Software Processes

Essential attributes of good software

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

Components of Software

There are **three components of the software**: Program, Documentation, and Operating Procedures.

- **Program –**

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

- **Documentation –**

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

- **Operating Procedures –**

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

Software Development

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

Software engineering: fundamental principle

Some fundamental principles apply to all types of software system, irrespective of the development techniques used:

- Systems should be developed using a **managed and understood development process**.
- **Different processes** are used for **different types of software**.
- **Dependability and performance** are important for all types of system.
- Understanding and managing the **software specification and requirements** (what the software should do) are important.
- Where appropriate, you should **reuse software** that has already been developed rather than write new software.

The software process

- A structured set of activities required to develop a software system.
- Many different **software processes** but all involve:
  - **Specification**
    - **- REQUIREMENT**
  - **Design and implementation**
    - **– CODING AND TESTING**
  - **Validation**
    - **– MATCHING THE PRODUCT WITH REQUIREMENTS**
  - **Evolution**
    - **– MAINTENANCE AND UPDATES**

Software Process Activities

There are four basic key process activities:

- **Software Specifications –** defining what the system should do
  - In this process, detailed description of a software system to be developed with its functional and non-functional requirements.
  - Here customers and engineers define the software that is to be produced and the constraints on its operation.
- **Software Development –** defining the organization of the system and implementing the system
  - In this process, designing, programming, documenting, testing, and bug fixing is done.
- **Software Validation –** checking that it does what the customer wants
  - In this process, evaluation software product is done to ensure that the software meets the business requirements as well as the end users needs.
  - Here the software is checked to ensure that it is what the customer requires.
- **Software Evolution –** changing the system in response to changing customer needs.
  - It is a process of developing software initially, then timely updating it for various reasons.
  - Here the software is modified to reflect changing customer and market requirements.

Software process descriptions

- When we describe and discuss processes, we usually talk about the activities in these processes such as specifying a **data model**, **designing a user interface**, etc. and **the ordering of these activities**.
- **Process descriptions** may also include:
  - **Products**,
    - which are the outcomes of a process activity;
  - **Roles**,
    - which reflect the responsibilities of the people involved in the process;
  - **Pre- and post-conditions**,
    - which are statements that are true before and after a process activity has been enacted or a product produced.

Processes

- **Plan-driven processes**
- **Agile processes**

Plan-driven and agile processes

- **Plan-driven processes**
  - Processes where all of the process activities are planned in advance and progress is measured against this plan.
- **Agile processes**,
  - In agile processes, **planning is incremental** and it is easier to change the process to reflect changing customer requirements.
- In practice, most practical processes include elements of both plan-driven and agile approaches.
- There are no right or wrong software processes.

Software Process and Software Development Lifecycle Model

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

Software Process and Software Development Lifecycle Model

✧ 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.

Software Process

Software Development Lifecycle Model

- Basic notions of the **software development process** is **SDLC** models which stands for Software Development Life Cycle
- There are many development life cycle models that have been developed in order to achieve different required objectives.
- The models specify the various stages of the process and the order in which they are carried out.

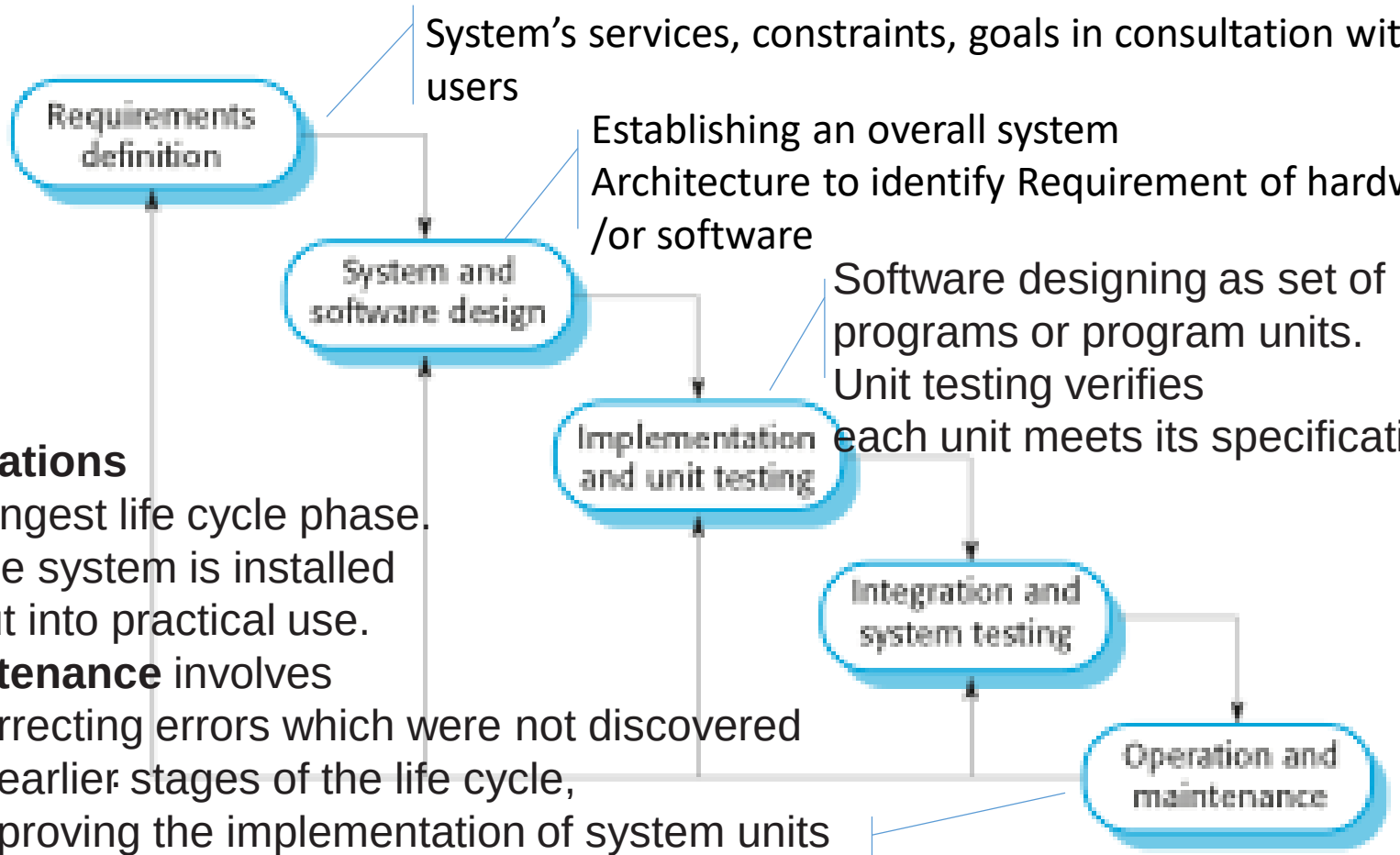
Software Process Models

- **The waterfall model**
- **Prototyping Model**
- **Incremental development**
- **Reuse-oriented software engineering**
- **Iterative Model**
- **Spiral Model**

Software Process Models

- **The waterfall model**
  - First published model of software development process
  - Derived from more general system engineering processes (Royce,1970)
  - **Plan-driven model.**
  - **Separate and distinct phases of specification and development**
    - **Cascade effect**

The waterfall model



Operations

- Longest life cycle phase.
- The system is installed
- Put into practical use.

Maintenance involves

- correcting errors which were not discovered in earlier stages of the life cycle,
- improving the implementation of system units and
- enhancing the system's services as new requirements are discovered.

Waterfall model phases

- There are separate identified phases in the waterfall model:
  - Requirements analysis and definition
  - System and software design
  - Implementation and unit testing
  - Integration and system testing
  - Operation and maintenance

Waterfall Model

- In the olden days, applications developed in Waterfall Model like
  - CRM Systems,
  - Supply Chain Management Systems etc
- Would usually take a year or longer to develop.

Waterfall Model

- There were cases where large scale enterprise systems were developed over a period of 2 to 3 years but were redundant by the time they were completed.
- WHY?
- The evolution of technology
  - By the time the applications were developed in C, C++ etc, new languages (relatively speaking) like Java, .Net etc would replace them with web based functionality.

Waterfall Model

- There were cases where large scale enterprise systems were developed over a period of 2 to 3 years but were redundant by the time they were completed.
- WHY?
- Even if the application was developed using a new technology, factors like
  - more competitors entering the market,
  - cheaper alternatives becoming available,
  - better functionality using newer technologies,
  - changes in customers requirement etc.

increased the risk of developing an application over several years.

Waterfall Model

- However, there are some areas where Waterfall model was continued to be preferred.
  - Consider a system where human life is on the line, where a system failure could result in one or more deaths.
  - In some countries, such mishaps could lead to imprisonment for those who are accountable.
  - Consider a system where time and money were secondary considerations and human safety was first.

Situations where Waterfall model was the preferred approach

- **Development of Department Of Defense (DOD), military and aircraft programs followed Waterfall model in many organizations.**
- This is because of the strict standards and requirements that have to be followed.
- In such industries, the requirements are known well in advance and contracts are very specific about the deliverable of the project.
- DOD Agencies typically considered Waterfall model to be compatible with their acquisition process and rigorous oversight process required by the government.

- Waterfall model was also used in banking, healthcare, control system for nuclear facilities, space shuttles etc

Advantages of Waterfall Model

- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model phases are processed and completed one at a time. Phases do not overlap.
- Waterfall model works well for smaller projects where requirements are clearly defined and very well understood.
- In Waterfall model, very less customer interaction is involved during the development of the product. Once the product is ready then only it can be demonstrated to the end users.

Waterfall model phases - DRAWBACK

The main drawback of the waterfall model:

- Difficulty of **accommodating change** after the process is underway.
- In principle, a phase has to be complete before moving onto the next phase.

Waterfall model problems

- Inflexible partitioning of the project into distinct stages **makes it difficult to respond to changing customer requirements**.
  - Once the product is developed and if any failure occurs then the cost of fixing such issues are very high, because we need to update everything from document till the logic.
  - Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
  - Few business systems have stable requirements.

Waterfall model phases - DRAWBACK

- Once an application is in the **testing** stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
- 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.

When to use the waterfall model

- This model is used only when the requirements are very well known, clear and fixed.
- Product definition is stable.
- Technology is understood.
- There are no ambiguous requirements
- Ample resources with required expertise are available freely
- The project is short.

Software process models

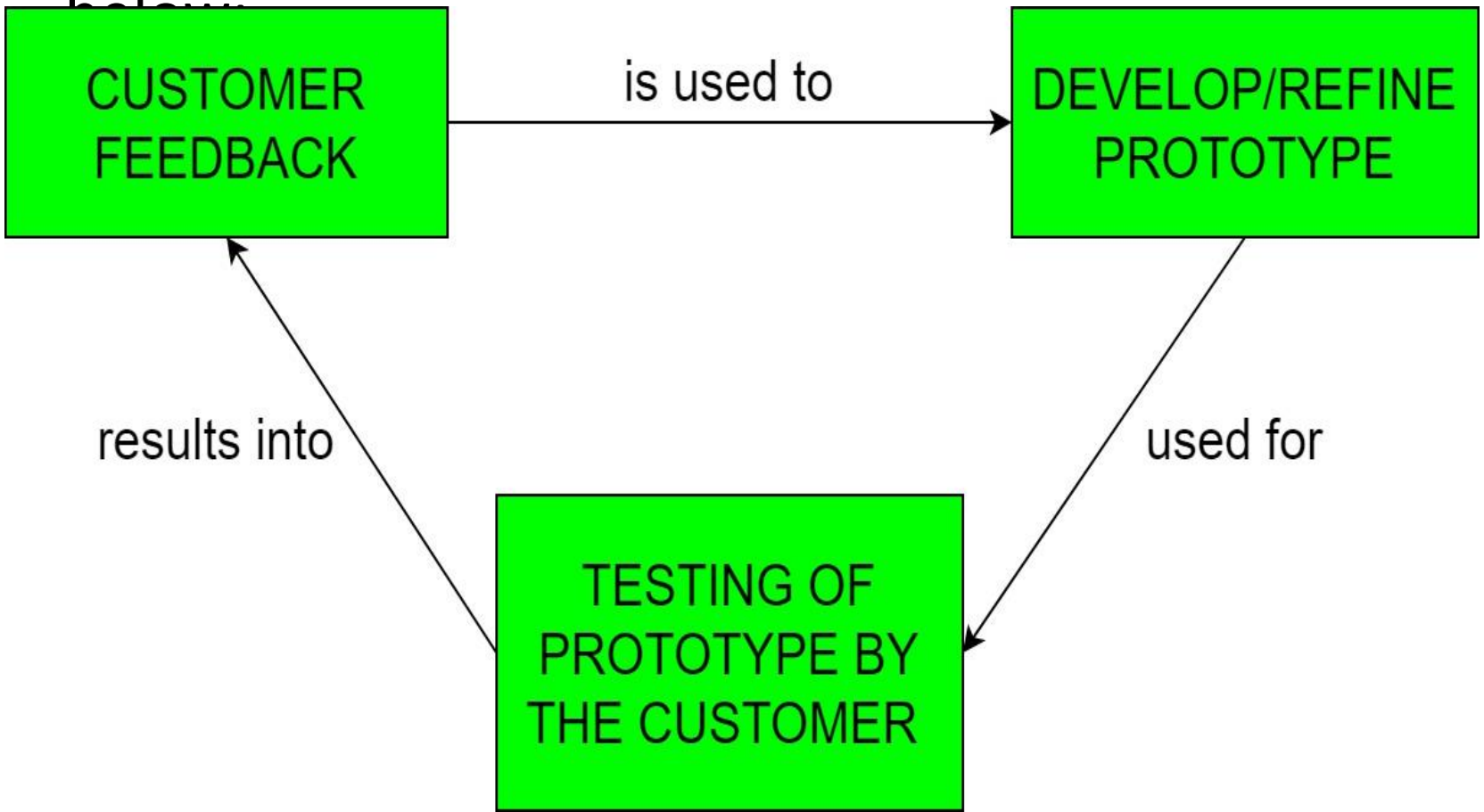
- The waterfall model
  - Plan-driven model. Separate and distinct phases of specification and development.
- Prototyping Model
  - The process of developing a working replication of a product or system
  - This model is used when the customers do not know the exact project requirements beforehand

Prototyping Model

- The Prototyping Model is one of the most popularly used Software Development Life Cycle Models (SDLC models).
- This model is used when the customers do not know the exact project requirements beforehand.
- In this model,
  - a prototype of the end product is first developed,
  - tested and
  - refined
  - as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product.

Prototyping Model

- It offers a small scale duplicate of the end product and is used for obtaining customer feedback as described below



Prototyping Model Phases

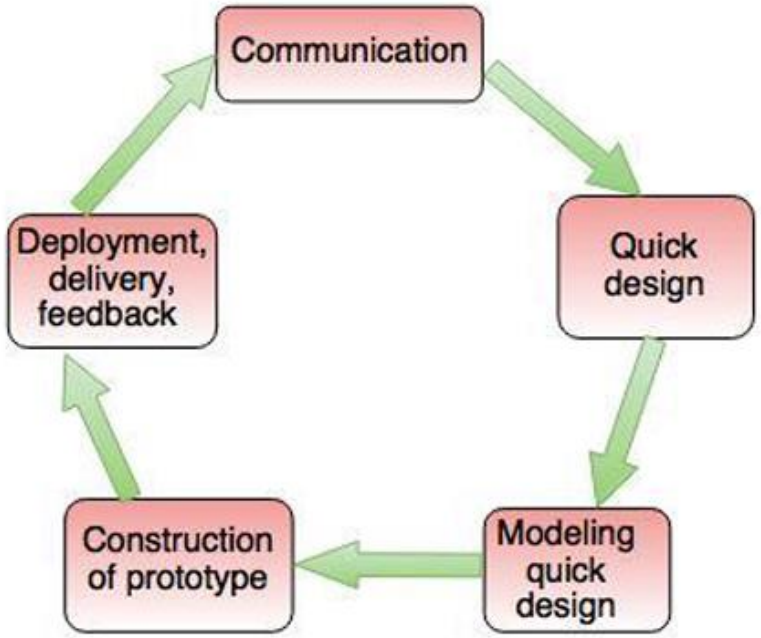


Fig. - The Prototyping Model

Types of prototyping

- **Sketches and diagrams**
- **3D printing or rapid model**
- **Physical model**
- **Wireframe**
  - A wireframe acts as a digital diagram or layout of the product. This is a common prototype used for websites, software or other digital tools. It can be used by anyone working on the project — from copywriters to developers — to navigate the structure and placement of different content.
- **Walk through virtual or augmented reality**
- **Video Prototype**

Features of Prototyping Model

- This model is used when the customers do not know the exact project requirements beforehand.
- In this model, a prototype of the end product is first developed, tested and refined as per customer feedback repeatedly till a final acceptable prototype is achieved which forms the basis for developing the final product.

Features of Prototyping Model

- In this process model,
  - the system is partially implemented before or during the analysis phase thereby giving the customers an opportunity to see the product early in the life cycle.
- The process starts by
  - interviewing the customers and
  - developing the incomplete high-level paper model.
- This document is used to build the initial prototype supporting only the basic functionality as desired by the customer.
- Once the customer figures out the problems, the prototype is further refined to eliminate them.
- The process continues until the user approves the prototype and finds the working model to be satisfactory.

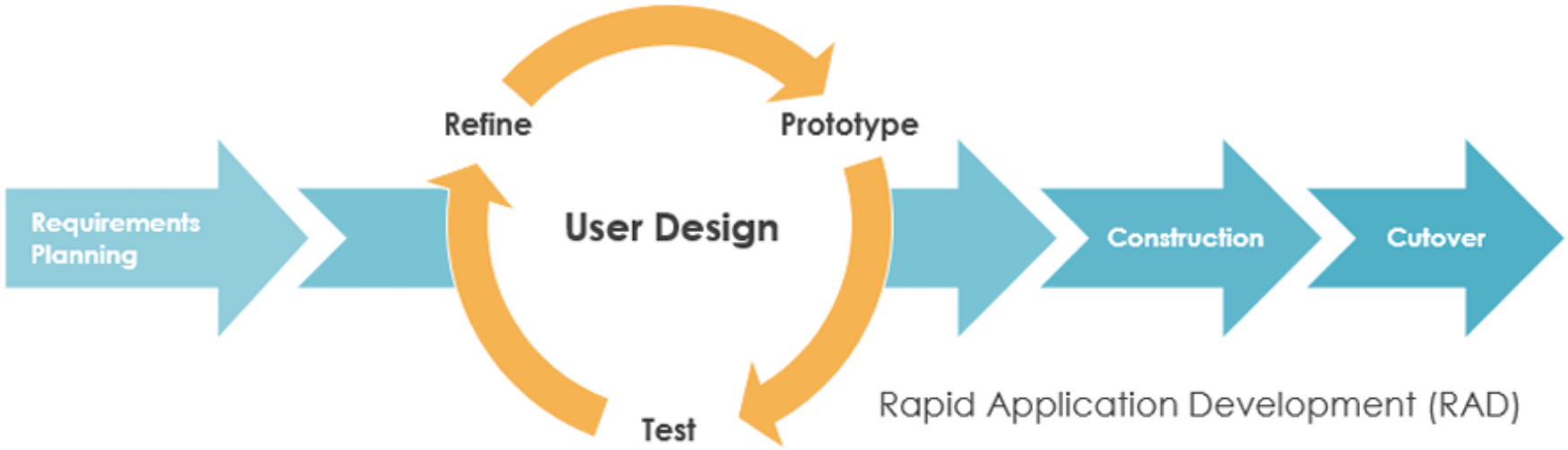
Types of Prototyping Model

- There are four types of models available:
  - **A) Rapid Throwaway Prototyping**
  - **B) Evolutionary Prototyping**
  - **C) Incremental Prototyping**
  - **D) Extreme Prototyping**

A) Rapid Throwaway Prototyping

- This technique offers a useful method of exploring ideas and getting customer feedback for each of them.
- In this method, a developed prototype need not necessarily be a part of the ultimately accepted prototype.
- Customer feedback helps in preventing unnecessary design faults and hence, the final prototype developed is of better quality.

Rapid Application Design



B) Evolutionary Prototyping

- In this method, the prototype developed initially is incrementally refined on the basis of customer feedback till it finally gets accepted.
- In comparison to Rapid Throwaway Prototyping, it offers a better approach which saves time as well as effort.
- This is because developing a prototype from scratch for every iteration of the process can sometimes be very frustrating for the developers.

C. Incremental Prototyping

- In this type of incremental Prototyping, the final expected product is broken into different small pieces of prototypes and being developed individually.
- In the end, when all individual pieces are properly developed, then the different prototypes are collectively merged into a single final product in their predefined order.

C. Incremental Prototyping

- It’s a very efficient approach that reduces the complexity of the development process, where the goal is divided into sub-parts and each sub-part is developed individually.

C. Incremental Prototyping

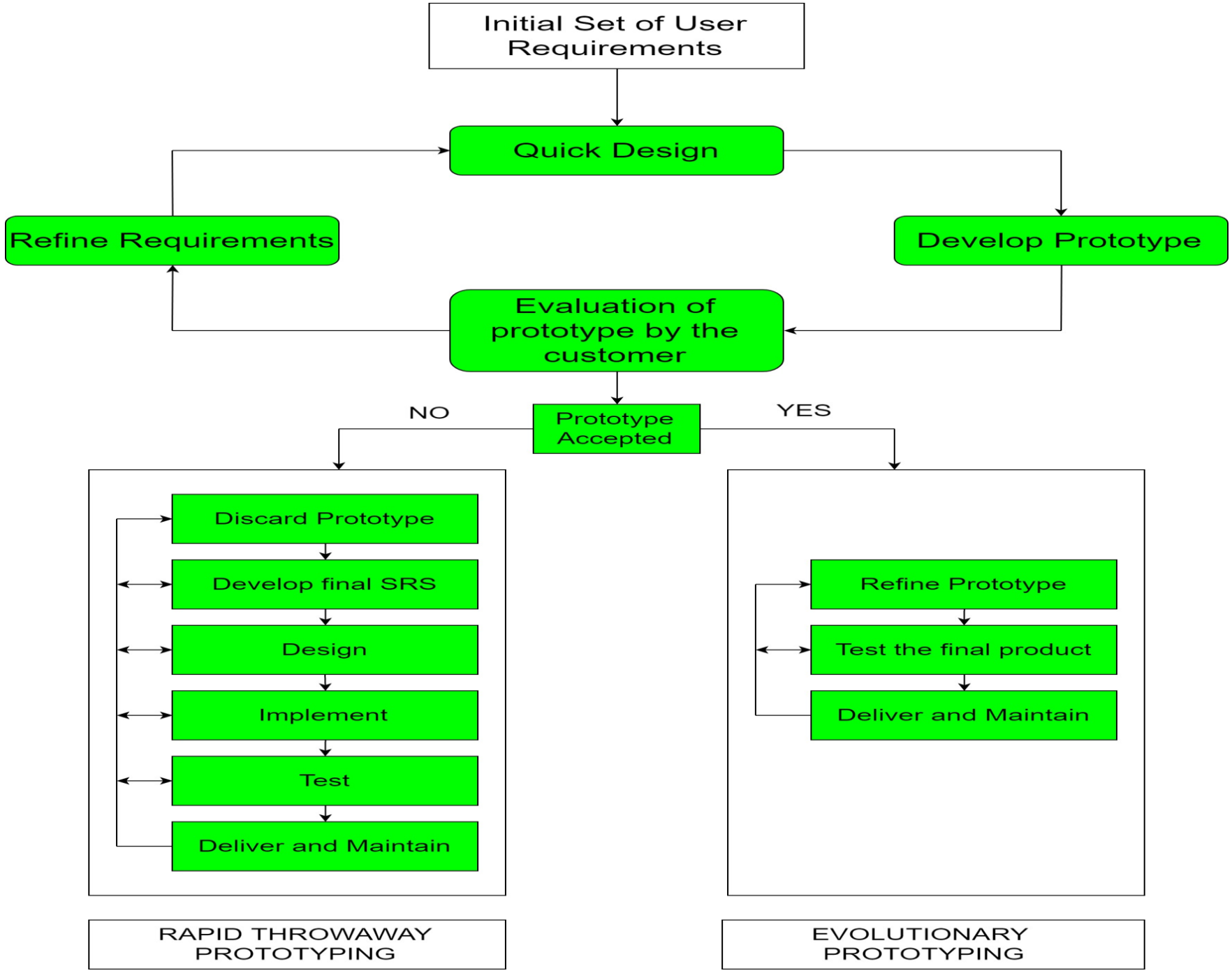
- It’s a very efficient approach that reduces the complexity of the development process, where the goal is divided into sub-parts and each sub-part is developed individually.
- The time interval between the project’s beginning and final delivery is substantially reduced because all parts of the system are prototyped and tested simultaneously.
- Of course, there might be the possibility that the pieces just do not fit together due to some lack of ness in the development phase – this can only be fixed by careful and complete plotting of the entire system before prototyping starts.

D. Extreme Prototyping

- This method is mainly used for **web development**.
- It is consists of three sequential independent phases:
- **D.1)** In this phase a basic prototype with all the existing static pages are presented in the HTML format.
- **D.2)** In the 2nd phase, Functional screens are made with a simulated data process using a prototype services layer.
- **D.3)** This is the final step where all the services are implemented and associated with the final prototype.

D. Extreme Prototyping

- This Extreme Prototyping method makes the project cycling and delivery **robust and fast**, and **keeps the entire developer team focus centralized on products deliveries** rather than discovering all possible needs and specifications and adding un-necessitated features.


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

Advantages –

- The customers get to see the partial product early in the life cycle. This ensures a greater level of customer satisfaction and comfort.
- New requirements can be easily accommodated as there is scope for refinement.
- Missing functionalities can be easily figured out.
- Errors can be detected much earlier thereby saving a lot of effort and cost, besides enhancing the quality of the software.
- The developed prototype can be reused by the developer for more complicated projects in the future.
- Flexibility in design.

Disadvantages –

- Costly w.r.t time as well as money.
- There may be too much variation in requirements each time the prototype is evaluated by the customer.
- Poor Documentation due to continuously changing customer requirements.
- It is very difficult for developers to accommodate all the changes demanded by the customer.
- There is uncertainty in determining the number of iterations that would be required before the prototype is finally accepted by the customer.

Disadvantages –

- After seeing an early prototype, the customers sometimes demand the actual product to be delivered soon.
- Developers in a hurry to build prototypes may end up with sub-optimal solutions.
- The customer might lose interest in the product if he/she is not satisfied with the initial prototype.

Use –

- The Prototyping Model should be used when **the requirements of the product are not clearly understood or are unstable**.
- It can also be used **if requirements are changing quickly**.
- This model can be successfully used **for developing user interfaces, high technology software-intensive systems, and systems with complex algorithms and interfaces**.
- It is also a very good choice **to demonstrate the technical feasibility of the product**.

Software process models

- **The waterfall model**
- **Prototyping Model**
- **Incremental Model development**

Software process models

- Incremental Prototyping

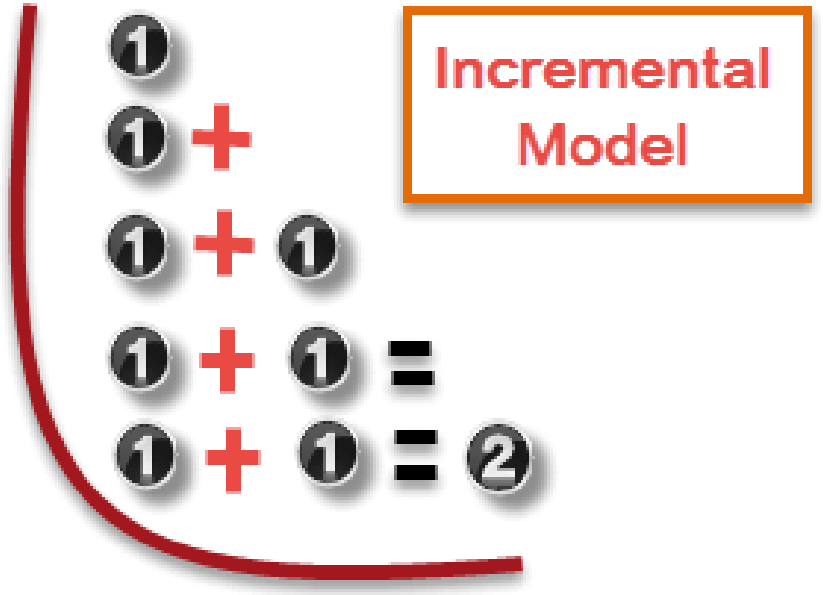
  VS

- Incremental development
  - Specification, development and validation are interleaved. May be plan-driven or agile.

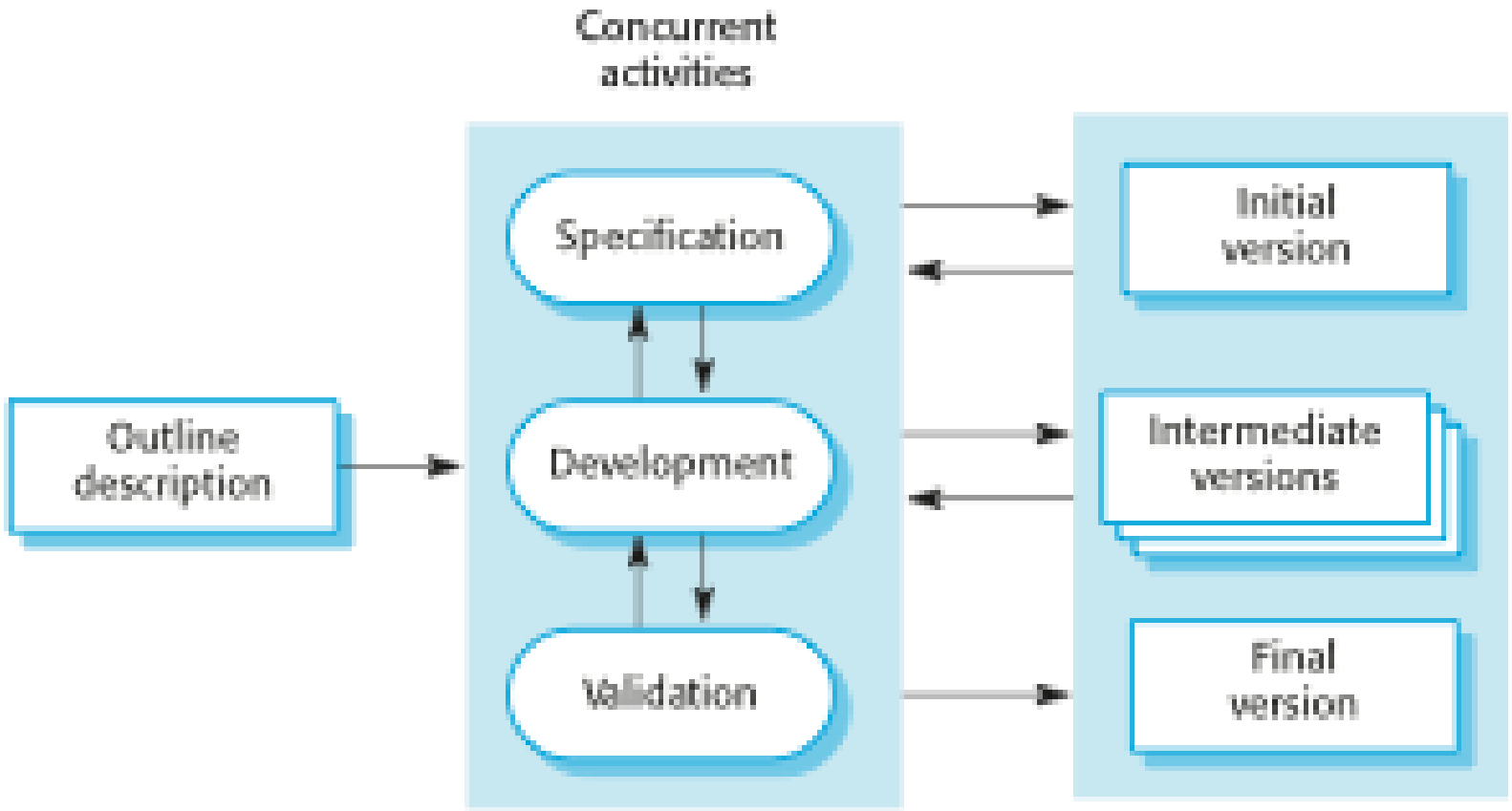
Incremental Model

- Incremental Model is a process of software development where **requirements are broken down into multiple standalone modules** of software development cycle.
- Incremental development is done in steps from analysis design, implementation, testing/verification, maintenance.

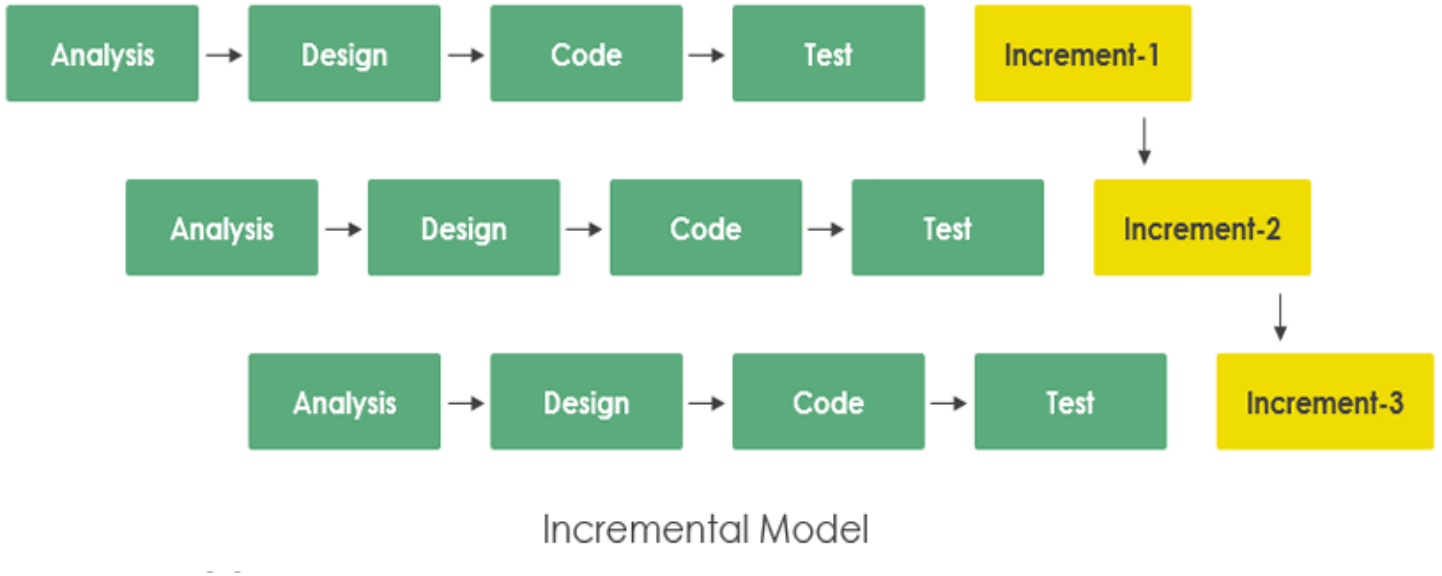
Incremental Model



Incremental Model development



Incremental Model



Incremental Model

Incremental Model development

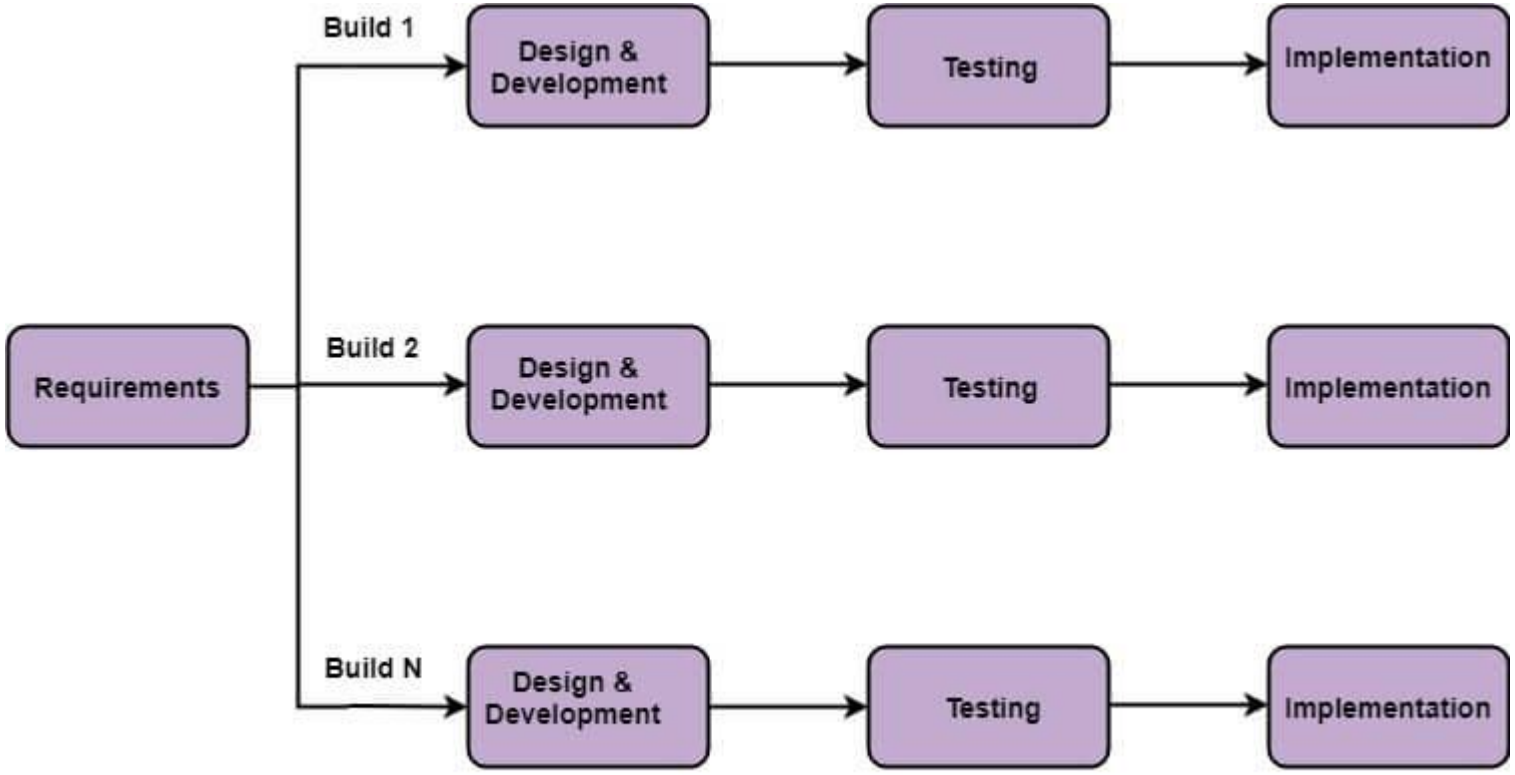


Fig: Incremental Model

Incremental development benefits

- **Module by Module Working**
- **Maximum Customer Interaction**
- **Good model for Large Products**
  - **Web Apps, Mobile Apps, LMS etc**
- **Faster Release of Products in Demand**
- **Flexible to Change**

Incremental development benefits

- The cost of accommodating changing customer requirements is reduced.
  - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
- It is easier to get customer feedback on the development work that has been done.
  - Customers can comment on demonstrations of the software and see how much has been implemented.
- More rapid delivery and deployment of useful software to the customer is possible.
  - Customers are able to use and gain value from the software earlier than is possible with a waterfall process.

Incremental development problems

- **The process is not visible**.
  - **Managers need regular deliverables** to measure progress.
  - If systems are developed quickly, it is **not cost-effective to produce documents** that reflect every version of the system.
- **System structure tends to degrade as new increments are added**.
  - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure.
  - Incorporating further software changes becomes increasingly difficult and costly.

When to use Incremental models?

- Requirements of the system are clearly understood
- When demand for an early release of a product arises
- When software engineering team are not very well skilled or trained
- When high-risk features and goals are involved
- Such methodology is more in use for web application and product based companies

Difference between Prototype Model and Incremental Model :

S.No.	Prototype Model	Incremental Model
1.	Prototype model is a software development model where a prototype is built, tested and then refined as per customer needs.	Incremental Model is a software development model where the product is, analyzed, designed, implemented and tested incrementally until the product is finished
2.	It is suitable for high-risk projects.	Incremental model can’t handle large project.
3.	Cost of prototype model is Low.	Cost of incremental model is also Low.
4.	Flexibility to change in prototype model is Easy.	Flexibility to change in incremental model is Easy.
5.	It does not support automatic code generation.	It supports automatic code generation as. results in minimal code writing.
6.	It does not give emphasis on risk analysis.	It give emphasis on risk analysis.
7.	In this user involvement is high.	In this user Involvement is only at the beginning.

Software process models

- **The waterfall model**
- **Prototyping Model**
- **Incremental development**
- **Reuse-oriented software engineering**
  - The system is assembled from existing components. May be plan-driven or agile.

Reuse-oriented software engineering

- **Reuse Oriented Model (ROM)**,
- **Also known as reuse-oriented development (ROD)**

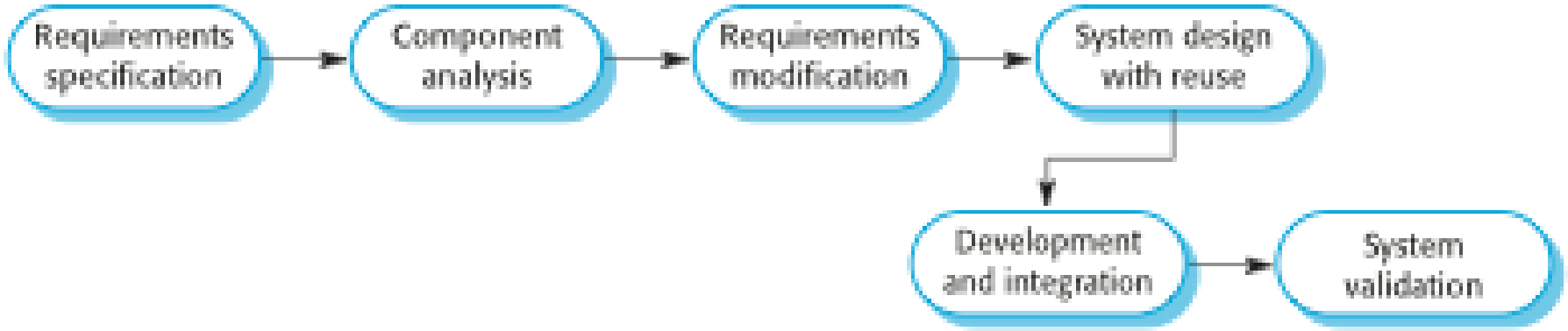
Reuse-oriented software engineering

- The reuse model has 4 **fundamental steps** which are followed :
  - **To identify components of old system that are most suitable for reuse.**
  - **To understand all system components.**
  - **To modify old system components to achieve new requirements.**
  - **To integrate all of modified parts into new system.**

Reuse-oriented software engineering

- Based on systematic reuse where systems are integrated from existing components or COTS (Commercial-off-the-shelf) systems.
- Process stages
  - Requirement analysis
  - **Component analysis**;
  - **Requirements modification**;
  - **System design with reuse**;
  - Development and integration.
- Reuse is now the standard approach for building many types of business system

Reuse-oriented software engineering



Advantages :

- It can reduce total cost of software development.
- The risk factor is very low.
- It can save lots of time and effort.
- It is very efficient in nature.

Disadvantages :

- Reuse-oriented model is not always worked as a practice in its true form.
  - due to cause of an entire repertoire of reusable additives that might not be available. In such cases, several new system components need to be designed.
  - If it is not done, ROM has to compromise in perceived requirements, leading to a product that does not meet exact requirements of user.
- Compromises in requirements may lead to a system that does not fulfill requirement of user.
- Sometimes using old system component, that is not compatible with new version of component, this may lead to an impact on system evolution.

Software process models

- **The waterfall model**
- **Prototyping Model**
- **Incremental development**
- **Reuse-oriented software engineering**
- **Iterative Model**
  - Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented.

Iterative Model

- In the Iterative model,
- iterative process starts with a simple implementation of a small set of the software requirements and
- iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

Iterative Model

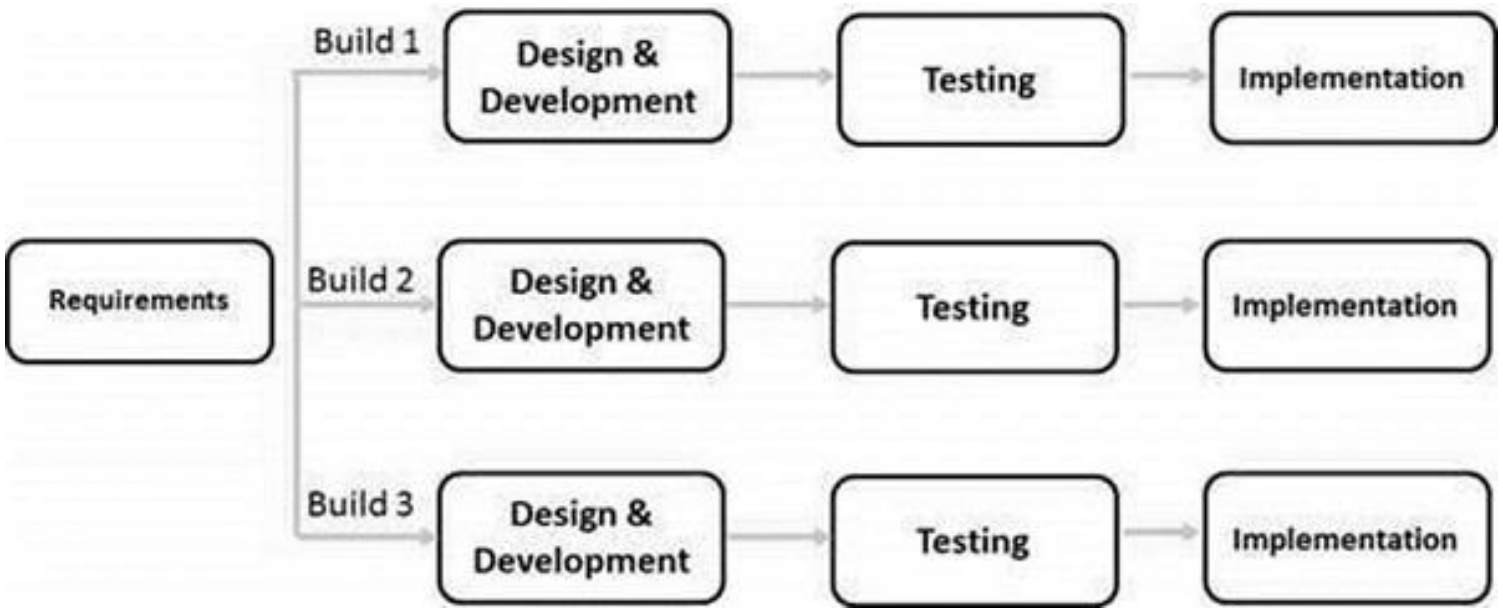
- An iterative life cycle model does not attempt to start with a full specification of requirements.
- Instead, development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements.
- This process is then repeated, producing a new version of the software at the end of each iteration of the model.

Iterative Model - Design

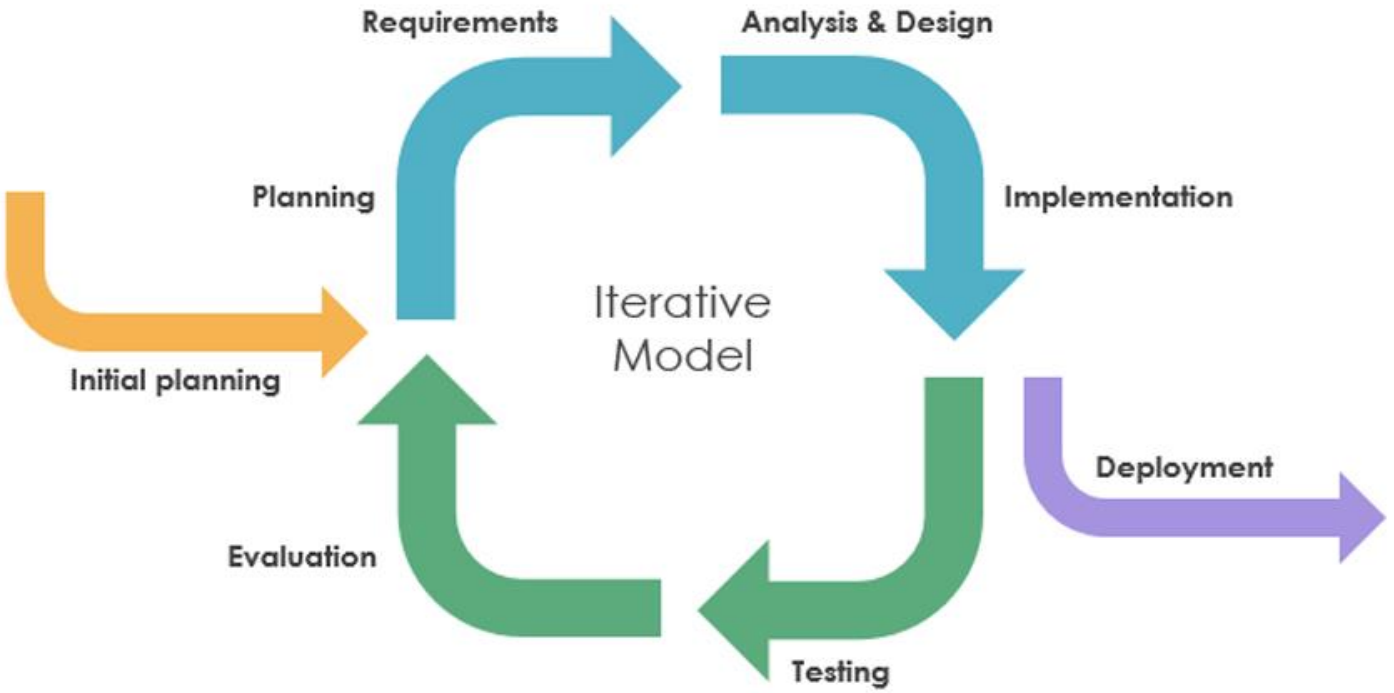
- The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).
- At each iteration, design modifications are made and new functional capabilities are added.

Iterative Model - Design

- The following illustration is a representation of the Iterative and Incremental model –



Iterative Model



Iterative Model - Design

- **Iterative and Incremental development** is a **combination of both iterative design** or **iterative method and incremental build model for development**.

Difference between iterative and an incremental life cycle

- The main difference between iterative and an incremental life cycle is:
- An iterative process makes progress through continuous refinement while an incremental process makes progress through small increments.
- For example,
- In a software project, an **iterative approach** would be to **build the overall solution/product** and **then refine some of the areas that require improvement**.
- Instead, using an **incremental approach** involves **building and releasing one feature at a time** depending on priorities defined by the customer.

Iterative Model - Application

- Like other SDLC models, Iterative and incremental development has some specific applications in the software industry.
- **This model is most often used in the following scenarios** −
  - **Requirements** of the complete system are **clearly defined and understood**.
  - **Major requirements must be defined**; however, some functionalities or requested enhancements may evolve with time.
  - There is a **time to the market** constraint.
  - A **new technology is being used** and is **being learnt** by the development team while working on the project.
  - **Resources** with needed skill sets are **not available** and are planned to be used on contract basis for specific iterations.
  - There are some **high-risk features and goals which may change in the future**.

Advantages

- The advantages of the Iterative and Incremental SDLC Model are as follows −
  - Some working functionality can be developed quickly and early in the life cycle.
  - Results are obtained early and periodically.
  - Parallel development can be planned.
  - Progress can be measured.
  - Less costly to change the scope/requirements.
  - Testing and debugging during smaller iteration is easy.

Advantages

- The advantages of the Iterative and Incremental SDLC Model are as follows –
  - Risks are identified and resolved during iteration; and each iteration is an easily managed milestone.
  - Easier to manage risk - High risk part is done first.
  - With every increment, operational product is delivered.
  - Issues, challenges and risks identified from each increment can be utilized/applied to the next increment.
  - Risk analysis is better.
  - It supports changing requirements.
  - Initial Operating time is less.
  - Better suited for large and mission-critical projects.
  - During the life cycle, software is produced early which facilitates customer evaluation and feedback.

Disadvantages

- The disadvantages of the Iterative and Incremental SDLC Model are as follows −
  - More resources may be required.
  - Although cost of change is lesser, but it is not very suitable for changing requirements.
  - More management attention is required.
  - System architecture or design issues may arise because not all requirements are gathered in the beginning of the entire life cycle.
  - Defining increments may require definition of the complete system.
  - Not suitable for smaller projects.
  - Management complexity is more.
  - End of project may not be known which is a risk.
  - Highly skilled resources are required for risk analysis.
  - Projects progress is highly dependent upon the risk analysis phase.

Software process models

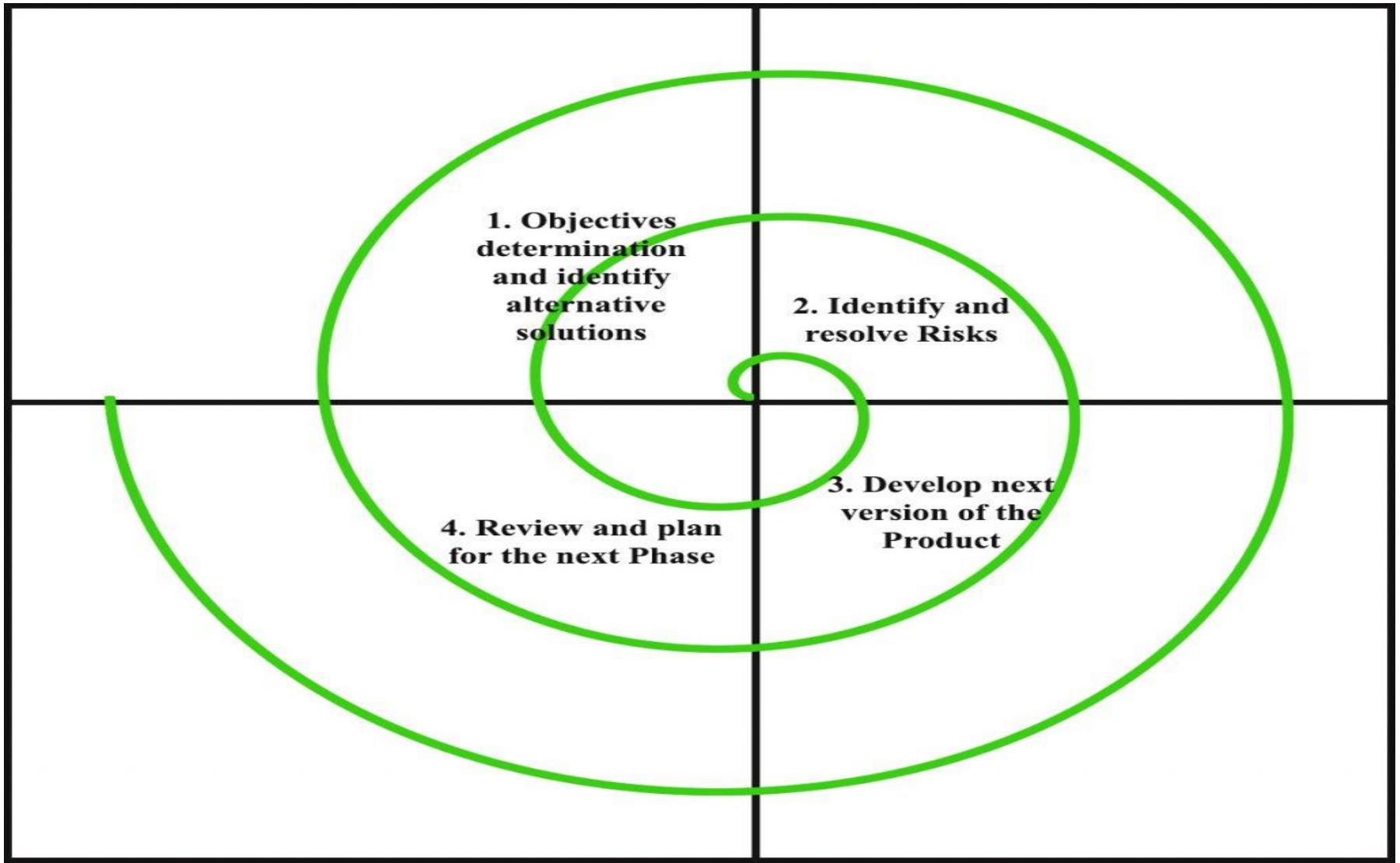
- **The waterfall model**
- **Prototyping Model**
- **Incremental development**
- **Reuse-oriented software engineering**
- **Iterative Model**
- **Spiral Model**
  - **First described by Barry Boehm in 1986, is a risk-driven software development process model which was introduced for dealing with the shortcomings in the traditional waterfall model.**

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.**

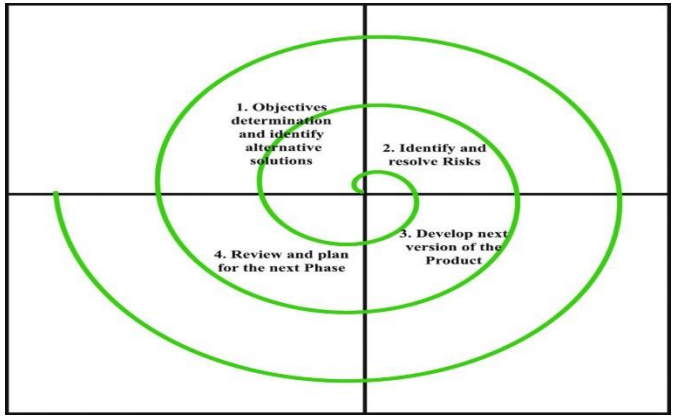
Diagram of the Spiral Model showing the different phases :-

Each phase of the Spiral Model is divided into four quadrants



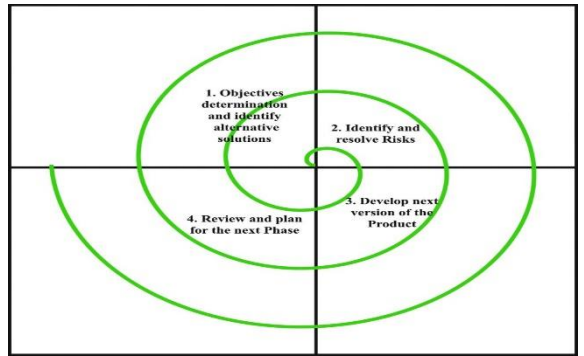
Functions of these four quadrants

- The functions of these four quadrants -
- **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.



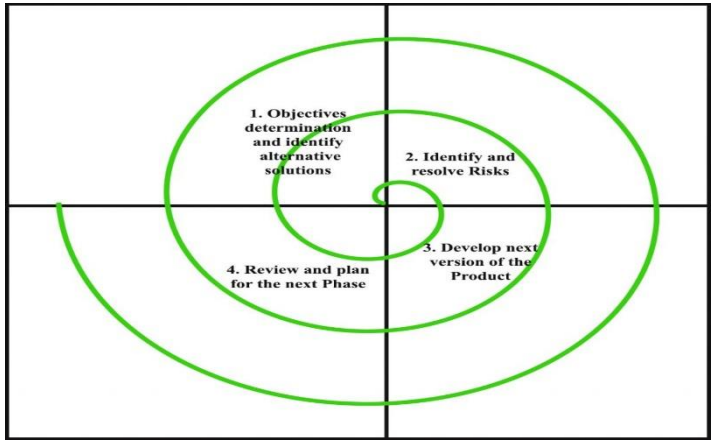
Functions of these four quadrants

- The functions of these four quadrants -
- **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.



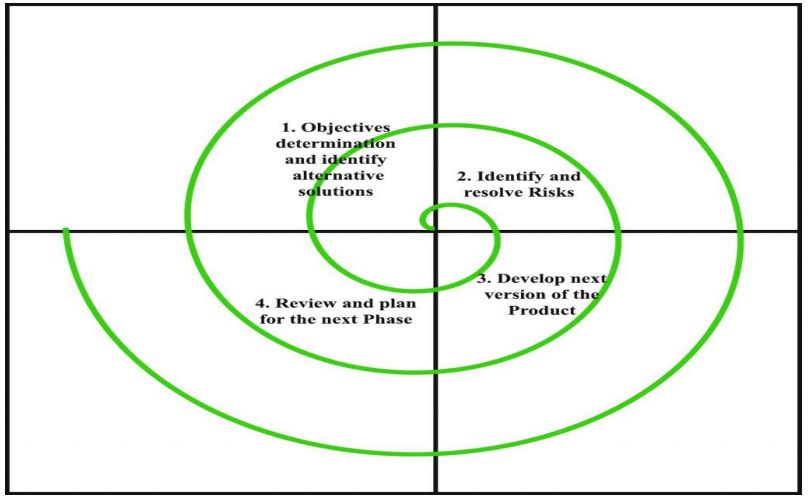
Functions of these four quadrants

- The functions of these four quadrants -
- **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.



Functions of these four quadrants

- The functions of these four quadrants -
- **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.



Framework

- The framework found in the risk activities of the spiral model are as shown in the following figure

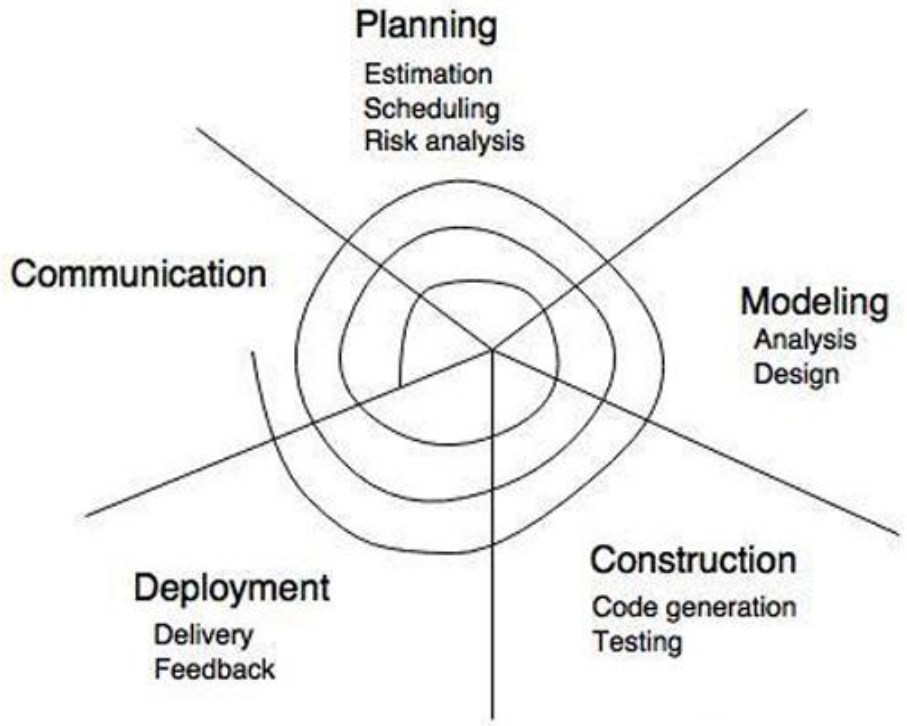


Fig. - The Spiral Model

Spiral Model is called Meta Model

- **Why?**
  - The Spiral model is called a Meta-Model because it **subsumes all the other SDLC models**.
  - For example, a **single loop spiral** actually represents the **Iterative Waterfall Model**.
  - The spiral model **incorporates the stepwise approach** of the **Classical Waterfall Model**.
  - The spiral model uses the approach of the **Prototyping Model** by building a prototype at the start of each phase as a risk-handling technique.
  - Also, the spiral model can be considered as supporting the **Evolutionary model** – the iterations along the spiral can be considered as evolutionary levels through which the complete system is built.

Advantages of Spiral Model

- It reduces high amount of risk.
- It is good for large and critical projects.
- It gives strong approval and documentation control.
- In spiral model, the software is produced early in the life cycle process.

Disadvantages of Spiral Model

- It can be costly to develop a software model.
- It is not used for small projects.

Critical Elements of Software

So with software, the elements that are critical:

- **Security** – information in the system is secure and protected
- **Robust-performance** – it'll handle rigorous use, it'll handle unexpected occurrences
- **Integration** – it doesn't stand alone, it connects to existing systems to simplify work
- **Maintainability** – it can be built upon and maintained without much effort and worry of collapse

“Industrial Strength Software”

- Unusually strong or effective;
- Able to withstand great strain or use
- QUALITY MATTERS

“Industrial Strength Software” – Why Quality Matters

- Electrical glitch forces Maruti to make its largest ever recall involving over 1.8 lakh cars; Ciaz, Ertiga, S-Cross among models in list - Sep 03, 2021

  Read more at: https://economictimes.indiatimes.com/industry/auto/auto-news/maruti-suzuki-makes-its-largest-ever-recall-involving-over-1-8-lakh-cars-after-electrical-glitch/articleshow/85893451.cms?utm_source=contentofinterest&utm_medium=text&utm_campaign=cppst

“Industrial Strength Software” – Why Quality Matters

- It’s a little bit like cars, some are engineered to higher standards of performance, efficiency and safety than others.
- You might pay a higher cost upfront, but **the value is in the long run** where the benefits come through or in the case of an unforeseen event, i.e. an accident.

“Industrial Strength Software” – Why Quality Matters

- In the new world of app development, **Industrial Strength** is more important than ever.
- Regardless of whether we’re doing Android development or in iOS developer mode the apps we build will in most cases have a cloud backend or need to integrate to an existing system.
- **Being secure and robust are key.**
- If it gets **broken** into or can’t cope it means significant loss of users, reputation and eventually revenue.

Assignment

- Study the following paper and make notes on “Industrial Strength Software” – Why Quality Matters
- industrial strength software and quality - paper.pdf