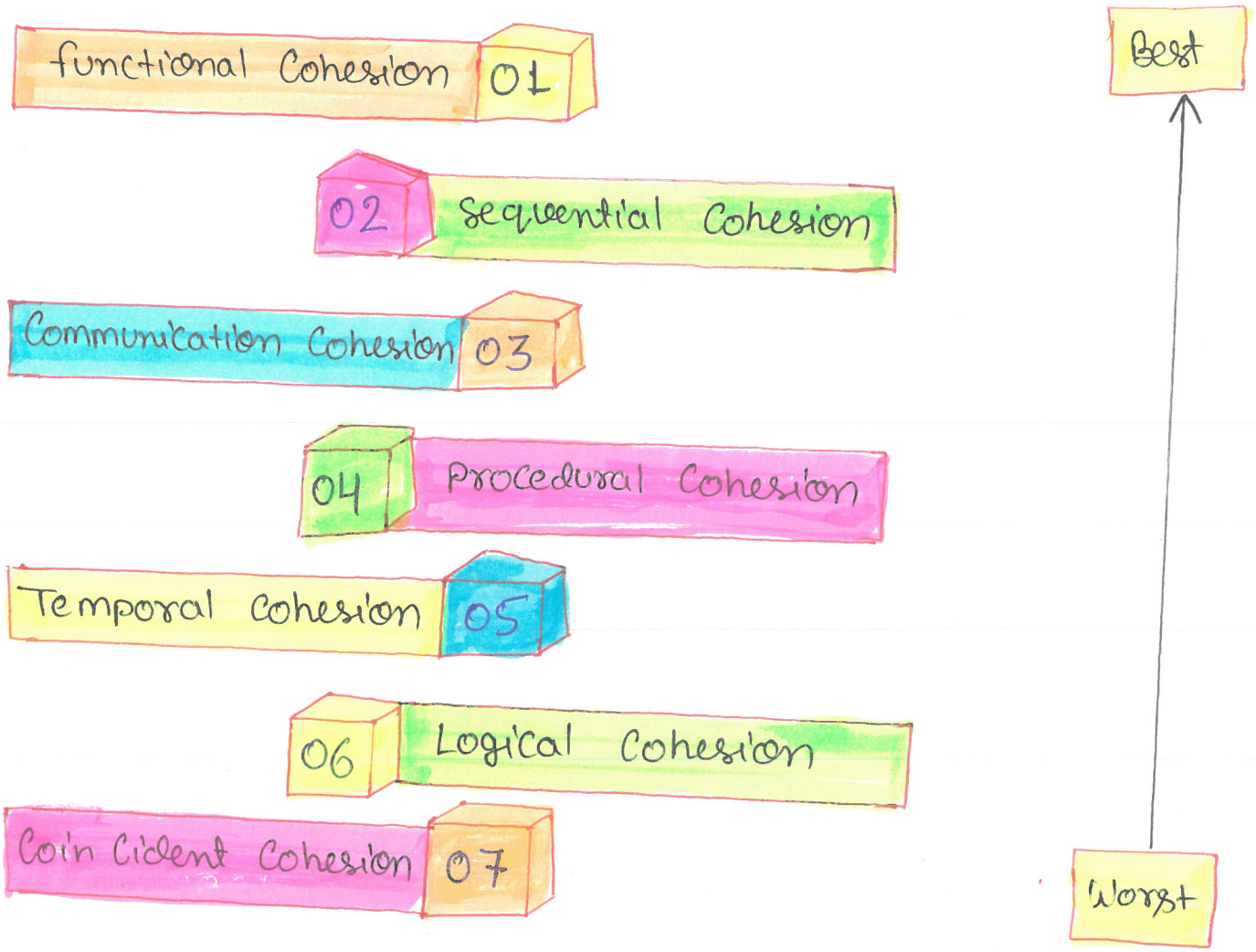
Ex ⇒ a branch from one module into another module.

Cohesion ⇒

- Cohesion is a measure of the degree to which the elements of the module are functionally related.
- It is the degree to which all elements directed towards performing a single task are contained in the Component.
- Cohesion is the internal glue that keeps the module together.
- A good software design will have high Cohesion.



Types of Cohesion



1 ⇒ functional Cohesion ⇒ functional Cohesion is said to exist if the different elements of a module, cooperate to achieve a single function.

2 ⇒ sequential Cohesion ⇒ A module is said to posses sequential Cohesion if the Element of a module from the Components of the sequence, where the output from one component of the sequence is input to the next.

3 ⇒ Communicational Cohesion ⇒ A module is said to have Communicational Cohesion, if all tasks of the module refer to or update the same data structure e.g, the set of functions defined on an array or a stack.

4 ⇒ Procedural Cohesion ⇒ A module is said to be procedural Cohesion if the set of purpose of the module are all parts of a procedure in which particular sequence of steps has to be carried out for achieving a goal.

5 ⇒ Temporal Cohesion ⇒ When a module includes functions that are associated by the fact that all the methods must be executed in the same time, the module is said to exhibit temporal Cohesion.

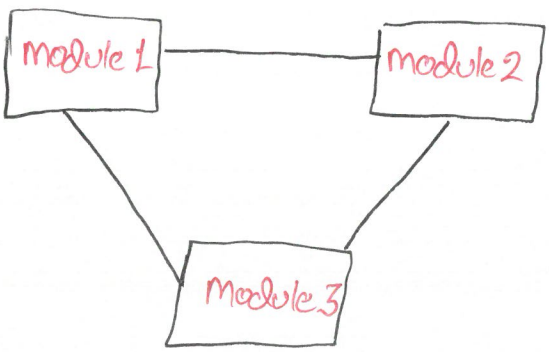
6 ⇒ Logical Cohesion ⇒ A module is said to be Logically Cohesive if all the elements of the module perform a similar operation.
for Example ⇒ Error handling, data input & data outputetc.

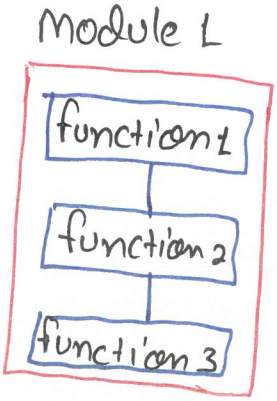
7 ⇒ Coincidental Cohesion ⇒ A module is said to have Coincidental Cohesion if it performs a set of tasks that are associated with each other very Loosely, if at all.

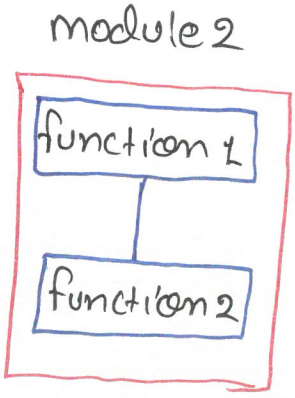
—— * ——

Difference blw Coupling & Cohesion

Coupling	Cohesion
• Coupling is the concept of inter module.	• Cohesion is the concept of Intra module.
• Coupling represents the relationships blw modules.	• Cohesion represents the relationship within module.
• Increasing in Coupling is avoided for software.	• Increasing in Cohesion is good for software.
• Coupling represents the independence among modules.	• Cohesion represents the functional strength of modules
• Where as Loosely coupling gives the best software	• Highly Cohesive gives the best Software.
• In Coupling, modules are connected to other modules.	• In Cohesion, module focuses on the single thing.
• Coupling shows the relative independence blw the modules	• Cohesion shows the module's relative functional strength.





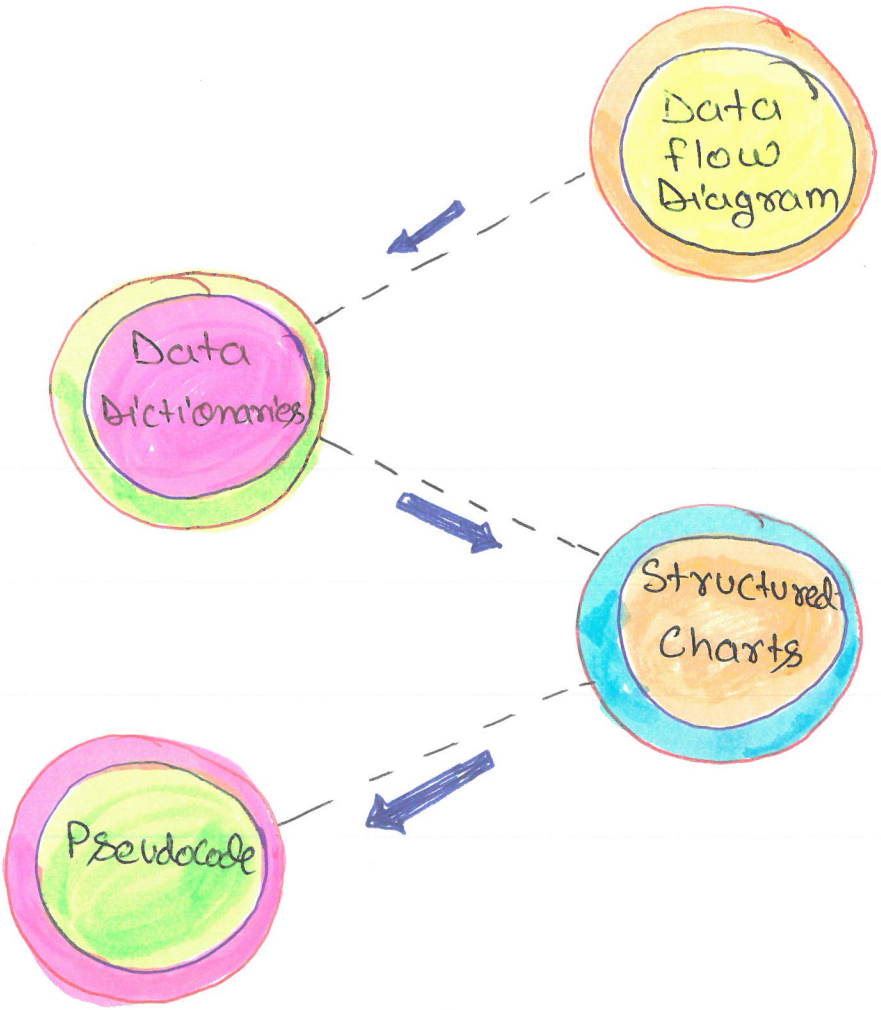


Design Strategies

function Oriented Design

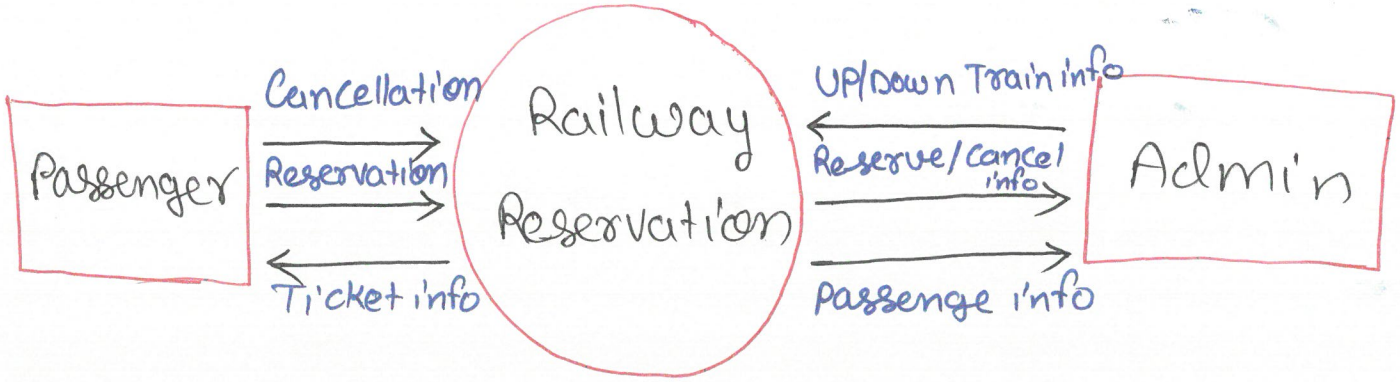
- function oriented Design is a S/w design method in which the model is broken into a series of interacting pieces or modules, each with a distinct function. As a result, the system is functionally designed.
- In function oriented design, the system is comprised of many smaller sub-systems known as functions. These functions are capable of performing significant task in the system. The system is considered as top view of all functions.
- Function oriented design inherits some properties of structured design where divide and conquer methodology is used.
- This design mechanism divides the whole S/m into smaller functions, which provides means of abstraction by concealing the information and their operation.
- These functional modules can share information among themselves by means of information passing and using information available globally.

Function Oriented Design Strategies ⇒



1 ⇒ Data Flow Diagram (DFD) ⇒ • It maps out the flow of information for any process or system.

- It uses difined symbols like rectangles Circles and arrows, plus short text Labels, to show data inputs, outputs, storage points and the routes blw each destination.



2 ⇒ Data Dictionaries ⇒ • Data Dictionaries are simply repositories to store information about all items defined in DFDs.

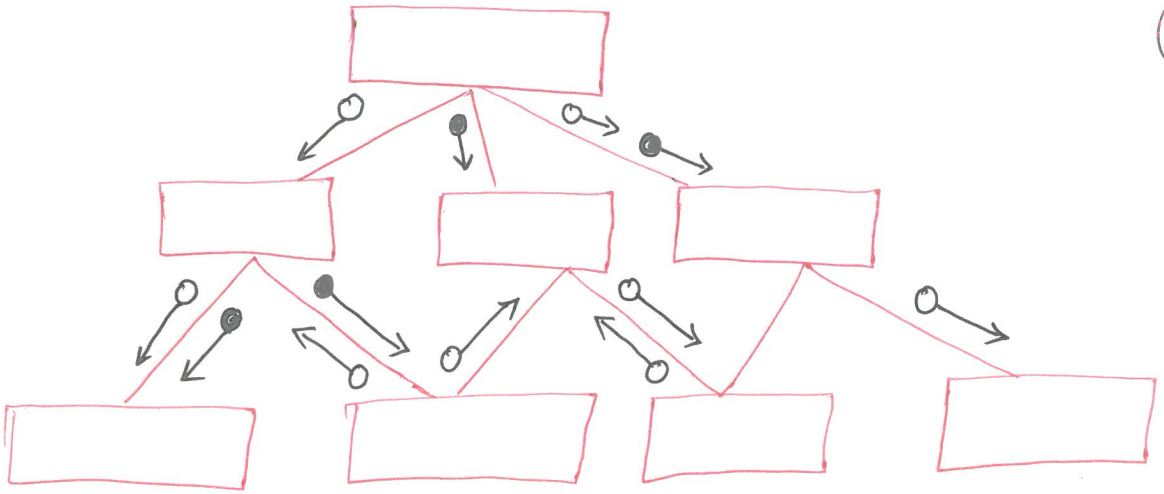
- At the requirement stage data Dictionaries Contains data items.
- Data Dictionaries include Name of the item, Aliases (other names for items), Description/ purpose, Related data items, Range of values, Data structure definition/ form.

Name	Alias	Use	Content Description	Additional Information
PNR Number	None	Enquiry Cancellation Reservation	PNR Num = Dehradun Uttarakhand -360003	None

3⇒ Structure Charts ⇒ • Structure Chart is the hierarchical representation of system which partitions the system into black boxes (functionality is known to users, but inner details are known).

- Components are read from top to bottom and Left to right.
- when a module calls another, it views the called module as black Box, passing required parameters

and receiving results.



4 ⇒ Pseudo Code ⇒ Pseudo code is system description in short English like phrases describing the function.

- It uses keyword and indentation.
- Pseudocodes are used as replacement for flow charts.
- It decreases the amount of documentation required.

Ex ⇒ Add two number Pseudo code

- Start
- Get two numbers
  - Get first no.
  - Get second no.
- Add them
- Print the answer
- End

—— * ——

Object Oriented Design

- Object oriented design works around the entities and their characteristics instead of functions involved in the software system.
- This design strategies focuses on entities and its characteristics.
- It is a Software Engineering methodology that Employs object oriented principles to model and design Complex systems.
- It involves analyzing the problem domain, representing it using objects and their interactions, and then designing a modular and scalable solution
- It helps create systems that are easier to understand, maintain, and extend by organizing functionality into reusable and interconnected Components.

Important terms Related to object-oriented Design

Objects

Classes

Messages

Abstraction

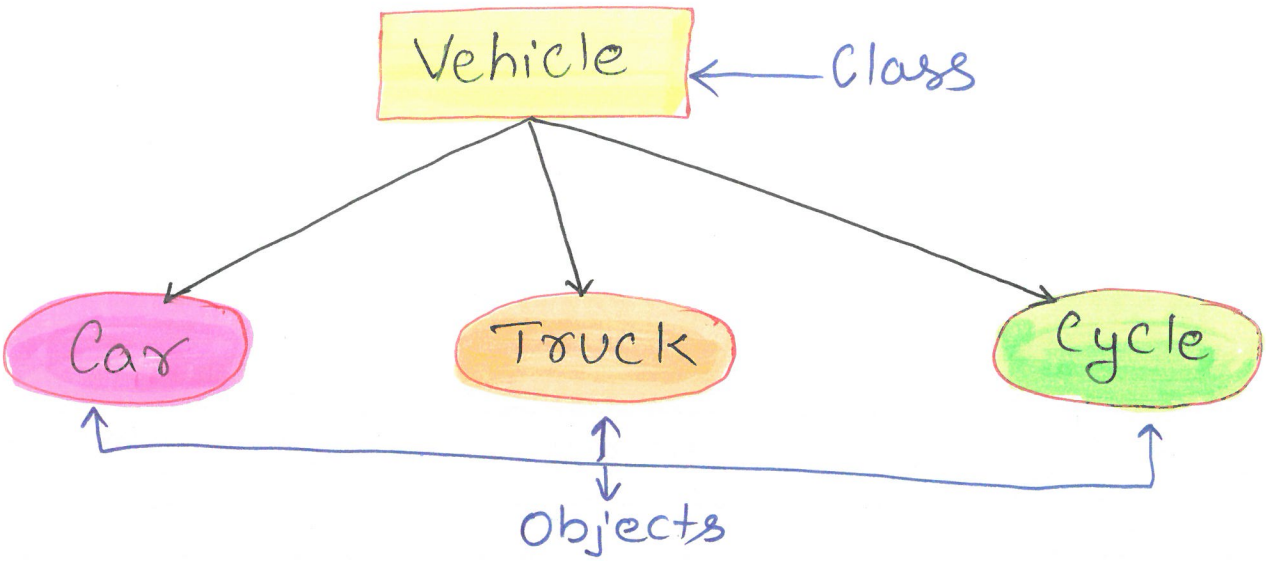
Encapsulation

Inheritance

Polymorphism

- **Objects:⇒** Objects are all the entities involved in the solution design.
- Persons, banks, companies and users are all Example of objects.
- Every object has some properties associated with it, along with some methods for performing operations on those attributes.

- **Class:⇒** Classes are generic description of Objects
- An object is a class instance.
- A Class defines all the properties on object Can have and the methods that represent the object's functionality.



- **Messages:⇒** Objects Communicate by passing messages.
- Messages Contain the target object's integrity, the name of the requested operation, and any

other action required to complete the function.

- Messages are frequently implemented through procedure or function calls.

- Abstraction ⇒ • Abstraction is used in object-oriented design to deal with complexity
- Abstraction is the removal of the unnecessary and the amplification of the necessary.

- Encapsulation ⇒ • It is also known as information concealing.
- Encapsulation is the processes and data are tied to a single unit.
- Encapsulation not only groups together an object's vital information but also restricts access to the data and operations from the outside world.

- Inheritance ⇒ • Object oriented Design (OOD) allows similar classes to be stacked hierarchically, with Lower or Sub-classes being able to import, implement, and reuse variables and functions from their immediate Superclass.
- The OOD characteristics is known as inheritance.
- This facilitates the definition of Specialized classes as well as the Creating of generic classes.

- polymorphism ⇒ OOD (object-oriented Design Languages provide a mechanism where methods performing similar tasks but vary in arguments, can be assigned the same name. (16)
- This is know as polymorphism, which allows a single interface is performing functions for different types.
- Depending upon how the service is invoked, the respective portion of the code gets executed.

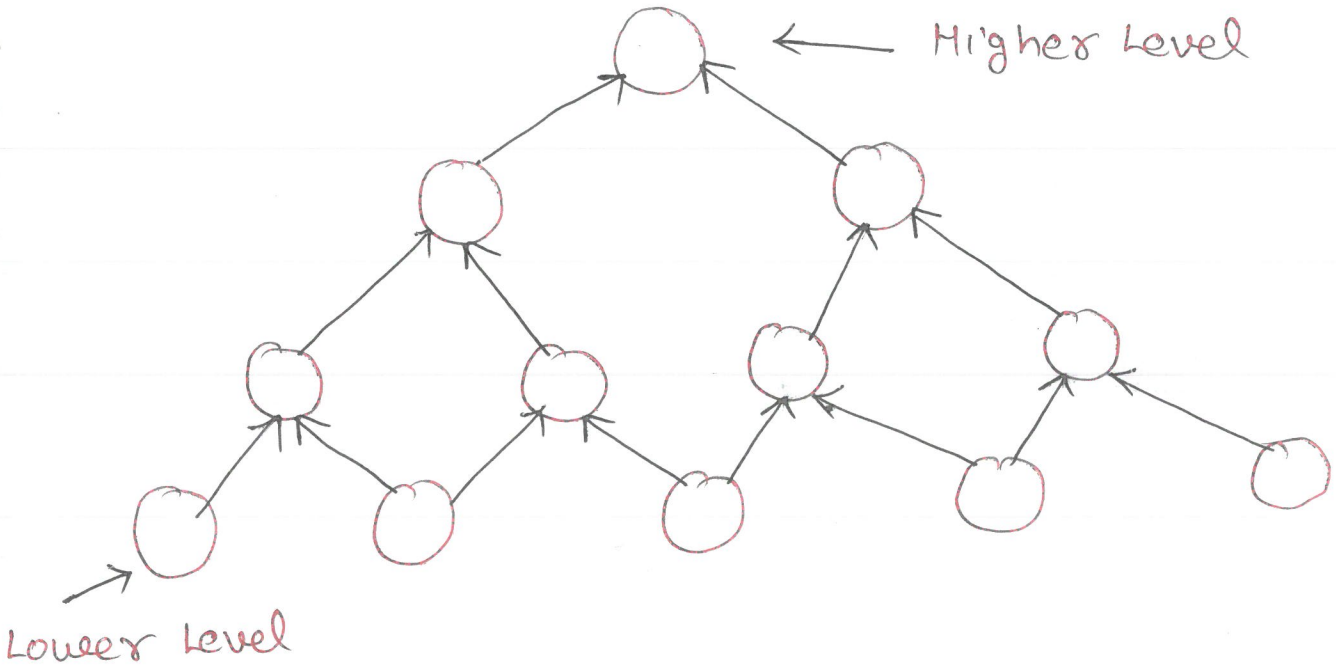
Top Down and Bottom-Up Design

- Software Engineering is the process of designing building, testing and maintaining software.
- The goal of Software Engineering is to create Software that is reliable, efficient and easy to maintain.
- System design is a critical component of SW Engineering and involves making decisions about the architecture, components, modules, interfaces, and data for a Software System.

Bottom-Up Design ⇒

- This Strategy involves starts with individual components and builds the system up, piece by piece.
- The design starts with the Lowest Level components and subsystems. By using these components, the next immediate higher-Level components and subsystems are created or composed.
- The process is continued till all the components and subsystems are composed into a single component; which is considered as the Complete System.

- The amount of abstraction grows high as the design moves to more high levels.
- By using the basic information existing S/m, when a new system needs to be created, the bottom-up strategy suits the purpose.



Advantages of Bottom-up approach: ⇒

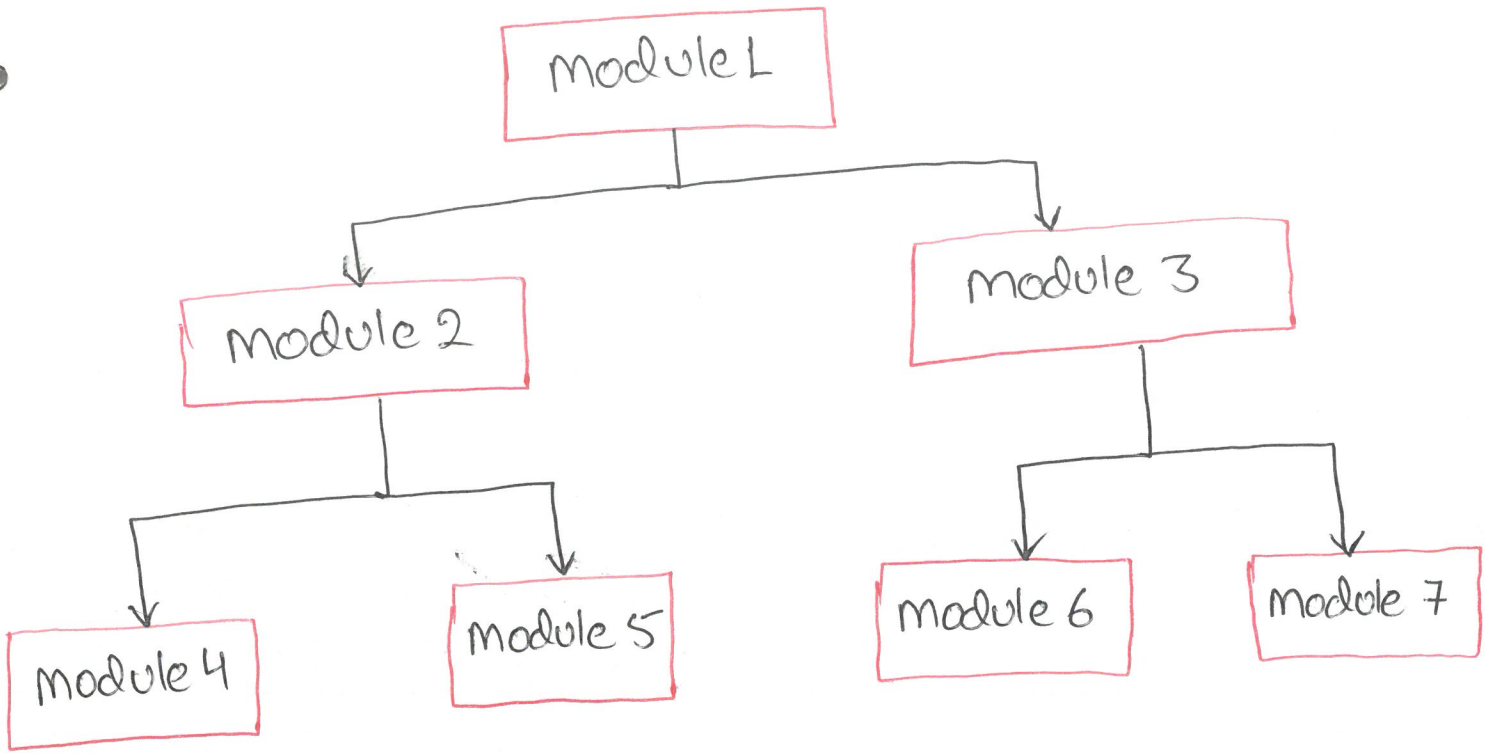
- The economics can result when general solution can be reused.
- It can be used to hide the low-level details of implementation and be merged with top-down techniques.

Disadvantages of Bottom-up approach: ⇒

- It is not so closely related to the structure of the problem.
- High-quality bottom-up solutions are very hard to construct.
- It leads to the proliferation of potentially useful functions rather than the most appropriate ones.

Top-down approach: ⇒

- Each System is divided into several subsystems and components. Each of the subsystem is further divided into a set of subsystems and components.
- This process is division facilitates forming a s/m hierarchy structure.
- The Complete s/w s/m is considered a single entity and in relation to the characteristics, the system is split into sub-systems and components. The same is done with each of the sub-systems.



- This process is continued until the lowest level of the system is reached.

- The design is started initially be defining the SIm as a whole and then keeps on adding definitions of the subsystems and components. when all the definitions are combined, it turns out to be a complete SIm.

for the solution of the SIw that need to be developed from the ground Level, a top-down design best suits the purpose.

Advantages of Top-down approach: ⇒

- The main advantages of the top-down approach is that its strong focus on requirements helps to make a design responsive according to its requirements.

Disadvantages of Top-down approach: ⇒

- Project and system boundaries tend to be application Specification-oriented. Thus it is move likely that the advantages of component reuse will be missed.
- The system is likely to miss, the benefits of a well structured, simple architecture.
- Hybrid design:
  It is a combination of both top-down and bottom-up design strategies. In this, we can reuse the modules.

Cyclomatic Complexity

- Cyclomatic Complexity (CYC) is a SLW metric used to determine the complexity of a program.
- Cyclomatic Complexity is a count of the number of decisions in the source code. The higher the count, the more complex the code.
- Thomas J. McCabe developed this metric in 1976. McCabe interprets a computer program as a set of a strongly connected directed graph.
- Nodes represent parts of the source code having no branches and arcs represent possible control flow transfers during program execution.
- The notation of program graph has been used for this measure, and it is used to measure and control the no. of paths through a program.

Cyclomatic Complexity can be used in two ways, such as: ⇒

- Limit code Complexity
- Determine the number of test cases required.

How to Calculate Cyclomatic Complexity ⇒

McCabe proposed the cyclomatic number, V(G) of graph theory as an indicator of S/w complexity. The cyclomatic number is equal to the number of linearly independent paths through a program in its graphs representation.

for a program control graph G, Cyclomatic number, V(G) is given as:-

$$V(G) = E - N + 2*P$$

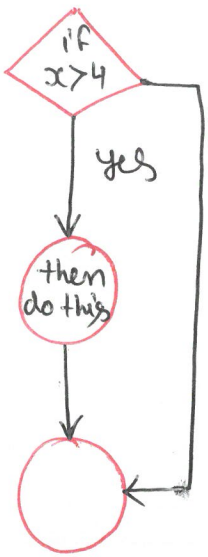
E = The no. of edges in graph G

N = The no. of nodes in graph G

P = The no. of Connected components in graph G.

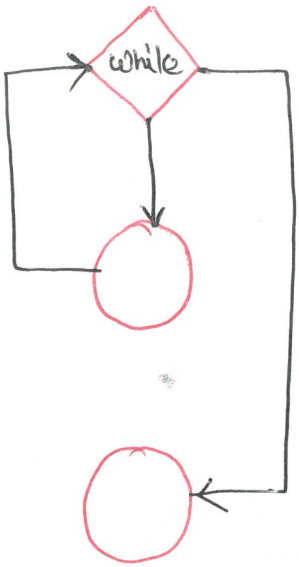
Sequence:

V(G) = 1 - 2 + 2 = 1



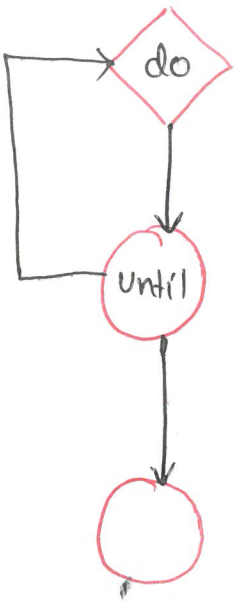
if/then

V(G) = 3 - 3 + 2 = 2



while loop:

V(G) = 3 - 3 + 2 = 2



until loop:

V(G) = 3 - 3 + 2 = 2

Ex ⇒ (1)

```
Void foo(void)
{
    if (a&&b)
        x=1;
    else
        x=2;
}
```

Cyclomatic Complexity ⇒ V(G) = 3-3+2 = 2

(2)

```
Void foo(void)
{
    if(a)
        if(b)
            x=1;
    else
        x=2;
}
```

Cyclomatic Complexity ⇒ V(G) = 4-3+2 = 1+2 = 3

Control Flow Graph (CFG)

- A Control flow Graph (CFG) is a graphical representation of a program's execution paths, Typically used in Compiler optimization and S/w Analysis
- Each node in the graph represents a basic block of code, which is a sequence of instructions with a single entry and exit point.
- The directed edges blw these nodes indicate how control can transfer blw the blocks, representing branching decisions, loops, and other flow control structures.

Characteristics of control flow Graph ⇒

- The Control flow graph is process-oriented.
- The Control flow graph shows all the paths that can be traversed during a program execution.
- A Control flow graph is a directed graph.
- Edges in CFG portray control flow paths and the nodes in CFG portray basic blocks.

There exist 2 designated blocks in the Control flow Graph:

1⇒ Entry Block:⇒ The entry block allows the Control to enter into the Control flow graph.

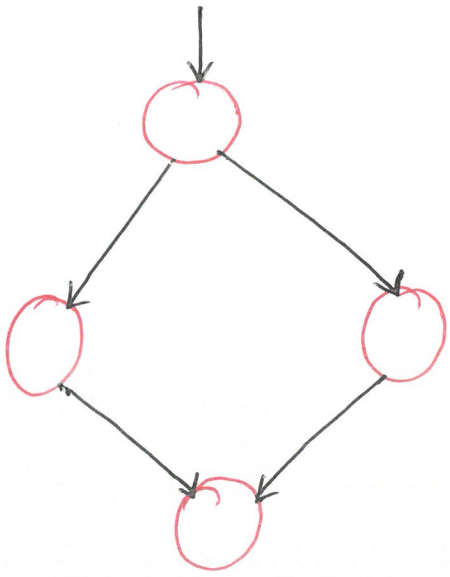
2⇒ Exit Block:⇒ Control flow leaves through the Exit Block.

Control flow graph Comprises all the building blocks involved in a flow diagram such as the start node, end node and flows between the nodes.

General Control flow Graphs ⇒

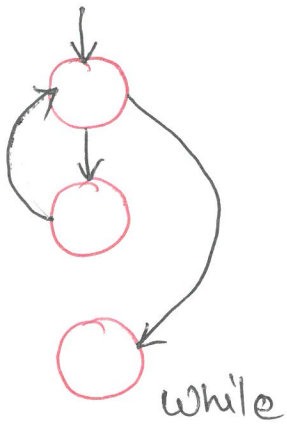
Control flow Graph is represented differently for all Statements and loops.

1. if-else ⇒



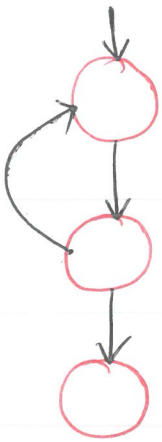
If-then-else

2. while ⇒



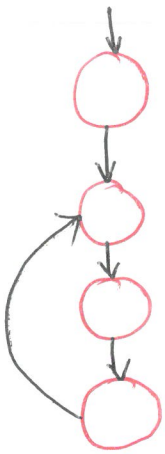
while

3. do-while ⇒



do-while

4. for



for

Advantage of CFG ⇒

There are many advantages of a control flow graph.

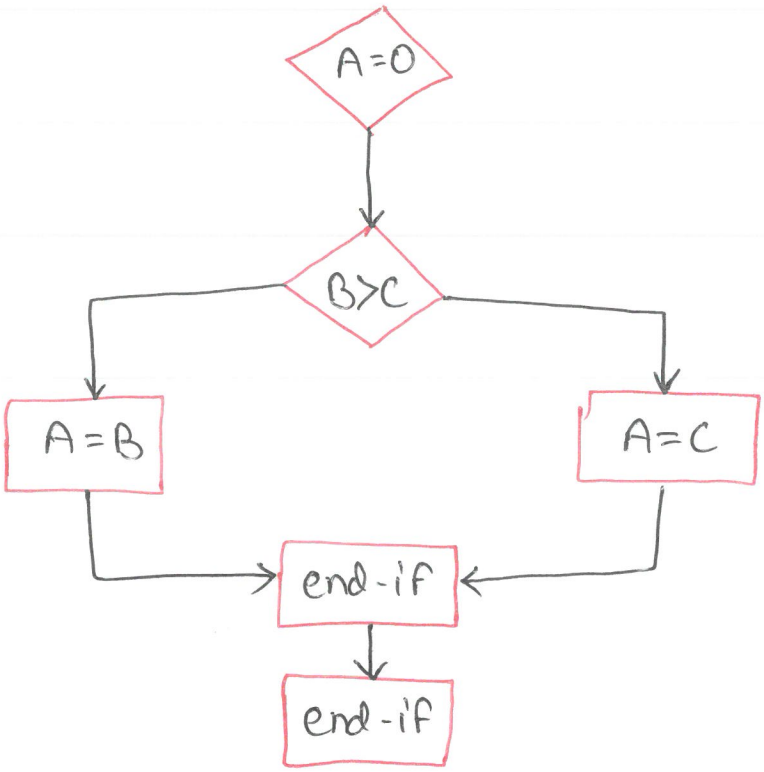
1 ⇒ It can easily encapsulate the information per each basic block.

2 ⇒ It can easily locate inaccessible codes of a program and syntatic structures such as loops are easy to find in a control flow graph.

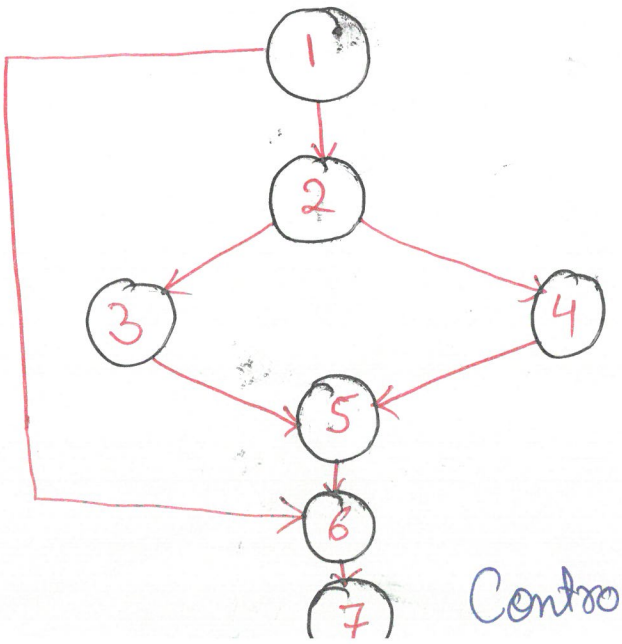
Example ⇒

```
if A =10 then
if B > C
   A = B
else A = C
endif
endif
print A, B, C
```

flowchart of above example ⇒



Control flow Graph of above example ⇒



Control flow Graph