Chapter 1: Introduction to software and software engineering

By

Er. Raj Kiran Chhatkuli

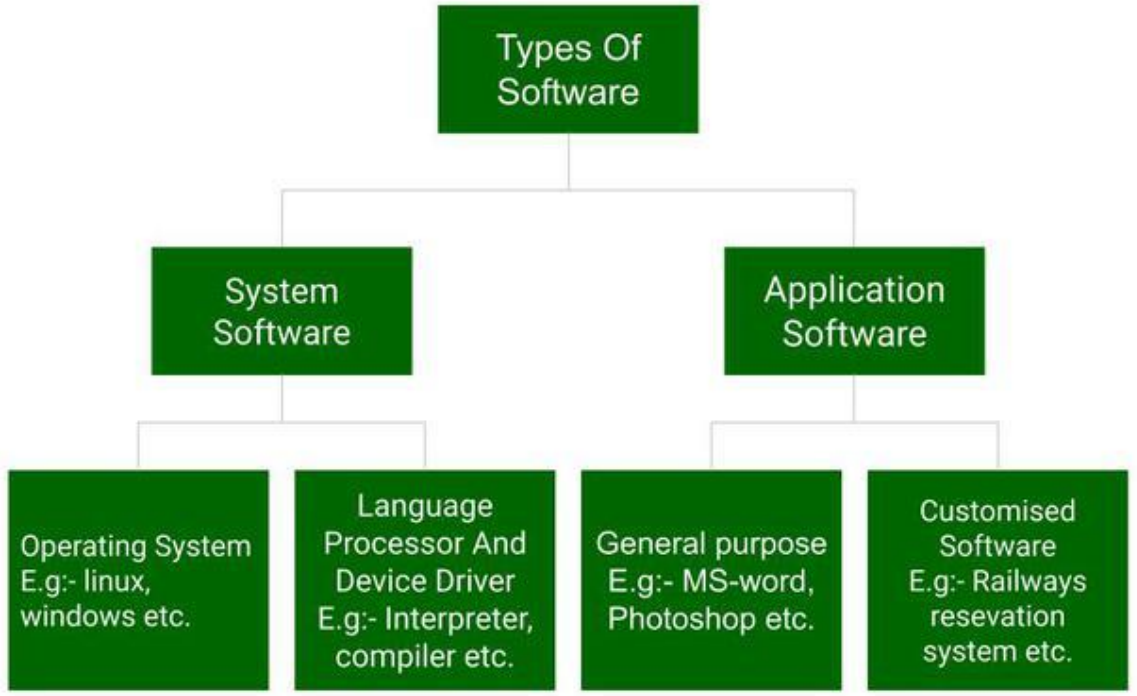
Software and Software engineering

- The term ‘Software’ is more than just a program code. A program is an executable code, which serves some computational purpose.
- Software is considered to be collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called *software product*.
- The other term ‘Engineering’ is all about developing some quality products by using some well-defined, scientific principles and standard within the specified budget and time.

Software and Software engineering

- Hence, Software engineering is defined as: It is a branch of computer science which is associated with development of quality software product using well-defined scientific principles, methods and procedures.
- The outcome of software engineering is an efficient and reliable software product that support over the long term.

Types of Software


Types Of Software
System Software
Application Software
Operating System E.g:- linux, windows etc.
Language Processor And Device Driver E.g:- Interpreter, compiler etc.
General purpose E.g:- MS-word, Photoshop etc.
Customised Software E.g:- Railways resevation system etc.

1. Application Software

- It can be divided into Generic and Specific

a. Generic (or Packaged) Software

- General purposed software
- The application software which is designed to fulfill the needs of large group of users is known as generic or packaged software.
- Example: MS-Word, MS-Excel.

b*. *Tailored (or Specific) Software

- The application software which is designed to fulfill the needs of a particular user/company/organization is known as tailored or specific software.
- Example: Software used in department stores, hospitals, schools etc

2. System Software:

- The software which can directly control the hardware of the computer are known as system software.
- Example: OS, Video driver, audio driver.

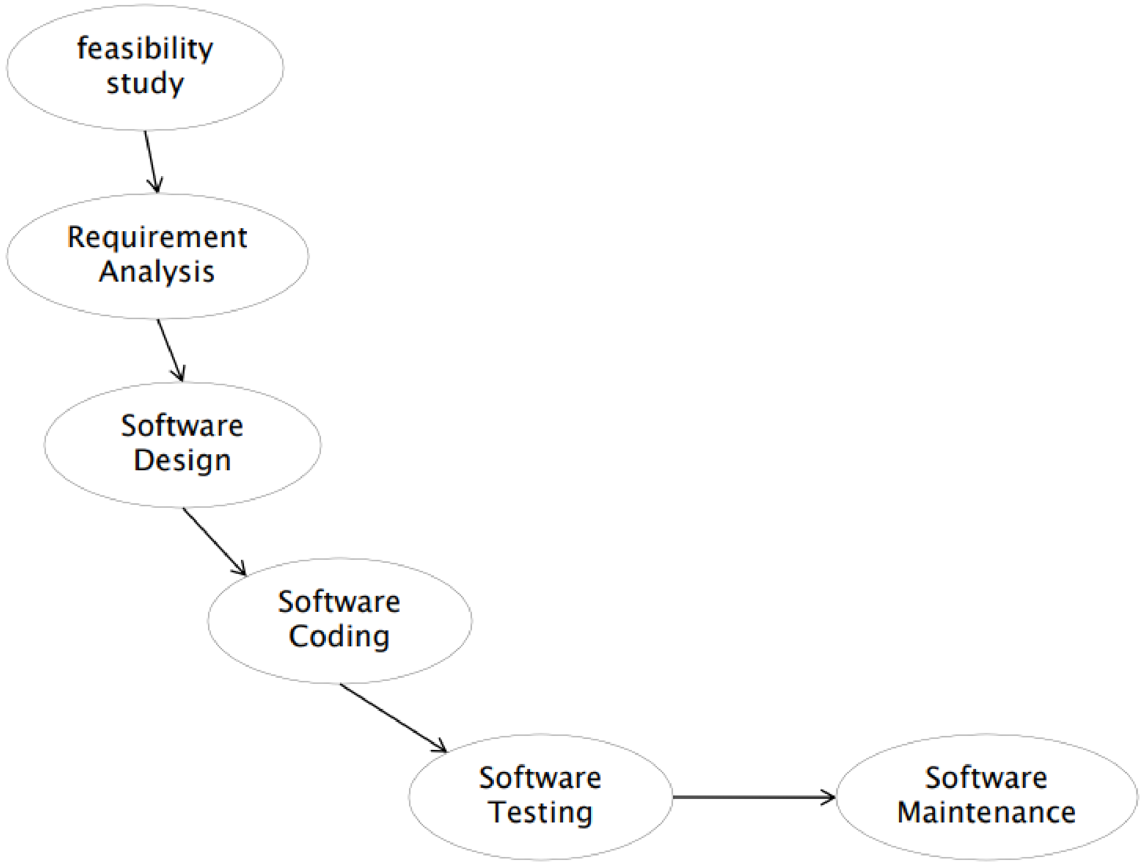
3. Utility Software:

- Small software that usually performs some useful tasks is known as utility software.
- Example: Win Zip, JPEG Compressor, PDF Merger, PDF to Word Converter etc.

Software process and software process model

- Since the *prime objective of software engineering* is to develop methods for large systems, which ***produce high quality software at low cost and in reasonable time***.
- So it is essential to ***perform software development in phases***. This phased development of software is often referred to as software development life cycle (SDLC) or software process.
- And the models used to achieve these goals are termed as Software Process Models.

Software development process



1.Feasibility study/ Preliminary investigation

- Feasibility study decides whether the new system should be developed or not.
- There are three constraints, which decides the go or no-go decision.

a. Technical Feasibility:

- ❑ determines whether technology needed for proposed system is available or not.
- ❑ determines whether the existing system can be upgraded to use new technology whether the organization has the expertise to use it or not.

b. Time Feasibility:

- ❑ determines the time needed to complete a project.
- ❑ Time is an important issue as cost increases with an increase in the time period of a project.

c. Budget Feasibility:

- ❑ This evaluation looks at the financial aspect of the project.
- ❑ determines whether the investment needed to implement the system will be recovered at later stages or not.

2. Requirement Analysis/Software Analysis:

- Studies the problem or requirements of software in detail.
- After analyzing and elicitations of the requirements of the user, a requirement statement known as software requirement specification (SRS) is developed.

3. Software Design

- Most crucial phase in the development of a system. The SDLC process continues to move from the what questions of the analysis phase to the how.
- Logical design is turned into a physical design.
- Based on the user requirements and the detailed analysis the system must be designed.
- Input, output, databases, forms, processing specifications etc. are drawn up in detail.
- Tools and techniques used for describing the system design are: ***Flowchart, ERD, Data flow diagram (DFD), UML diagrams like Use case, Activity, Sequence etc.***

4. Software Coding

- Physical design into software code.
- Writing a software code requires a prior knowledge of programming language and its tools. Therefore, it is important to choose the appropriate programming language according to the user requirements.
- A program code is efficient if it makes optimal use of resources and contains minimum errors.

5. Software Testing

- Software testing is performed to ensure that software produces the correct outputs i.e. free from errors. This implies that outputs produced should be according to user requirements.
- Efficient testing improves the quality of software.
- Test plan is created to test software in a planned and systematic manner.

6. Software Maintenance

- This phase comprises of a set of software engineering activities that occur after software is delivered to the user.
- After the software is developed and delivered, it may require changes. Sometimes, changes are made in software system when user requirements are not completely met.
- To make changes in software system, software maintenance process evaluates, controls, and implements changes.

Class Work

- Mention different phases of Software Development life cycle(SDLC), under the project of Library Management system.

Software process

What is Software process?

- When you work to build a product or system, it’s important to go through a series of predictable steps—*a road map* that helps you to create a timely, high-quality result. The road map that you follow is called a “software process.”

Who does it?

- *Software engineers and their managers* adapt the process to their needs and then follow it. In addition, *the people who have requested the software* have in the process of defining, building, and testing it.

Why is it important?

- Because it provides *path, stability, control over your project.*

Software process

What are the steps?

- At a detailed level, the process that you adopt *depends on the software that you’re building*. One process might be appropriate for creating software for an aircraft avionics system, while an entirely different process would be indicated for the creation of a website.

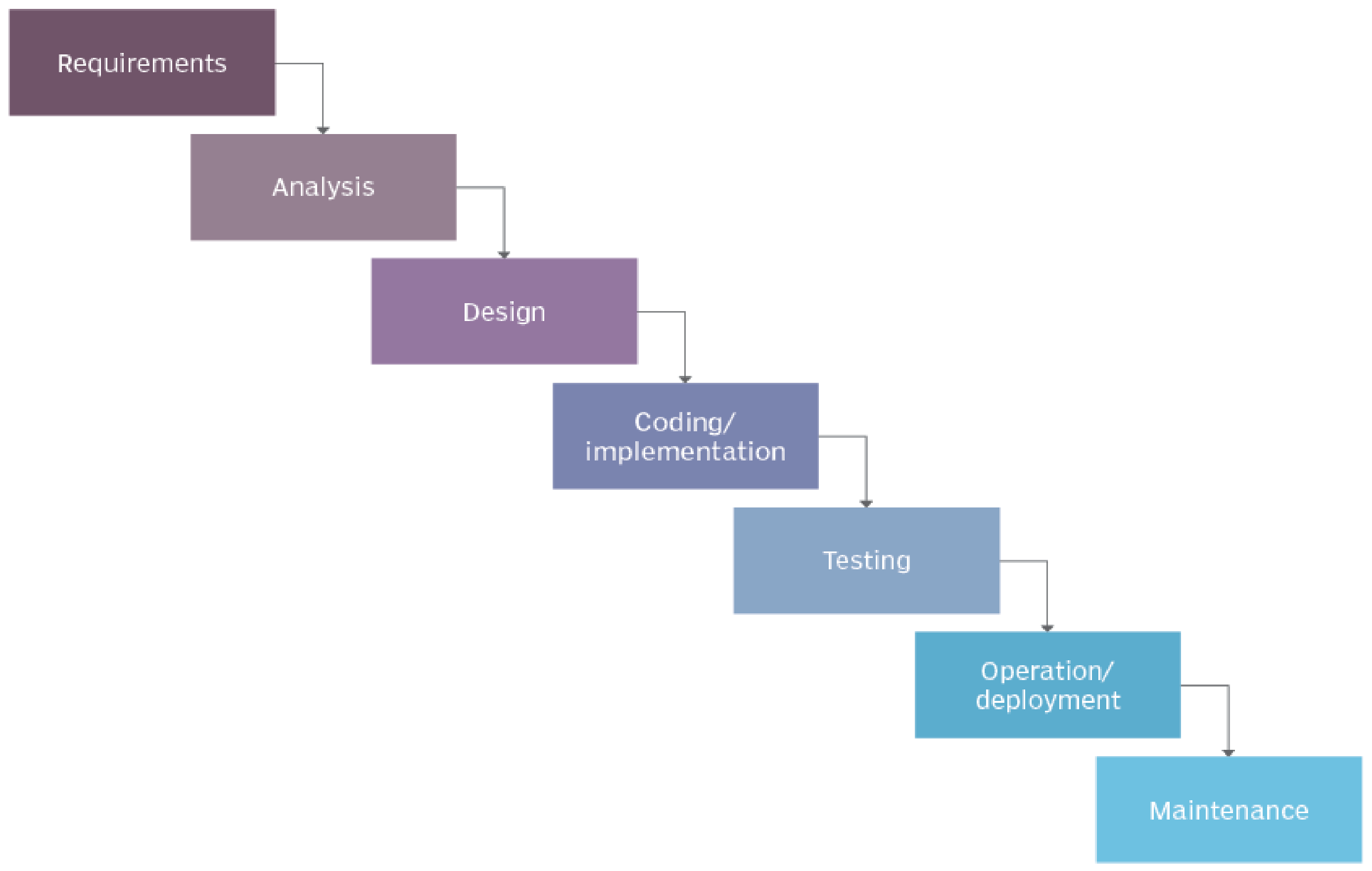
From a technical point of view:

- A software process is a *framework for the activities*, *actions*, *and tasks* that are required to build high-quality software.

The Waterfall Model

- The Waterfall Model is a *traditional and linear approach* to software development, where the development process flows sequentially through a set of distinct phases.
- Each phase in the Waterfall Model must be completed before moving on to the next, and changes are typically not revisited once a phase is finished.
- The model is called "waterfall" because it resembles a cascading waterfall where progress flows in one direction—downward through the phases.

Waterfall model



i. Feasibility study

- Technical, Financial & Time Feasibility etc.

ii. Requirement specification

- To specify the requirements' users specification should be clearly understood and the requirements should be analyzed.
- This phase involves interaction between *user and software engineer*, and produces a document known as *software requirement specification (SRS).*

iii. Design

- Determines the detailed process of developing software after the requirements are analyzed.
- *It utilizes software requirements defined by the user and translates them into a software representation*.
- In this phase, the emphasis is on finding a solution to the problems defined in the requirement analysis phase.
- The software engineer, in this phase is mainly concerned with the data structure, algorithmic detail, and interface representations.

iv. Coding

- Emphasizes on translation of design into a programming language using the coding style and guidelines. The programs created should be easy to read and understand. All the programs written are documented according to the specification.

v. Testing:

- Ensures that the *product is developed according to the requirements of the user*. Testing is performed to verify that the product is functioning efficiently with minimum errors. It focuses on the internal logics and external functions of the software .

vi. Implementation and maintenance

- Delivers fully functioning operational software to the user. Once the software is accepted and deployed at the user's end, various changes occur due to changes in external environment (these include upgrading new operating system or addition of a new peripheral device). The changes also occur due to changing requirements of the user and the changes occurring in the field of technology. This phase focuses on *modifying software, correcting errors, and improving the performance of the software.*

ADVANTAGES
OF WATERFALL

- ✓ Simple method and easy to use
- ✓ Phases are clear
- ✓ Suitable for smaller projects
- ✓ Easy to manage

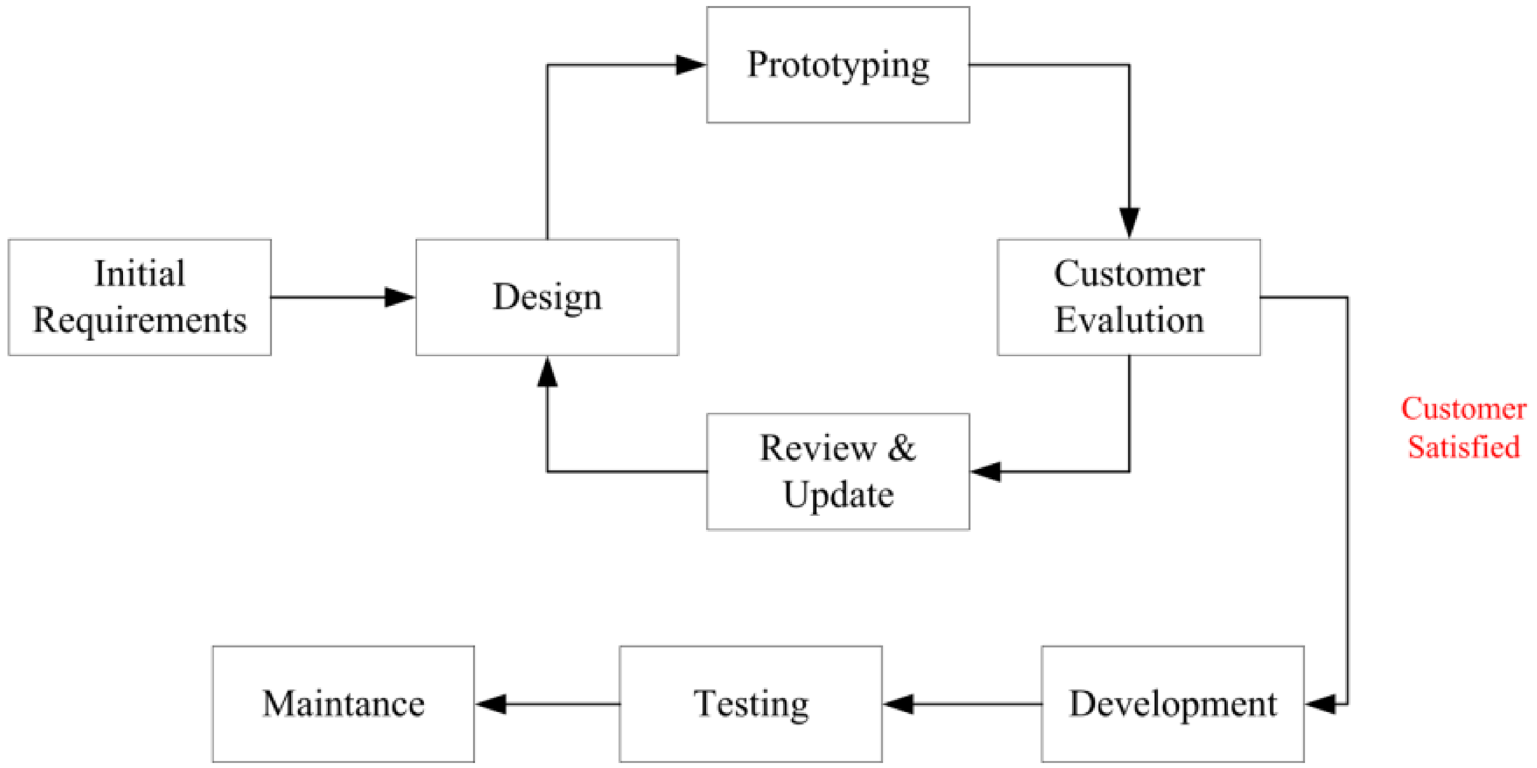
DISADVANTAGES
OF WATERFALL

- ✓ Does not allow much revision
- ✓ Not suitable for complex projects
- ✓ Risk and uncertainty are high
- ✓ Does not include a feedback path

The Prototype Model

- The Prototype Model is an iterative software development model in which a prototype (an early approximation of a final system or product) is built, tested, and then reworked until an acceptable prototype is achieved.
- It involves creating a working model of parts of the system or the entire system with the goal of understanding user requirements, validating design decisions, and refining the final product.

The Prototype Model



i. Requirements gathering and analysis

- Prototyping model begins with requirements analysis, and the requirements of the system are defined in detail.
- The *user is interviewed* to know the requirements of the system.

ii. Quick design

- When requirements are known, a preliminary design or a quick design for the system is created.
- It is not a detailed design, however, it includes the important aspects of the system, which gives an idea of the system to the user.
- Quick design helps in developing the prototype.

iii. Build prototype

- Information gathered from quick design is modified to form a prototype.
- The first prototype of the required system is developed from quick design. It represents a ‘*rough’ design* of the required system.

iv. User evaluation

- Next, the proposed system is presented to the user for consideration as a part of development process.
- The users thoroughly evaluate the prototype and recognize its *strengths and weaknesses*, such as *what is to be added or removed*.
- Comments and suggestions are collected from the users and are provided to the developer.

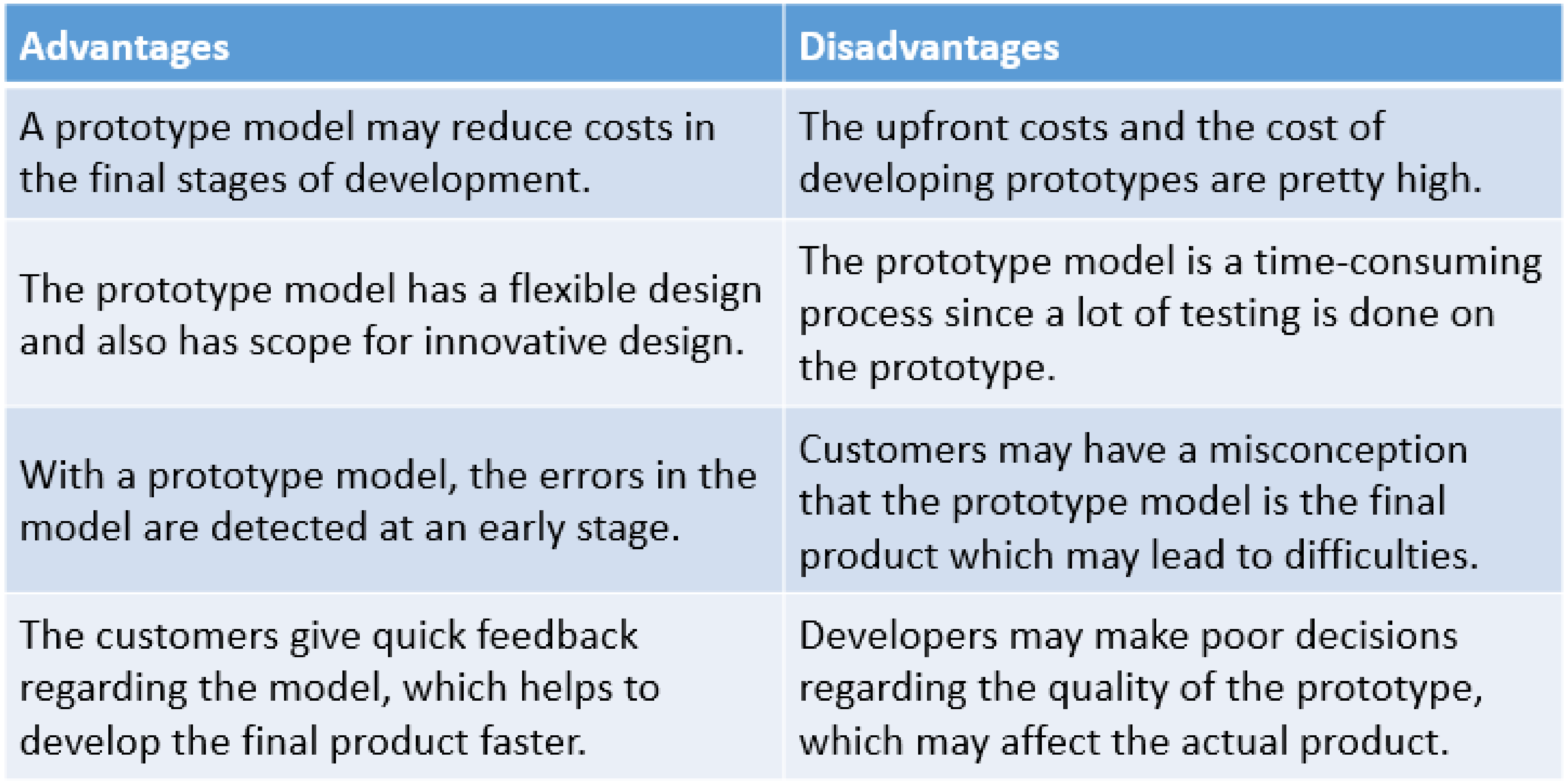
v. Prototype refinement

- Once the user evaluates the prototype, it is refined according to the requirements. The developer revises the prototype to make it more effective and efficient according to the user requirement.
- If the user is not satisfied with the developed prototype, a new prototype is developed with the additional information provided by the user. The new prototype is evaluated in the same manner as the previous prototype, process continues until all the requirements specified by the user are met. Once the user is satisfied a final system is developed.

vi. Engineer product

- Once the requirements are completely known, user accepts the final prototype. The final system is thoroughly evaluated and tested followed by routine maintenance on continuing basis to prevent large-scale failures and to minimize downtime.

Advantages & Disadvantages of Prototype Model

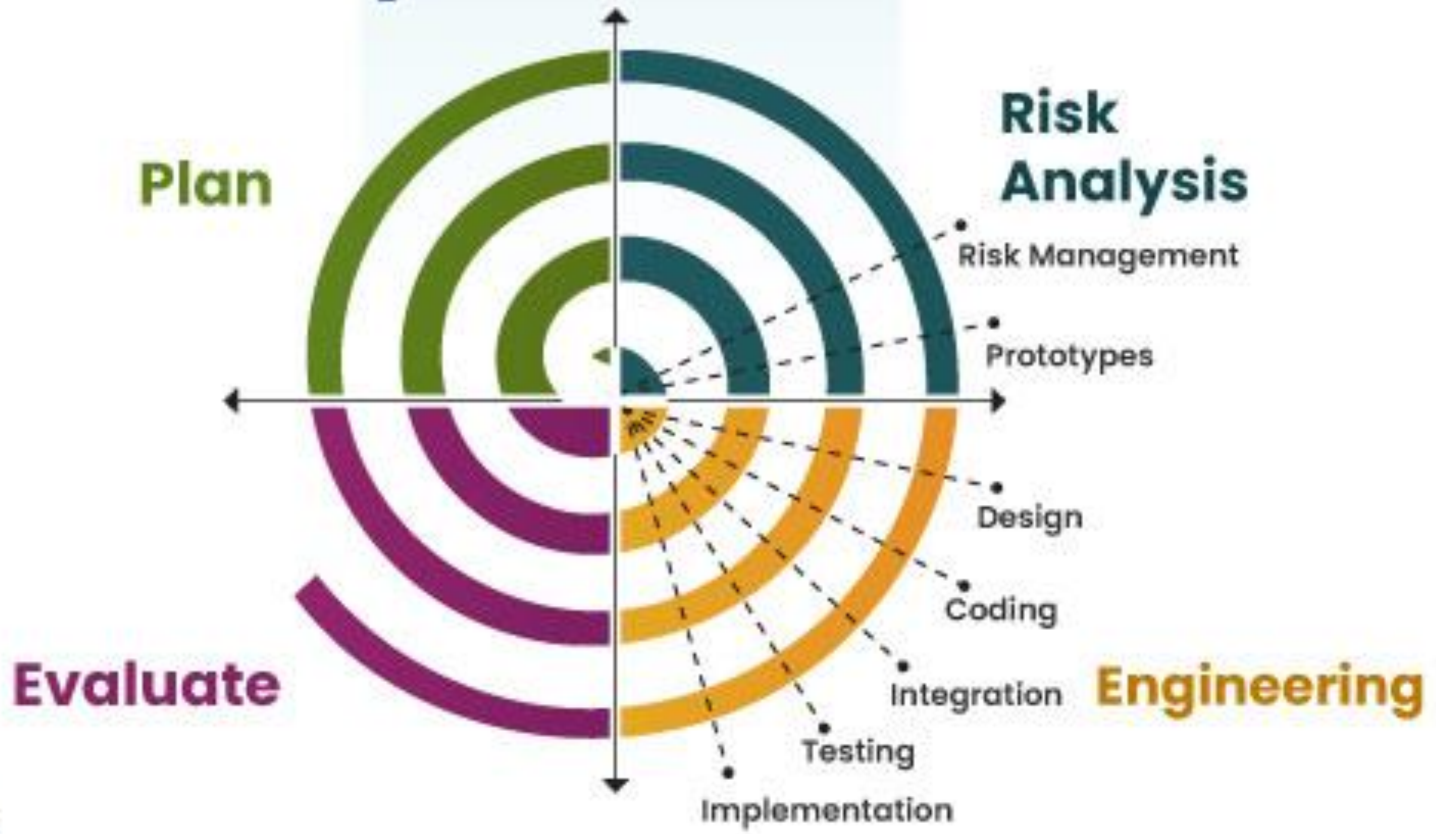


Advantages	Disadvantages
A prototype model may reduce costs in the final stages of development.	The upfront costs and the cost of developing prototypes are pretty high.
The prototype model has a flexible design and also has scope for innovative design.	The prototype model is a time-consuming process since a lot of testing is done on the prototype.
With a prototype model, the errors in the model are detected at an early stage.	Customers may have a misconception that the prototype model is the final product which may lead to difficulties.
The customers give quick feedback regarding the model, which helps to develop the final product faster.	Developers may make poor decisions regarding the quality of the prototype, which may affect the actual product.

The Spiral Model

- The Spiral Model is a software development model that *combines the idea of iterative development (prototyping) with the systematic aspects of the waterfall model.*
- It was proposed by Barry Boehm in 1986. The model is called the "Spiral" because it visualizes the software development process as a spiral with successive cycles, each representing a phase in the software development life cycle (SDLC).
- The key feature of the Spiral Model is *risk management*, with a focus on identifying and mitigating risks throughout the development process.

What Is Spiral Model



1. Planning:

- Define the project objectives, requirements, constraints, and deliverables.
- Identify potential risks and establish a plan for risk management.
- Develop a preliminary schedule and budget.

2. Risk Analysis:

- Evaluate identified risks and assess their impact on the project.
- Prioritize risks based on their severity and likelihood of occurrence.
- Develop strategies to mitigate or manage the identified risks.

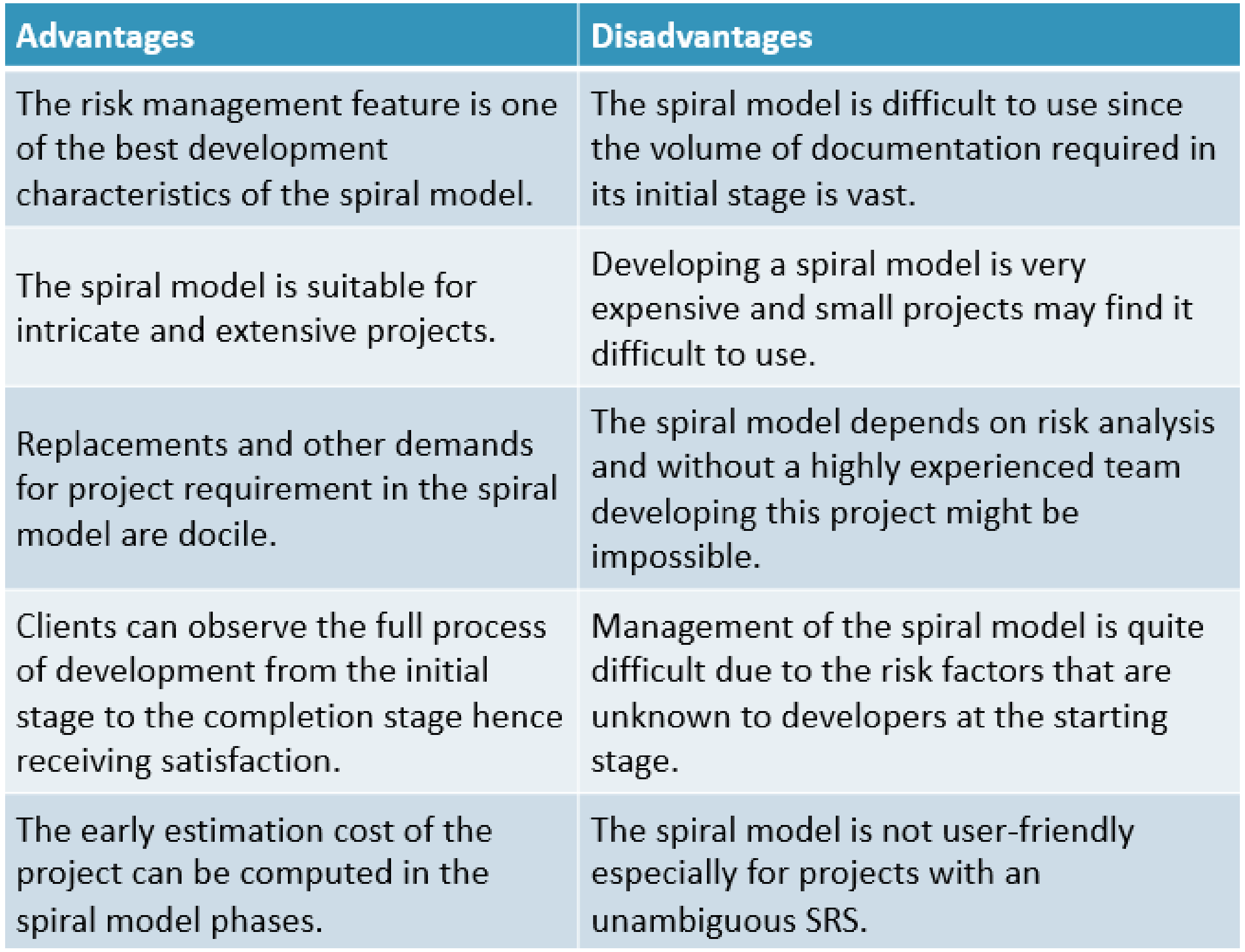
3. Engineering (Development and Testing):

- Implement the project based on the requirements and design specifications.
- Conduct testing during and after the implementation phase.
- Each cycle in the spiral represents a portion of the product, and these cycles may involve building prototypes, designing components, or implementing specific features.

4. Evaluation:

- Review the progress made in the current iteration.
- Assess the project against the defined objectives and constraints.
- Determine if the project is ready to proceed to the next iteration or if further refinement is necessary.

Advantages & Disadvantages of Spiral Model

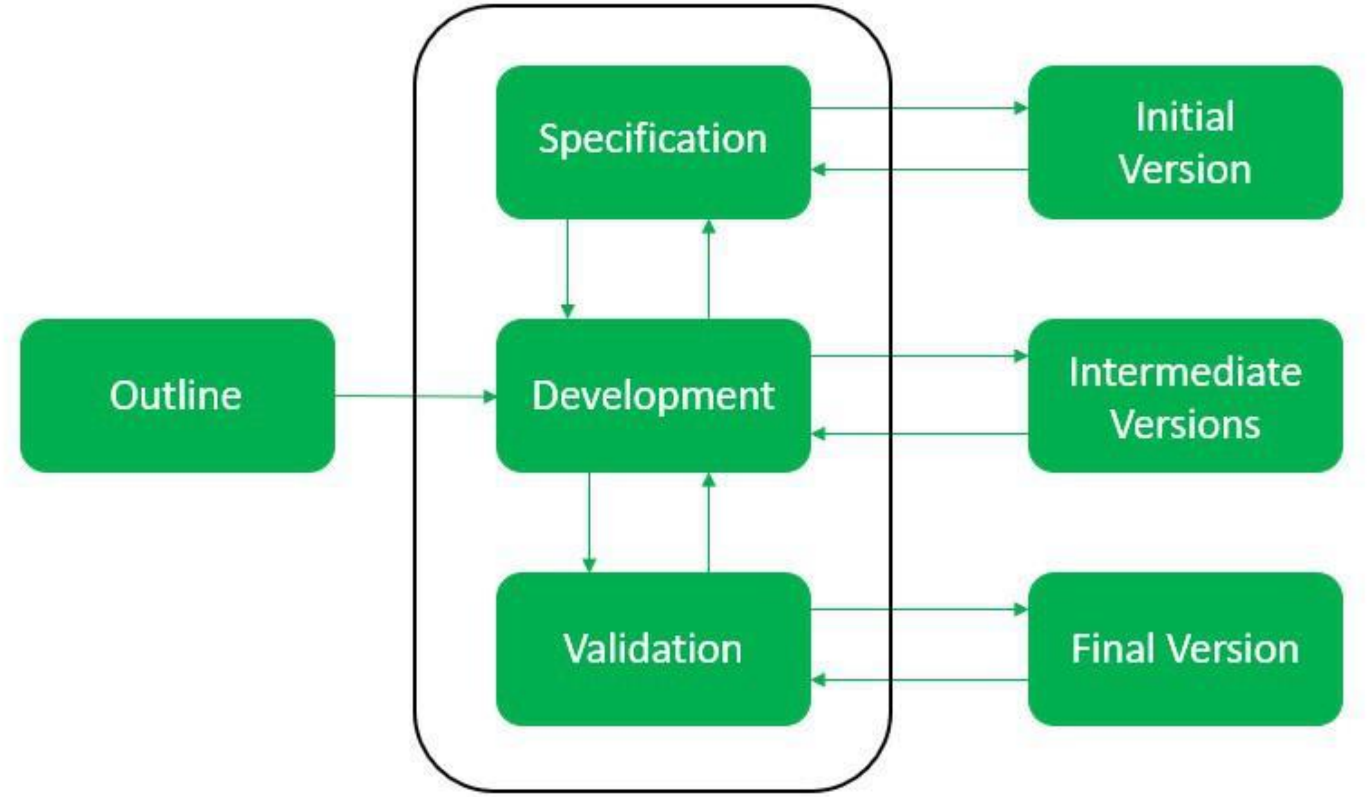


Advantages	Disadvantages
The risk management feature is one of the best development characteristics of the spiral model.	The spiral model is difficult to use since the volume of documentation required in its initial stage is vast.
The spiral model is suitable for intricate and extensive projects.	Developing a spiral model is very expensive and small projects may find it difficult to use.
Replacements and other demands for project requirement in the spiral model are docile.	The spiral model depends on risk analysis and without a highly experienced team developing this project might be impossible.
Clients can observe the full process of development from the initial stage to the completion stage hence receiving satisfaction.	Management of the spiral model is quite difficult due to the risk factors that are unknown to developers at the starting stage.
The early estimation cost of the project can be computed in the spiral model phases.	The spiral model is not user-friendly especially for projects with an unambiguous SRS.

The Evolutionary Model

- The Evolutionary Model is a software development model that emphasizes the iterative and incremental nature of the development process.
- It involves the *successive development of the software in small portions*, each *adding new features or refining existing ones*.
- The Evolutionary Model is particularly well-suited for projects where requirements are not well-understood initially or are expected to change frequently. This model allows for the adaptation of the software as it evolves over time.

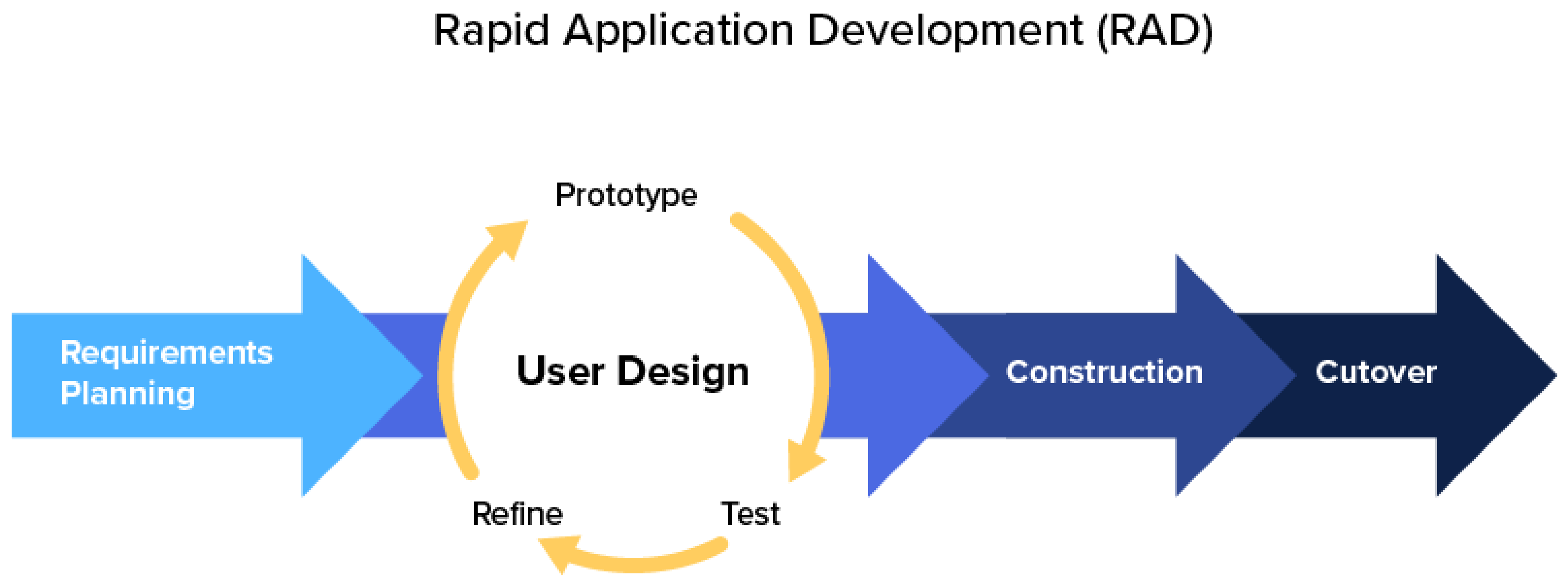
The Evolutionary Model



The Rapid Application Development (RAD)

- The Rapid Application Development (RAD) model is an incremental software development process model that *prioritizes rapid prototyping and quick feedback over strict planning and extensive testing*.
- It is *designed to deliver software systems in a short span of time*, emphasizing user involvement and iterative development.
- The RAD model was introduced to address the limitations of traditional waterfall models that tended to have long development cycles with minimal user involvement until the end.

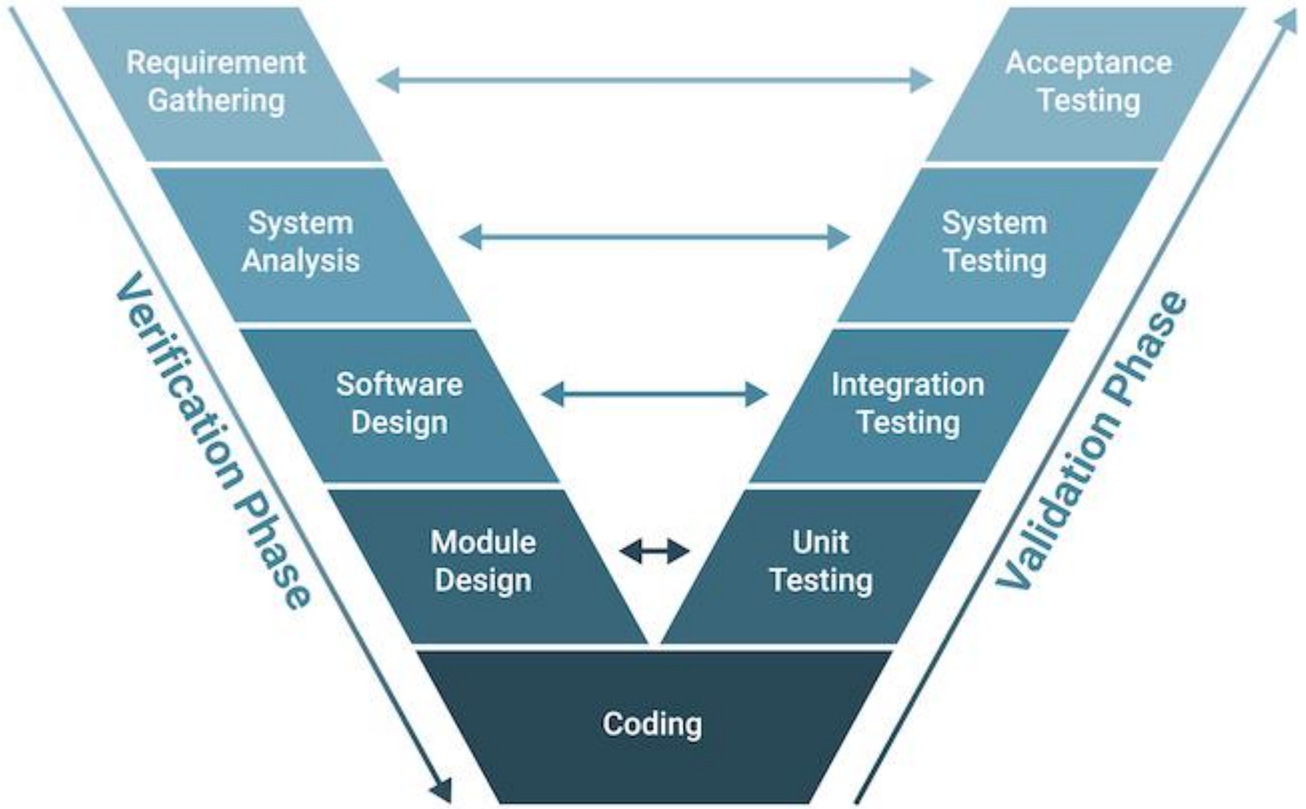
The Rapid Application Development (RAD)



The V-Model

- The V-Model, also known as the **Verification and Validation Model**, is a software development and testing model that emphasizes a systematic and parallel approach to the development life cycle.
- It is an *extension of the traditional waterfall model and is named after the V-shaped representation of the project's life cycle*.
- The V-Model incorporates testing activities into each phase of the development process, making it a testing-centric model.
- Each stage of the development process has a corresponding testing phase. Testing activities are planned in parallel with development activities so that the necessary resources are available to support testing.

The V-Model


Requirement Gathering
Acceptance Testing
System Analysis
System Testing
Software Design
Integration Testing
Module Design
Unit Testing
Coding
Verification Phase
Validation Phase

Agile software development

- Agile software development is an *iterative and flexible approach* to software development that prioritizes *collaboration*, *adaptability*, *and customer satisfaction*.
- It emerged as a response to the limitations of traditional, plan-driven methodologies, such as the Waterfall Model.
- Agile methodologies focus on delivering small, functional increments of a product quickly and regularly, allowing for continuous improvement and adaptation throughout the development process. Here are some key principles and practices associated with Agile development:

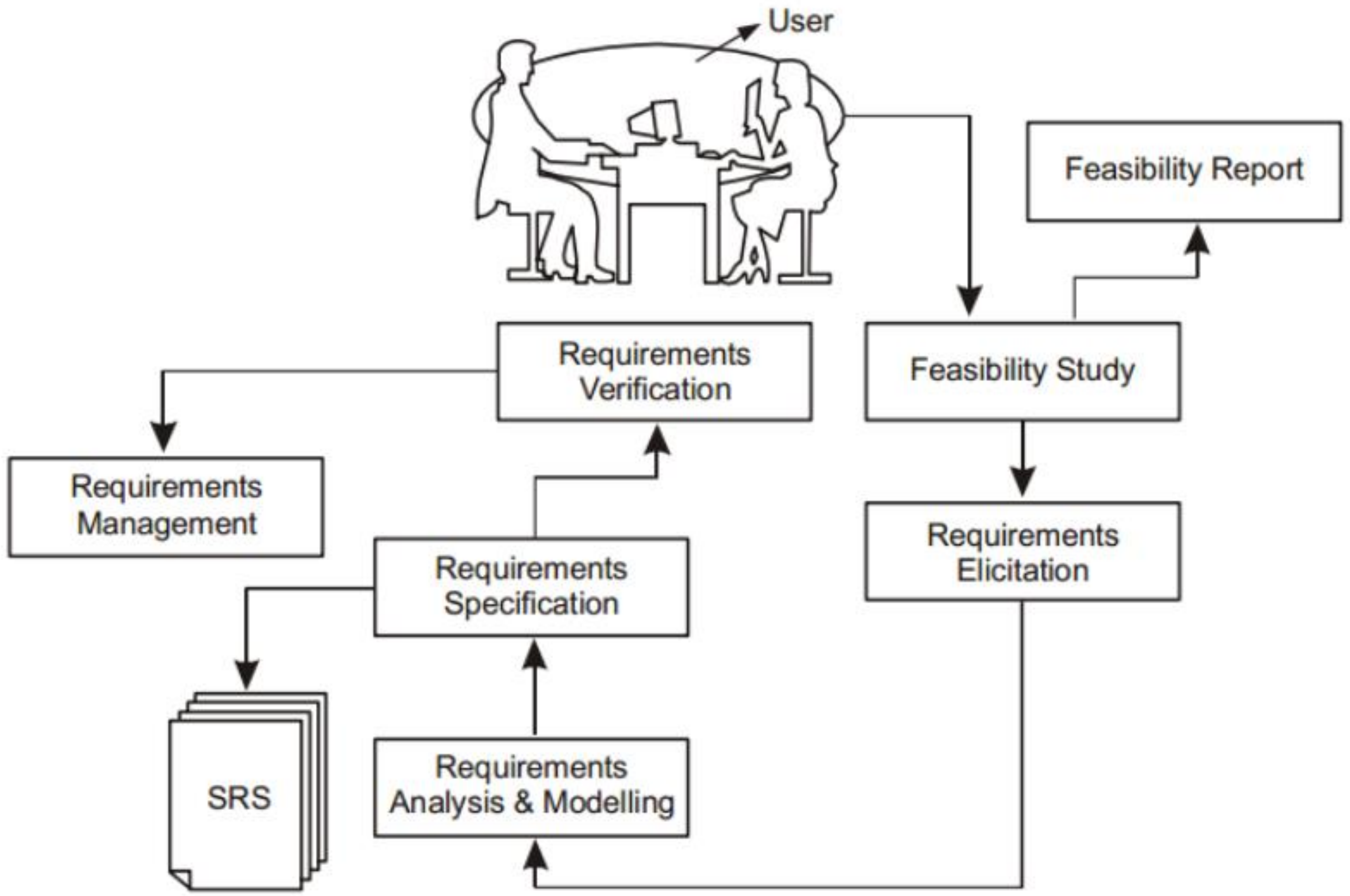
Agile Principles:

1. ***Individuals and Interactions over Processes and Tools:***
   - Emphasizes the importance of effective communication and collaboration among team members.
2. ***Working Software over Comprehensive Documentation:***
   - Values delivering a functional product over extensive documentation.
   - Documentation is important but should not be a barrier to development.
3. ***Customer Collaboration over Contract Negotiation:***
   - Encourages active involvement of customers or stakeholders throughout the development process.
   - Prioritizes collaboration over strict contractual agreements.
4. ***Responding to Change over Following a Plan:***
   - Acknowledges the inevitability of changing requirements and encourages flexibility and responsiveness to change.

Requirements Engineering Process

- The requirements engineering (RE) process is a series of activities that are performed in requirements phase in order to express requirements in software requirements specification (SRS) document.
- These steps include feasibility study, requirements elicitation, requirements analysis, requirements specification, requirements validation, and requirement management.

Requirements Engineering Process



STEP 1: FEASIBILITY STUDY

Objectives of feasibility study:

- To determine whether the software can be implemented using current technology and within the specified budget and schedule or not.
- To determine whether the software can be integrated with other existing software or not.
- To minimizes project failure.

Types of feasibility study

Technical

- technical skills and capabilities of development team.
- Assure that the technology chosen, has large number of users so that they can be consulted when problems arise.

Operational

- solution suggested by software development team is acceptable or not.
- whether users will adapt to new software or not.

Economic feasibility/ Budget

- whether the required software is capable of generating financial gains for an organization or not.
- cost incurred on software development team
- estimated cost of hardware and software.
- cost of performing feasibility study.

Time

- Whether the project will be completed on pre-specified time or not.

Feasibility Study Process

1.Information assessment:

- verifies that the system can be implemented using new technology and within the budget.

2. Information collection:

- Specifies the sources from where information about software can be obtained.

Sources:

❑users (who will operate the software)

❑organization (where the software will be used).

❑software development team (who understands user requirements and knows how to fulfill them in software).

3. Report writing:

Information about changes in software scope, budget, schedule, and suggestion of any requirements in the system.

Step 2: Requirements Elicitation

- Process of collecting information about software requirements from different stakeholders (users, developer, project manager etc.)

Issues in Requirement Elicitation

1. Problems of understanding

- *Users are not certain about their requirements* and thus are unable to express what they require in software and which requirements are feasible.
- This problem also arises when users have no or little knowledge of the problem domain and are unable to understand the limitations of computing environment of software.

2. Problems of volatility

- This problem arises when *requirements change over time*.

Elicitation Techniques

- The commonly followed elicitation techniques are listed below:

1.Interviews:

- Ways for eliciting requirements, it helps software engineer, users, & development team to understand the problem and suggest solution for the problem.
- *An effective interview should have characteristics listed below:*

❑Individuals involved in interviews should be able to accept new ideas, focus on listening to the views of stakeholders & avoid biased views.

❑Interviews should be conducted in defined context to requirements rather than in general terms.

- E.g. a set of a questions or a requirements proposal.

2. Scenarios:

- Helps to determine possible future outcome before implementation.
- In Generally, a scenario comprises of:

❑Description of what users expect when scenario starts.

❑Description of how to handle the situation when software is not operating correctly.

❑Description of the state of software when scenario ends.

3.Prototypes:

- helps to clarify unclear requirements.
- helps users to understand the information they need to provide to software development team.

Step 3: Requirement Analysis

- It is the process of studying and refining requirements

Tasks performed in requirements analysis are:

- Understand the problem for which software is to be developed.
- Develop analysis model to analyze the requirements in the software.
- Detect and resolve conflicts that arise due to unclear and unstated requirements.
- Determine operational characteristics of software and how it interacts with its environment.

Step 4: Requirements Specification

- Development of SRS document (software requirement specification document.
- Characteristics of SRS

1. Correct: SRS is correct when

- all user requirements are stated in the requirements document.
- The stated requirements should be according to the desired system.

2. Unambiguous:

- SRS is unambiguous when every stated requirement has only one interpretation i.e. each requirement is uniquely interpreted.

3. Complete:

- SRS is complete when the requirements clearly define what the software is required to do.

4. Modifiable:

- The requirements of the user can change, hence, requirements document should be created in such a manner where those changes can be modified easily.

5. Ranked for importance and stability:

- All requirements are not equally important.

6. Verifiable:

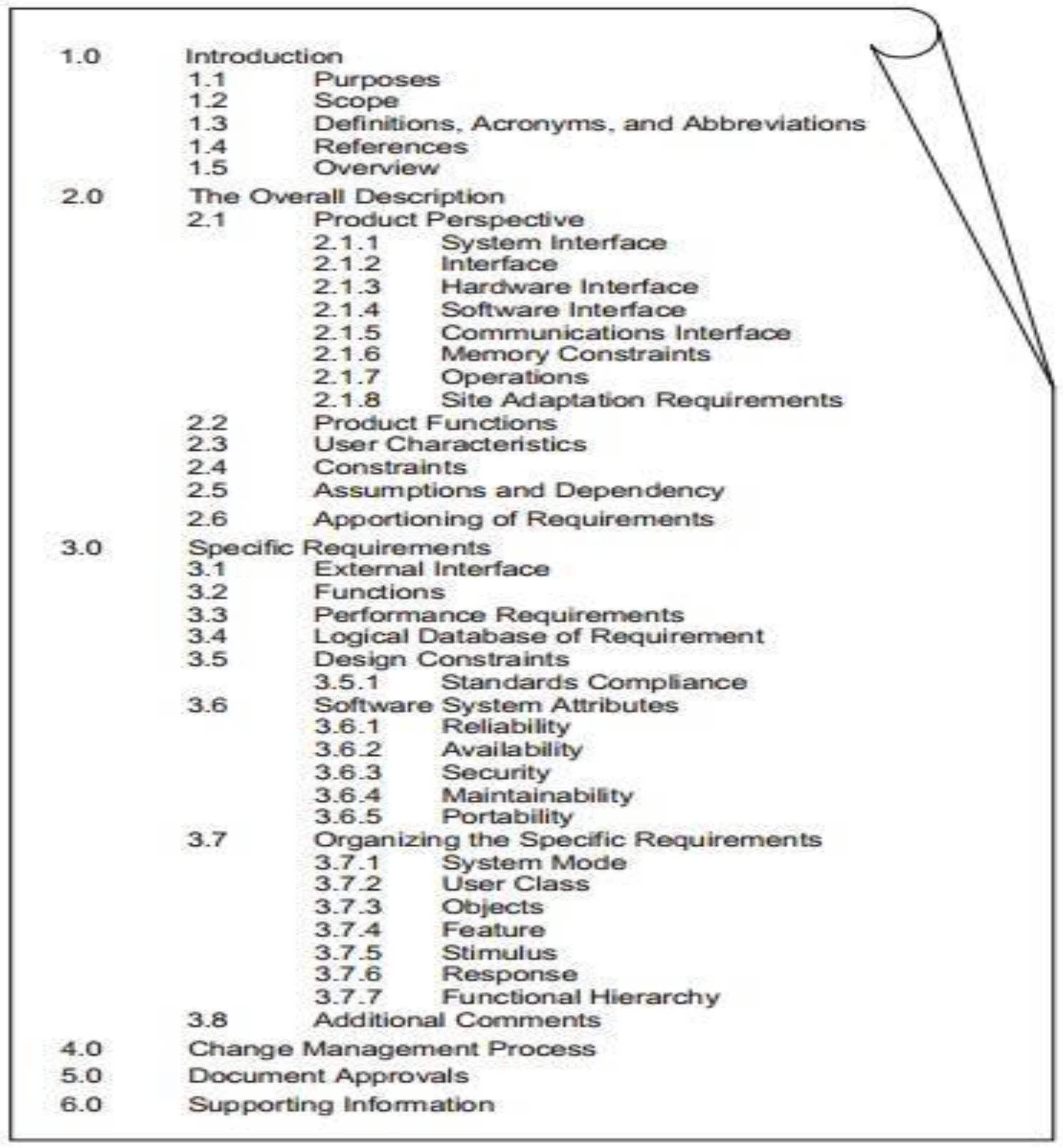
- SRS is verifiable when the specified requirements can be verified with a cost-effective process to check whether the final software meets those requirements or not.

7. Consistent:

- SRS is consistent when the individual requirements defined does not conflict with each other.
- e.g., a requirement states that an event 'a' is to occur before another event 'b'. But then another set of requirements states that event 'b' should occur before event 'a'.

8. Traceable:

- SRS is traceable when the source of each requirement is clear and it facilitates the reference of each requirement in future.



1.0 Introduction
- 1.1 Purposes
- 1.2 Scope
- 1.3 Definitions, Acronyms, and Abbreviations
- 1.4 References
- 1.5 Overview

2.0 The Overall Description
- 2.1 Product Perspective
  - 2.1.1 System Interface
  - 2.1.2 Interface
  - 2.1.3 Hardware Interface
  - 2.1.4 Software Interface
  - 2.1.5 Communications Interface
  - 2.1.6 Memory Constraints
  - 2.1.7 Operations
  - 2.1.8 Site Adaptation Requirements
- 2.2 Product Functions
- 2.3 User Characteristics
- 2.4 Constraints
- 2.5 Assumptions and Dependency
- 2.6 Apportioning of Requirements

3.0 Specific Requirements
- 3.1 External Interface
- 3.2 Functions
- 3.3 Performance Requirements
- 3.4 Logical Database of Requirement
- 3.5 Design Constraints
  - 3.5.1 Standards Compliance
- 3.6 Software System Attributes
  - 3.6.1 Reliability
  - 3.6.2 Availability
  - 3.6.3 Security
  - 3.6.4 Maintainability
  - 3.6.5 Portability
- 3.7 Organizing the Specific Requirements
  - 3.7.1 System Mode
  - 3.7.2 User Class
  - 3.7.3 Objects
  - 3.7.4 Feature
  - 3.7.5 Stimulus
  - 3.7.6 Response
  - 3.7.7 Functional Hierarchy
- 3.8 Additional Comments

4.0 Change Management Process

5.0 Document Approvals

6.0 Supporting Information

Fig : SRS Document template

Step 5 : Requirements Validation

Why Validation ?

Errors present in the SRS will adversely affect the cost if they are detected later in the development process or when the software is delivered to the user.

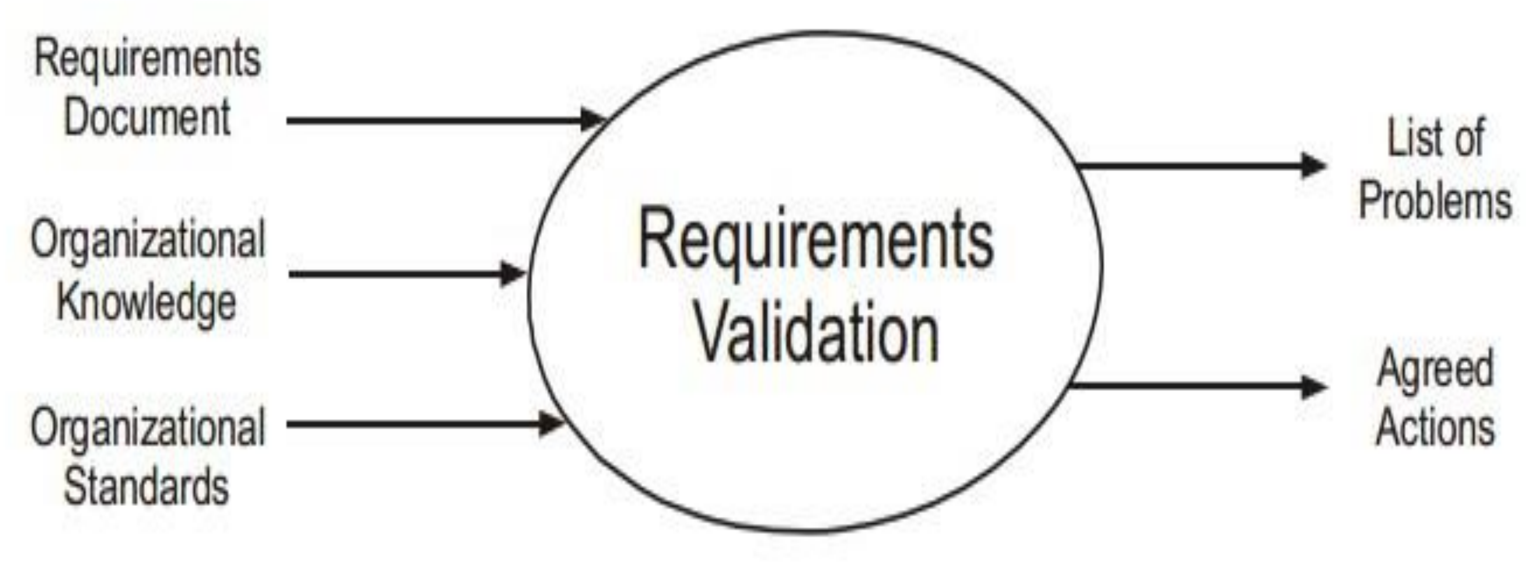


Fig: Requirement Validation

Step 6: Requirements Management

Why ??

- To understand and control changes to system requirements.

Advantages of requirements management

Better control of complex projects:

- ✓ Provides overview to development team with a clear understanding of what, when and why software is to be delivered.

Improves software quality:

- ✓ Ensures that the software performs according to requirements to enhance software quality.

Reduced project costs and delays:

- ✓ Minimizes errors early in the development cycle, as it is expensive to 'fix' errors at the later stages of the development cycle. As a result, the project costs also reduced.

Improved team communications:

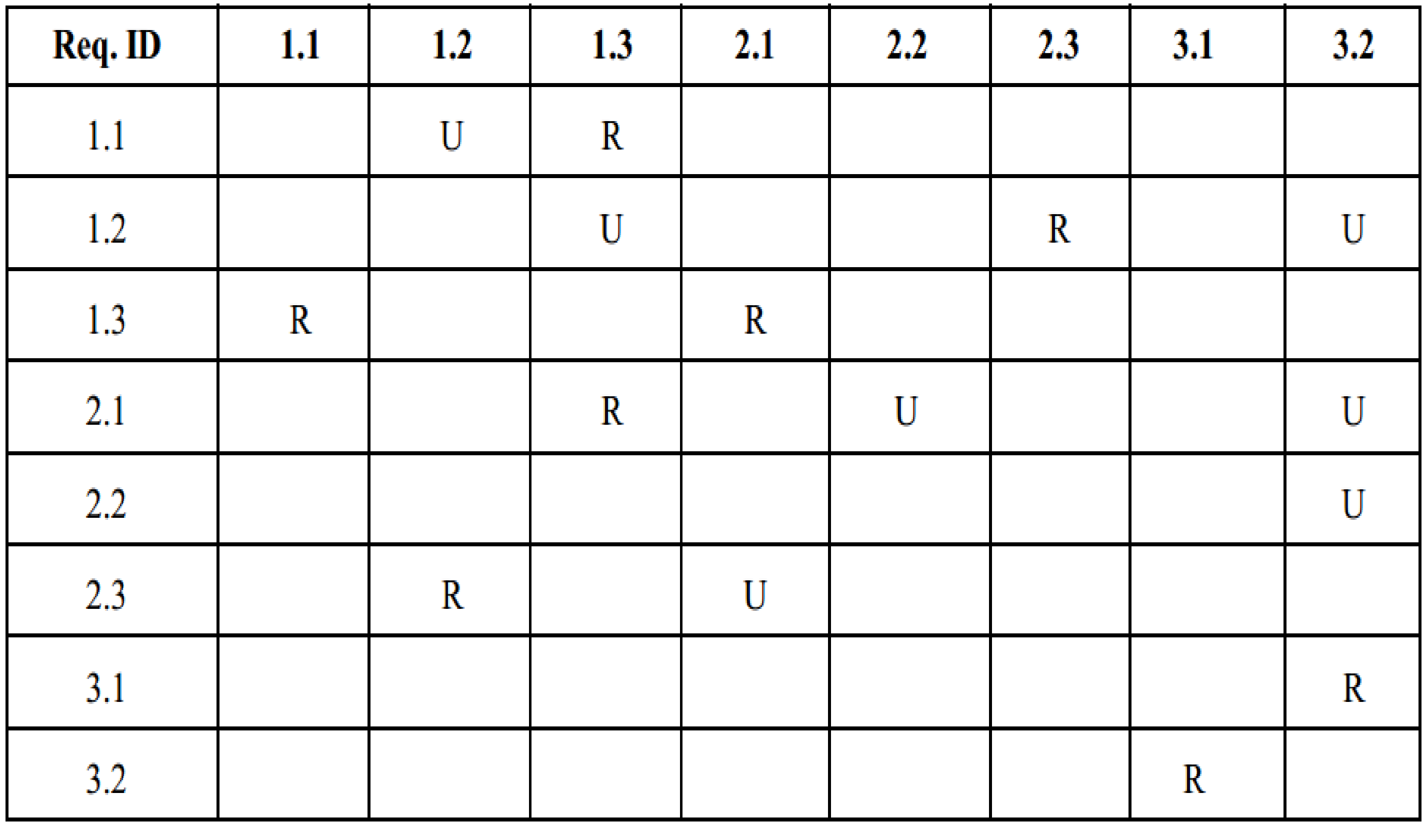
- ✓ Facilitates early involvement of users to ensure that their needs are achieved.

Requirements Management Process

- Requirements management starts with *planning*
- Then, each requirement is assigned a unique ‘identifier’ so that it can be crosschecked by other requirements. Once requirements are *identified*, requirements tracing is performed.
- The objective of *requirement tracing* is to ensure that all the requirements are well understood and are included in test plans and test cases.
- Traceability information is stored in a traceability matrix, which relates requirements to stakeholders or design module. Traceability matrix refers to a table that correlates requirements.

Traceability Matrix

- A Traceability Matrix is a document that establishes a relationship between different stages of the development life cycle.
- It helps ensure that requirements are met through the various phases of a project and provides a means to track changes and updates.
- The key purpose of a traceability matrix is to trace and manage the flow of requirements from their origin to the final implementation.



Req. ID	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2
1.1		U	R					
1.2			U			R		U
1.3	R			R				
2.1			R		U			U
2.2								U
2.3		R		U				
3.1								R
3.2							R	

U → dependency

R → weaker Relationship

Requirements change management

- It is used when there is a request or proposal for a change to the requirements.

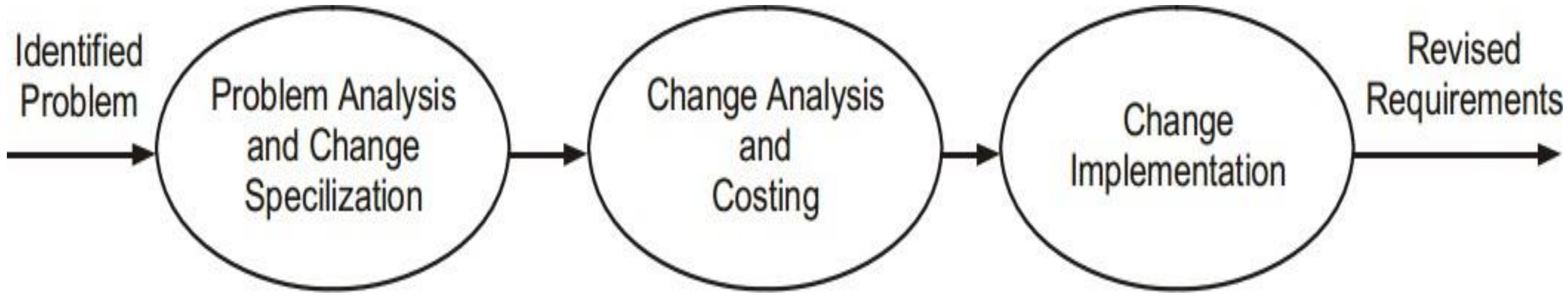

Identified Problem
Problem Analysis and Change Specilization
Change Analysis and Costing
Change Implementation
Revised Requirements

Fig: Requirement change management

Software prototyping

- A prototype is a version of a software product developed in the early stages of the product's life cycle for specific experimental purposes.
- A prototype enables you to fully understand how easy or difficult it will be to implement some of the features of the system.
- It also can give users a chance to comment on the usability and usefulness of the user interface design and it lets you assess the fit between the software tools selected, the functional specification, and the users' needs.

Software prototyping

- The purpose of the prototype review is threefold:

1. To demonstrate that the prototype has been developed according to the specification and that the final specification is appropriate.

2. To collect information about errors or other problems in the system, such as user interface problems that need to be addressed in the intermediate prototype stage.

3. To give management and everyone connected with the project the first (or it could be second or third …) glimpse of what the technology can provide.

System Modeling

- System modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system.
- It is about representing a system using some kind of graphical notation, which is now almost always based on notations in the Unified Modeling Language (UML).
- Models help the analyst to understand the functionality of the system; they are used to communicate with customers.

System Modeling

- Models can explain the system from different perspectives:
- An external perspective, where you model the context or environment of the system.
- An interaction perspective, where you model the interactions between a system and its environment, or between the components of a system.
- A structural perspective, where you model the organization of a system or the structure of the data that is processed by the system.
- A behavioral perspective, where you model the dynamic behavior of the system and how it responds to events.

System Modeling

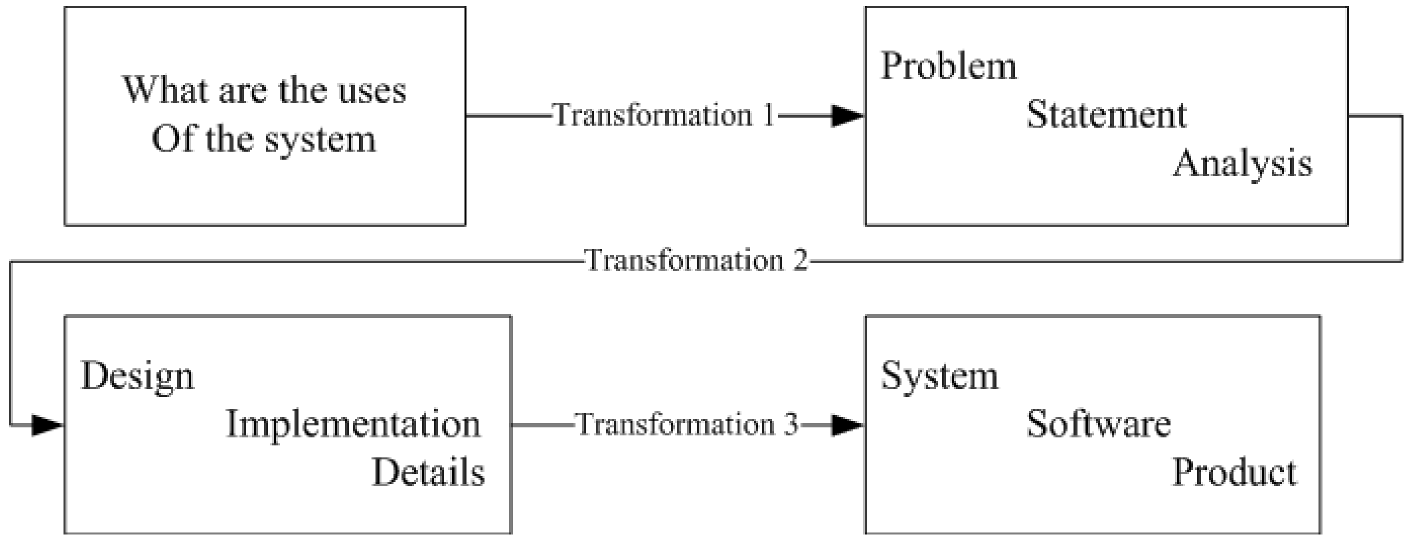
Five types of UML diagrams that are the most useful for system modeling:

- ***Activity diagrams***, which show the activities involved in a process or in data ***processing.***
- ***Use case diagrams***, which show the interactions between a system and its environment.
- ***Sequence diagrams***, which show interactions between actors and the system and between system components.
- ***Class diagrams***, which show the object classes in the system and the associations between these classes.
- ***State diagrams***, which show how the system reacts to internal and external events.

Object oriented Software development Cycle

- Object oriented systems development is a way to develop software by *building self – contained (independent) modules or objects that can be easily replaced, modified and reused.*
- In an object–oriented environment, software or application is a collection of discrete/separate objects that encapsulate their data as well as the functionality.
- It works by allocating tasks among the objects of the application.
- The software development approach for object oriented modeling goes through following stages:
- ***Analysis, Design, Implementation and Testing***

Object Oriented Software Development Cycle


What are the uses Of the system
Transformation 1
Problem Statement Analysis
Transformation 2
Design Implementation Details
Transformation 3
System Software Product

Transformation 1 (analysis):

- Translates the *users' needs into system requirements & responsibilities*.

Transformation 2 (design):

- Begins with a problem statement and ends with a detailed design that can be transformed into an operational system.

Transformation 3 (implementation)

- Refines the detailed design into the system deployment that will satisfy the users' needs.
- It represents embedding s/w product within its operational environment.