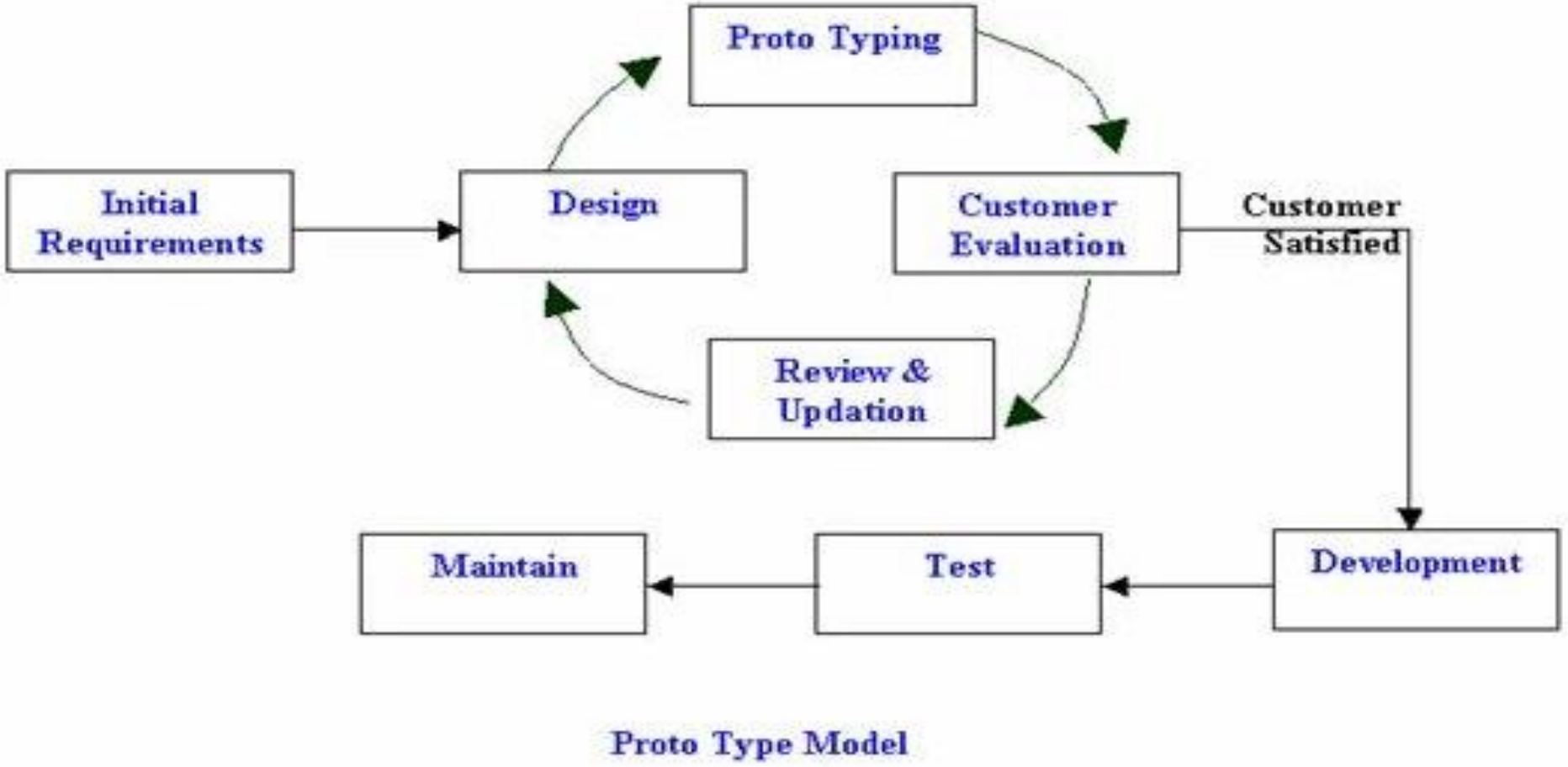
Prototyping Model
Spiral Model
specialized process models

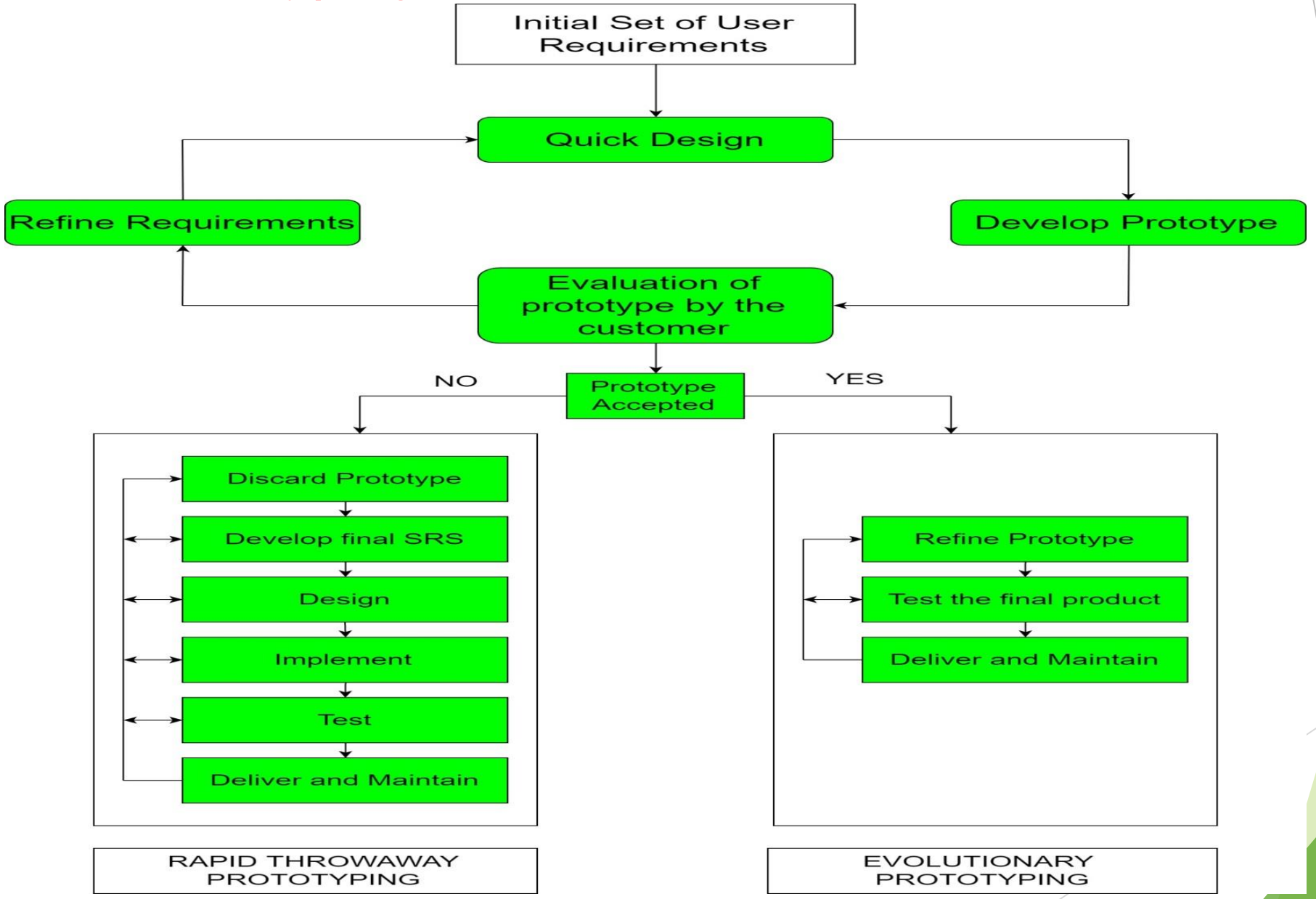
Prototyping Model



Proto Type Model

- The prototyping model can be considered to be an extension of the waterfall model.
- This model suggests building a working prototype of the system, before development of the actual software.
- A prototype is a crude implementation of a system.
- The prototype is developed based on the currently known requirements.
- Development of the prototype obviously undergoes design, coding, and test but each of these phases is not done very formally or thoroughly.
- Prototyping is an attractive idea for complicated and large systems for which there is no manual process or existing system to help determine the requirements.
- By using this prototype, the client can get an actual feel of the system, because the interactions with the prototype can enable the client to better understand the requirements of the desired system.
-

Prototyping Model



There are 2 approaches for this model:

- **Rapid Throwaway Prototyping**
- **Evolutionary Prototyping**

- **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 a better quality.

- **Evolutionary Prototyping -**

- 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.

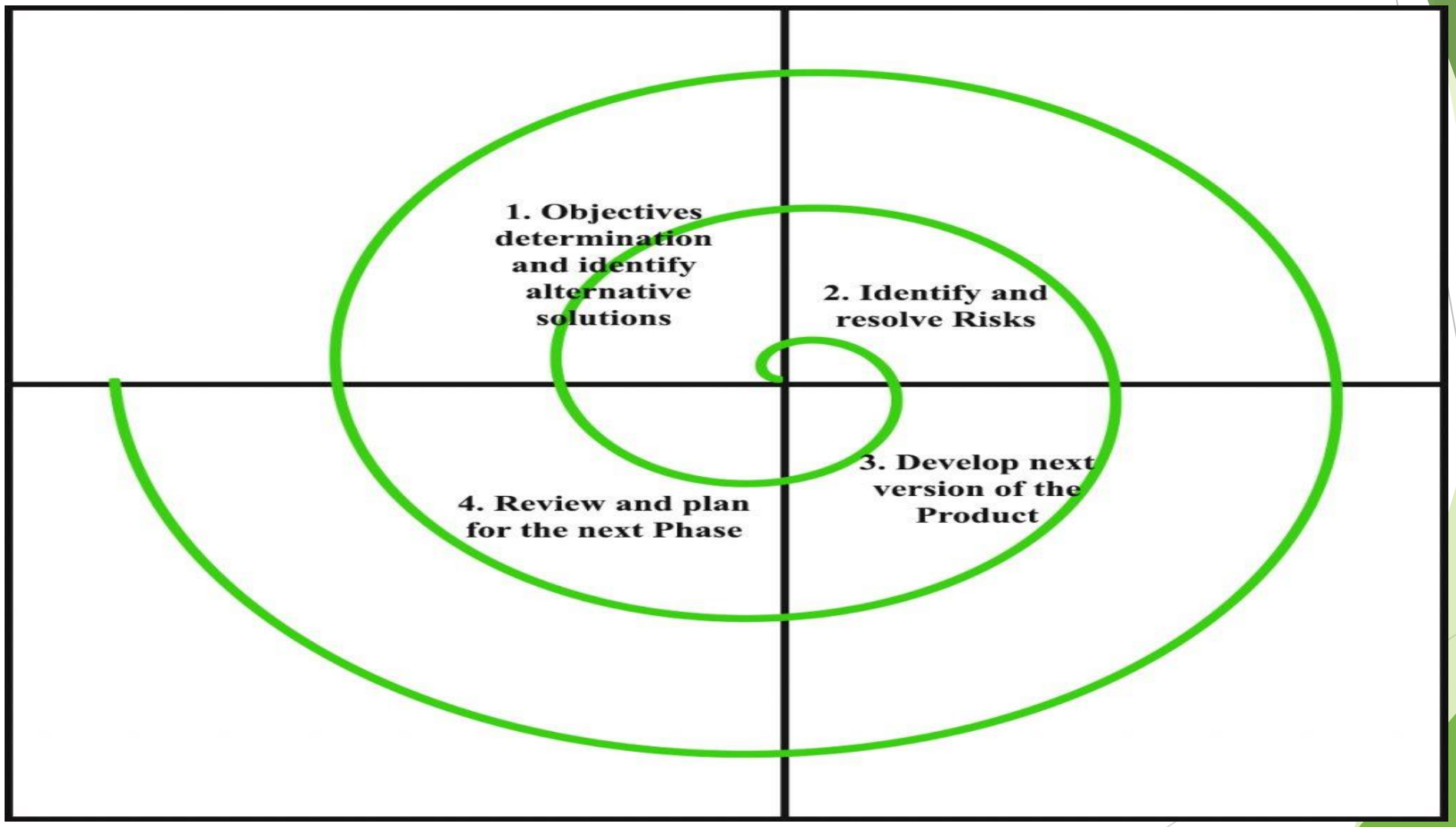
- Advantageous

- The resulting system is easier to use
- User needs are better accommodated
- Problems are detected earlier
- The design is of higher quality
- The resulting system is easier to maintain
- The development incurs less effort

- **Disadvantages of using Prototype Model :**

1. This model is costly.
2. It has poor documentation because of continuously changing customer requirements.
3. There may be too much variation in requirements.
4. Customers sometimes demand the actual product to be delivered soon after seeing an early prototype.
5. There may be sub-optimal solutions because of developers in a hurry to build prototypes.
6. Customers may not be satisfied or interested in the product after seeing the initial prototype.
7. There is certainty in determining the number of iterations.
8. There may be incomplete or inadequate problem analysis.
9. There may increase the complexity of the system.

Spiral Model

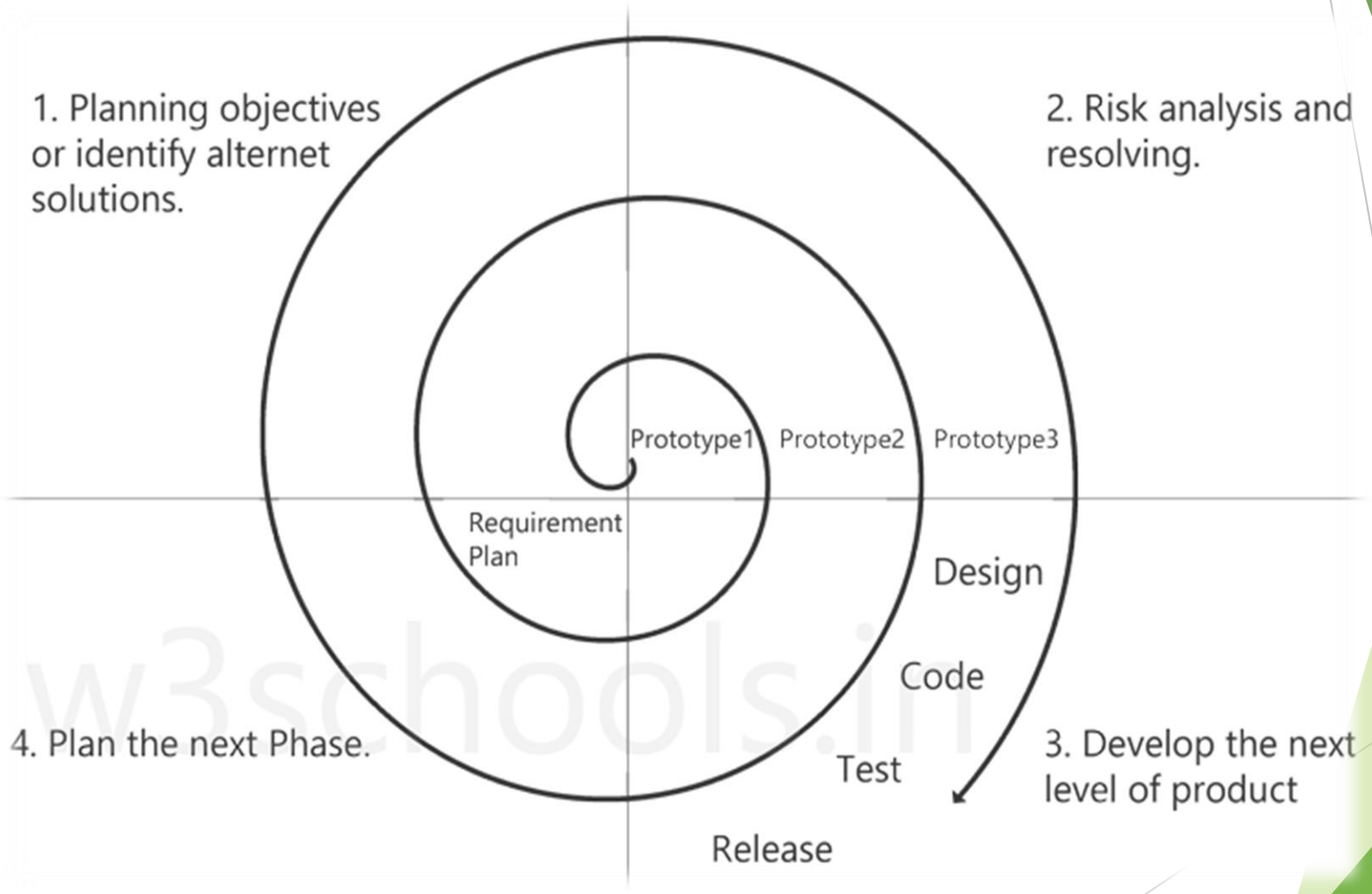


Spiral Model

- This Spiral model is a combination of iterative development process model and sequential linear development model
- The spiral model, originally proposed by Boehm in 1988,
- It is an evolutionary iterative software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the linear sequential model. (water fall model)
- Using the spiral model, software is developed in a series of incremental releases.

- During early iterations, the incremental release might be a model or prototype.
- During later iterations, increasingly more complete versions of the engineered system are produced.
- This model got its name from the appearance of its diagrammatic representation that looks like a spiral with many loops
- The exact number of loops of the spiral is not fixed and can vary from

Spiral Model



- Quadrant I: The objectives are investigated, elaborated, and analysed.
- Based on this, the risks involved in meeting the phase objectives are identified.
- In this quadrant, alternative solutions possible for the phase under consideration are proposed.

- Quadrant 2: During the second quadrant, the alternative solutions are evaluated to select the best possible solution.
- To be able to do this, the solutions are evaluated by developing an appropriate prototype.

- Quadrant 3: Activities during the third quadrant consist of developing and verifying the next level of the software.
- At the end of the third quadrant, the identified features have been implemented and the next version of the software is available.
- Quadrant 4: Activities during the fourth quadrant concern reviewing the results of the stages traversed so far (i.e. the developed version of the software) with the customer and planning the next iteration of the spiral.

Advantages

- **Risk Handling:** 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.
- **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

- **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 dependable on Risk Analysis:** The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced expertise, 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.

Specialized Process Models

- Specialized process models adopt many of the characteristics of one or more of the traditional models and are generally applied when a specialized or narrowly defined software engineering approach is chosen.
- The following are some of the specialized process models:
- Component-Based Development
- The Formal Methods Model
- Aspect-Oriented Software Development

Specialized software process models

Specialized software process models

- Component based development-------Reuse
- Formal ----Mathematical Specifications
- Aspect oriented s/w development

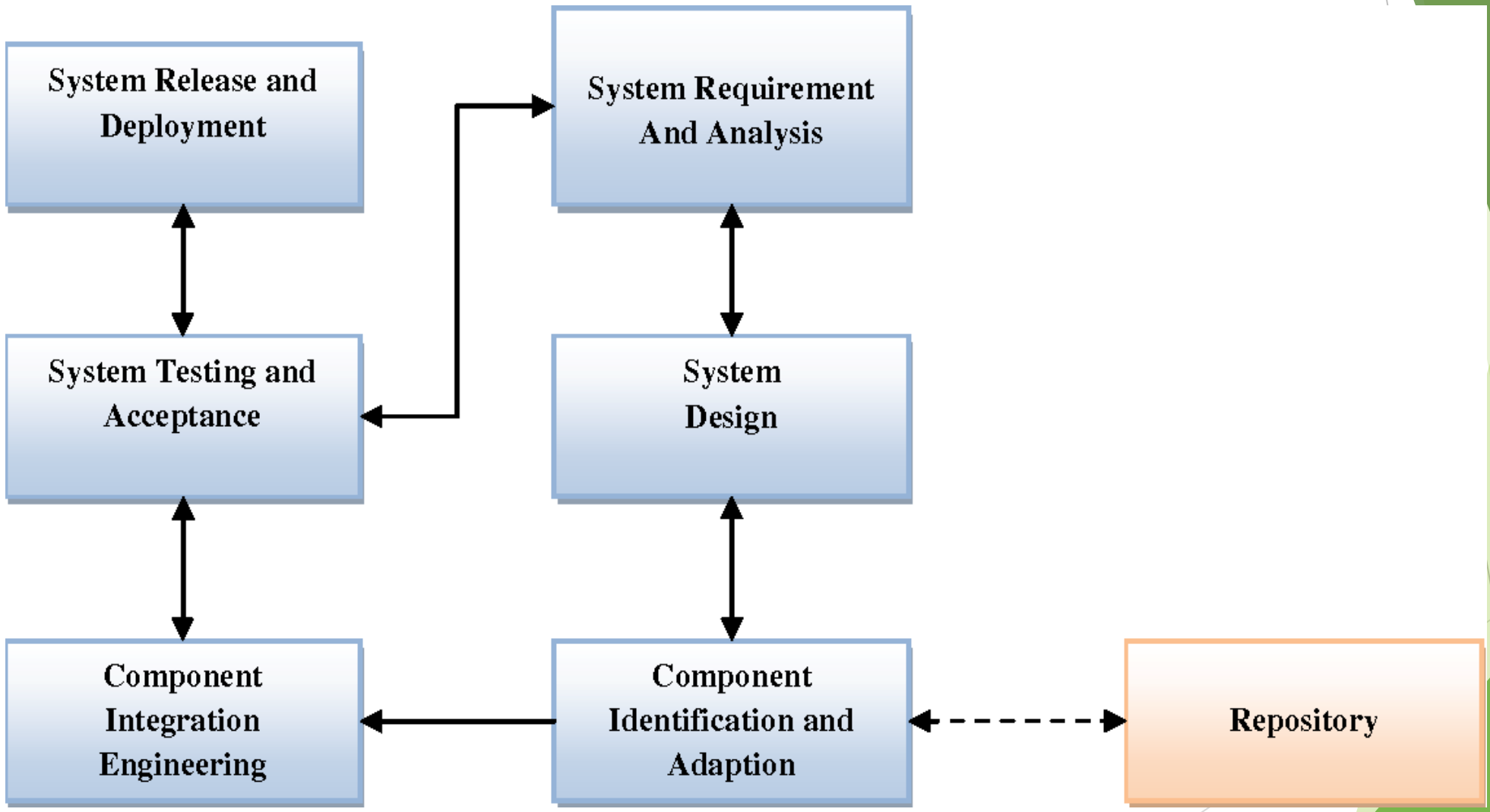
  methodological approach
- Unified Method--------diagrammatic representation(UML,Usecase)

Component-based software engineering

- It is a reuse-based approach to defining, implementing and composing loosely coupled independent components into systems.
- Component-based development emphasizes the design and development of software systems with the help of reusable software components.
- The primary objective of component-based architecture is to ensure component reusability.
- A component is a modular, portable, replaceable, and reusable set of well-defined functionality that encapsulates its implementation and transferring it as a higher-level interface.

- A set of pre-built, standardized software components are made available to fit a specific architecture for some application domain and the application is then assembled using these components.
- There are many standard component frameworks such as COM/DCOM, JavaBean, EJB, CORBA, .NET, web services, and grid services.

Component-based software engineering



- Advantages:
- Components-based systems are easier to assemble and therefore less costly to build than systems constructed from discrete components.
- Components-based systems also offers increased quality, accelerated development and reduced risk.
- It allows for component reuse
- A component-based approach accelerates development
- It easily integrates into the development process
- Component-based development optimizes the requirements design process
- It speeds up the transition from design to development.

- Disadvantages:
- Finding suitable components which fit the architectural design of the software to be developed may be sometimes difficult because gaps exist between the component features and the software requirements.
- Component-based development has not been widely adopted in domains of embedded systems because of the inability of this technology to cope with the important concerns of embedded systems like resource constraints, real time or dependability requirements
- Component-based software development is young, therefore long term maintainability is still unknown.

- **formal methods**
- The Formal Methods Model is an approach to software development that applies mathematical methods or techniques to the process of developing complex software systems.
- The approach uses a formal specification language to define each characteristic of the system.
- Such formal methods provide frameworks within which software developer can specify, develop, and verify systems in a systematic, rather than ad hoc manner.
- For improving reliability and robustness of design

- When formal methods are used during development, they provide a mechanism for eliminating many of the problems like ambiguity, incompleteness, and inconsistency that are difficult to overcome using other software engineering paradigms through the application of mathematical analysis.
- When formal methods are used during design, they serve as a basis for program verification and therefore enable you to discover and correct errors that might otherwise go undetected.
- Formal specifications can function as a guide to requirements.
- In analysis, formal methods provide the description of functions by which the program can be verified.

- Different types of Formal Specification Languages
- 1. Model Based Languages:
- In Model Based Languages, the specification is expressed as a system state model using well understood mathematical entities such as sets, relations, sequences and functions.
- Operations of a system are specified by defining how they affect the state of the system model. The most widely used notations for developing model based languages are Vienna Development Method (VDM).

- 2. Algebraic Specification Languages: Algebraic specification languages describe the behaviour of a system in terms of axioms that characterize its desired properties. Examples of algebraic specification languages include OBJ and the Common Algebraic Specification Language (CASL).
- 3. Process oriented Languages: In these process oriented languages processes are denoted and built up by expressions and elementary expressions, respectively, which describe particularly simple processes. Example of process oriented languages is Communicating Sequential Processes (CSP).

- Formal methods-------- levels
- 1. Formal Specification: During the formal specification phase, the Software Engineer rigorously defines a system using a formal specification language that has its own syntax and semantics and a set of relation.
- It describes what the system should do, not (necessarily) how the system should do it.
- Given such a specification, it is possible to use formal verification techniques to demonstrate that a candidate system design is correct with respect to the specification.
- This process of formal specification is similar to the process of converting a word problem into algebraic notation.

- 2. Formal development and verification: Formal development process involves iteratively refining a formal specification to produce the finished system.
- Formal Methods differ from other specification systems by their heavy emphasis on provability and correctness.
- By building a system using a formal specification, the designer is actually developing a set of theorems about his system.
- By proving these theorems correct, the formal methods ensures the correctness of the system.
- The process of proving or disproving properties of the software system against a formal specification is known as formal verification.

- 3. Implementation: Once the model has been specified and verified, it is implemented by converting the specification into code..

- Advantages:
- Discovers ambiguity, incompleteness, and inconsistency in the software.
- Offers defect-free software.
- Incrementally grows in effective solution after each iteration.
- This model does not involve high complexity rate.
- Formal specification language semantics verify self-consistency.

- Disadvantages:
- The development of formal models is currently quite time consuming and expensive.
- Because few software developers have the necessary background to apply formal methods, extensive training is required.
- It is difficult to use the models as a communication mechanism for technically unsophisticated customers.