Dr. SNS RAJALAKSHMI

COLLEGE OF ARTS AND SCIENCE

(AUTONOMOUS)

Accredited by NAAC (Cycle-III) with 'A+' Grade

DEPARTMENT OF COMPUTER APPLICATIONS (PG)

Course Name : SOFTWARE PROJECT MANAGEMENT

Class : I MCA

Semester : I

Year : 2022-2023

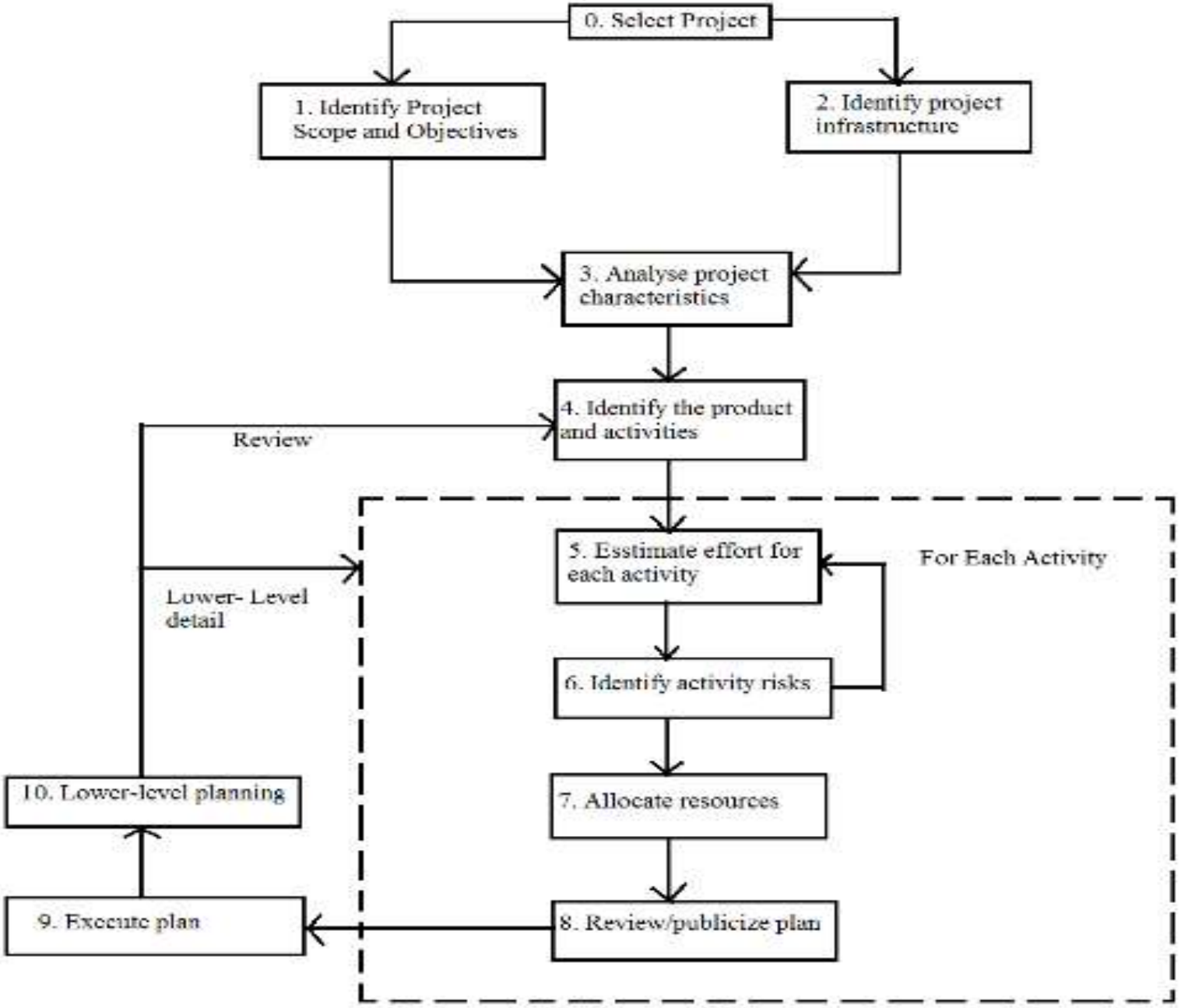
UNIT-II

Software Project Management

Topic: STEP WISE : An overview of project planning

Step Wise Project Planning

- Planning is the most difficult process in project management. The framework described is called the Stepwise method to help to distinguish it from other method.


0. Select Project
1. Identify Project Scope and Objectives
2. Identify project infrastructure
3. Analyse project characteristics
4. Identify the product and activities
Review
5. Esstimate effort for each activity
For Each Activity
Lower- Level detail
6. Identify activity risks
10. Lower-level planning
7. Allocate resources
9. Execute plan
8. Review/publicize plan

Step 0: Select Project

Step 1: Identify project scope and objectives

Step 1.1 : Identify objectives and practical measures of the effectiveness in meeting those objectives

Step 1:2 : Establish a project authority

Step 1.3 : Stakeholder analysis - identify all stakeholders in the project and their interests.

Step1.4: Modify objectives in the light of stakeholder analysis.

Step 1.5 : Establish methods of communication with all parties.

Step 2 : Identify project infrastructure

Step 2.2 : Identify installation standard and procedures

Step 2.3 : Identify project team organization

Step 3 : Analyse project characteristics

Step 3.1 : Distinguish the project as either objectives- or product-driven.

Step 3.2 : Analyse other project characteristics

Step 3.3 : Identify high-level project risks

Step 3.4 : Take into account use requirements concerning implementation

Step 3.5 : Select development methodology and life-cycle approach

Step 3.6 : Review overall resource estimates

Step 4 : Identify project products and activities

Step 4.1 : Identify and describe project products

Step 4.2 : Document generic product flows

Step 4.3 : Recognize product instances

Step 4.4 : Produce ideal activity network

Step 4.5 : Modify the ideal to take into account need for stages and checkpoints

Step 5 : Estimate effort for each activity

Step 5.1 : Carry out bottom-up estimates

- distinguish carefully between effort and elapsed time

Step 5.2 : Revise plan to create on trollable activities

- breakup very long activities into a series of smaller ones
- bundle up very short activities

Step 6 : Identify activity risks

Step 6.1 : Identify and quantify activity based risks

- damage if risk occurs
- likelihood if risk occurring

Step 6.2 : Plan risk reduction and contingency measures

- risk reduction : activity to stop risk occurring
- contingency : action if risk does occurs

Step 6.3 : Adjust overall plans and estimates to take account of risks

Step 7 : Allocate resources

Step 7.1 : Identify and allocate resources

Step 7.2 : Revise plans and estimates to take into account resource constraints

Step 8 : Review/ Publicize plans

Step 8.1 : Review quality aspects of the project plan

Step 8.2 : Documenter plans and obtain agreement

Step 9 and 10 : Execute plan. Lower levels of planning

Process Models

What is a Software Process Model?

Software Processes is a coherent set of activities for specifying, designing, implementing and testing software systems. A software process model is an abstract representation of a process that presents a description of a process from some particular perspective.

There are many different software processes but all involve:

- Specification – defining what the system should do;
- Design and implementation – defining the organization of the system and implementing the system;
- Validation – checking that it does what the customer wants;
- Evolution – changing the system in response to changing customer needs.

Types of Software Process Model

Software processes, methodologies and frameworks range from specific prescriptive steps that can be used directly by an organization in day-to-day work, to flexible frameworks that an organization uses to generate a custom set of steps tailored to the needs of a specific project or group. In some cases a “sponsor” or “maintenance” organization distributes an official set of documents that describe the process.

Software Process and Software Development Lifecycle Model

One of the basic notions of the software development process is SDLC models which stands for Software Development Life Cycle models. There are many development life cycle models that have been developed in order to achieve different required objectives.

The most used, popular and important SDLC models are given below:

- Waterfall model
- V model
- Incremental model
- RAD model
- Agile model
- Iterative model
- Spiral model
- Prototype model

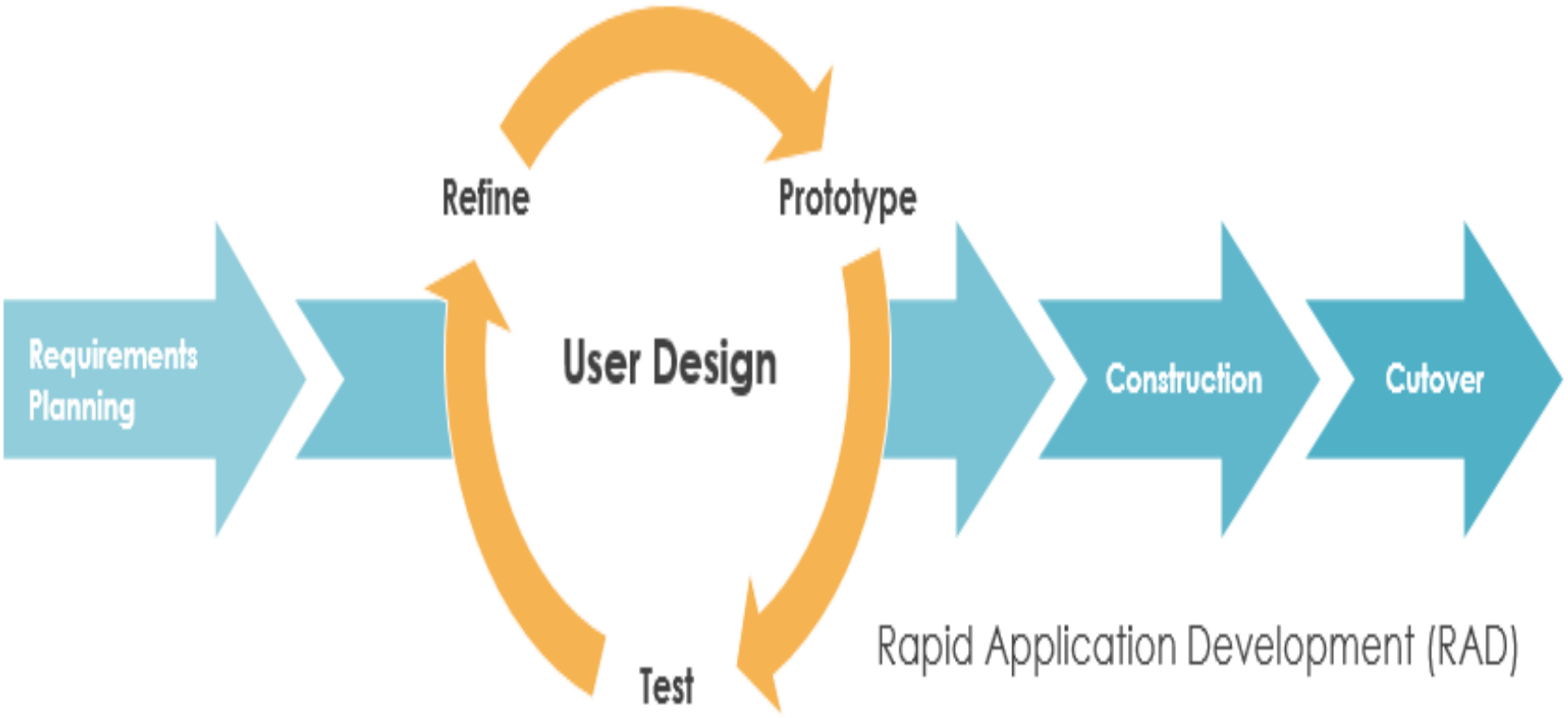
Rapid Application Development

- **RAD model**

- The Rapid Application Development Model was first proposed by IBM in 1980's. The critical feature of this model is the use of powerful development tools and techniques.
- A software project can be implemented using this model if the project can be broken down into small modules wherein each module can be assigned independently to separate teams. These modules can finally be combined to form the final product.

Rapid Application Development

- **RAD model**
- RAD is an incremental prototyping approach to software development that end users can produce better feedback when examining a live system, as opposed to working strictly with documentation. It puts less emphasis on planning and more emphasis on an adaptive process.
- RAD may resulted in a lower level of rejection when the application is placed into production, but this success most often comes at the expense of a dramatic overruns in project costs and schedule. RAD approach is especially well suited for developing software that is driven by user interface requirements. Thus, some GUI builders are often called rapid application development tools.


Refine
Prototype
Requirements Planning
User Design
Construction
Cutover
Test
Rapid Application Development (RAD)

This model consists of 4 basic phases:

- **Requirements Planning –**

  It involves the use of various techniques used in requirements elicitation like brainstorming, task analysis, form analysis, user scenarios, FAST (Facilitated Application Development Technique), etc. It also consists of the entire structured plan describing the critical data, methods to obtain it and then processing it to form final refined model.

- **User Description –**

  This phase consists of taking user feedback and building the prototype using developer tools. In other words, it includes re-examination and validation of the data collected in the first phase. The dataset attributes are also identified and elucidated in this phase.

This model consists of 4 basic phases:

- **Construction –**

  In this phase, refinement of the prototype and delivery takes place. It includes the actual use of powerful automated tools to transform process and data models into the final working product. All the required modifications and enhancements are too done in this phase.

- **Cutover –**

  All the interfaces between the independent modules developed by separate teams have to be tested properly. The use of powerfully automated tools and subparts makes testing easier. This is followed by acceptance testing by the user.

- **Advantages –**
- Use of reusable components helps to reduce the cycle time of the project.
- Feedback from the customer is available at initial stages.
- Reduced costs as fewer developers are required.
- Use of powerful development tools results in better quality products in comparatively shorter time spans.
- The progress and development of the project can be measured through the various stages.
- It is easier to accommodate changing requirements due to the short iteration time spans.

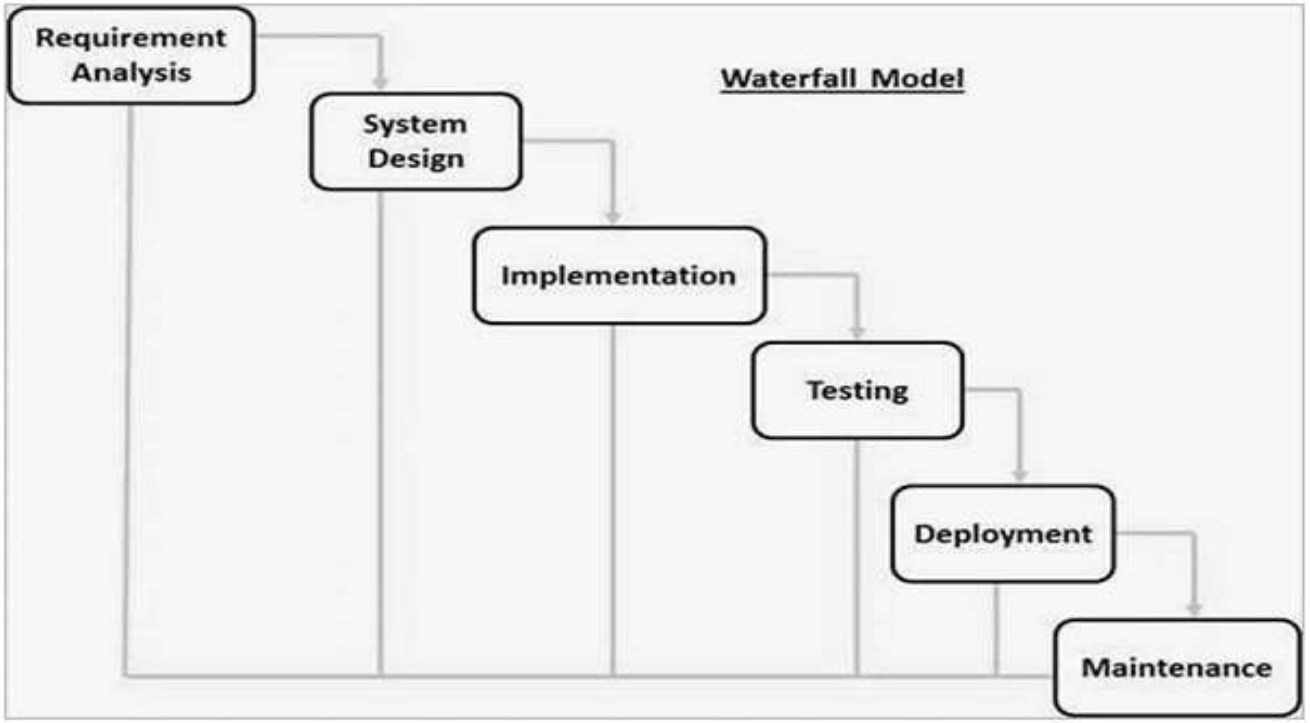
- **Disadvantages –**
- The use of powerful and efficient tools requires highly skilled professionals.
- The absence of reusable components can lead to failure of the project.
- The team leader must work closely with the developers and customers to close the project in time.
- The systems which cannot be modularized suitably cannot use this model.
- Customer involvement is required throughout the life cycle.
- It is not meant for small scale projects as for such cases, the cost of using automated tools and techniques may exceed the entire budget of the project.

Applications

- This model should be used for a system with known requirements and requiring short development time.
- It is also suitable for projects where requirements can be modularized and reusable components are also available for development.
- The model can also be used when already existing system components can be used in developing a new system with minimum changes.
- This model can only be used if the teams consist of domain experts. This is because relevant knowledge and ability to use powerful techniques is a necessity.
- The model should be chosen when the budget permits the use of automated tools and techniques required.

Waterfall Model

In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially



Each of these phases produces one or more documents that need to be approved before the next phase begins. However, in practice, these phases are very likely to overlap and may feed information to one another.

The waterfall model is easy to understand and follow. It doesn't require a lot of customer involvement after the specification is done. Since it's inflexible, it can't adapt to changes. There is no way to see or try the software until the last phase.

The waterfall model has a rigid structure, so it should be used in cases where the requirements are understood completely and unlikely to radically change.

The sequential phases in Waterfall model are −

- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

The sequential phases in Waterfall model are −

- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Waterfall Model - Application

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Waterfall Model - Advantages

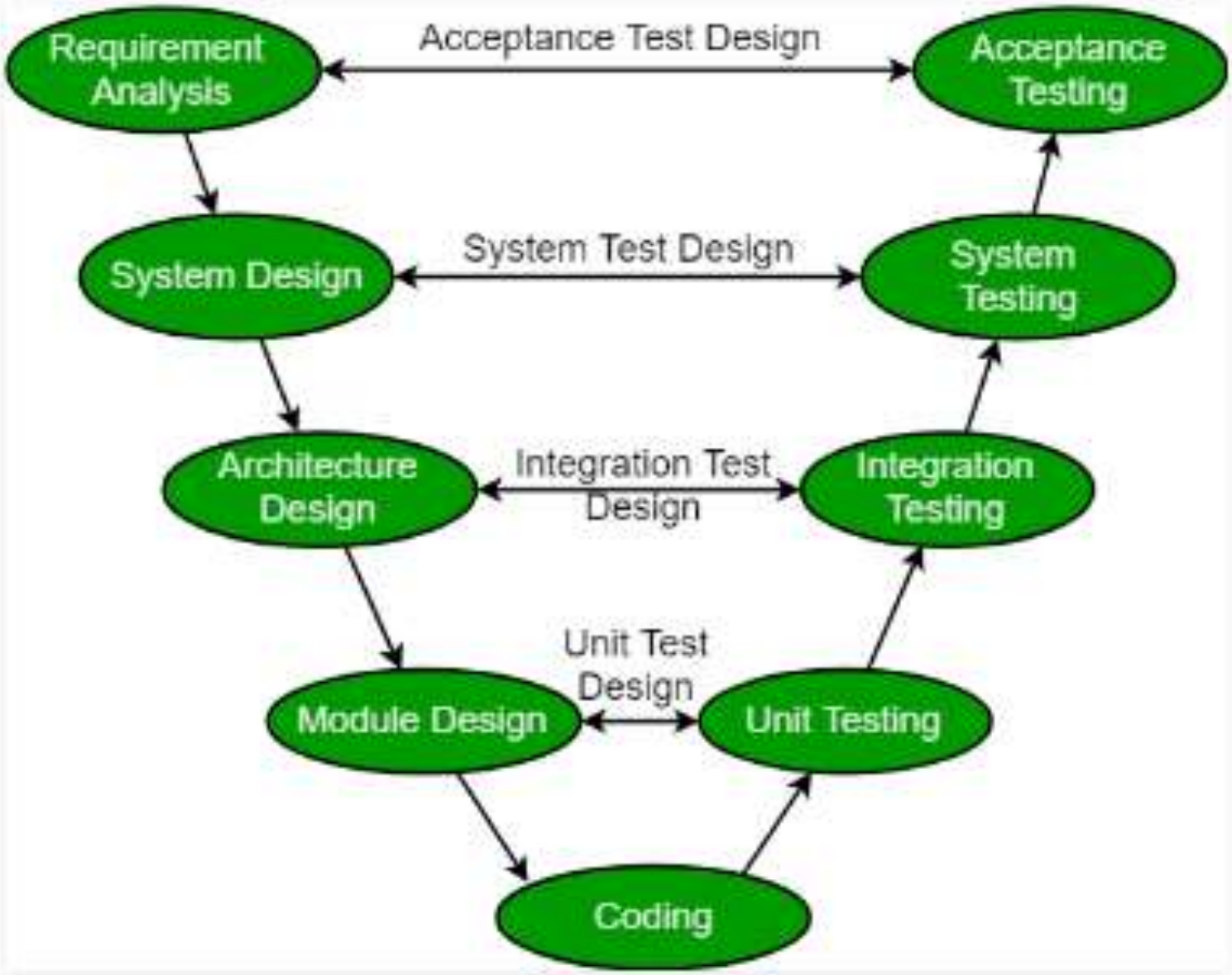
- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Waterfall Model – Disadvantages

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.

The V-process model

The V-model is a type of SDLC model where process executes in a sequential manner in V-shape. It is also known as Verification and Validation model. It is based on the association of a testing phase for each corresponding development stage. Development of each step directly associated with the testing phase. The next phase starts only after completion of the previous phase i.e. for each development activity, there is a testing activity corresponding to it.


Requirement Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Unit Test Design
Module Design
Unit Testing
Coding

- **Validation:** It involves dynamic analysis technique (functional, non-functional), testing done by executing code. Validation is the process to evaluate the software after the completion of the development phase to determine whether software meets the customer expectations and requirements.
- So V-Model contains Verification phases on one side of the Validation phases on the other side. Verification and Validation phases are joined by coding phase in V-shape. Thus it is called V-Model.

Design Phase:

- **Requirement Analysis:** This phase contains detailed communication with the customer to understand their requirements and expectations. This stage is known as Requirement Gathering.
- **System Design:** This phase contains the system design and the complete hardware and communication setup for developing product.
- **Architectural Design:** System design is broken down further into modules taking up different functionalities. The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood.
- **Module Design:** In this phase the system breaks dowm into small modules. The detailed design of modules is specified, also known as Low-Level Design (LLD).

Principles of V-Model:

- **Large to Small:** In V-Model, testing is done in a hierarchical perspective, For example, requirements identified by the project team, create High-Level Design, and Detailed Design phases of the project. As each of these phases is completed the requirements, they are defining become more and more refined and detailed.
- **Data/Process Integrity:** This principle states that the successful design of any project requires the incorporation and cohesion of both data and processes. Process elements must be identified at each and every requirements.

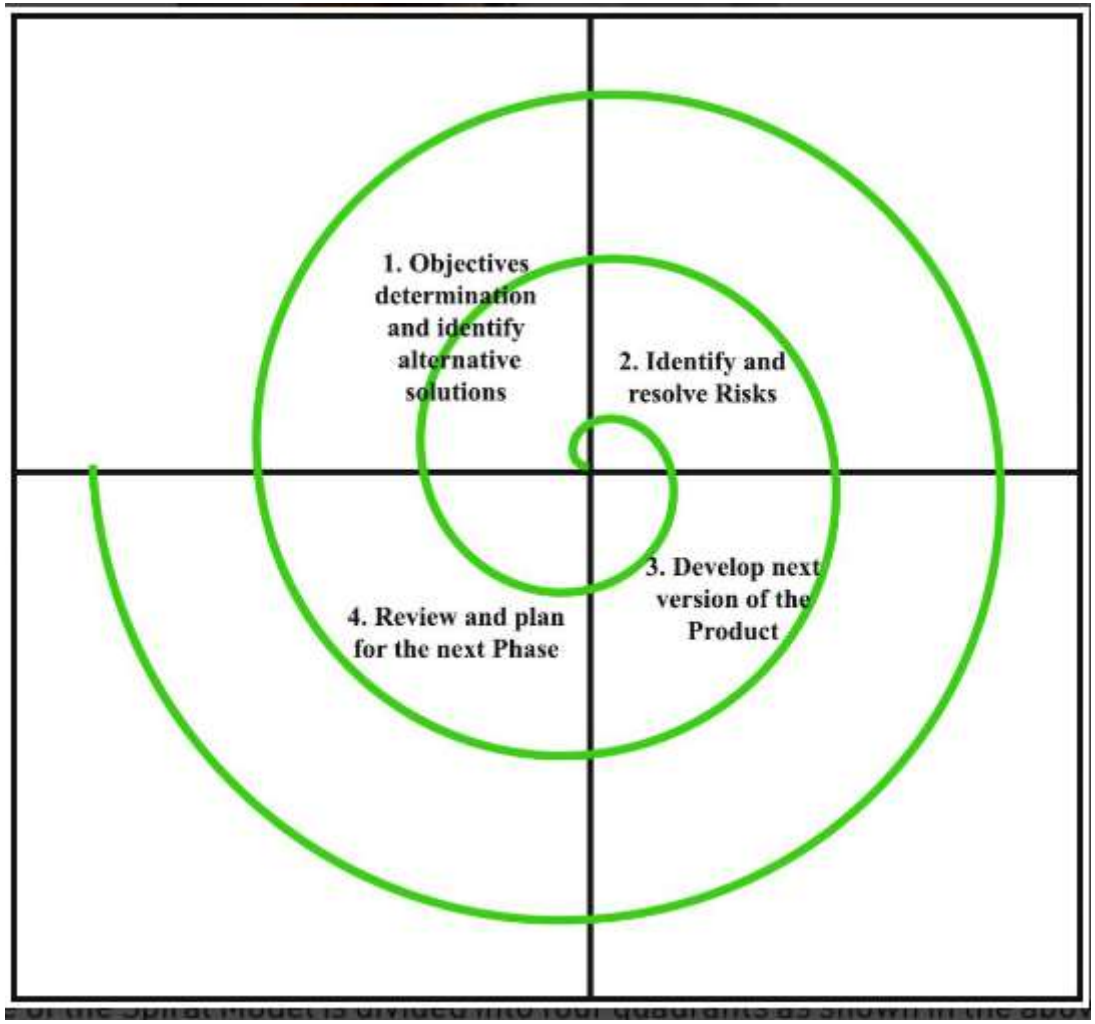
Principles of V-Model:

- **Scalability:** This principle states that the V-Model concept has the flexibility to accommodate any IT project irrespective of its size, complexity or duration.
- **Cross Referencing:** Direct correlation between requirements and corresponding testing activity is known as cross-referencing.
- **Tangible Documentation:** This principle states that every project needs to create a document. This documentation is required and applied by both the project development team and the support team. Documentation is used to maintaining the application once it is available in a production environment.

The Spiral model

- **Spiral model** is one of the most important Software Development Life Cycle models, which provides support for **Risk Handling**. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a **Phase of the software development process.** The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using the spiral model.

The below diagram shows the different phases of the Spiral Model



Each phase of the Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below

- **Objectives determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated, and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
- **Identify and resolve Risks:** During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.

- **Develop next version of the Product:** During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.

- **Review and plan for the next Phase:** In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.

- **Risk Handling in Spiral Model**

  A risk is any adverse situation that might affect the successful completion of a software project. The most important feature of the spiral model is handling these unknown risks after the project has started. Such risk resolutions are easier done by developing a prototype. The spiral model supports copying up with risks by providing the scope to build a prototype at every phase of the software development.

Advantages of Spiral Model:

Below are some advantages of the Spiral Model.

- **Risk Handling:** The projects with many unknown risks that occur as the development proceeds, in that case, Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.
- **Good for large projects:** It is recommended to use the Spiral Model in large and complex projects.

Advantages of Spiral Model:

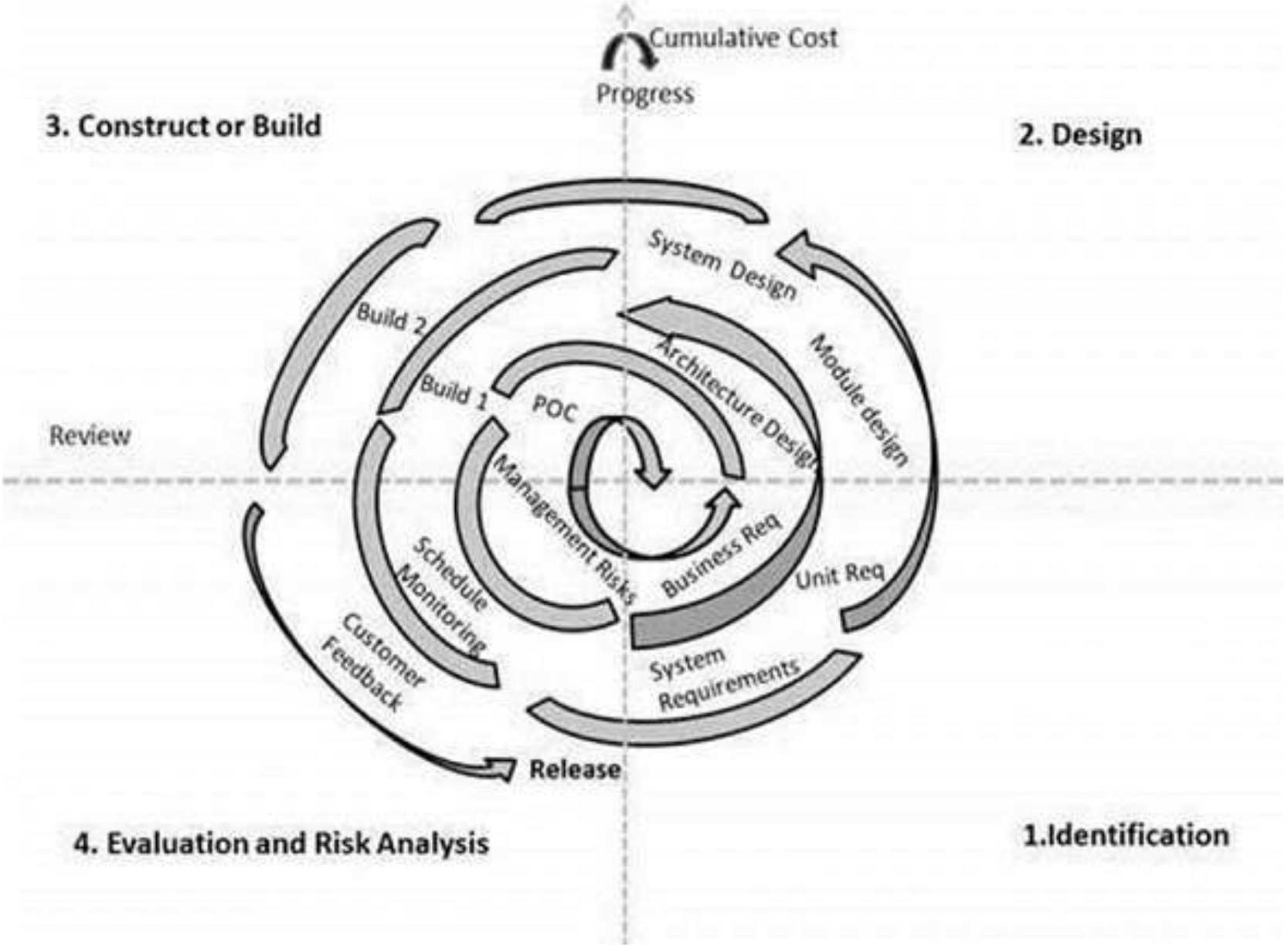
Below are some advantages of the Spiral Model.

- **Flexibility in Requirements:** Change requests in the Requirements at later phase can be incorporated accurately by using this model.
- **Customer Satisfaction:** Customer can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product.

Disadvantages of Spiral Model:

Below are some main disadvantages of the spiral model.

- **Complex:** The Spiral Model is much more complex than other SDLC models.
- **Expensive:** Spiral Model is not suitable for small projects as it is expensive.
- **Too much dependability on Risk Analysis:** The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced experts, it is going to be a failure to develop a project using this model.
- **Difficulty in time management:** As the number of phases is unknown at the start of the project, so time estimation is very difficult.


Cumulative Cost
Progress
3. Construct or Build
2. Design
Review
Release
4. Evaluation and Risk Analysis
1.Identification
System Design
Architecture Design
Module design
Unit Req
System Requirements
Business Req
Management Risks
POC
Build 1
Build 2
Schedule Monitoring
Customer Feedback

Software prototyping.

Prototyping is the process that enables developer to create a small model of software.

What is Software Prototyping?

- Prototype is a working model of software with some limited functionality. The prototype does not always hold the exact logic used in the actual software application and is an extra effort to be considered under effort estimation.
- Prototyping is used to allow the users evaluate developer proposals and try them out before implementation. It also helps understand the requirements which are user specific and may not have been considered by the developer during product design.
- Following is a stepwise approach explained to design a software prototype.

- Basic Requirement Identification

  This step involves understanding the very basics product requirements especially in terms of user interface. The more intricate details of the internal design and external aspects like performance and security can be ignored at this stage.

- Developing the initial Prototype

  The initial Prototype is developed in this stage, where the very basic requirements are showcased and user interfaces are provided. These features may not exactly work in the same manner internally in the actual software developed. While, the workarounds are used to give the same look and feel to the customer in the prototype developed.

- Basic Requirement Identification

  Review of the Prototype

  The prototype developed is then presented to the customer and the other important stakeholders in the project. The feedback is collected in an organized manner and used for further enhancements in the product under development.

- Revise and Enhance the Prototype

  The feedback and the review comments are discussed during this stage and some negotiations happen with the customer based on factors like – time and budget constraints and technical feasibility of the actual implementation. The changes accepted are again incorporated in the new Prototype developed and the cycle repeats until the customer expectations are met.

- Software Prototyping - Types

  There are different types of software prototypes used in the industry. Following are the major software prototyping types used widely −

- Throwaway/Rapid Prototyping

  Throwaway prototyping is also called as rapid or close ended prototyping. This type of prototyping uses very little efforts with minimum requirement analysis to build a prototype. Once the actual requirements are understood, the prototype is discarded and the actual system is developed with a much clear understanding of user requirements.

- Evolutionary Prototyping

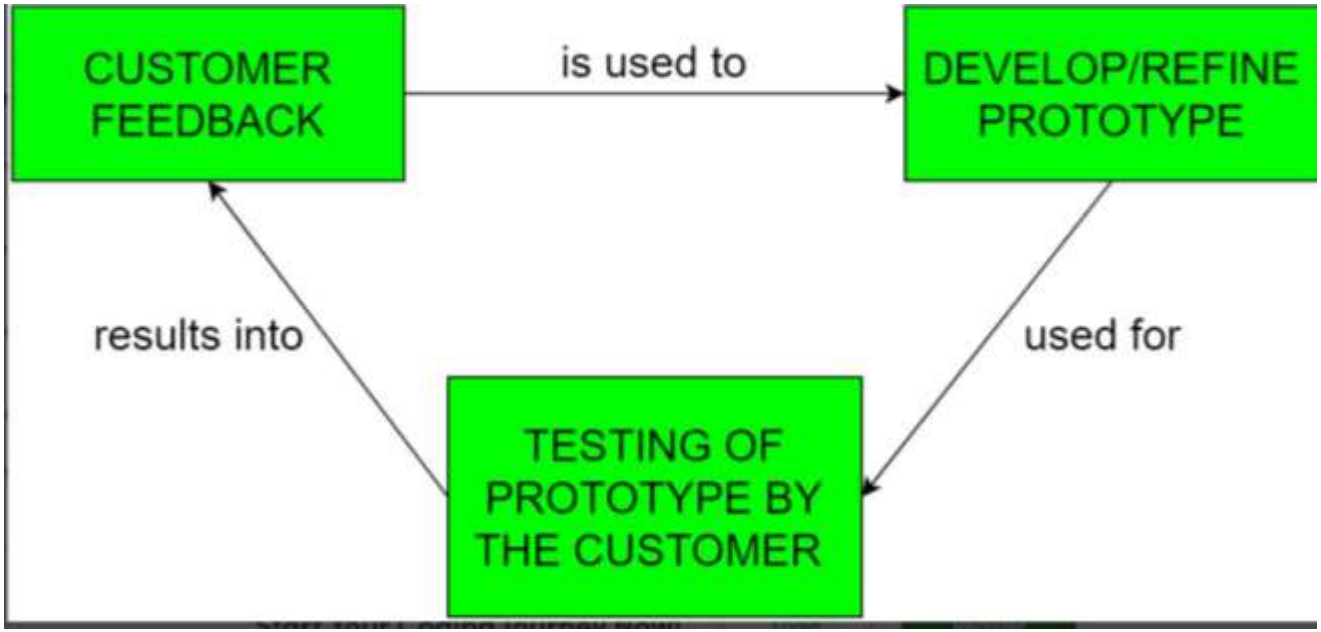
  Evolutionary prototyping also called as breadboard prototyping is based on building actual functional prototypes with minimal functionality in the beginning. The prototype developed forms the heart of the future prototypes on top of which the entire system is built. By using evolutionary prototyping, the well-understood requirements are included in the prototype and the requirements are added as and when they are understood.

- Incremental Prototyping

  Incremental prototyping refers to building multiple functional prototypes of the various sub-systems and then integrating all the available prototypes to form a complete system.

- Extreme Prototyping

  Extreme prototyping is used in the web development domain. It consists of three sequential phases. First, a basic prototype with all the existing pages is presented in the HTML format. Then the data processing is simulated using a prototype services layer. Finally, the services are implemented and integrated to the final prototype. This process is called Extreme Prototyping used to draw attention to the second phase of the process, where a fully functional UI is developed with very little regard to the actual services.


CUSTOMER FEEDBACK
is used to
DEVELOP/REFINE PROTOTYPE
results into
used for
TESTING OF PROTOTYPE BY THE CUSTOMER


Initial Set of User Requirements
Quick Design
Refine Requirements
Develop Prototype
Evaluation of prototype by the customer
NO
Prototype Accepted
YES
Discard Prototype
Develop final SRS
Design
Implement
Test
Deliver and Maintain
Refine Prototype
Test the final product
Deliver and Maintain
RAPID THROWAWAY PROTOTYPING
EVOLUTIONARY PROTOTYPING