Unit 1

Software Engineering separately consists two words that is **Software And Engineering.**

To understand **Software Engineering** we have to understand **what is software** and **what is Engineering first.**

what is software:-

Software:

It refers to a program or set of programs and applications used to manage and control various functions of a device such as a **computer**, unlike hardware, which represents a physical part of a device, software is virtual.

or

Computer program and associated documentation. Software products may be developed for a particular customer or maybe developed general market. **By Ian Sommerville.**

or

Software is-

1) computer program that when executed provide desired features functions and performance.

2) data structure that enable the program to adequately manipulate information, and

3) detective information in both hard copy and virtual form that describes the operation and use of program. **By Rogers pressman.**

Software:

It is more than just a program code. A program code which serves some computational purpose. Software is considered to be collection of executable programming codes, Associated libraries and documentation.

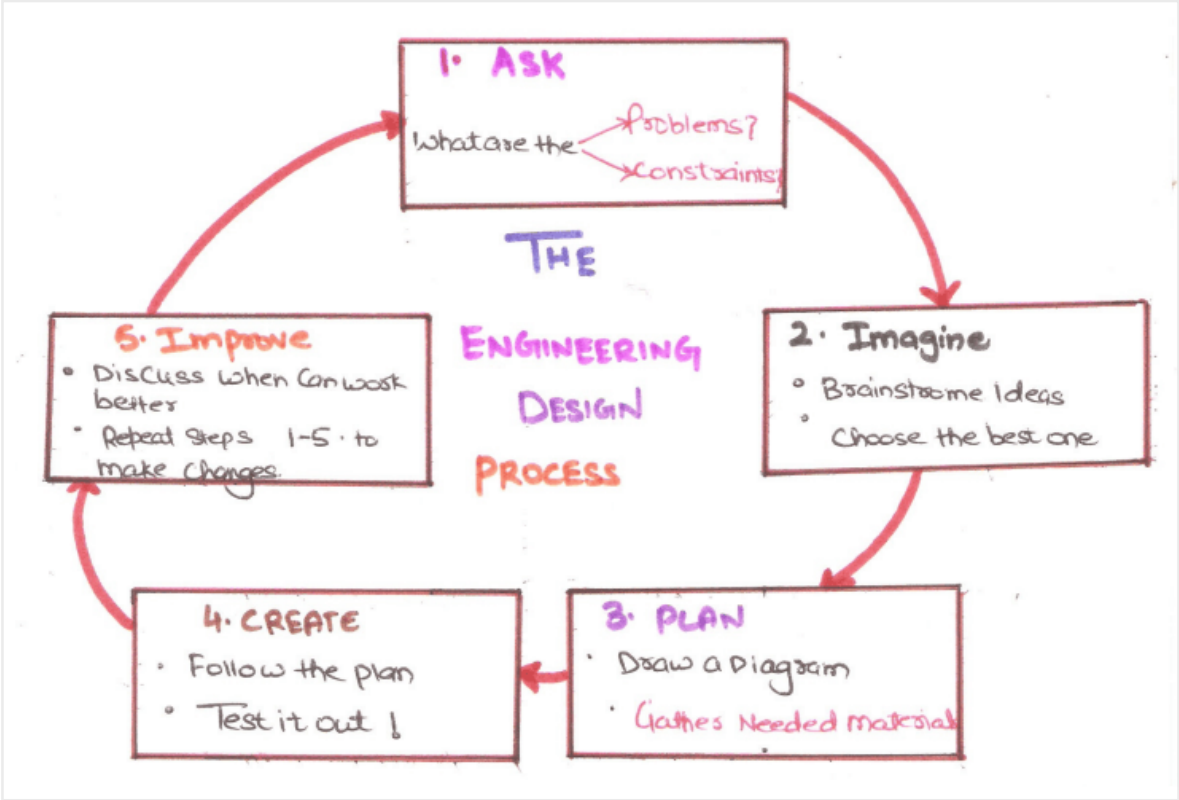
Software, when made for a specific requirement is called **Software Product.**

What is Engineering?

Engineering is all about developing products, using well defined, scientific methods and principles.

Engineering in the application of scientific knowledge problem in

Real world.



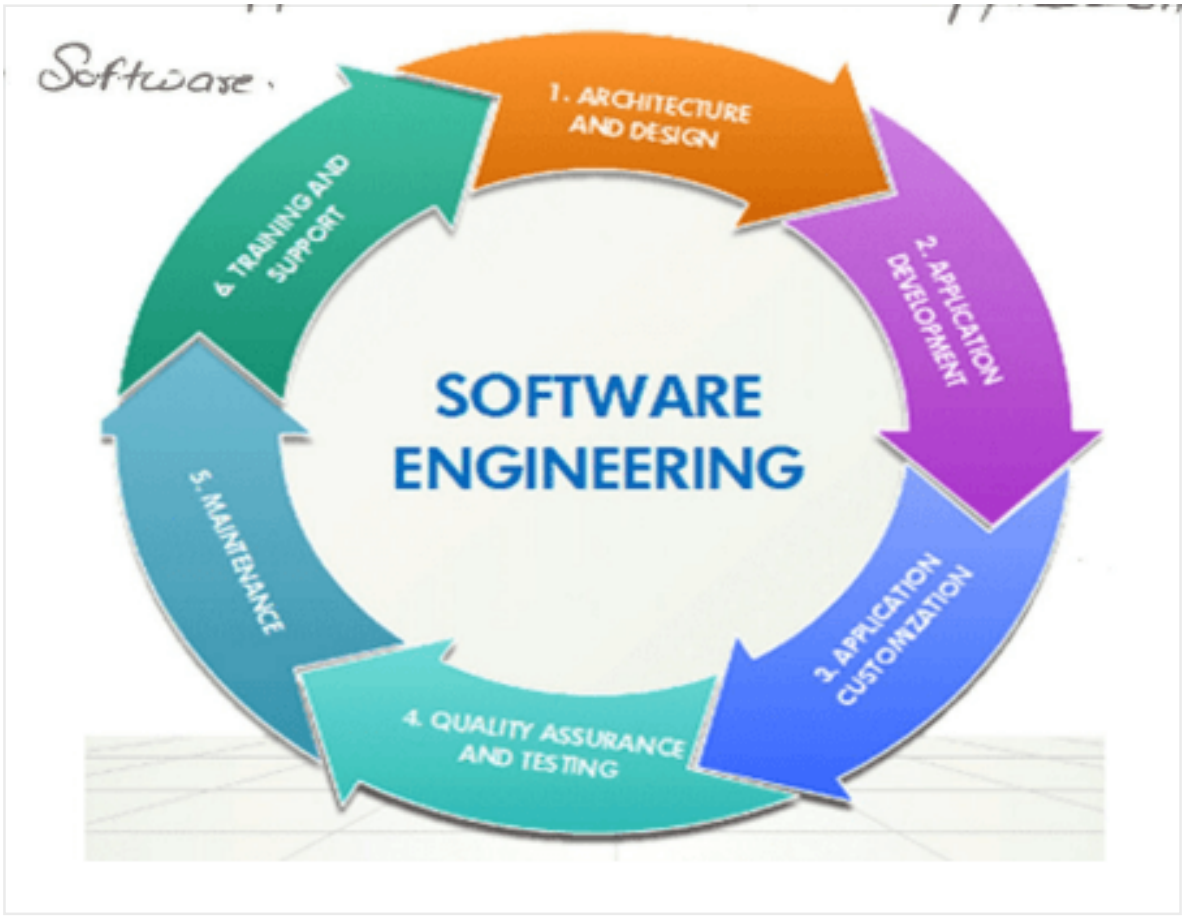
What is software engineering?

It is an **Engineering branch** associated with the development of software product using well-defined **scientific principles, method, and procedure.** The outcome of software engineering is an efficient and reliable software product.

So now in brief:

Engineering forces us to focus on Systematic, scientific and well defined processes to produce "A Good Quality Product."

The application of a **Systematic, Disciplined, Quantifiable** approach, to the development, operation and maintenance of software, and the study of these approaches, that is the application of Engineering to software.

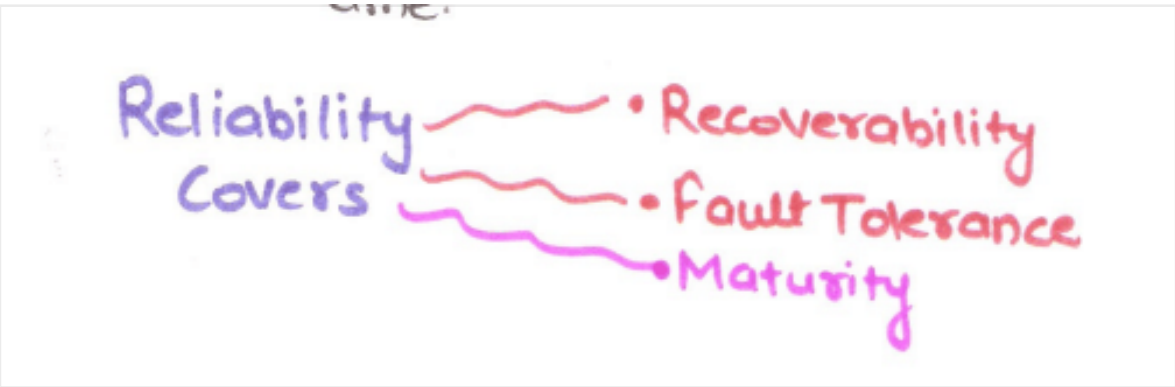


Software Characteristics:

Software characteristics are classified into 6 major components:

Functionality: It refers to the degree of performance of the software against its intended purpose. It basically means are the required functions.

Reliability:A set of attribute that Bear on the capability of software to maintain its level of performances understated conditions for a stated period of time.

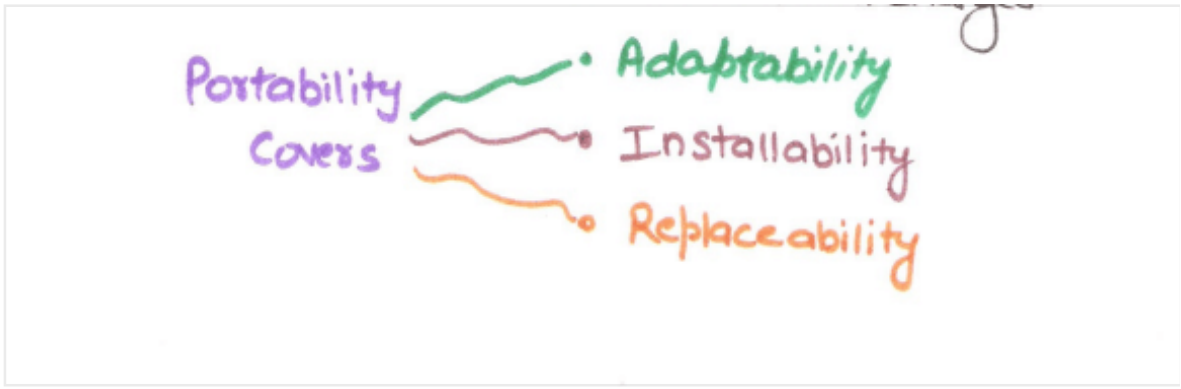


Efficiency: It refers to the ability of the software to use System Resources in the most Effective and Efficient Manner. The software should make effective use of storage space and executive commands as per desired timing requirement.

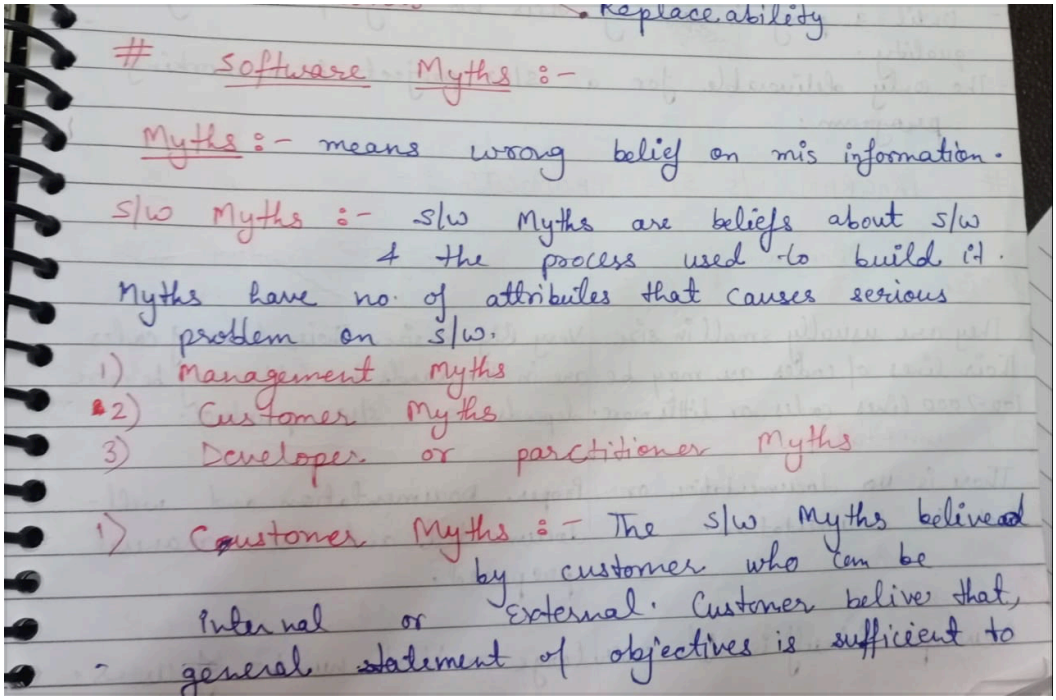
Usability: It refers to the extent to which the software can be used with ease. Or the amount of effort or time required to learn how to use the software should be less.

Maintainability: Refers to the ease with which the modifications can be made in a software system to extend its functionality, improvement, performance or correct errors.

Portability: A set of attributes that bears on the ability of the software to be transferred from one environment to another, without or minimum changes.



Robustness and integrity are also important:
Robustness: It refers to the degree to which the software can keep on functioning in spite of being provided with invalid data.
Integrity: It refers to the degree to which Unauthorized Access to the software data can be prevented.


Replaceability

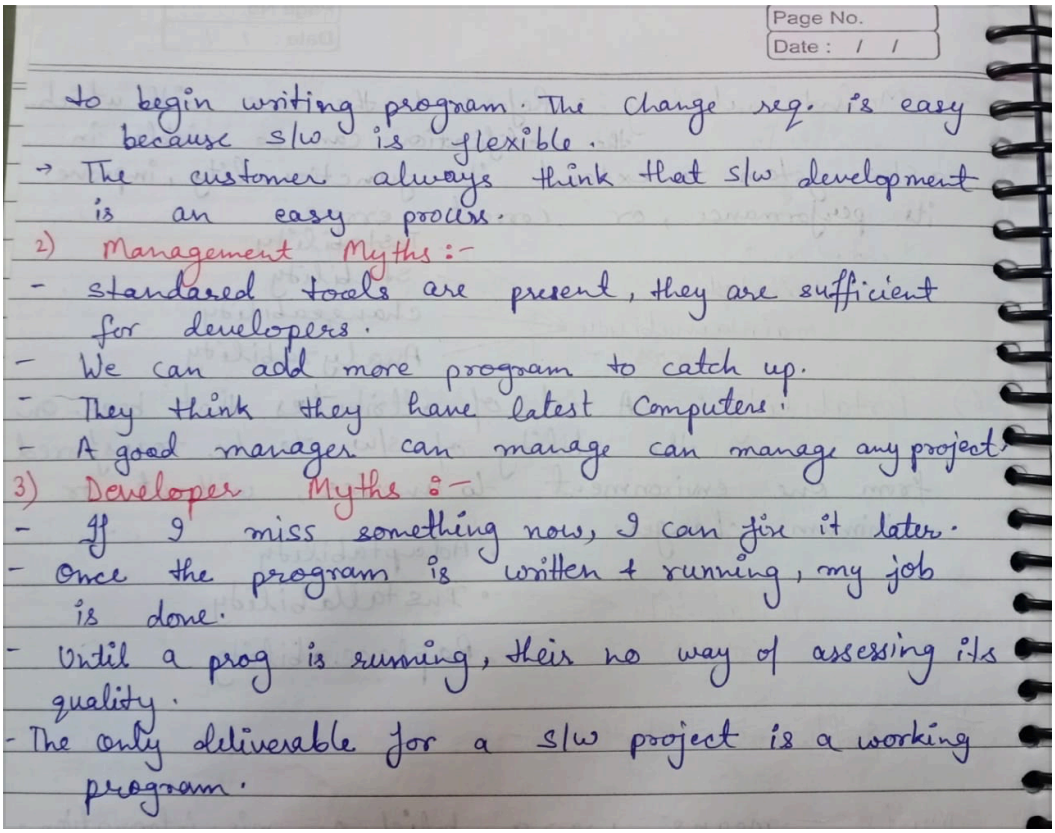
Software Myths :-

Myths :- means wrong belief on mis information.

S/w Myths :- S/w Myths are beliefs about s/w & the process used to build it. Myths have no of attributes that causes serious problem on s/w.

1) Management Myths
2) Customer Myths
3) Developer or parctitioner Myths

1) Customer Myths :- The s/w Myths believed by customer who can be Internal or External. Customer belive that, general statement of objectives is sufficient to


Page No.
Date : / /

to begin writing program. The change req. is easy because s/w is flexible.
→ The customer always think that s/w development is an easy process.
2) Management Myths :-
- standared tools are present, they are sufficient for developers.
- We can add more program to catch up.
- They think they have latest Computers.
- A good manager can manage can manage any project.
3) Developer Myths :-
- If I miss something now, I can fix it later.
- Once the program is written & running, my job is done.
- Until a prog is running, their no way of assessing it's quality.
- The only deliverable for a s/w project is a working program.

Applications Of Software

The most significant factor in determining which software engineering methods and techniques are most important is the type of application that is being developed.

Different Types of Software Application

System Software: A collection of programs written to service other programs. Compiler, device driver, editors, file management.

Application software or stand alone program: It solves a specific Business needs. It is needed to convert the business function in real time. Example -point of sale, Transaction processing, real time manufacturing control.

Scientific / Engineering Software: Applications like based on astronomy, automative stress analysis , molecular Biology, volcanology, space Shuttle orbital dynamic, automated manufacturing.

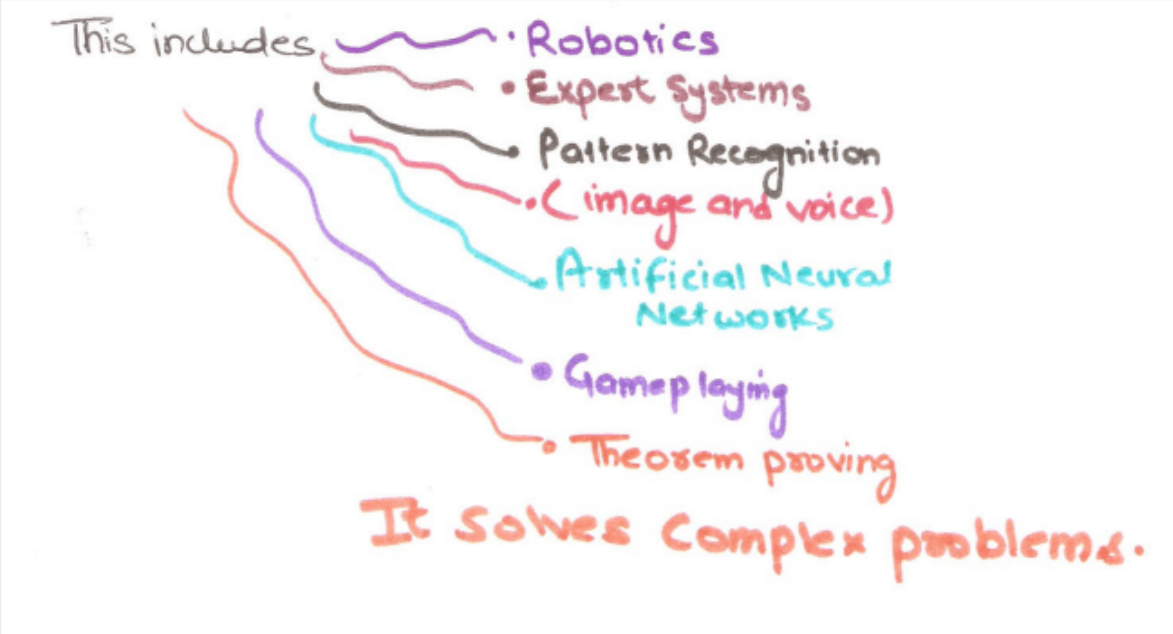
Embedded Software: There are software control systems that control and manage hardware devices. Example- software in mobile phone, software in Anti Lock Braking in car, software in microwave oven to control the cooking process.

Product Line Software: It is designed to provide a specific capability for used by many different customers. It can focus on

unlimited or esoteric Marketplace like inventory control products. Or address mass market place like : Spreadsheets, computer graphics, multimedia, entertainment, database management, personal, business financial applications.

Web application: It is also called " web apps ", are evolving into sophisticated computing environment that not only provide stand alone features, computing functions, and content to the end user but also are integrated with corporate database and business applications.

Artificial intelligence software: This include- robotic, expert system, pattern recognition, image and voice, artificial neural network, game playing, theorem proving ... It solves Complex problems.



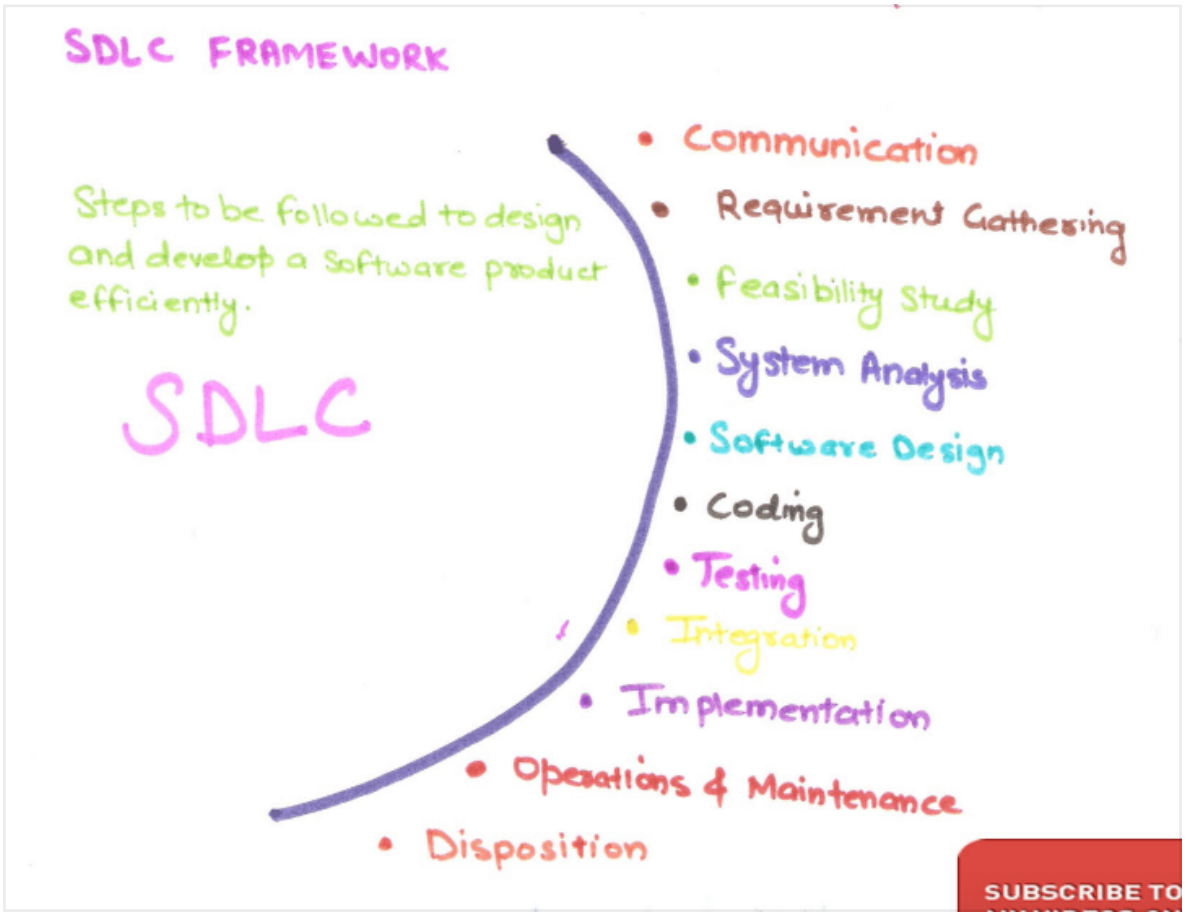
Software Development Life Cycle

Software development life cycle Or sdlc:

It is a sequence of activities that leads to the production of a software product.

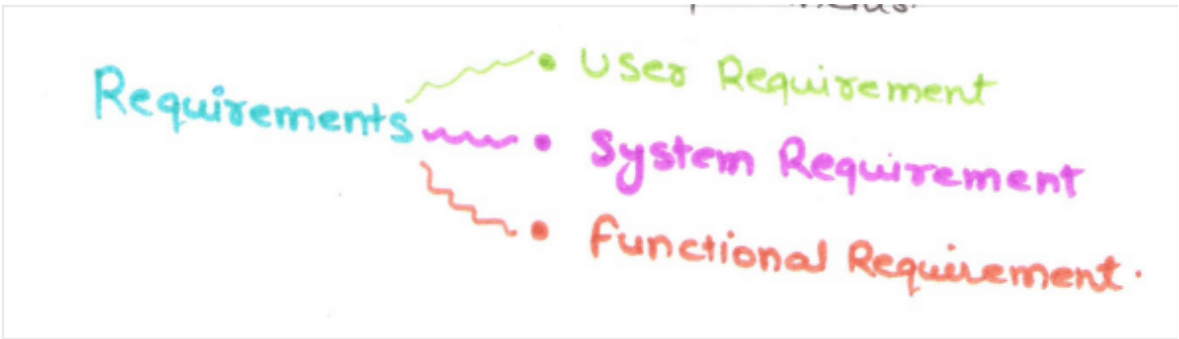
or

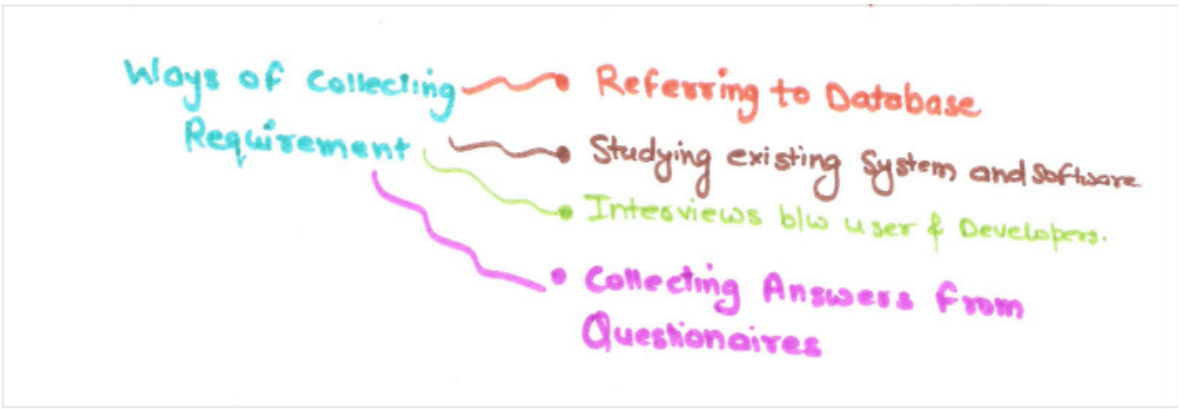
It is a well defined, structured sequence of stages in software engineering to develop the intended software product.



Communication: Very first step where user contact the service provider i.e, software organisation and initiate the request for a desired software product in writing and tries to negotiate the terms.

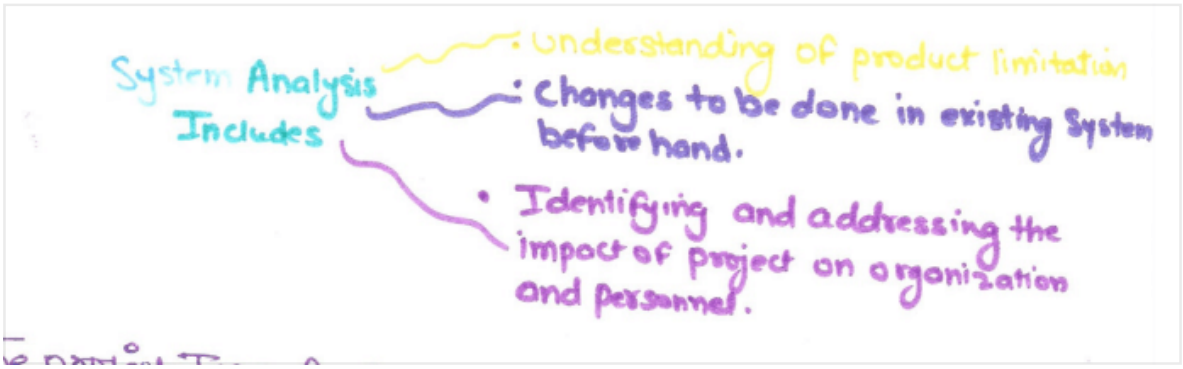
Requirement Gathering: A team of software developer holds discussion with various stakeholders from problem domain and bring out as much as information as possible on the requirements.





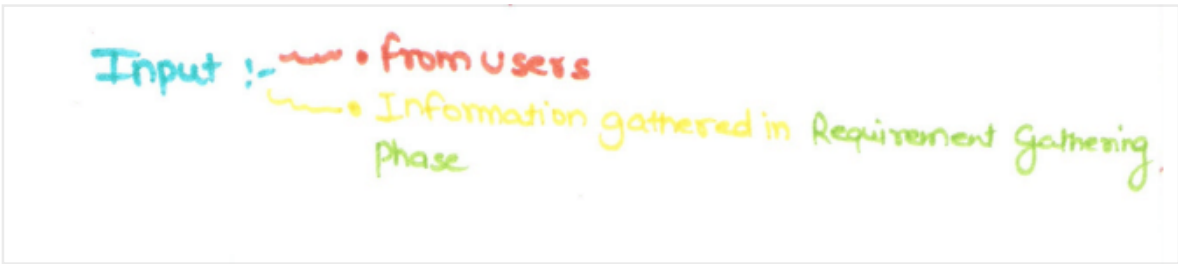
Feasibility Study: After requirement gathering , with the help of many algorithms, team analyses if a software can be designed to fulfill all requirements of the user and also analyze if the product is financially, practically and technologically feasible for the organization to take up.

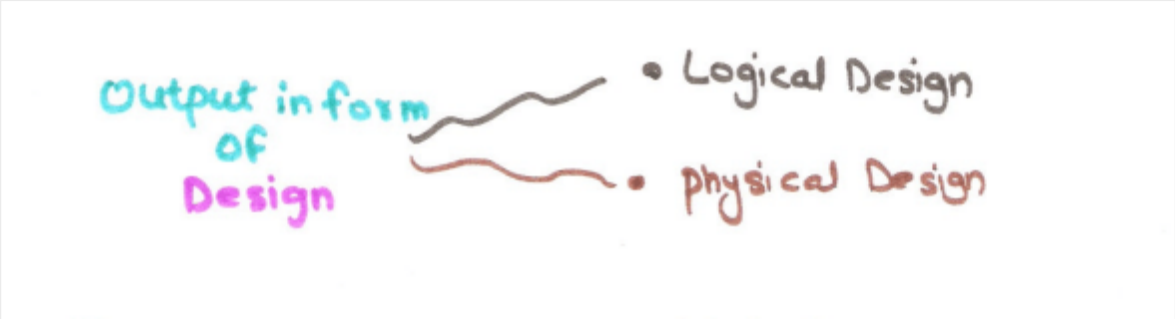
System Analysis - A Planning Phase: Developer decide a road map of their plan and try to bring up the best software model suitable for project.



The project team analyzes the scope of the project and then plans the schedule and the resources accordingly.

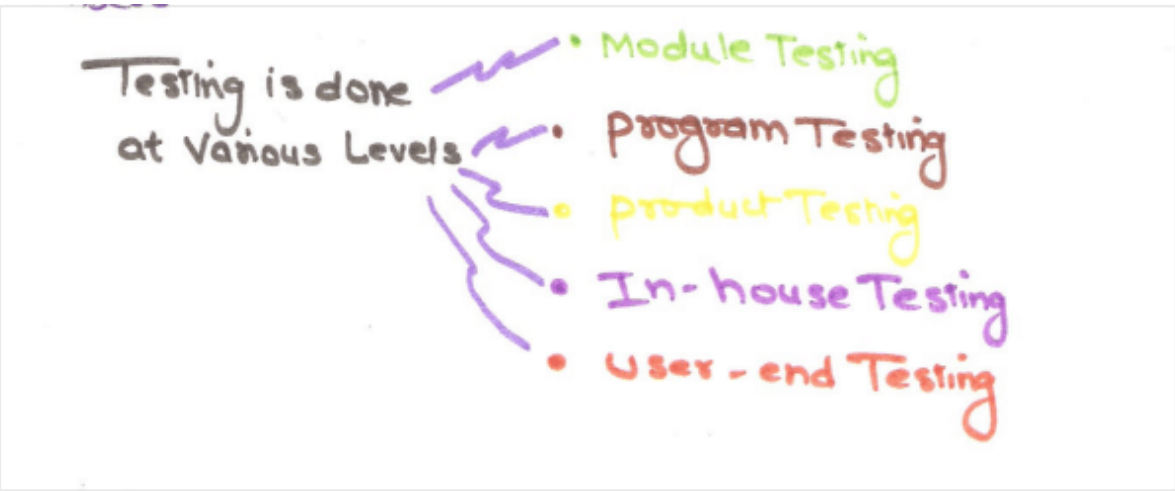
Software Design:- All knowledge of requirement and analysis are taken together to plan-up design the software product.





Coding: Programming Phase- This step is also known as programming phase. The implementation of software design starts in terms of writing program code in the suitable programming language and developing error-free executable programming efficiently.

Testing: Software testing is done while coding, by the tester that is developing team members.



Integration: Software is integrated with libraries, databases and other programs.

Implementation: We install software on user machine. Software is tested for portability, adaptability integration.

Operation And Maintenance: This phase confirms the software operations In terms of more efficiency and less errors. If required, the user are trained on, are aided with the documentation on how

to operate the software and how to keep the software operational.
The software is maintained timely by updating the code according to the changes taking place in user and environment or Technology.
This phase may take challenges from hidden bugs and real word undefined problems.

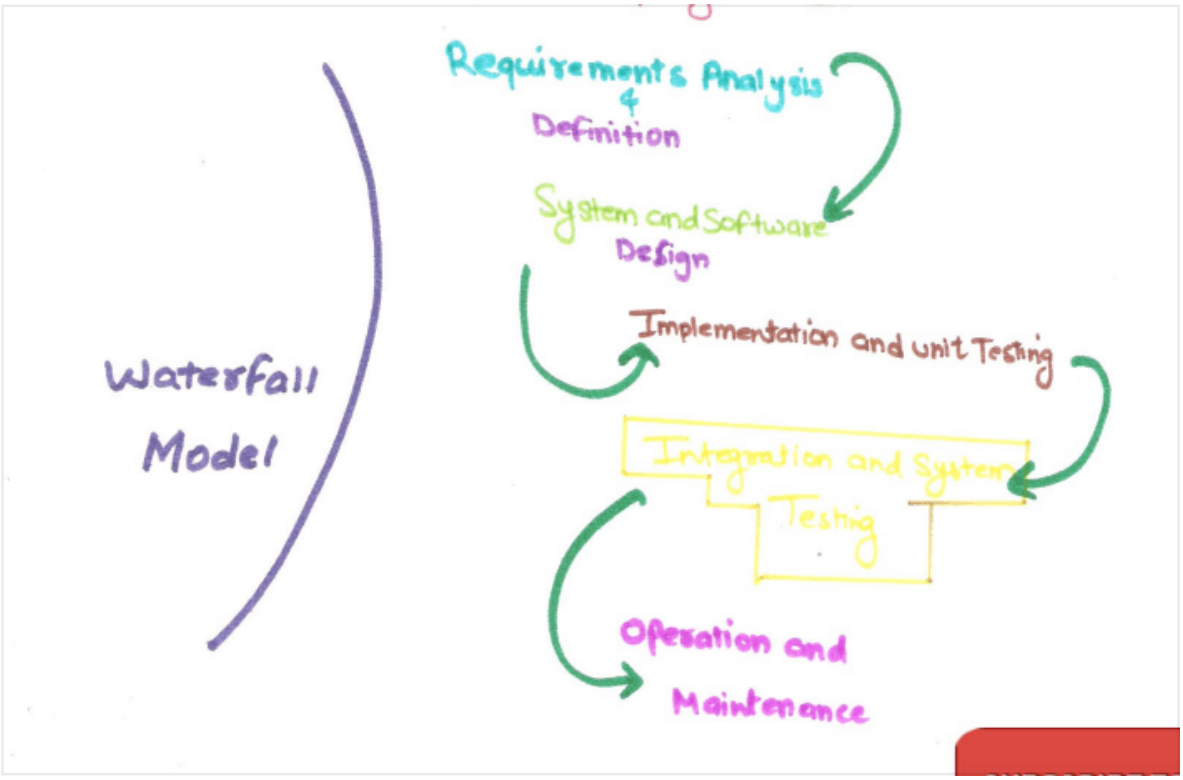
The Software Process Model

A software process model is a specified definition of a software process, which is presented from a particular perspective. Models, by their nature, are a simplification, so a software process model is an abstraction of the actual process, which is being described. Process models may contain activities, which are part of the software process, software product, and the roles of people involved in software engineering. Some examples of the types of software process models that may be produced are:

1. **A workflow model:** This shows the series of activities in the process along with their inputs, outputs and dependencies. The activities in this model perform human actions.
2. **2. A dataflow or activity model:** This represents the process as a set of activities, each of which carries out some data transformations. It shows how the input to the process, such as a specification is converted to an output such as a design. The activities here may be at a lower level than activities in a workflow model. They may perform transformations carried out by people or by computers.
3. **3. A role/action model:** This means the roles of the people involved in the software process and the activities for which they are responsible.

Waterfall Model OR Classical Waterfall Model:

• Waterfall model is a simplest model of software development paradigm.
• All the phases of sdlc will function one after in linear manner. That is, when the first phase is finished then only the second phase will start and so on.
The Waterfall model was first process model to be introduce and also called **"Linear- Sequential Life Cycle Model.**



This type of model is basically used for the project which is small and there are **"No Uncertain Requirements".**

At the end of a phase, a review take place to determine if the project is on the right path and whether or not to continue or discard the project.

In this testing starts only after the development is completed and phases do not overlap.

Advantages of WaterFall Model:

1) Discipline: This staged development cycle enforces discipline. Every phase has a start and end point, and progress can be conclusively identified(through the use of milestones) by both vendor and client.

2) minimal wastage:- the emphasis on requirement and design before writing a single line of code ensure **minimal wastage of time and efforts and reduces the risk of schedule slippage.**

3) improves quality:- It is much easier to catch and correct possible flows at the design stage then at the testing stage after all the component have been **integrated and tracking down specific error is more complex.**

Disadvantages of WaterFall Model:

Uncertain Nature of Customer Needs:- due to this estimating time and costs with any degree of accuracy(as the model subjects) is often extremely difficult.

Implicit Assumption to Roadblock: Implicit assumption that designs can be feasibly translated into real products, they

sometimes runs into roadblock when developer begin implementation.

No Good Model: Not a good model for Complex and object oriented project as world changing fast as technology changes.

- **It haas High amount of risk and uncertainty.**
- **It is somewhere poor model for long and ongoing projects.**
- **It is Not suitable for the projects where requirements are at a moderate to higher risk of changing.**

Software Process Model: Spiral Model

Spiral Model was originally proposed by **"Boehm".** Rather than represent the software process as a **sequence of activities** with some **backtracking** from one activity to the another, the process is represented as a **SPIRAL.**

Each loop in spiral model represents a phase of the Software Process.

In this model, developers define and implement features in order of Decreasing Priority.

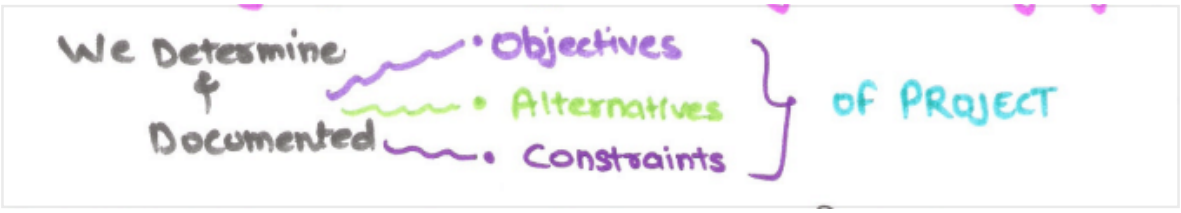
Exact numbers of loops in the spiral is **"Not fixed".**

This model considers **RISK,** which often goes **"UN-Noticed"** by most other models.

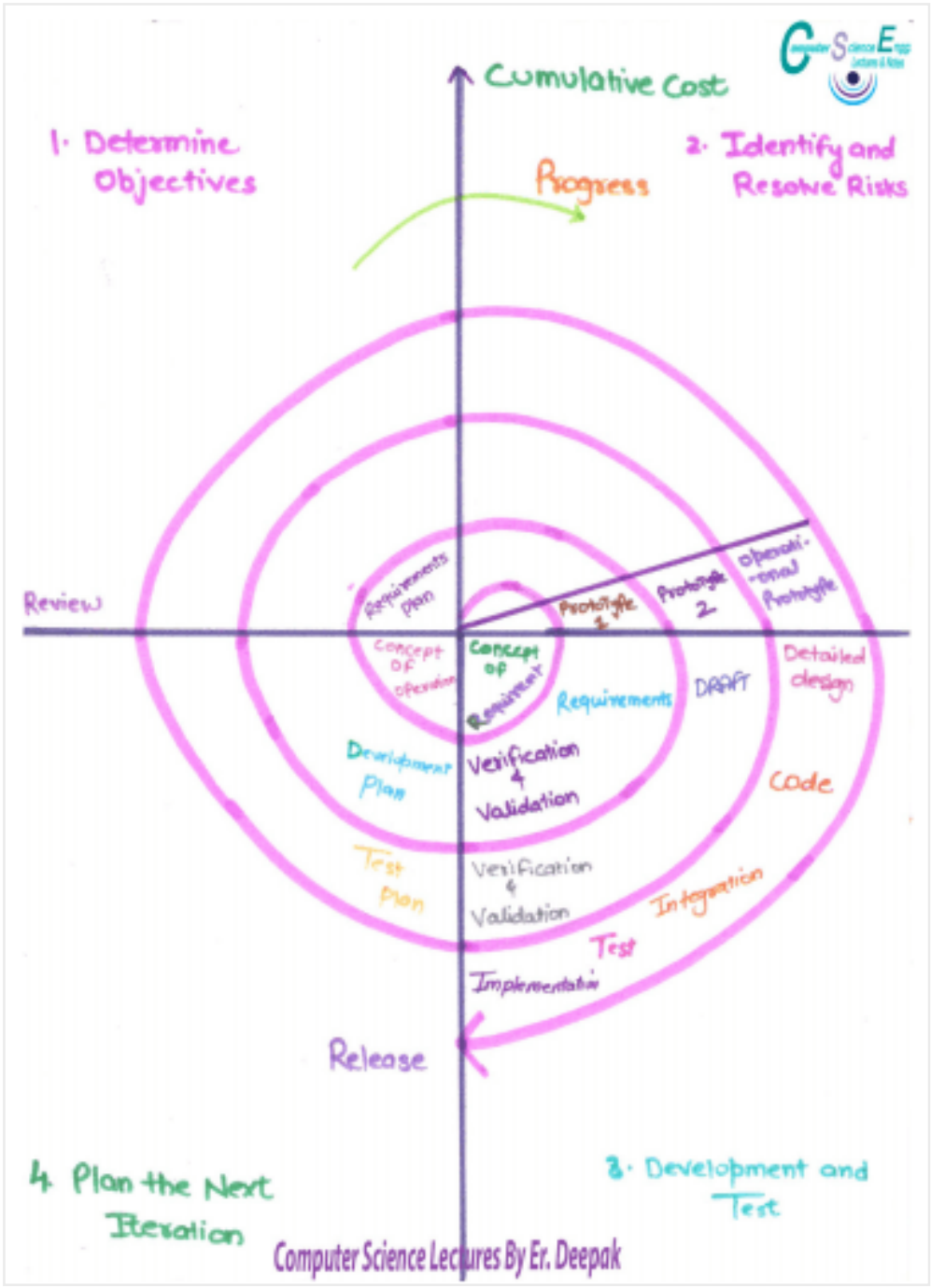
Each loop of the spiral represent a phase of the software process. And it has four phases.

The Four Phases of Spiral Model

1) Determining Object or **Objective Setting** or **Planning Object**



A detailed management plan is drawn up and "Project Risk" are identified.

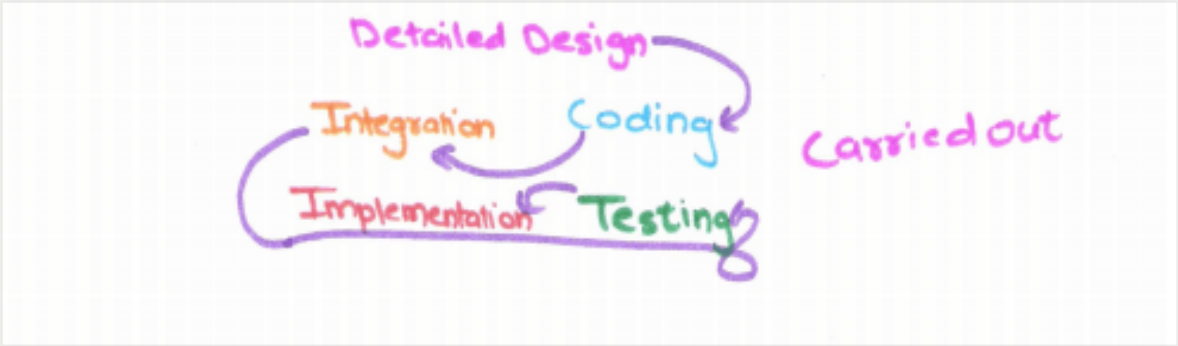


2) Identify And Resolve Risks or **Risk Assessment And Reduction** or **Risk Analysis:**

This phase has been added specially in order to identify and resolve all possible risks in project development. If RISKS indicate any kind of UNCERTAINLY in requirement, Prototyping may be used to proceed with the available data and find out solution in order to deal with the Potential changes in the Requirements.

3) Development and Test or **Engineering** or **Development And Validation:**

After resolving the identified risk, the actual development of the project is carried out in this..



4) Plan Next Iteration or **Planning** or **Reviewing And Planning:** The prototype is reviewed and decision made whether to continue with a further loop of the spiral. It is decided to continue, plans are **drawn up for the next phase of the project.**

Advantages Of Spiral Model:

Later Stage Changes: Addition functionality or changes can be done at a later stages as proper planning is done at each Iteration.

Easy Cost Estimation: As **Prototype** building is done in **small fragments** by which cost estimation becomes easy.

Good Risk Management: As **Risk Analysis Stage** comes in very early stages and **continuous or repeated development** helps in **Risk Management.**

Systematic Development: Delopment is fast and features can be added in a systematic way.

Space For Customer Feedback: There is continuous feedback is taken at each **ITERATION.**

Large Project Oriented: It is very good model for handling large and **Mission Critical Projects.**

Better Project Monitoring: Project Monitoring is very "Easy And Effective". Each Phase as well as "Each Loop, requires a Review from concerned people. This makes people model more Transparent.

Disdvantages Of Spiral Model:

High Cost: Cost involved in this model is quite **"HIGH"** as more planning, prototype version, risk management has to be done.

Need well expertise: Skills required to evaluate and review project from time to time, need expertise.

No Reusablity of prototype: Due to various customization from client, using same prototype for the other project in future is

difficult.

Not suitable for small project: It is quite difficult to follow this strategy for **Small Projects** due to

Low Budget
Time Constraint
Large Procedures.

What is the iterative process?

The iterative process is the practice of building, refining, and improving a project, product, or initiative. Teams that use the iterative development process create, test, and revise until they're satisfied with the end result. You can think of an iterative process as a trial-and-error methodology that brings your project closer to its end goal.

Iterative processes are a fundamental part of lean methodologies and Agile project management—but these processes can be implemented by any team, not just Agile ones. During the iterative process, you will continually improve your design, product, or project until you and your team are satisfied with the final project deliverable.

Example iterative processes

Engineering

Many engineering teams use the iterative process to develop new features, implement bug fixes, or A/B test new strategies. Often, an engineering team will create a few iterations that they think are equally promising, then test them with users. They'll note pain points and successes, and then continue building out the one that tested the best.

Product development

You might be surprised to realize that most product development is very iterative. Think of any personal technology you've ever purchased for yourself—there was likely a previous version before the one you bought, and maybe a version afterwards, as well. Think of the development of mobile phones throughout the years, how speakers have gotten smaller and more portable over time, or even the way refrigerators from the same brands have changed to adapt to new family needs. All of these are iterative processes.

Marketing

Some marketing teams embrace iterative processes, others not so

much. But to a certain extent, a lot of marketing is iterative. For example, some marketing teams might test different advertising copy to see which one gets better engagement, or send out two versions of an email newsletter to compare click-through rates. Alternatively, a brand marketing team could use iterative design processes to identify the imagery that works best for their target audience.

Sales

Though most of a sales team's customer-facing work isn't iterative, some of their tasks can benefit from iterative processes. For example, a sales team might take an iterative approach to sending cold emails. They might have their reps send a few different email subject lines and analyze the results. Then, the team can implement the most successful subject lines moving forward.

The 5 steps of the iterative process

The iterative process can help you during the lifecycle of a project. During the steps of the iterative process, your goals and requirements will serve as the project's starting point. Then, your team will use testing, prototyping, and iteration to achieve the best possible result. Here's how:

1. Planning and requirements

During this step in the iterative process, you will define your project plan and align on your overall project objectives. This is the stage where you will outline any hard requirements—things that must happen in order for your project to succeed. Without this step, you run the risk of iterating but not hitting your goals.

2. Analysis and design

During this step, you and your team will focus on the business needs and technical requirements of your project. If step one was the process of outlining your goals, step two is when you brainstorm a design that will help you ultimately hit those goals.

3. Implementation

During the third step, your team will create the first iteration of your project deliverable. This iteration will be informed by your analysis and design, and should work to hit your ultimate project objective. The level of detail and time you spend on this iteration will depend on the project.

4. Testing

Now that you have an iteration, you will test it in whatever way makes the most sense. If you're working on an improvement to a web page,

for example, you might want to A/B test it against your current web page. If you're creating a new product or feature, consider doing usability testing with a set of potential customers.
In addition to testing, you should also check in with your project stakeholders. Ask them to weigh in on the iteration, and provide any feedback.
Read: What is the Plan-Do-Check-Act (PDCA) cycle?

5. Evaluation and review

After testing, your team will evaluate the success of the iteration and align on anything that needs to change. Does this iteration achieve your project objectives? Why, or why not? If something needs to change, you can restart the iterative process by going back to step two to create the next iteration. Keep in mind that your initial planning and goals should remain the same for all iterations. Continue building upon the previous iteration until you get to a deliverable you're happy with.
If you restart the iterative process, make sure everyone is still aligned on your project goals. The iterative process can take weeks or months, depending on how many iterations you run through. Centering your iteration on your project objectives every time you restart the iterative process can help you ensure you don't lose track of your north star.

Iterative Model

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 software 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) process.

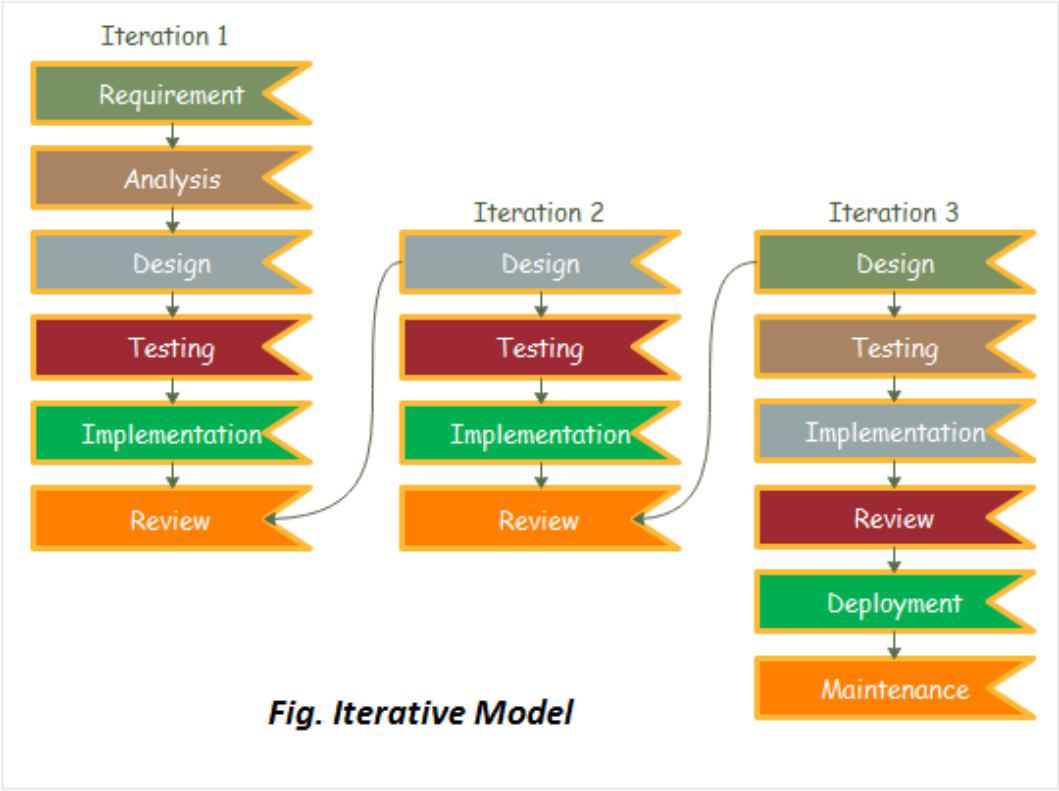


Fig. Iterative Model

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 budget or not. After all of this, the software team skips to the next phase.

2. Design: In the design phase, team design the software 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 Software.

4. Testing: After completing the coding phase, software testing starts using different test methods. There are many test methods, 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 deployment, 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 software in the working environment there may be some bugs, some errors or new updates are required. Maintenance involves debugging and new addition options.

When to use the Iterative Model?

1. When requirements are defined clearly and easy to understand.
2. When the software application is large.
3. When there is a requirement of changes in future.

Advantage(Pros) of Iterative Model:

1. Testing and debugging during smaller iteration is easy.
2. A Parallel development can plan.
3. It is easily acceptable to ever-changing needs of the project.
4. Risks are identified and resolved during iteration.
5. Limited time spent on documentation and extra time on designing.

Disadvantage(Cons) of Iterative Model:

1. It is not suitable for smaller projects.
2. More Resources may be required.
3. Design can be changed again and again because of imperfect requirements.
4. Requirement changes can cause over budget.
5. Project completion date not confirmed because of changing requirements.

Software Processes Activities

Software is the set of instructions in the form of programs to govern the computer system and to process the hardware components. To produce a software product the set of activities is used. This set is called a software process.

or

The process that deals with the technical and management issues of the software development is called software process.

or

Software process is the set of activities and associated

results that produce a software product.

Components of Software:
There are three components of the software:

1. **Program:**

A computer program is a list of instructions that tell a computer what to do.

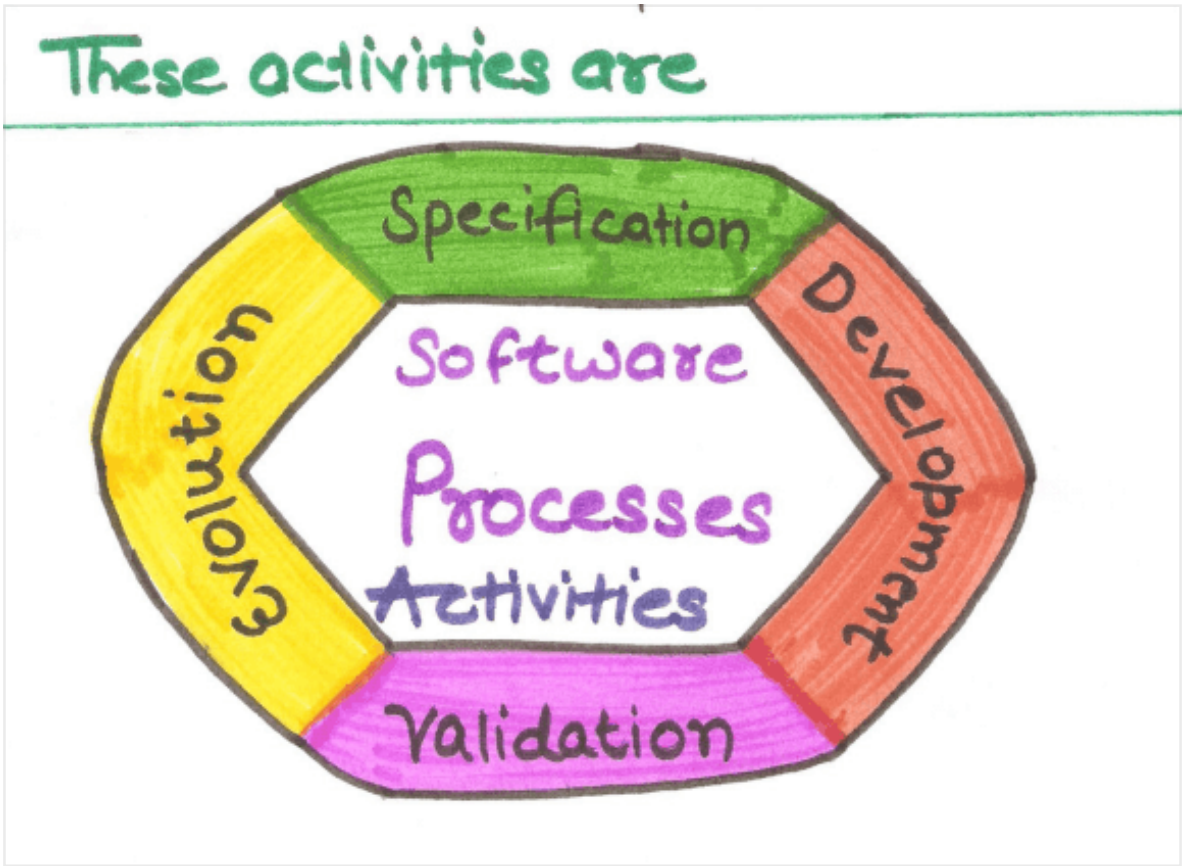
2. **Documentation:**

Source information about the product contained in design documents, detailed code comments, etc.

3. **Operating Procedures:**

Set of step-by-step instructions compiled by an organization to help workers carry out complex routine operations.

There are four basic key process activities:



1. **Software Specifications:**

In this process, detailed description of a software system to be developed with its functional and non-functional requirements.

2. **Software Development:**

In this process, designing, programming, documenting, testing, and bug fixing is done.

3. **Software Validation:**

In this process, evaluation software product is done to ensure that the software meets the business requirements as well as the end users needs.

4. **Software Evolution:**

It is a process of developing software initially, then timely updating it for various reasons.

CASE TOOLS

Computer aided software engineering (CASE) is the implementation of computer facilitated tools and methods in software development. CASE is used to ensure a high-quality and defect-free software. CASE ensures a check-pointed and disciplined approach and helps designers, developers, testers, managers and others to see the project milestones during development.

CASE can also help as a warehouse for documents related to projects, like business plans, requirements and design specifications. One of the major advantages of using CASE is the delivery of the final product, which is more likely to meet real-world requirements as it ensures that customers remain part of the process.

CASE Tools:

The essential idea of CASE tools is that in-built programs can help to analyze developing systems in order to enhance quality and provide better outcomes.

Throughout the 1990, CASE tool became part of the software lexicon, and big companies like IBM were using these kinds of tools to help create software.

Various tools are incorporated in CASE and are called CASE tools, which are used to support different stages and milestones in a software development life cycle.

CASE tools are set of software application programs, which are used to automate SDLC activities. CASE tools are used by

software project managers, analysts and engineers to develop software system.

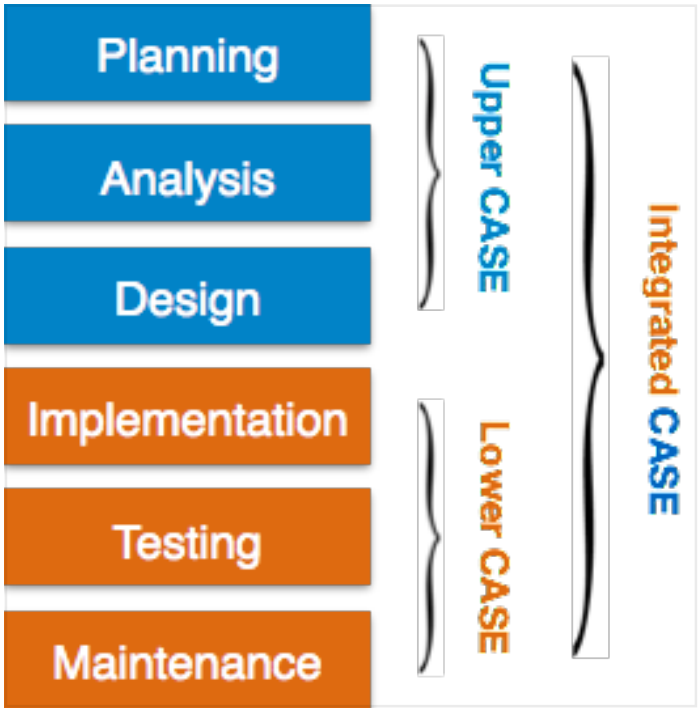
There are number of CASE tools available to simplify various stages of Software Development Life Cycle such as Analysis tools, Design tools, Project management tools, Database Management tools, Documentation tools are to name a few.

Use of CASE tools accelerates the development of project to produce desired result and helps to uncover flaws before moving ahead with next stage in software development.

Components of CASE Tools

CASE tools can be broadly divided into the following parts based on their use at a particular SDLC stage:

• **Central Repository** - CASE tools require a central repository, which can serve as a source of common, integrated and consistent information. Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as data dictionary.



• **Upper Case Tools** - Upper CASE tools are used in planning, analysis and design stages of SDLC.

• **Lower Case Tools** - Lower CASE tools are used in implementation, testing and maintenance.

• **Integrated Case Tools** - Integrated CASE tools are helpful in all the stages of SDLC, from Requirement gathering to Testing and documentation.

CASE tools can be grouped together if they have similar

functionality, process activities and capability of getting integrated with other tools.

Scope of Case Tools

The scope of CASE tools goes throughout the SDLC.

Diagram tools

These tools are used to represent system components, data and control flow among various software components and system structure in a graphical form. For example, Flow Chart Maker tool for creating state-of-the-art flowcharts.

Process Modeling Tools

Process modeling is method to create software process model, which is used to develop the software. Process modeling tools help the managers to choose a process model or modify it as per the requirement of software product. For example, EPF Composer

Project Management Tools

These tools are used for project planning, cost and effort estimation, project scheduling and resource planning. Managers have to strictly comply project execution with every mentioned step in software project management. Project management tools help in storing and sharing project information in real-time throughout the organization. For example, Creative Pro Office, Trac Project, Basecamp.

Documentation Tools

Documentation in a software project starts prior to the software process, goes throughout all phases of SDLC and after the completion of the project.

Documentation tools generate documents for technical users and end users. Technical users are mostly in-house professionals of the development team who refer to system manual, reference manual, training manual, installation manuals etc. The end user documents describe the functioning and how-to of the system such as user manual. For example, Doxygen, DrExplain, Adobe RoboHelp for documentation.

Analysis Tools

These tools help to gather requirements, automatically check for any inconsistency, inaccuracy in the diagrams, data redundancies or erroneous omissions. For example, Accept 360, Accompa, CaseComplete for requirement analysis, Visible Analyst for total analysis.

Design Tools

These tools help software designers to design the block structure

of the software, which may further be broken down in smaller modules using refinement techniques. These tools provides detailing of each module and interconnections among modules. For example, Animated Software Design

Configuration Management Tools

An instance of software is released under one version. Configuration Management tools deal with –

- Version and revision management
- Baseline configuration management
- Change control management

CASE tools help in this by automatic tracking, version management and release management. For example, Fossil, Git, Accu REV.

Change Control Tools

These tools are considered as a part of configuration management tools. They deal with changes made to the software after its baseline is fixed or when the software is first released. CASE tools automate change tracking, file management, code management and more. It also helps in enforcing change policy of the organization.

Programming Tools

These tools consist of programming environments like IDE (Integrated Development Environment), in-built modules library and simulation tools. These tools provide comprehensive aid in building software product and include features for simulation and testing. For example, Cscope to search code in C, Eclipse.

Prototyping Tools

Software prototype is simulated version of the intended software product. Prototype provides initial look and feel of the product and simulates few aspect of actual product.

Prototyping CASE tools essentially come with graphical libraries. They can create hardware independent user interfaces and design. These tools help us to build rapid prototypes based on existing information. In addition, they provide simulation of software prototype. For example, Serena prototype composer, Mockup Builder.

Web Development Tools

These tools assist in designing web pages with all allied elements like forms, text, script, graphic and so on. Web tools also provide live preview of what is being developed and how will it look after completion. For example, Fontello, Adobe Edge Inspect,

Foundation 3, Brackets.

Quality Assurance Tools

Quality assurance in a software organization is monitoring the engineering process and methods adopted to develop the software product in order to ensure conformance of quality as per organization standards. QA tools consist of configuration and change control tools and software testing tools. For example, SoapTest, AppsWatch, JMeter.

Maintenance Tools

Software maintenance includes modifications in the software product after it is delivered. Automatic logging and error reporting techniques, automatic error ticket generation and root cause Analysis are few CASE tools, which help software organization in maintenance phase of SDLC. For example, Bugzilla for defect tracking, HP Quality Center.

They are either open source or are paid tools. Some of them are listed below:

1. **Analyst4j tool** is based on the Eclipse platform and available as a stand-alone Rich Client Application or as an Eclipse IDE plug-in. It features search, metrics, analyzing quality, and report generation for Java programs.
2. **CCCC is an open source command-line tool**. It analyzes C++ and Java lines and generates reports on various metrics, including Lines of Code and metrics proposed by Chidamber & Kemerer and Henry & Kafura.
3. **Chidamber & Kemerer Java Metrics** is an open source command-line tool. It calculates the C&K object-oriented metrics by processing the byte-code of compiled Java.
4. **Dependency Finder** is an open source. It is a suite of tools for analyzing compiled Java code. Its core is a dependency analysis application that extracts dependency graphs and mines them for useful information. This application comes as a command-line tool, a Swing-based application, and a web application.
5. **Eclipse Metrics Plug-in 1.3.6** by Frank Sauer is an open source metrics calculation and dependency analyzer plugin for the Eclipse IDE. It measures various metrics and detects cycles in package and type dependencies.
6. **Eclipse Metrics Plug-in 3.4** by Lance Walton is open source. It calculates various metrics during build cycles and warns, via the problems view, of metrics 'range violations'.
7. **OOMeter** is an experimental software metrics tool developed by Alghamdi. It accepts Java/C# source code and UML models in XMI

and calculates various metrics.
8. **Semmle** is an Eclipse plug-in. It provides an SQL like querying language for object-oriented code, which allows searching for bugs, measure code metrics, etc.

Advantages of the CASE approach:

• As special emphasis is placed on redesign as well as testing, the servicing cost of a product over its expected lifetime is considerably reduced.

• The overall quality of the product is improved as an organized approach is undertaken during the process of development.

• Chances to meet real-world requirements are more likely and easier with a computer-aided software engineering approach.

• CASE indirectly provides an organization with a competitive advantage by helping ensure the development of high-quality products.

Disadvantages of the CASE approach:

• Cost: Using case tool is a very costly. Mostly firms engaged in software development on a small scale do not invest in CASE tools because they think that the benefit of CASE are justifiable only in the development of large systems.
• Learning Curve: In most cases, programmers productivity may fall in the initial phase of implementation , because user need time to learn the technology. Many consultants offer training and on-site services that can be important to accelerate the learning curve and to the development and use of the CASE tools.
• Tool Mix: It is important to build an appropriate selection tool mix to urge cost advantage CASE integration and data integration across all platforms is extremely important.

Software Project Management

Project Management Concept:

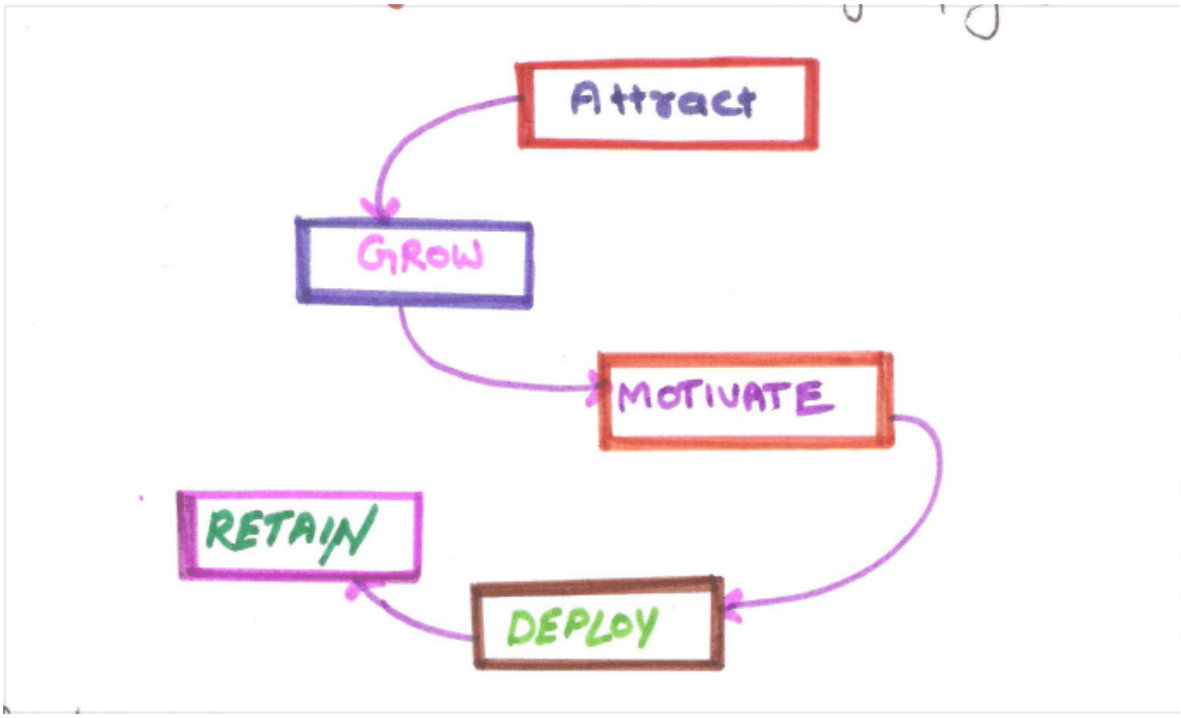
It is an art and science of planning and leading software projects.

Project Management-> software project- planned, implemented, Monitored, controlled.

Effective software project management focuses on the four P's:

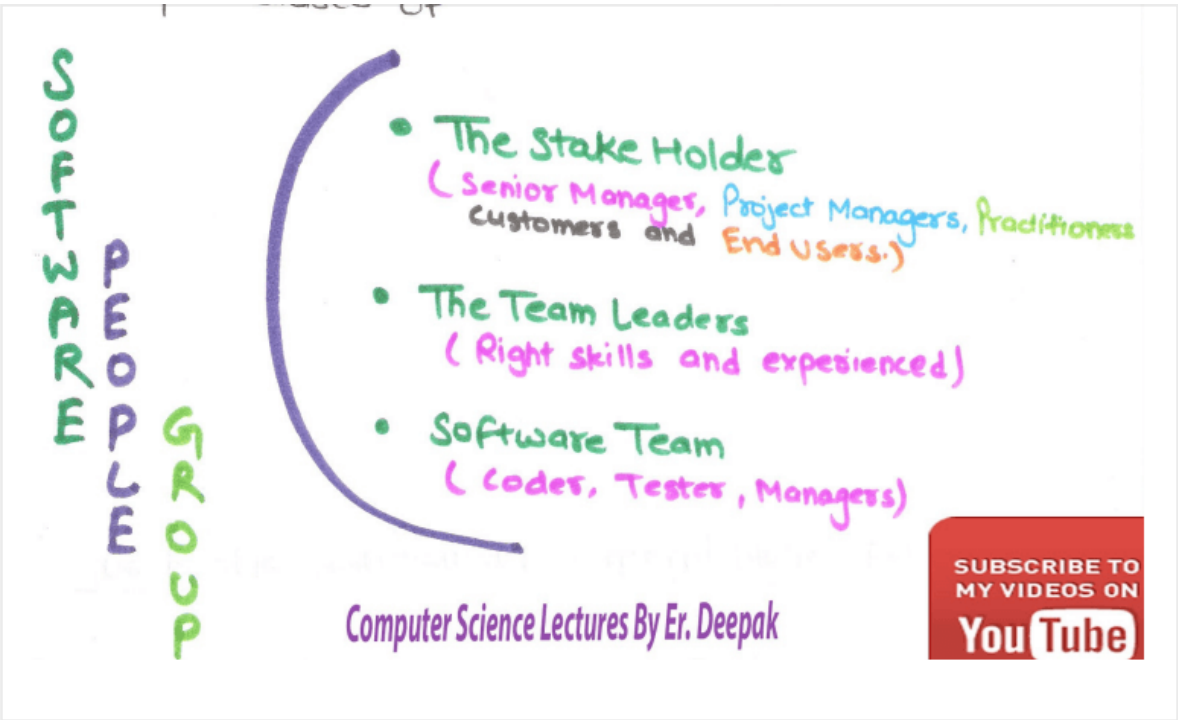
The People: It deals with the cultivation of motivated, highly skilled people.

Software engineering institute has developed a people management capability maturity model (PM-CMM) to enhance the readiness of the



software organizations to undertake increasingly Complex Applications by helping to the talented needed to improve their software development capability.
The People management maturity model focuses on

People consists of



The People: The Stakeholder:

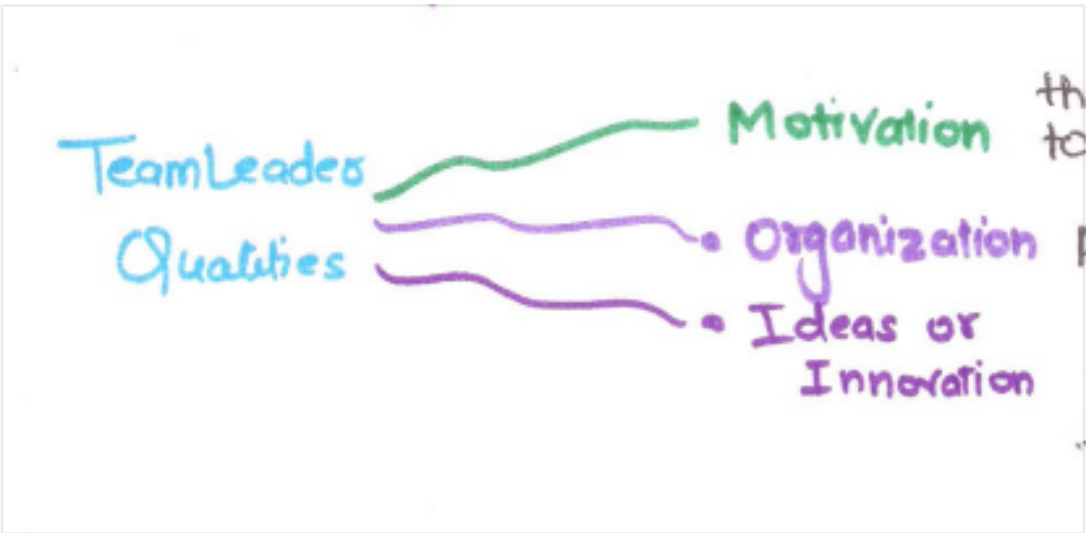
There are mainly 5 categories of stakeholder:

- **Senior Manager:** Define business issues that often have significant influence on the project.
- **Project (Technical) Managers:** They **"Plan, Motivate, Organised, and Control"** the practitioners who do the work.
- **Practitioner:** They deliver the technical skills that are necessary to engineer a product or application.
- **Customers:** Specify the requirement for the software to be engineered and other stakeholders who have a peripheral interest in the outcome.
- **End users:** Interact with the software once it is released for production use.

The people: Team leaders

Team Leader Qualities:

- **Motivational-** the ability to encourage technical people to produce to their best ability.
- **Organisation-** the ability to mold existing processes.



• **Ideas or innovations-** ability to encourage to go beyond their skills.

Another set of useful leadership Traits:

•- **problem solving:** Diagnose, structure a solution, apply lessons learned, remain flexible.

•- **Managerial Identity:** Take charge of the project, have confidence to assume control, have a assurance to allow good people to do their jobs.

•-**Achievement :** Reward initiative, demonstrate that control risk taking will not be punished.

•-**Influence and team building:** Be able to ' read' people, understand verbal and nonverbal signals. Able to react to the signals, remain under control in high- stress situations.

The people: The software team

Seven project factors to be considered when selecting a software development team.

Four Organizational paradigm for Software Development Teams :

• **1. Closed paradigm:** Traditional hierarchy of authority works well when producing software similar to past efforts, members are less likely to be innovative .

•**2. Random paradigm:** Depends on individual initiative of team members , work well for project requiring innovation technological break.

•**3. Open paradigm:** hybrid of the closed and random paradigm, works well for solving Complex problems, requiring collaboration, communication, and censensus among members.

4. synchronous paradigm: Organizes team members based on the network pieces of the problem, members have little communication outside of their subgroups.

Project Management Concept

The Product:

The scope of the software development must be established and bounded:

• **Context:** How does the software to the built fit into a larger system, product, or business context and what constraint are imposed as a result of the context?

• **Information Objective** - what customer visible data objects are produced as output from the software? What data object are required for input?

• **Function and performance:** What function does the software

performance to transform input data into output? Are there any special performance characteristics to be addressed.

Software Project scope must be unambiguous and understand at both the "Managerial and Technical levels".

The Process:

The project manager must decide process model is most appropriate based on

The customer who have requested the product and the people who will do the work.

The characteristics of the product itself

The project environment in which the software team works <

The Project :

Planning and controlling a software project is done for one the primary reason. It is the only known way to manage complexity.

W5HH principle

A series of questions that like to add definition of the key project characteristic and resultant project plan.

•- why is the system being developed?

Assesses the validity of business reason and justifications.

•- what will be done?

Establishing the task set required for the project.

• -When will be done -

Establishes a project schedule.

• -Where are they organizationally located?

->Notes the organisational location of the team members, customers and other stakeholder.

•-Who is responsible for functional-

defines the role and responsibility of each member.

•- How will the job is done technically and managerially?

-> establishes the management and Technical strategy for the project.

•- how: much of each resources is needed?

-> Establishes estimates based on the answer to the previous questions.

What is software project management?

Software project management is an art and discipline of planning and supervising software projects. It is a sub-discipline of software project management in which software projects planned, implemented, monitored and controlled.

It is a procedure of managing, allocating and timing resources to develop computer software that fulfills requirements.

17.4M
396
How to find Nth Highest Salary in SQL

In software Project Management, the client and the developers need to know the length, period and cost of the project.

Prerequisite of software project management?

There are three needs for software project management. These are:

1. Time
2. Cost
3. Quality

It is an essential part of the software organization to deliver a quality product, keeping the cost within the client?s budget and deliver the project as per schedule. There are various factors, both external and internal, which may impact this triple factor. Any of three-factor can severely affect the other two.

Project Manager

A project manager is a character who has the overall responsibility for the planning, design, execution, monitoring, controlling and closure of a project. A project manager represents an essential role in the achievement of the projects.

A project manager is a character who is responsible for giving decisions, both large and small projects. The project manager is used to manage the risk and minimize uncertainty. Every decision the project manager makes must directly profit their project.

Role of a Project Manager:

1. Leader

A project manager must lead his team and should provide them direction to make them understand what is expected from all of them.

2. Medium:

The Project manager is a medium between his clients and his team. He must coordinate and transfer all the appropriate information from the clients to his team and report to the senior management.

3. Mentor:

He should be there to guide his team at each step and make sure that the team has an attachment. He provides a recommendation to his team and points them in the right direction.

Responsibilities of a Project Manager:

1. Managing risks and issues.
2. Create the project team and assigns tasks to several team members.
3. Activity planning and sequencing.
4. Monitoring and reporting progress.
5. Modifies the project plan to deal with the situation.

Activities

Software Project Management consists of many activities, that includes planning of the project, deciding the scope of product, estimation of cost in different terms, scheduling of tasks, etc.

The list of activities are as follows:

1. Project planning and Tracking
2. Project Resource Management
3. Scope Management
4. Estimation Management
5. Project Risk Management
6. Scheduling Management
7. Project Communication Management
8. Configuration Management

Now we will discuss all these activities -

1. Project Planning: It is a set of multiple processes, or we can say that it a task that performed before the construction of the product starts.

Project Planning

Project planning is one of the most important activities and is an ongoing effort throughout the life of the project.

Project planning complete the essential activities:

•- Estimating the following attributes of the project

• **Project size:** What will be problem complexity in terms of the effort and time required to develop the product.

•**Cost:** How much is it going to cost to develop the project?

•**Duration:** How long is it going to take to complete the software project development.

•**Effort:** How much effort would be required?

•-Scheduling manpower and other resources.

•- Staff Organization and Staffing Plans.

•- Risk Identification, Analysis, and Abatement Planning.

•- Miscellaneous plans such as quality assurance plan,

configuration management plan,etc.

Who does Project Management:

Software Project Managers- using information selected from customers and software engineer and software metrics data collected from the past projects.

How to do that?

A simple table delineating the task to be performed, the functions to be implemented and the cost, efforts and time involved for each is generated. A list of required for that resources is also produced .

Project Planning key tasks: Set goal and scope, select life cycle,set organisational team form, start team selection ,determine risks, create wbs, identify tasks, estimate size, estimate effort,identify task

dependencies, assign resources, schedule work.17.7M

2. Scope Management: It describes the scope of the project. Scope management is important because it clearly defines what would do and what would not. Scope Management create the project to contain restricted and quantitative tasks, which may merely be documented and successively avoids price and time overrun.

3. Estimation management: This is not only about cost estimation because whenever we start to develop software, but we also figure out their size(line of code), efforts, time as well as cost.

If we talk about the size, then Line of code depends upon user or software requirement.

If we talk about effort, we should know about the size of the software, because based on the size we can quickly estimate how big team required to produce the software.

If we talk about time, when size and efforts are estimated, the time required to develop the software can easily determine.

And if we talk about cost, it includes all the elements such as:

o Size of software
o Quality
o Hardware
o Communication
o Training
o Additional Software and tools
o Skilled manpower

4. Scheduling Management: Scheduling Management in software refers to all the activities to complete in the specified order and within time slotted to each activity. Project managers define multiple tasks and arrange them keeping various factors in mind.

For scheduling, it is compulsory -

o Find out multiple tasks and correlate them.
o Divide time into units.
o Assign the respective number of work-units for every job.
o Calculate the total time from start to finish.
o Break down the project into modules.

Project Scheduling

Project-task scheduling is a significant project planning activity. It comprises deciding which functions would be taken up when. To schedule the project plan, a software project manager wants to do

the following:
1. Identify all the functions required to complete the project.
2. Break down large functions into small activities.
3. Determine the dependency among various activities.
4. Establish the most likely size for the time duration required to complete the activities.
5. Allocate resources to activities.
6. Plan the beginning and ending dates for different activities.
7. Determine the critical path. A critical way is the group of activities that decide the duration of the project.

The first method in scheduling a software plan involves identifying all the functions required to complete the project. A good judgment of the intricacies of the project and the development process helps the supervisor to identify the critical role of the project effectively. Next, the large functions are broken down into a valid set of small activities which would be assigned to various engineers. The work breakdown structure formalism supports the manager to breakdown the function systematically after the project manager has broken down the purpose and constructs the work breakdown structure; he has to find the dependency among the activities. Dependency among the various activities determines the order in which the various events would be carried out. If an activity A necessary the results of another activity B, then activity A must be scheduled after activity B. In general, the function dependencies describe a partial ordering among functions, i.e., each service may precede a subset of other functions, but some functions might not have any precedence ordering describe between them (called concurrent function). The dependency among the activities is defined in the pattern of an activity network.

Once the activity network representation has been processed out, resources are allocated to every activity. Resource allocation is usually done using a Gantt chart. After resource allocation is completed, a PERT chart representation is developed. The PERT chart representation is useful for program monitoring and control. For task scheduling, the project plan needs to decompose the project functions into a set of activities. The time frame when every activity is to be performed is to be determined. The end of every action is called a milestone. The project manager tracks the function of a project by audit the timely completion of the milestones. If he examines that the milestones start getting delayed, then he has to handle the activities carefully so that the complete deadline can still be met.

5. Project Resource Management: In software Development, all the

elements are referred to as resources for the project. It can be a human resource, productive tools, and libraries.
Resource management includes:
o Create a project team and assign responsibilities to every team member
o Developing a resource plan is derived from the project plan.
o Adjustment of resources.
6. Project Risk Management: Risk management consists of all the activities like identification, analyzing and preparing the plan for predictable and unpredictable risk in the project.
Several points show the risks in the project:
o The Experienced team leaves the project, and the new team joins it.
o Changes in requirement.
o Change in technologies and the environment.
o Market competition.

Risk Management

Risk management is the identification, assessment and prioritization of risks followed by coordinated and economical application of resources to minimize, monitor and control the probability and or impact of unfortunate events or to maximize the realization of opportunities.

Principles Of Risk Management:
Risk Management should
- **create values**
- **be an integral part of organisational process**
- **be part of decision making**
- **be systematic and structured.**

Process or steps for Risk Management:
- **1) Establishing the context:**

It involves
1) identification of risk in a selected domain of interest.
2) Planning the remainder of process.
3) mapping out the following-
a) the social scope of risk management
b) the Identity and objective of stakeholders.
- **2) Identification:** After establishing the context, the next step in process of managing risk is to identify potential risks.

Objective based risk identification: Organisation and project team have objectives. **Scenario based identification:** Different scenarios are created. The scenarios maybe the alternative ways to achieve an objective.

Common-Risk checking : In several industries, Lest with know -risks are available.

• **3) Assessment:** Once Risk has been identified, then must then be assessed as to their potential severity of loss and to the probability of occurrence.

• **4)Potential risk Treatments:** Once Risks have been identified assessed, all techniques to manage the risk fall into one or more of these four major categories. **A) Avoidance:** This include not performing an activity that could carry risk .

B) Reduction: Reduction of Optimization involve reducing the severity of the loss of the likelihood of the loss from occurring.

C)sharing: It defined as sharing with another party the burden of loss or benefit of gain, from a risk and the measure to reduce risk .

D) Retention: Involve accepting the loss or benefit of Gain, from a risk when it occurs.

5) Create a risk management plan: Select appropriate control or counter measures to measure each risk. The risk management plan should propose applicable and effective security controls for managing the risk.

Implementation: It follows all the planned methods for mitigating the effect of the risks. Purchase insurance policies for the risks that have been decided to be transferred to an insurer, avoid all risk that can be avoided without sacrificing the entity's goal, reduce other and retain the first.

Risk Management Activities

Risk management consists of three main activities, as shown in fig:

Risk Assessment

The objective of risk assessment is to division the risks in the condition of their loss, causing potential. For risk assessment, first, every risk should be rated in two methods:

o The possibility of a risk coming true (denoted as r).

o The consequence of the issues relates to that risk (denoted as s).

Based on these two methods, the priority of each risk can be estimated:

$$p = r * s$$

Where p is the priority with which the risk must be controlled, r is the probability of the risk becoming true, and s is the severity of loss caused due to the risk becoming true. If all identified risks are set up, then the most likely and damaging risks can be controlled first, and more comprehensive risk abatement methods can be designed for these risks.

1. Risk Identification: The project organizer needs to anticipate the risk in the project as early as possible so that the impact of risk can be reduced by making effective risk management planning.

A project can be of use by a large variety of risk. To identify the significant risk, this might affect a project. It is necessary to categories into the different risk of classes.

There are different types of risks which can affect a software project:

1. **Technology risks:** Risks that assume from the software or

hardware technologies that are used to develop the system.
2. **People risks:** Risks that are connected with the person in the development team.
3. **Organizational risks:** Risks that assume from the organizational environment where the software is being developed.
4. **Tools risks:** Risks that assume from the software tools and other support software used to create the system.
5. **Requirement risks:** Risks that assume from the changes to the customer requirement and the process of managing the requirements change.
6. **Estimation risks:** Risks that assume from the management estimates of the resources required to build the system

2. Risk Analysis: During the risk analysis process, you have to consider every identified risk and make a perception of the probability and seriousness of that risk.

There is no simple way to do this. You have to rely on your perception and experience of previous projects and the problems that arise in them.

It is not possible to make an exact, the numerical estimate of the probability and seriousness of each risk. Instead, you should authorize the risk to one of several bands:

1. The probability of the risk might be determined as very low (0-10%), low (10-25%), moderate (25-50%), high (50-75%) or very high (+75%).
2. The effect of the risk might be determined as catastrophic (threaten the survival of the plan), serious (would cause significant delays), tolerable (delays are within allowed contingency), or insignificant.

Risk Control

It is the process of managing risks to achieve desired outcomes. After all, the identified risks of a plan are determined; the project must be made to include the most harmful and the most likely risks. Different risks need different containment methods. In fact, most risks need ingenuity on the part of the project manager in tackling the risk.

There are three main methods to plan for risk management:

1. **Avoid the risk:** This may take several ways such as discussing with the client to change the requirements to decrease the scope of the work, giving incentives to the engineers to avoid the risk of human resources turnover, etc.
2. **Transfer the risk:** This method involves getting the risky element developed by a third party, buying insurance cover, etc.

3. **Risk reduction:** This means planning method to include the loss due to risk. For instance, if there is a risk that some key personnel might leave, new recruitment can be planned.

Risk Leverage: To choose between the various methods of handling risk, the project plan must consider the amount of controlling the risk and the corresponding reduction of risk. For this, the risk leverage of the various risks can be estimated.

Risk leverage is the variation in risk exposure divided by the amount of reducing the risk.

Risk leverage = (risk exposure before reduction - risk exposure after reduction) / (cost of reduction)

1. Risk planning: The risk planning method considers each of the key risks that have been identified and develop ways to maintain these risks.

For each of the risks, you have to think of the behavior that you may take to minimize the disruption to the plan if the issue identified in the risk occurs.

You also should think about data that you might need to collect while monitoring the plan so that issues can be anticipated.

Again, there is no easy process that can be followed for contingency planning. It rely on the judgment and experience of the project manager.

2. Risk Monitoring: Risk monitoring is the method king that your assumption about the product, process, and business risks has not changed.

7. Project Communication Management: Communication is an essential factor in the success of the project. It is a bridge between client, organization, team members and as well as other stakeholders of the project such as hardware suppliers.

From the planning to closure, communication plays a vital role. In all the phases, communication must be clear and understood. Miscommunication can create a big blunder in the project.

8. Project Configuration Management: Configuration management is about to control the changes in software like requirements, design, and development of the product.

The Primary goal is to increase productivity with fewer errors.

Some reasons show the need for configuration management:

o Several people work on software that is continually update.

o Help to build coordination among suppliers.

o Changes in requirement, budget, schedule need to accommodate.

o Software should run on multiple systems.

Tasks perform in Configuration management:
o Identification
o Baseline
o Change Control
o Configuration Status Accounting
o Configuration Audits and Reviews
People involved in Configuration Management:

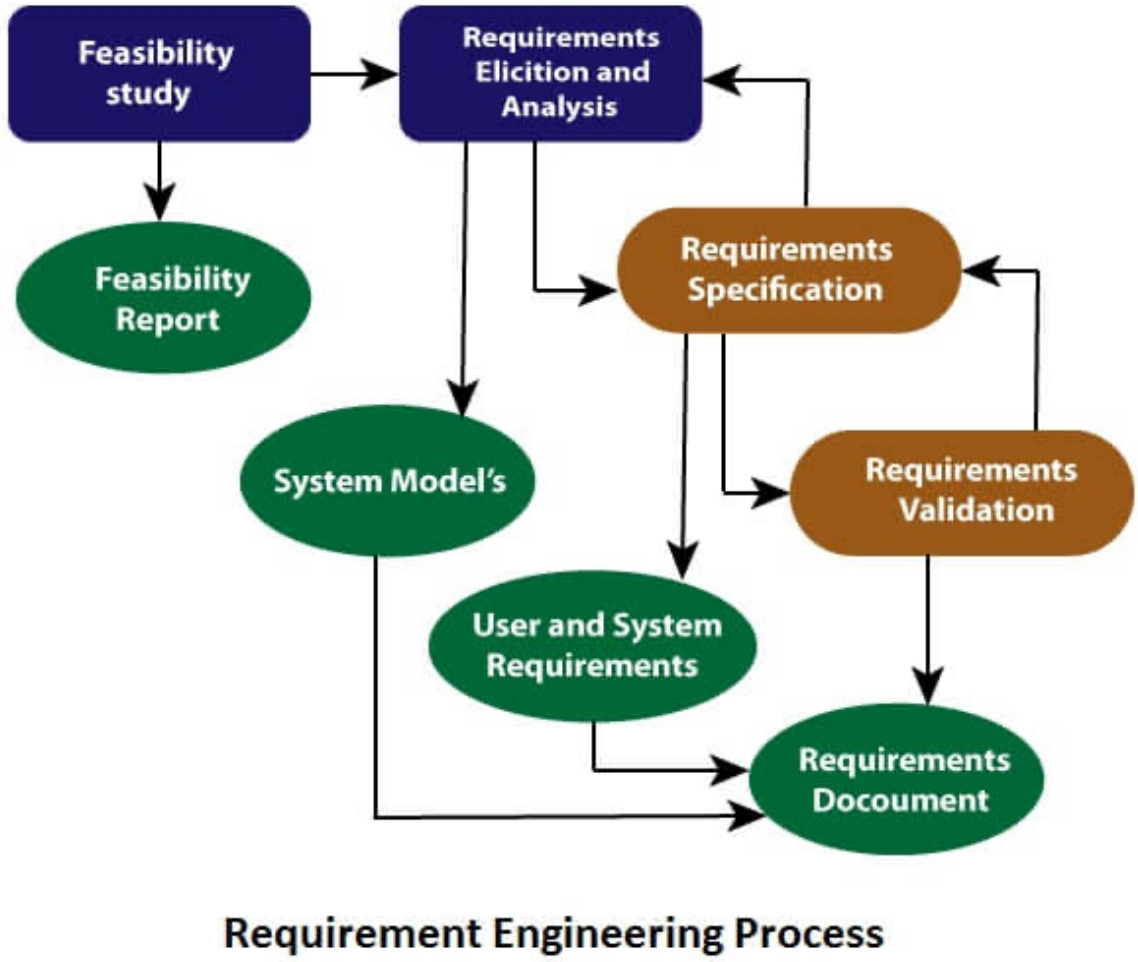
Requirement Engineering

Requirements engineering (RE) refers to the process of defining, documenting, and maintaining requirements in the engineering design process. Requirement engineering provides the appropriate mechanism to understand what the customer desires, analyzing the need, and assessing feasibility, negotiating a reasonable solution, specifying the solution clearly, validating the specifications and managing the requirements as they are transformed into a working system. Thus, requirement engineering is the disciplined application of proven principles, methods, tools, and notation to describe a proposed system's intended behavior and its associated constraints.

Requirement Engineering Process

It is a four-step process, which includes -
1. Feasibility Study
2. Requirement Elicitation and Analysis
3. Software Requirement Specification

4. Software Requirement Validation
5. Software Requirement Management



Requirement Engineering Process

1. Feasibility Study:

The objective behind the feasibility study is to create the reasons for developing the software that is acceptable to users, flexible to change and conformable to established standards.

Types of Feasibility:

20.7M
471
Exception Handling in Java - Javatpoint

1. **Technical Feasibility** - Technical feasibility evaluates the current technologies, which are needed to accomplish customer requirements within the time and budget.
2. **Operational Feasibility** - Operational feasibility assesses the range in which the required software performs a series of levels to solve business problems and customer requirements.
3. **Economic Feasibility** - Economic feasibility decides whether the

necessary software can generate financial profits for an organization.

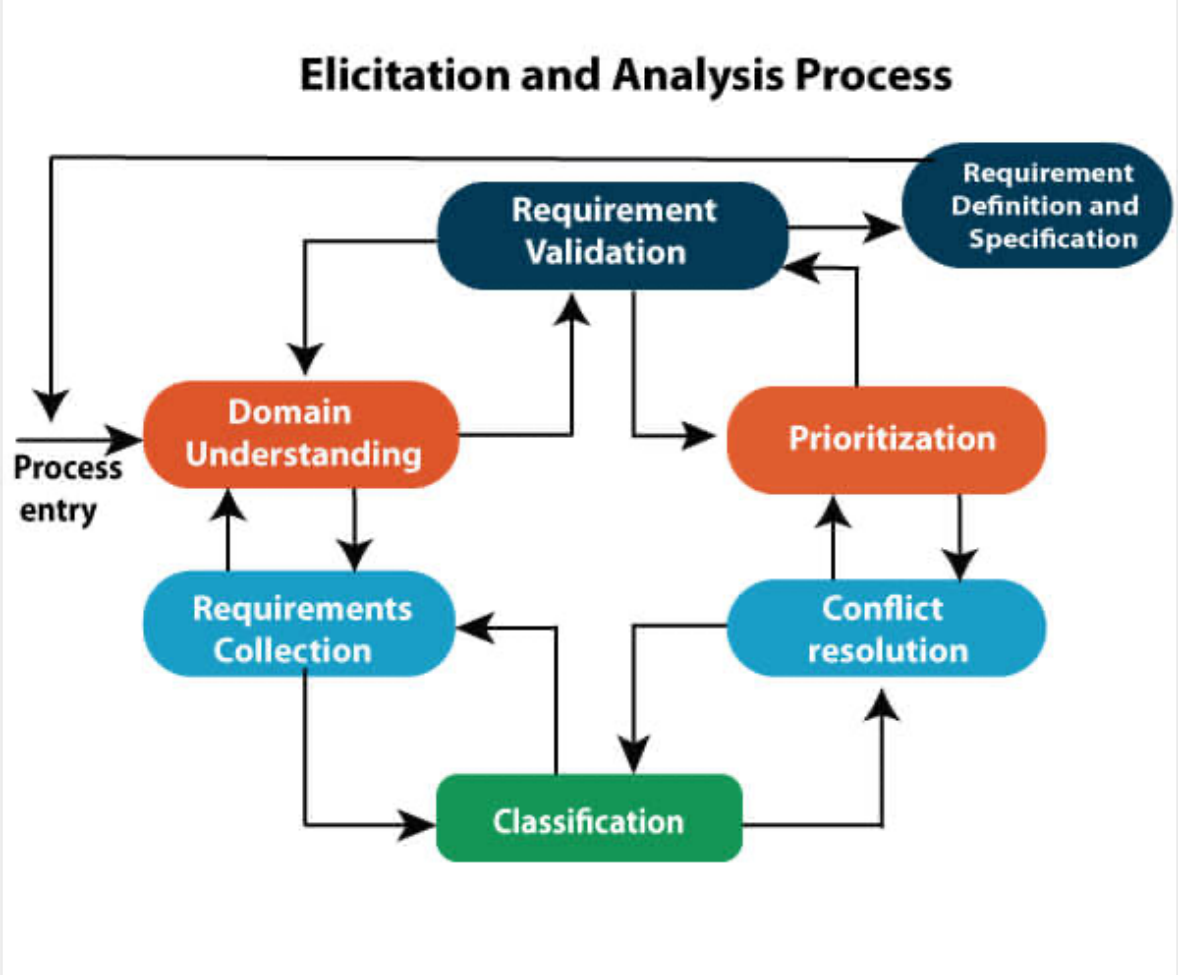
2. Requirement Elicitation and Analysis:

This is also known as the **gathering of requirements**. Here, requirements are identified with the help of customers and existing systems processes, if available.

Analysis of requirements starts with requirement elicitation. The requirements are analyzed to identify inconsistencies, defects, omission, etc. We describe requirements in terms of relationships and also resolve conflicts if any.

Problems of Elicitation and Analysis

o Getting all, and only, the right people involved.
o Stakeholders often don't know what they want
o Stakeholders express requirements in their terms.
o Stakeholders may have conflicting requirements.
o Requirement change during the analysis process.
o Organizational and political factors may influence system requirements.



3. Software Requirement Specification:

Software requirement specification is a kind of document which is created by a software analyst after the requirements collected from the various sources - the requirement received by the customer written in ordinary language. It is the job of the analyst to write the requirement in technical language so that they can be understood and beneficial by the development team.

The models used at this stage include ER diagrams, data flow diagrams (DFDs), function decomposition diagrams (FDDs), data dictionaries, etc.

o **Data Flow Diagrams:** Data Flow Diagrams (DFDs) are used widely for modeling the requirements. DFD shows the flow of data through a system. The system may be a company, an organization, a set of procedures, a computer hardware system, a software system, or any combination of the preceding. The DFD is also known as a data flow graph or bubble chart.

o **Data Dictionaries:** Data Dictionaries are simply repositories to store information about all data items defined in DFDs. At the requirements stage, the data dictionary should at least define customer data items, to ensure that the customer and developers use the same definition and terminologies.

o **Entity-Relationship Diagrams:** Another tool for requirement specification is the entity-relationship diagram, often called an "***E-R diagram***." It is a detailed logical representation of the data for the organization and uses three main constructs i.e. data entities, relationships, and their associated attributes.

4. Software Requirement Validation:

After requirement specifications developed, the requirements discussed in this document are validated. The user might demand illegal, impossible solution or experts may misinterpret the needs. Requirements can be the check against the following conditions -

o If they can practically implement

o If they are correct and as per the functionality and specially of software

o If there are any ambiguities

o If they are full

o If they can describe

Requirements Validation Techniques

o **Requirements reviews/inspections:** systematic manual analysis of the requirements.

o **Prototyping:** Using an executable model of the system to check

requirements.
o **Test-case generation:** Developing tests for requirements to check testability.
o **Automated consistency analysis:** checking for the consistency of structured requirements descriptions.

Software Requirement Management:

Requirement management is the process of managing changing requirements during the requirements engineering process and system development.
New requirements emerge during the process as business needs a change, and a better understanding of the system is developed.
The priority of requirements from different viewpoints changes during development process.
The business and technical environment of the system changes during the development.

Prerequisite of Software requirements

Collection of software requirements is the basis of the entire software development project. Hence they should be clear, correct, and well-defined.
A complete Software Requirement Specifications should be:
o Clear
o Correct
o Consistent
o Coherent
o Comprehensible
o Modifiable
o Verifiable
o Prioritized
o Unambiguous
o Traceable
o Credible source
Software Requirements: Largely software requirements must be categorized into two categories:
1. **Functional Requirements:** Functional requirements define a function that a system or system element must be qualified to perform and must be documented in different forms. The functional requirements are describing the behavior of the system as it correlates to the system's functionality.
2. **Non-functional Requirements:** This can be the necessities that specify the criteria that can be used to decide the operation instead

of specific behaviors of the system.
Non-functional requirements are divided into two main categories:
o **Execution qualities** like security and usability, which are observable at run time.
o **Evolution qualities** like testability, maintainability, extensibility, and scalability that embodied in the static structure of the software system.

Software Requirement Specifications

The production of the requirements stage of the software development process is **Software Requirements Specifications (SRS)** (also called a **requirements document**). This report lays a foundation for software engineering activities and is constructing when entire requirements are elicited and analyzed. **SRS** is a formal report, which acts as a representation of software that enables the customers to review whether it (SRS) is according to their requirements. Also, it comprises user requirements for a system as well as detailed specifications of the system requirements.
The SRS is a specification for a specific software product, program, or set of applications that perform particular functions in a specific environment. It serves several goals depending on who is writing it. First, the SRS could be written by the client of a system. Second, the SRS could be written by a developer of the system. The two methods create entirely various situations and establish different purposes for the document altogether. The first case, SRS, is used to define the needs and expectation of the users. The second case, SRS, is written for various purposes and serves as a contract document between customer and developer.

Characteristics of good SRS

Following are the features of a good SRS document:

1. 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.

20.6M

478

Features of Java - Javatpoint

2. Completeness: The SRS is complete if, and only if, it includes the following elements:

(1). All essential requirements, whether relating to functionality, performance, design, constraints, attributes, or external interfaces.

(2). Definition of their responses of the software to all realizable classes of input data in all available categories of situations.

Note: It is essential to specify the responses to both valid and invalid values.

(3). Full labels and references to all figures, tables, and diagrams in the SRS and definitions of all terms and units of measure.

3. Consistency: The SRS is consistent if, and only if, no subset of individual requirements described in its conflict. There are three types of possible conflict in the SRS:

(1). The specified characteristics of real-world objects may conflicts. For example,
(a) The format of an output report may be described in one requirement as tabular but in another as textual.
(b) One condition may state that all lights shall be green while another states that all lights shall be blue.
(2). There may be a reasonable or temporal conflict between the two specified actions. For example,
(a) One requirement may determine that the program will add two inputs, and another may determine that the program will multiply them.
(b) One condition may state that "A" must always follow "B," while other requires that "A and B" co-occurs.
(3). Two or more requirements may define the same real-world object but use different terms for that object. For example, a program's request for user input may be called a "prompt" in one requirement's and a "cue" in another. The use of standard terminology and descriptions promotes consistency.
4. Unambiguousness: SRS is unambiguous when every fixed requirement has only one interpretation. This suggests that each element is uniquely interpreted. In case there is a method used with multiple definitions, the requirements report should determine the implications in the SRS so that it is clear and simple to understand.
5. Ranking for importance and stability: The SRS is ranked for importance and stability if each requirement in it has an identifier to indicate either the significance or stability of that particular requirement.
Typically, all requirements are not equally important. Some prerequisites may be essential, especially for life-critical applications, while others may be desirable. Each element should be identified to make these differences clear and explicit. Another way to rank requirements is to distinguish classes of items as essential, conditional, and optional.
6. Modifiability: SRS should be made as modifiable as likely and should be capable of quickly obtain changes to the system to some extent. Modifications should be perfectly indexed and cross-referenced.
7. Verifiability: SRS is correct when the specified requirements can be verified with a cost-effective system to check whether the final software meets those requirements. The requirements are verified with the help of reviews.
8. Traceability: The SRS is traceable if the origin of each of the

requirements is clear and if it facilitates the referencing of each condition in future development or enhancement documentation.

There are two types of Traceability:

1. Backward Traceability: This depends upon each requirement explicitly referencing its source in earlier documents.

2. Forward Traceability: This depends upon each element in the SRS having a unique name or reference number.

The forward traceability of the SRS is especially crucial when the software product enters the operation and maintenance phase. As code and design document is modified, it is necessary to be able to ascertain the complete set of requirements that may be concerned by those modifications.

9. Design Independence: There should be an option to select from multiple design alternatives for the final system. More specifically, the SRS should not contain any implementation details.

10. Testability: An SRS should be written in such a method that it is simple to generate test cases and test plans from the report.

11. Understandable by the customer: An end user may be an expert in his/her explicit domain but might not be trained in computer science. Hence, the purpose of formal notations and symbols should be avoided too as much extent as possible. The language should be kept simple and clear.

12. The right level of abstraction: If the SRS is written for the requirements stage, the details should be explained explicitly. Whereas,for a feasibility study, fewer analysis can be used. Hence, the level of abstraction modifies according to the objective of the SRS.

Properties of a good SRS document

The essential properties of a good SRS document are the following:

Concise: The SRS report should be concise and at the same time, unambiguous, consistent, and complete. Verbose and irrelevant descriptions decrease readability and also increase error possibilities.

Structured: It should be well-structured. A well-structured document is simple to understand and modify. In practice, the SRS document undergoes several revisions to cope up with the user requirements. Often, user requirements evolve over a period of time. Therefore, to make the modifications to the SRS document easy, it is vital to make the report well-structured.

Black-box view: It should only define what the system should do and refrain from stating how to do these. This means that the SRS

document should define the external behavior of the system and not discuss the implementation issues. The SRS report should view the system to be developed as a black box and should define the externally visible behavior of the system. For this reason, the SRS report is also known as the black-box specification of a system.

Conceptual integrity: It should show conceptual integrity so that the reader can merely understand it. Response to undesired events: It should characterize acceptable responses to unwanted events. These are called system response to exceptional conditions.

Verifiable: All requirements of the system, as documented in the SRS document, should be correct. This means that it should be possible to decide whether or not requirements have been met in an implementation.