Tribhuvan University
Institute of Engineering
Pulchowk Campus
Department of Electronics and Computer Engineering

OBJECT ORIENTED SOFTWARE ENGINEERING

Chapter One

Introduction to software and software engineering

by
Santosh Giri
Lecturer, IOE, Pulchowk Campus.

Course Outline → 5 hours, 9 Marks

1.1. Introduction to software and software engineering

1.2. Software process and software process model

1.3. Agile software development

1.4.Requirement engineering process

1.5. System modeling

1.6. Software prototyping

1.7. Object oriented software development

Software and software engineering

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.**

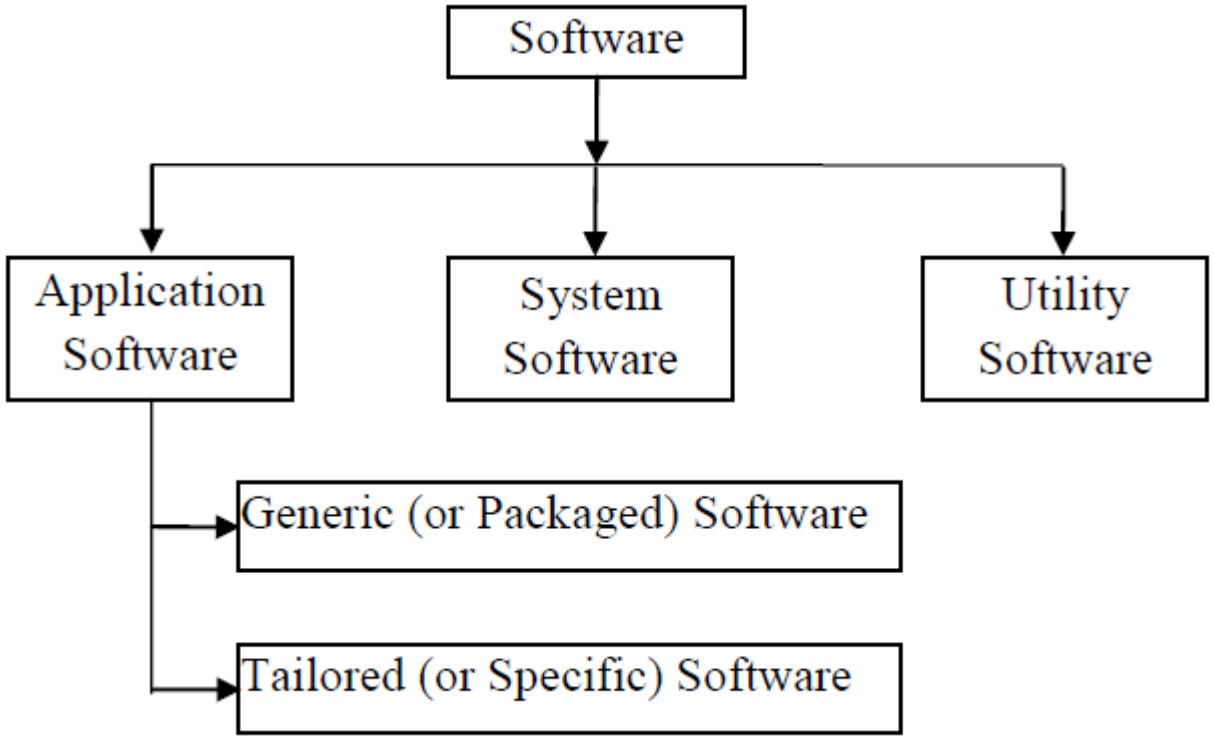
- **The other term 'Engineering'** is all about developing some quality products by using some well-defined, scientific principles and standard within the specified budget and time.

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.

Software types



1. Application Software

- It can be divided into Generic and Specific

a. Generic (or Packaged) 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, Adobe Reader, 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: 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 & software process model

1.2. 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**.

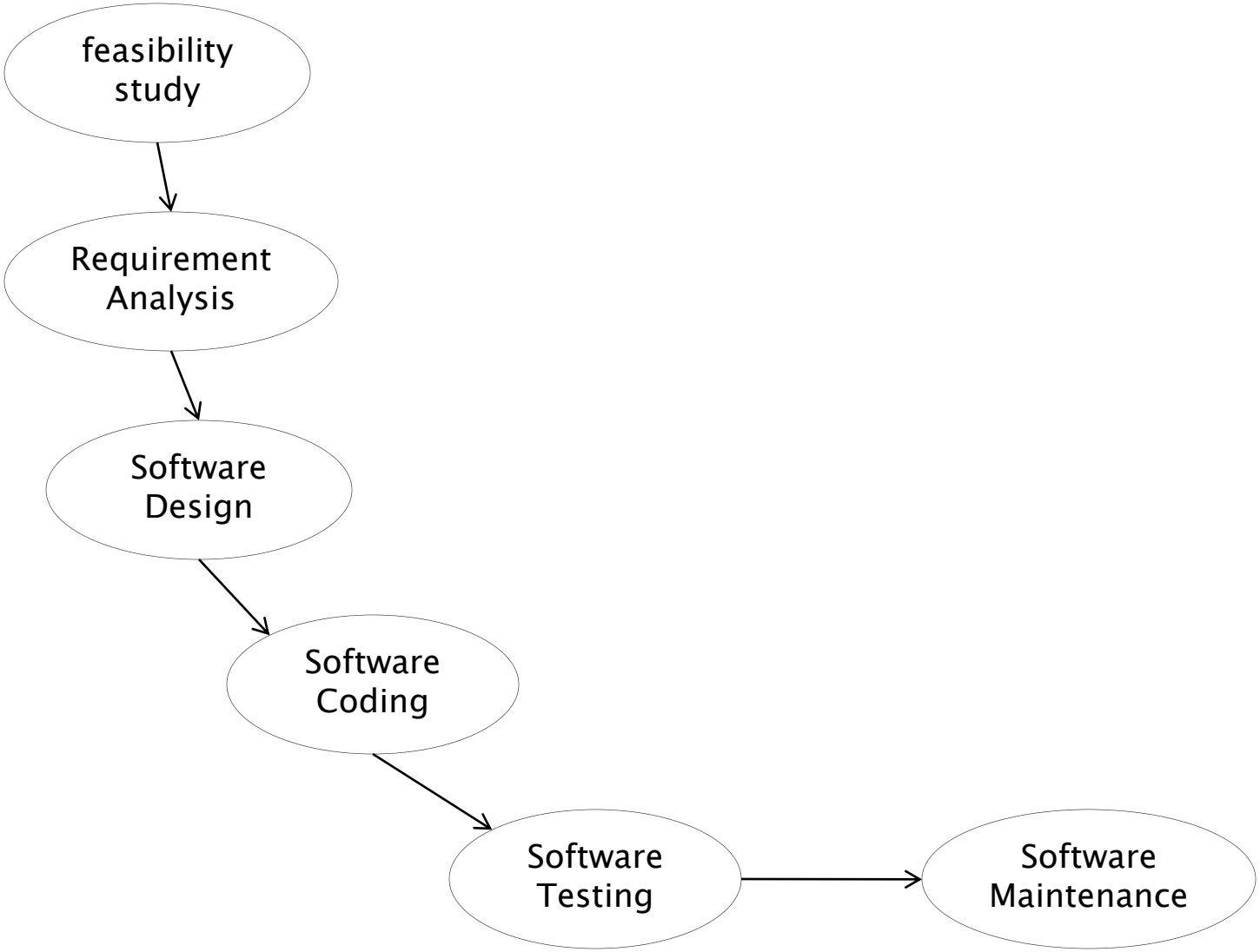


Fig 1 software development process

In fig 1,

- These phases **work in top to bottom** approach.
- The phases take inputs from the previous phases, add features, and then produce outputs.

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:
   - 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:

- 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:

- 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), if you are under the project of Library Management system.

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.

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.

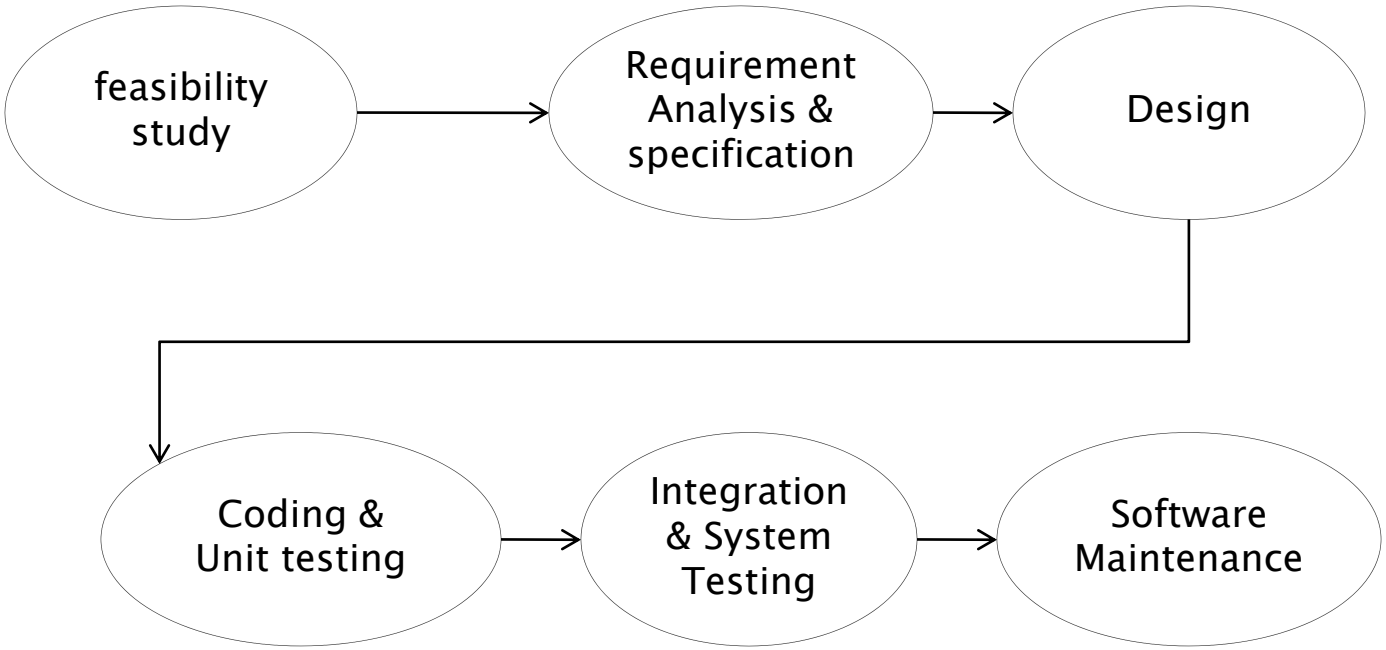
from a technical point of view:

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

software process model cont…

1. Waterfall model

Waterfall Model



i. Feasibility study

- √ Technical
- √ Financial
- √ Time 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.

Waterfall model

Advantages	Disadvantages
▪ Relatively simple to understand. ▪ Each phase of development proceeds sequentially. ▪ Allows managerial control where a schedule with deadlines is set for each stage of development. ▪ Helps in controlling schedules, budgets, and documentation.	▪ Requirements need to be specified before the development proceeds. ▪ The changes of requirements in later phases of the waterfall model cannot be done. This implies that once an application is in the testing phase, it is difficult to incorporate changes at such a late phase. ▪ No user involvement and working version of the software is available when the software is developed. ▪ Does not involve risk management. ▪ Assumes that requirements are stable and are frozen across the project span.

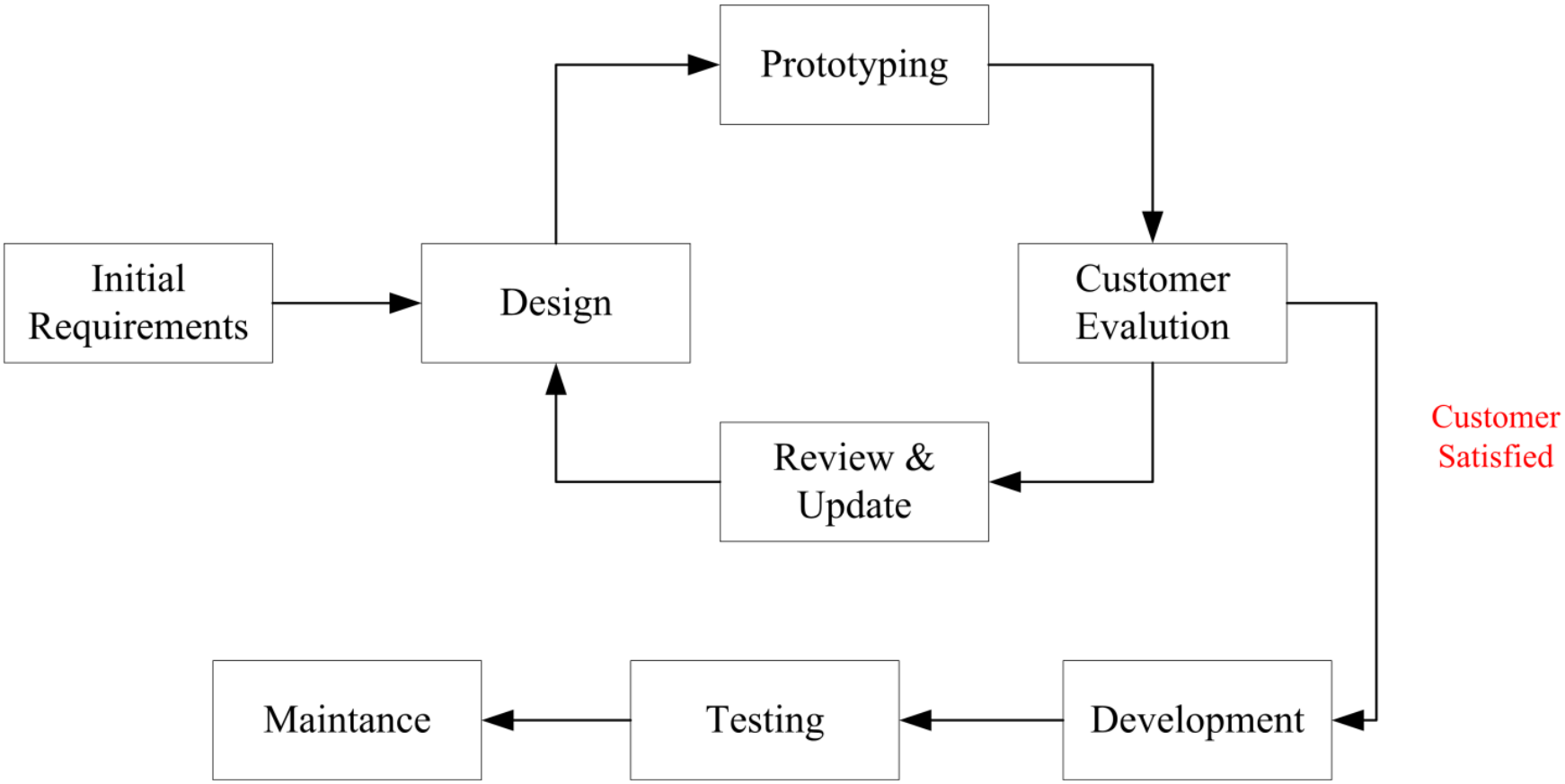
2. Prototype model

b. prototype model

- The prototyping model is applied when there is an absence of detailed information regarding input and output requirements in the software.

- It is used if the requirements are not preciously specified.

- Prototyping model increases flexibility of the development process by allowing the user to interact and experiment with a working representation of the product known as **prototype**. A prototype gives the user an actual feel of the system.



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.

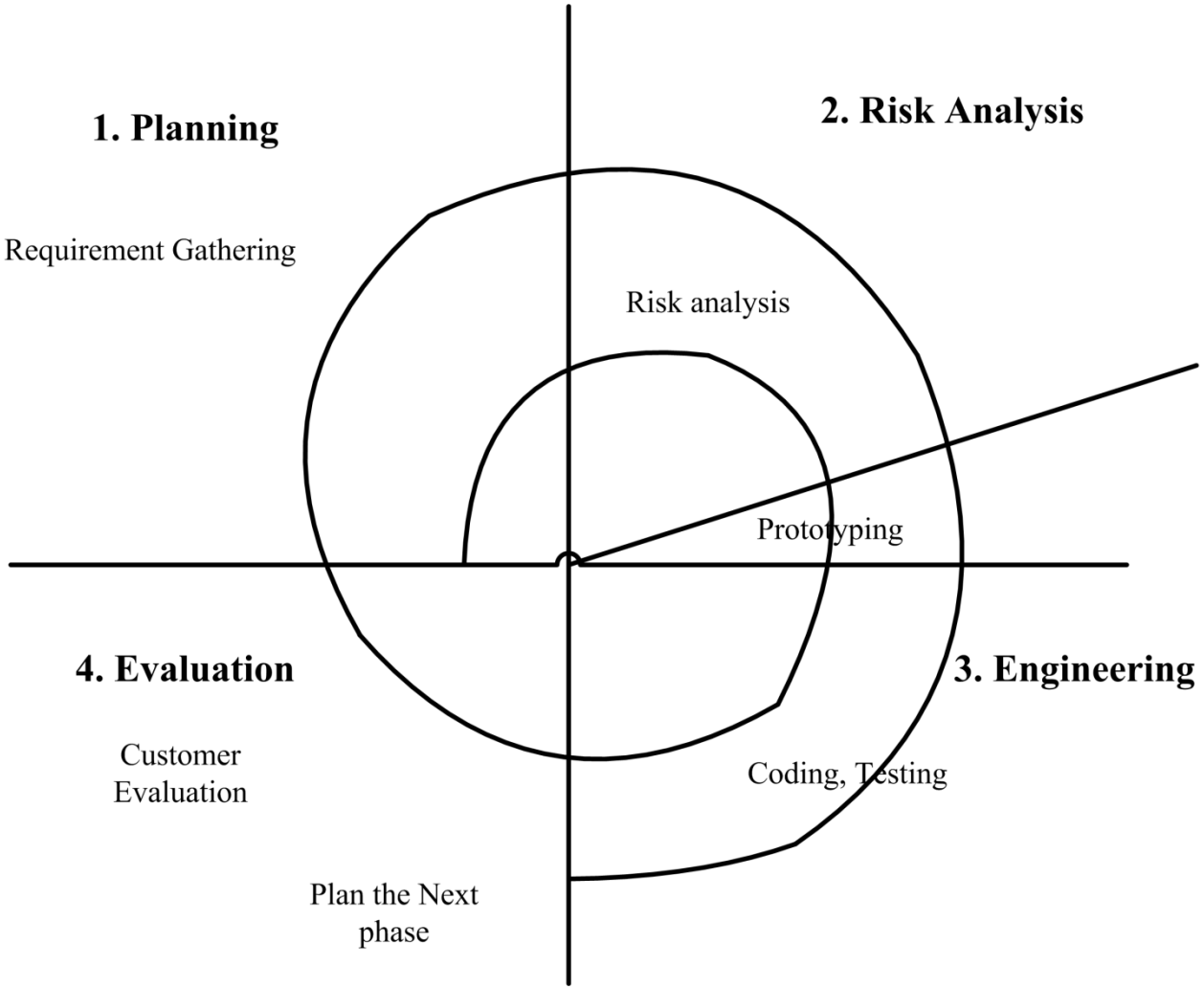
Advantages	Disadvantages
▪ Provides a working model to the user early in the process, enabling early assessment and increasing user confidence. ▪ Developer gains experience and insight by developing a prototype, thereby resulting in better implementation of requirements. ▪ Prototyping model serves to clarify requirements, which are not clear, hence reducing ambiguity and improving communication between developer and user. ▪ There is a great involvement of users in software development. Hence, the requirements of the users are met to the greatest extent. ▪ Helps in reducing risks associated with the project.	▪ If the user is not satisfied by the developed prototype, then a new prototype is developed. This process goes on until a perfect prototype is developed. Thus, this model is time consuming and expensive. ▪ Developer looses focus of the real purpose of prototype and compromise with the quality of the product. For example, they apply some of the inefficient algorithms or inappropriate programming languages used in developing the prototype. ▪ Prototyping can lead to false expectations. It often creates a situation where user believes that the development of the system is finished when it is not. ▪ The primary goal of prototyping is rapid development, thus, the design of system can suffer as it is built in a series of layers without considering integration of all the other components.

3. Spiral Model

C. Spiral Model

In 1980's Boehm introduced a process model known as spiral model. The spiral model comprises of activities organized in a spiral, which has many cycles. This model combines the features of prototyping model and waterfall model and is advantageous for large, complex and expensive projects which involves high risk.

C. Spiral Model



1. Each cycle of the **first quadrant** commences with identifying the goals for that cycle. In addition, it determines other alternatives, which are possible in accomplishing those goals.

2. Next step in the cycle evaluates alternatives based on objectives and constraints. This process identifies the project risks. Risk signifies that there is a possibility that the objectives of the project cannot be accomplished. If so the formulation of a cost effective strategy for resolving risks is followed. the strategy, which includes prototyping, simulation, benchmarking..

3. The development of the software depends on remaining risks. The third quadrant develops the final software while considering the risks that can occur. Risk management considers the time and effort to be devoted to each project activity, such as planning, configuration management, quality assurance, verification, and testing.

4. The last quadrant plans the next step, and includes planning for the next prototype and thus,comprises of requirements plan, development plan, integration plan, and test plan

Advantages	Disadvantages
▪ Avoids the problems resulting in risk-driven approach in the software. ▪ Specifies a mechanism for software quality assurance activities. ▪ Spiral model is utilised by complex and dynamic projects. ▪ Re-evaluation after each step allows changes in user perspectives, technology advances or financial perspectives. ▪ Estimation of budget and schedule gets realistic as the work progresses.	▪ Assessment of project risks and its resolution is not an easy task. ▪ Difficult to estimate budget and schedule in the beginning, as some of the analysis is not done until the design of the software is developed.

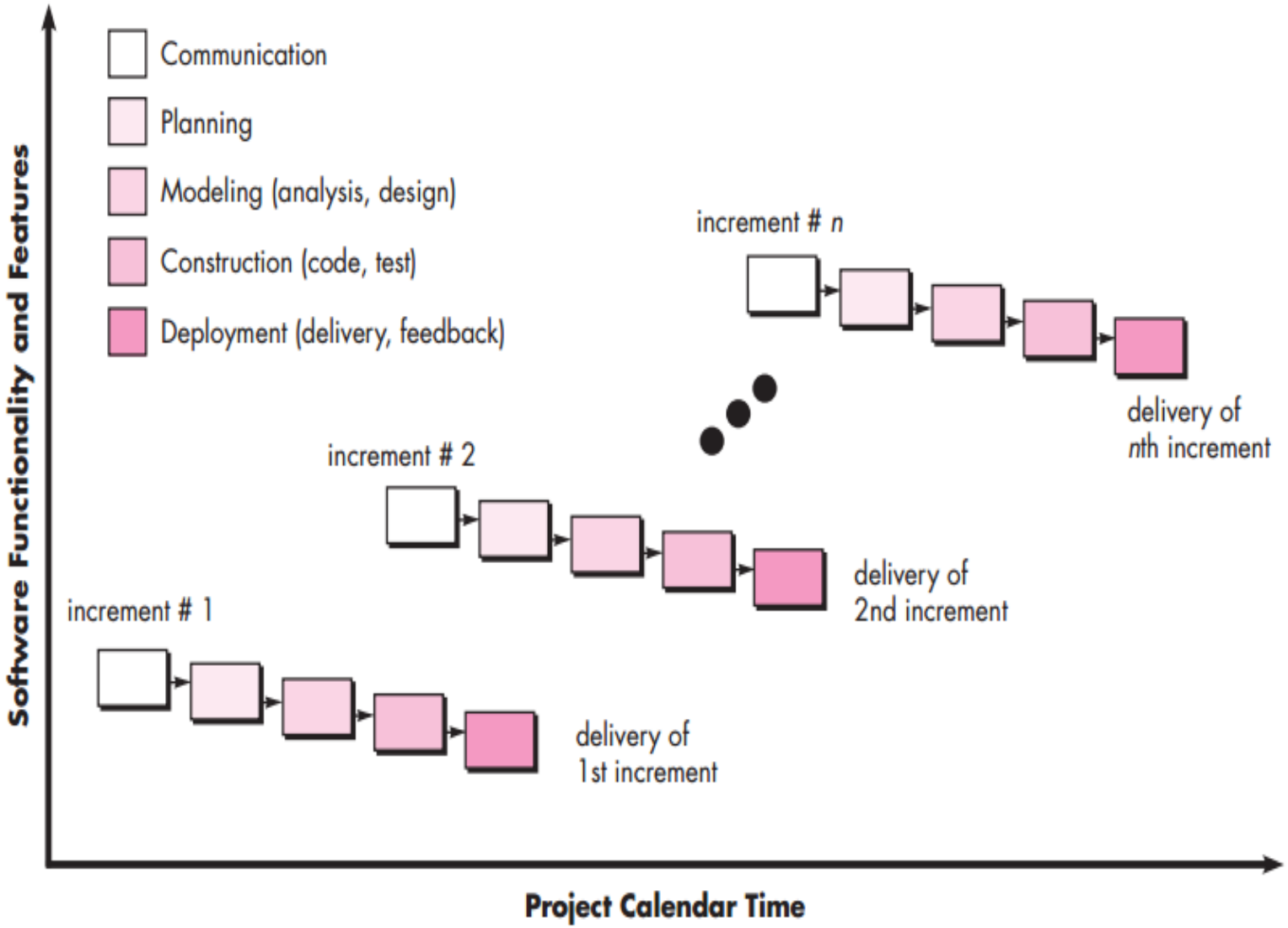
4. Evolutionary model

d. Evolutionary model

- An Evolutionary model breaks up an overall solution into increments of functionality and develops each increment individually.

- The evolution model **divides** the development cycle into **smaller, "Incremental Waterfall Model"** in which users are able to get access to the product at the end of each cycle.

- The users provide feedback on the product for planning stage of the next cycle and the development team responds, often by changing the product, plans or process.


Communication
Planning
Modeling (analysis, design)
Construction (code, test)
Deployment (delivery, feedback)
increment # n
delivery of
nth increment
increment # 2
delivery of
2nd increment
increment # 1
delivery of
1st increment
Software Functionality and Features
Project Calendar Time

5. RAD model

RAD model

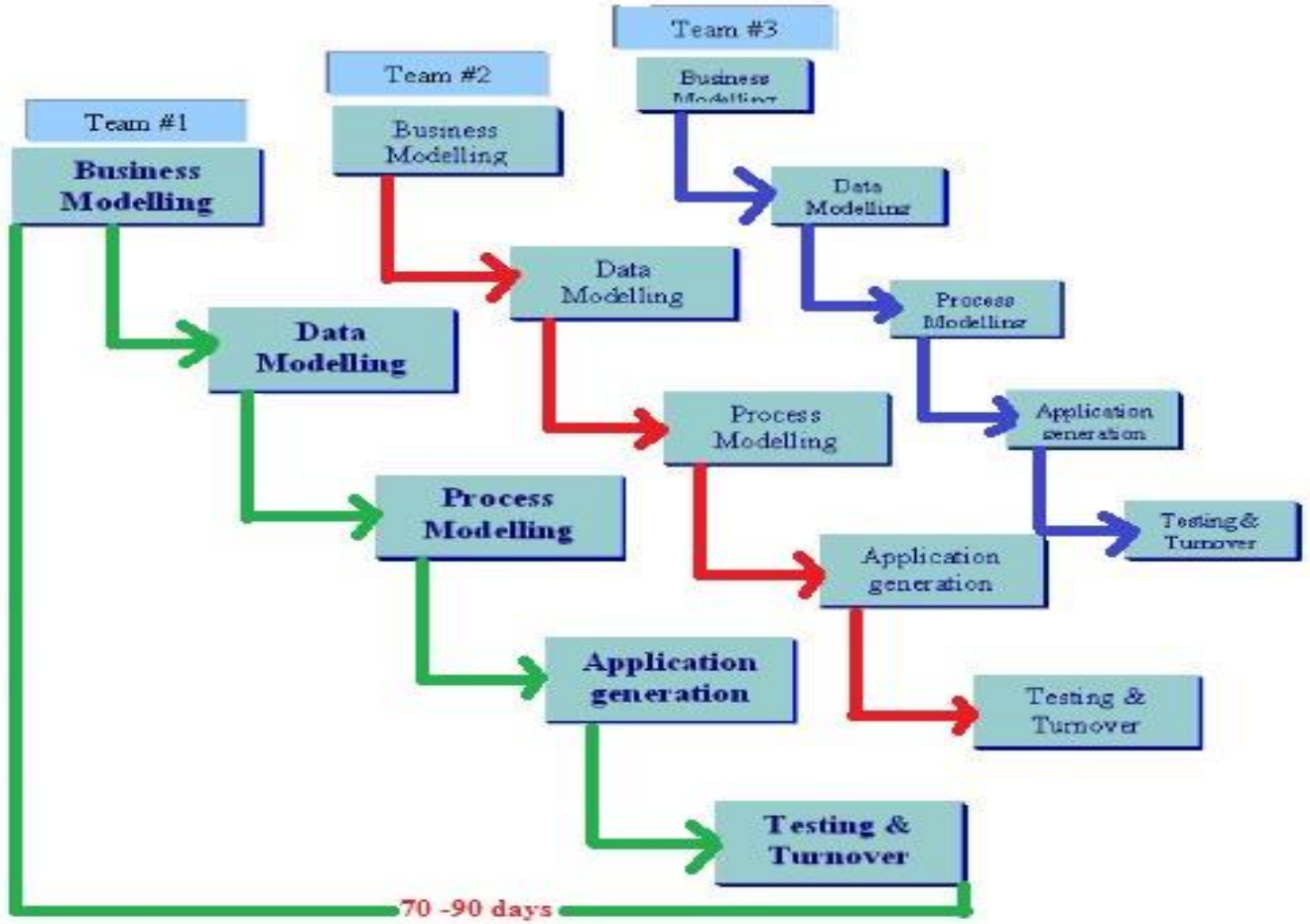
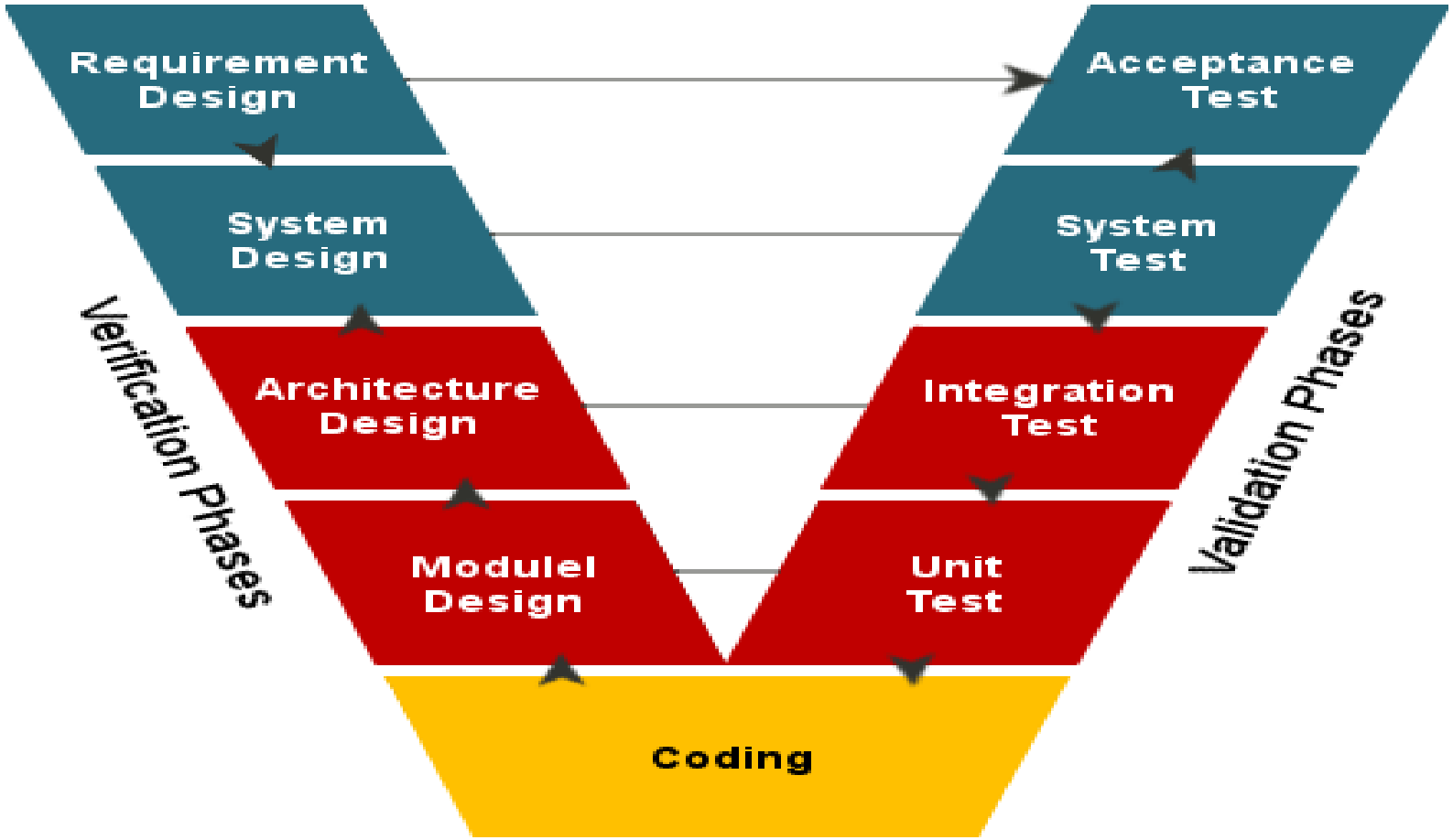


Fig:- RAD (Rapid Application Development) Model

6. V model

V model


Requirement Design
System Design
Architecture Design
Modulel Design
Coding
Unit Test
Integration Test
System Test
Acceptance Test
Verification Phases
Validation Phases

5. Agile Software development

Presentation topic → Group 1

Requirement Engineering Process

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.

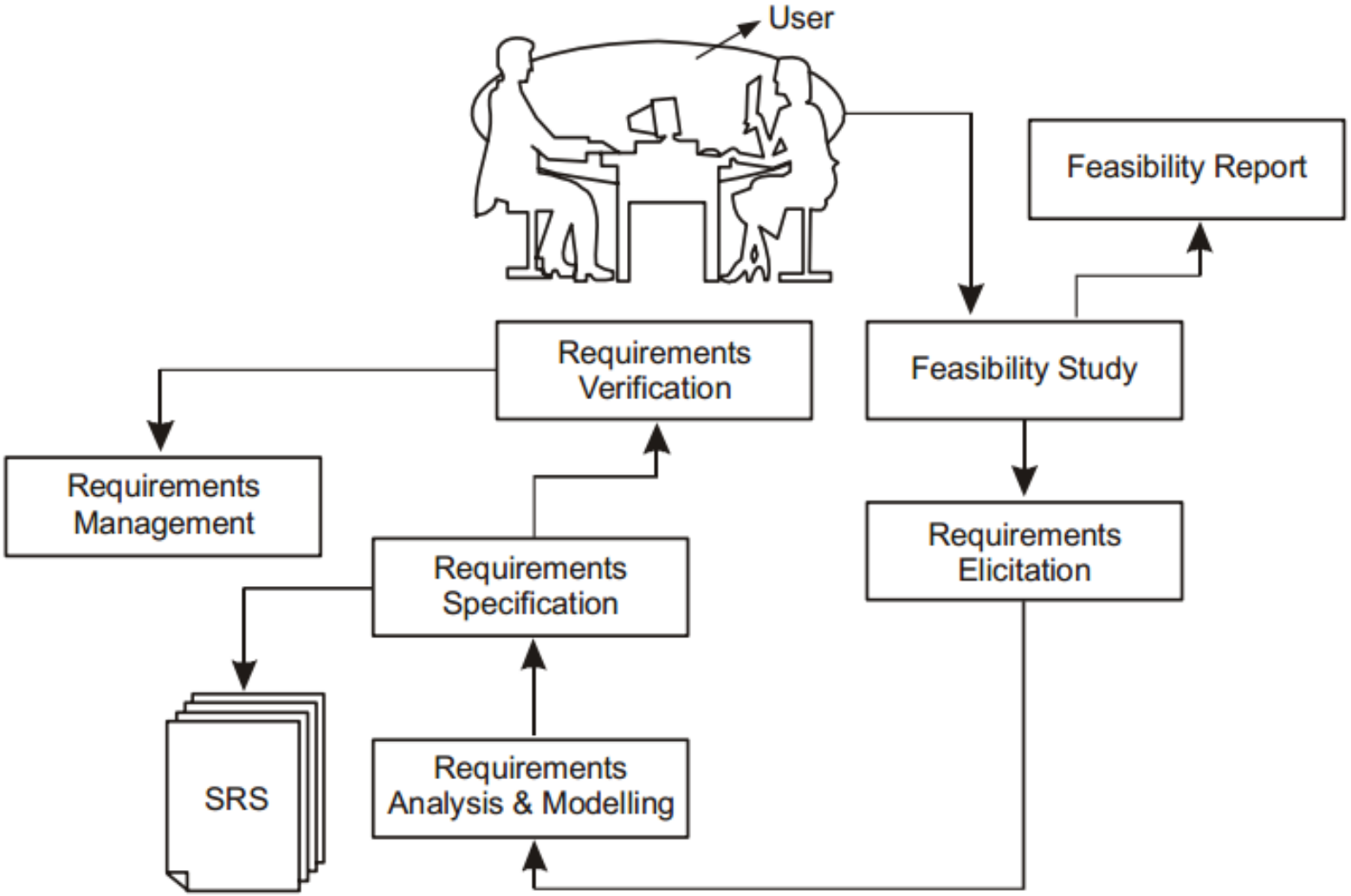


Fig: Requirement engineering process

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. *f*
  - ✓ 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.

4.Quality function deployment (QFD) →Assignment (3)

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.

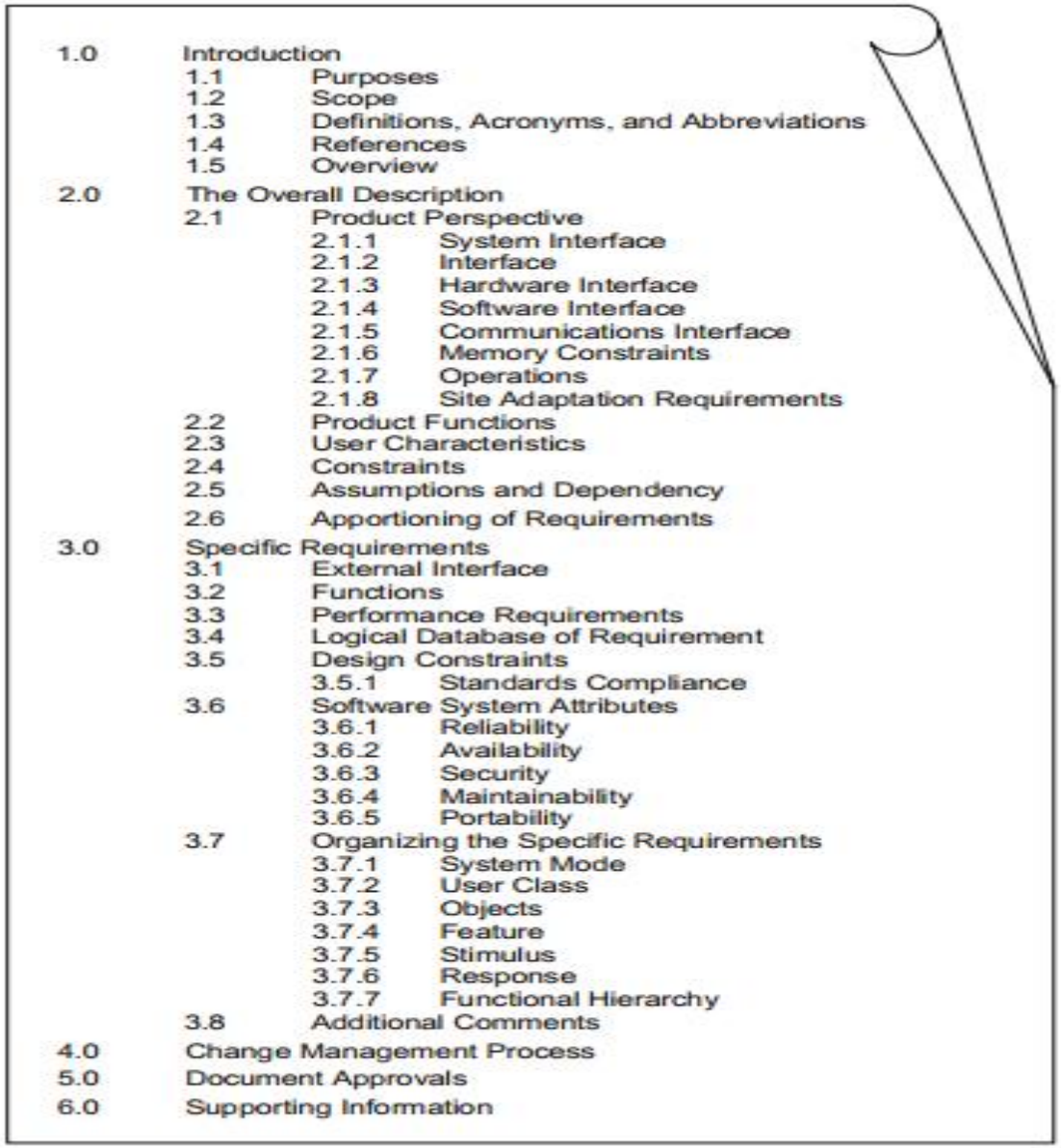


Fig : SRS Document template

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.

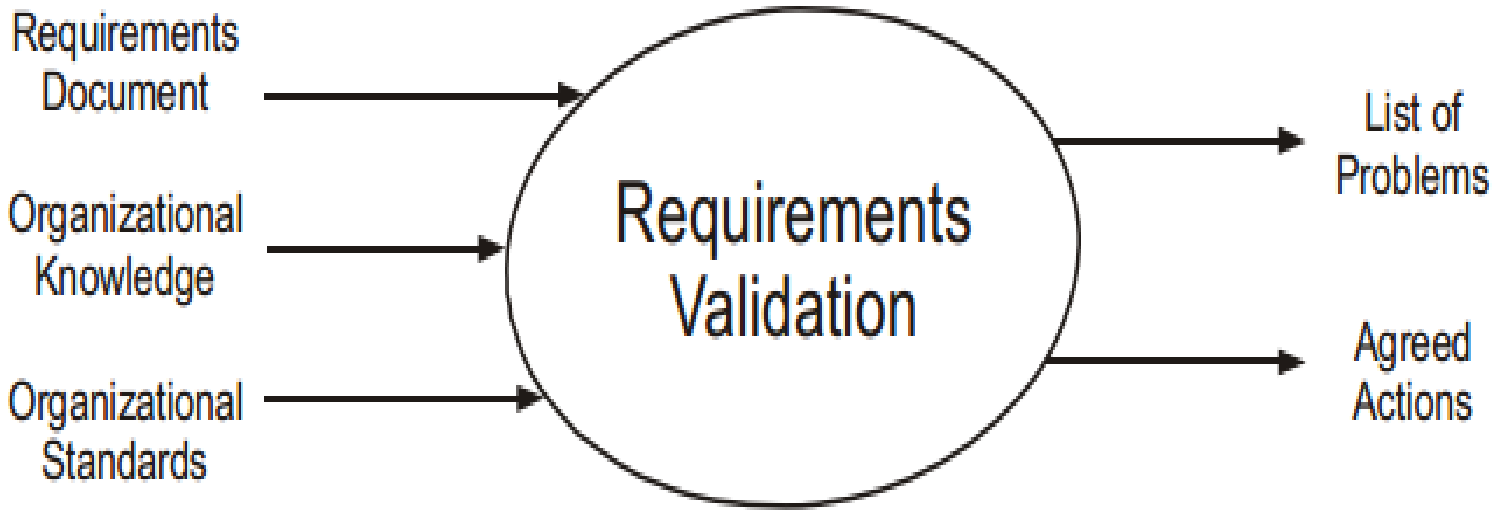


Fig: Requirement Validation

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.

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.

Fig: Requirement change management

Software Prototyping

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

- 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:

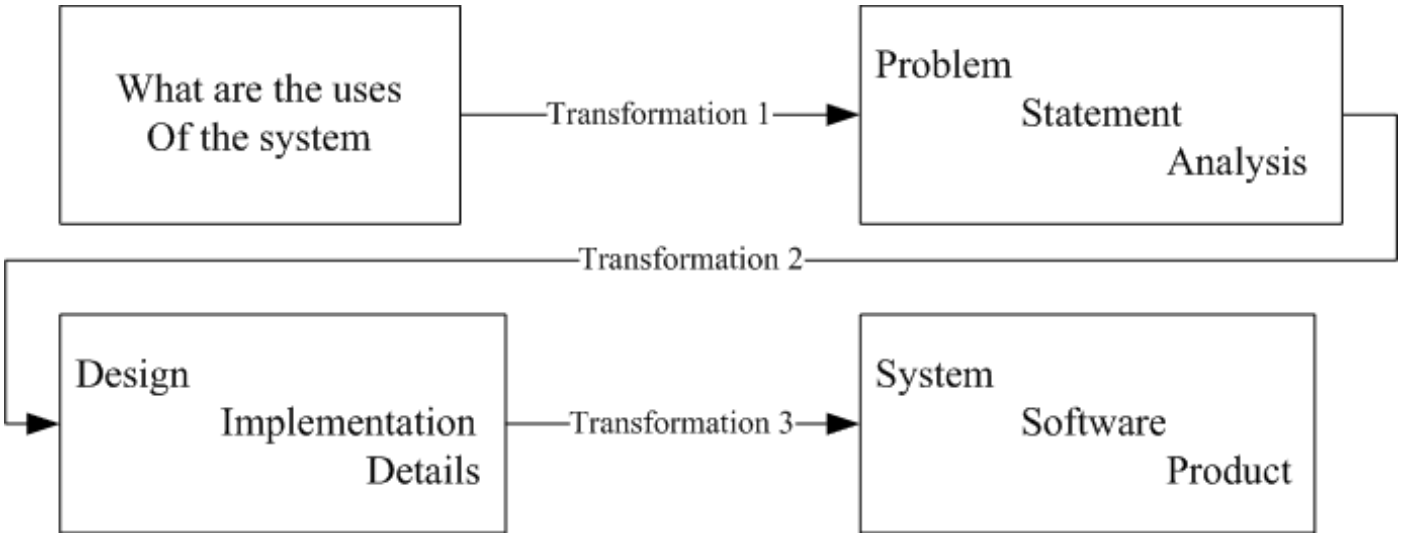
- Activity diagrams, which show the activities involved in a process or in data processing.
- Use case diagrams, which show the interactions between a system and its environment.
- Sequence diagrams, which show interactions between actors and the system and between system components.
- Class diagrams, 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

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



- **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.