UNIT-II

Software Life Cycle Models: A Few Basic Concepts, Waterfall Model and its Extensions – (Iterative, V Model, Prototyping, Incremental, Evolutionary Model), Rapid Application Development (RAD), Spiral Mode, Comparison of Different Life Cycle Models and Selecting an Appropriate Life cycle Model for a Project

- In contrast to the build and fix style, the software engineering approaches emphasise software development through a well-defined and ordered set of activities.
- These activities are graphically modelled (represented) as well as textually described and are variously called a s software life cycle model, software development life cycle (SDLC) model, and software development process model.

A FEW BASIC CONCEPTS

- Software life cycle
  - The life cycle of a software represents the series of identifiable stages through which it evolves during its life time.
    - Inception
    - developed and is released to the customers
    - start of the operation (also called maintenance) phase.
    - retired

Process versus methodology

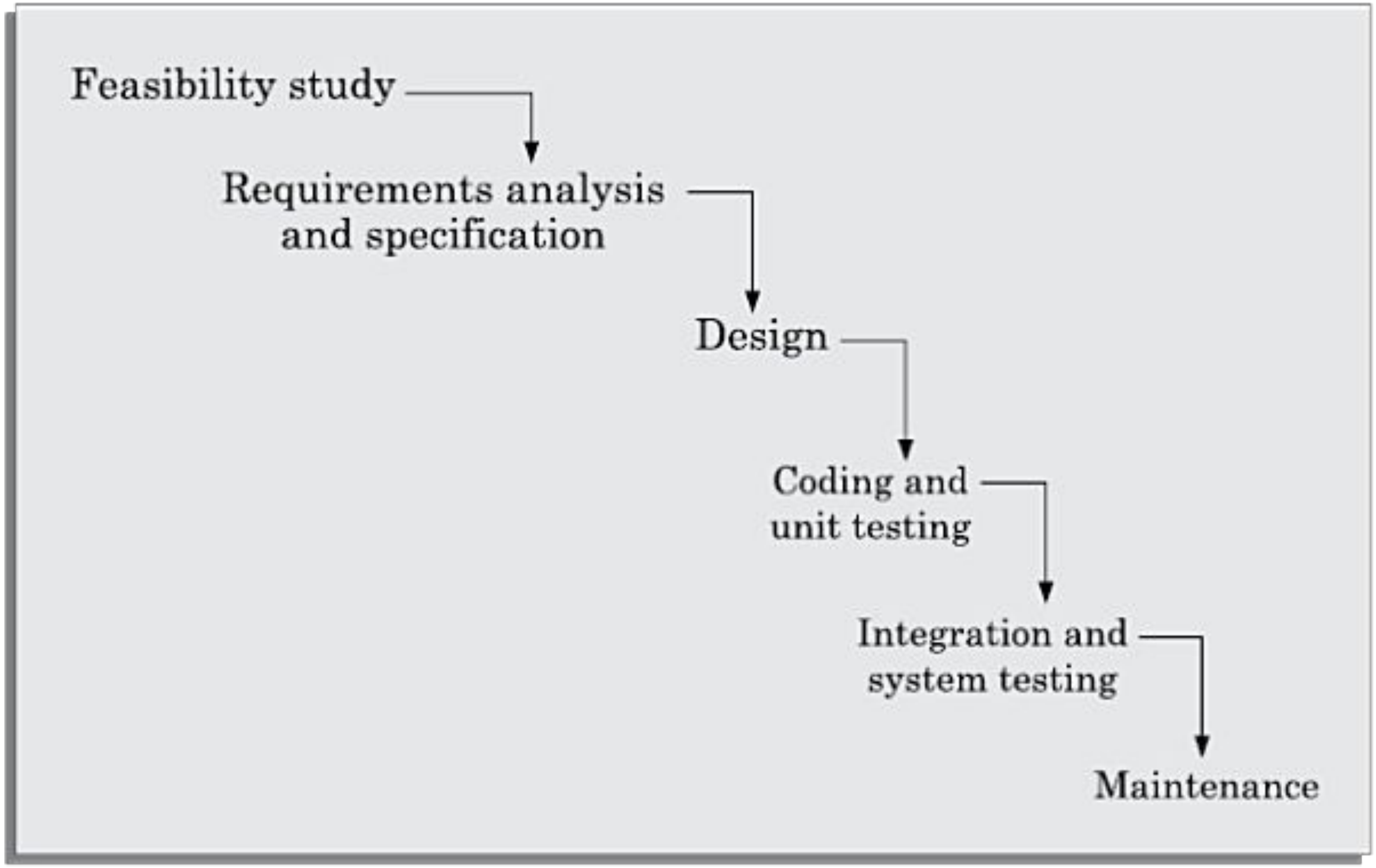
- A software development process has a much broader scope as compared to a software development methodology.
- A process usually describes all the activities starting from the inception of a software to its maintenance and retirement stages, or at least a chunk of activities in the life cycle.
- It also recommends specific methodologies for carrying out each activity.
- A methodology, in contrast, describes the steps to carry out only a single or at best a few individual activities.

- Why use a development process?
- Why document a development process?
- Phase entry and exit criteria

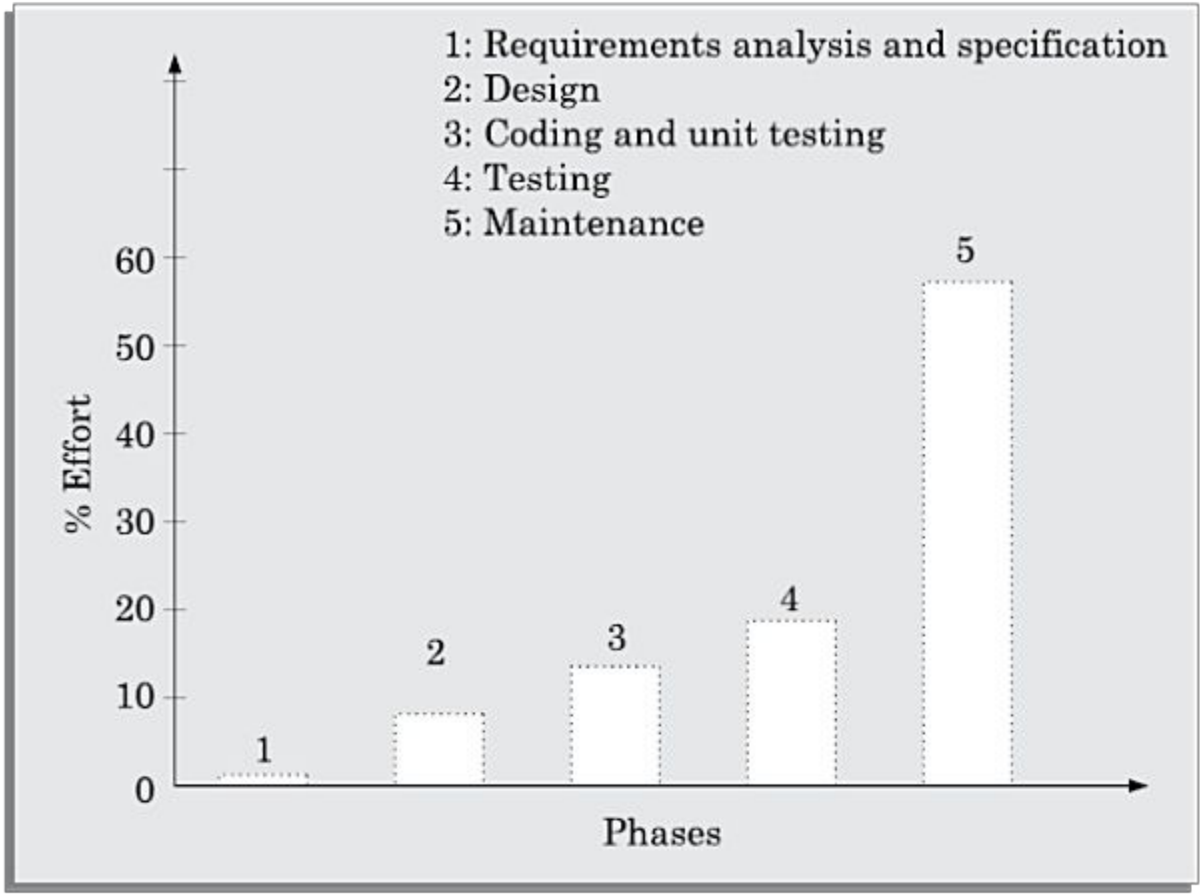
WATERFALL MODEL AND ITS EXTENSIONS

- Classical Waterfall Model
- Iterative Waterfall model
- V-Model
- Prototyping model
- Incremental Model
- Evolutionary Model

Classical Waterfall Model


Feasibility study
Requirements analysis and specification
Design
Coding and unit testing
Integration and system testing
Maintenance

Relative effort distribution among different phases of a typical product.



Feasibility study

- → The main focus of the feasibility study stage is to determine whether it would be financially and technically feasible to develop the software.
- → feasibility study involves carrying out several activities
  - Different data items that would be input to the system
  - The processing required to be carried out on these data
  - The output data required to be produced by the system
  - Various constraints on the development
- → These collected data are analysed to perform at the following
  - Development of an overall understanding of the problem
  - Formulation of the various possible strategies for solving the problem
  - Evaluation of the different solution strategies

Case Study

A mining company named Galaxy Mining Company Ltd. (GMC Ltd.) has mines located
at various places in India. It has about fifty different mine sites spread across eight states. The company employs a large number of miners at each mine site. Mining being a risky profession, the company intends to operate a special provident fund, which would exist in addition to the standard provident fund that the miners already enjoy. The main objective of having the special provident fund (SPF) would be to quickly distribute some compensation before the PF amount is paid.
According to this scheme, each mine site would deduct SPF installments from each miner every month and deposit the same to the central special provident fund commissioner (CSPFC). The CSPFC will maintain all details regarding the SPF installments collected from the miners. GMC Ltd. requested a reputed software vendor Adventure Software Inc. to undertake the task of developing the software for automating the maintenance of SPF records of all employees. GMC Ltd. has realised that besides saving manpower on book-keeping work, the software would help in speedy settlement of claim cases. GMC Ltd. indicated that the amount it can at best afford Rs. 1 million for this software to be developed and installed them

Adventure Software Inc. deputed their project manager to carry out the feasibility study. The project manager discussed with the top managers of GMC Ltd. to get an overview of the project. He also discussed with the field PF officers at various mine sites to determine the exact details of the project. The project manager identified two broad approaches to solve the problem. One is to have a central database which would be accessed and updated via a satellite connection to various mine sites. The other approach is to have local databases at each mine site and to update the central database periodically through a dial-up connection. This periodic updates can be done on a daily or hourly basis depending on the delay acceptable to GMC Ltd. in invoking various functions of the software. H e found that the second approach is very affordable and more fault-tolerant as the local mine sites can operate even when the communication link temporarily fails. In this approach, when a link fails, only the update of the central database gets delayed. Whereas in the first approach, all SPF work gets stalled at a mine site for the entire duration of link failure. The project manager quickly analysed t h e overall database functionalities required, the user- interface issues, and the software handling communication with the mine sites. From this analysis, he estimated the approximate cost to develop the software. He found that a solution involving maintaining local databases at the mine sites and periodically updating a central database is financially and technically feasible. The project manager discussed this solution with the president of GMC Ltd., who indicated that the proposed solution would be acceptable to

Requirements analysis and specification

- To understand the exact requirements of the customer and to document them properly.
- Requirements gathering and analysis
- Requirements specification (SRS)

Design

- During the design phase the software architecture is derived from the SRS document
- Popular Design approaches:

→ procedural design approach

→ object-oriented design approach

- **Procedural design approach:**

→ based on the data flow-oriented design approach

→ It consists of two important activities;

  - Structured analysis: of the requirements specification is carried out where the detailed structure of the problem is examined.
  - Structured design step: where the results of structured analysis are transformed into the software design.

- **Object-oriented design approach:**

→ Various objects that occur in the problem domain and the solution domain are first identified and the different relationships that exist among these objects are identified.

→ The object structure is further refined to obtain the detailed design.

→ The OOD approach is credited to have several benefits:

Coding and unit testing

- To translate a software design into source code and to ensure that individually each function is working correctly.
- Coding phase is also sometimes called the *implementation phase*
- *Each component of the design is implemented as a program module.*
- *The end-product of this phase is a set of program modules that have been individually unit tested.*
- *The main objective of unit testing is to determine the correct working of the individual modules.*
- *The specific activities carried out during unit testing include designing test cases, testing, debugging to fix problems, and management of test cases.*

Integration and system testing

- Integration testing is carried out to verify that the interfaces among different units are working satisfactorily.
- The goal of system testing is to ensure that the developed system conforms to the requirements that have been laid out in the SRS document.
- System testing usually consists of three different kinds of testing activities:
  - → α -testing: is the system testing performed by the development team.
  - → β-testing: This is the system testing performed by a friendly set of customers.
  - → Acceptance testing: After the software has been delivered, the customer performs system testing to determine whether t o accept the delivered software or to reject it.

Maintenance

- The total effort spent on maintenance of a typical software during its operation phase is much more than that required for developing the software itself.
- Many studies shown that the ratio of relative effort of developing a typical software product and the total effort spent on its maintenance is roughly 40:60.
- Maintenance is required in the following three types of situation
  - **Corrective maintenance**: This type of maintenance is carried out to correct errors that were not discovered during the product development phase.
  - **Perfective maintenance:** This type of maintenance is carried out to improve the performance of the system, or to enhance the functionalities of the system based on customer's requests.
  - **Adaptive maintenance:** Adaptive maintenance is usually required for porting the software to work in a new environment.
    - For example, porting may be required to get the software to work on a new computer platform or with a new operating system

Shortcomings of the classical waterfall model

- No feedback paths
- Difficult to accommodate change requests
- Inefficient error corrections
- No overlapping of phases

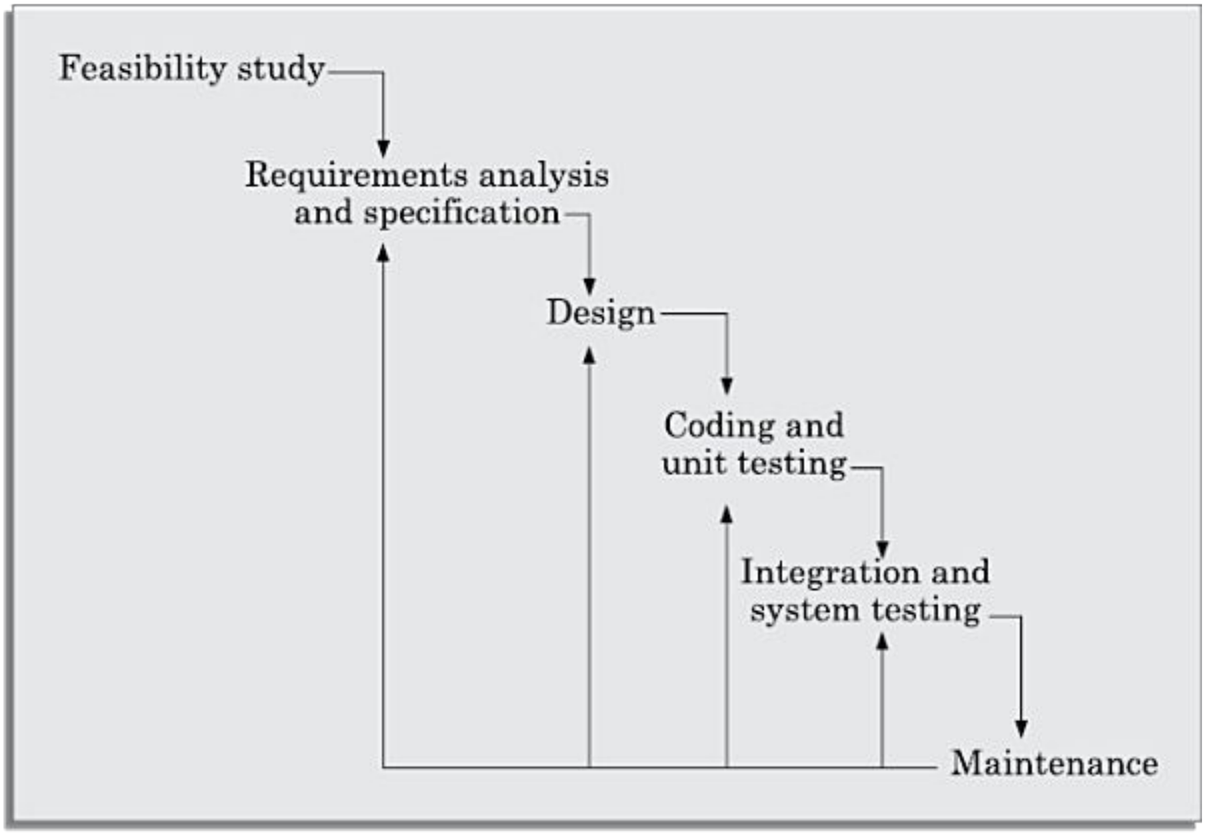
Is the classical waterfall model useful at all?

- Irrespective of the life cycle model that is actually followed for a product development, the final documents are always written to reflect a classical waterfall model of development,
- so that comprehension of the documents becomes easier for any one reading the document.

Iterative Waterfall Model

- Provides feedback paths from every phase to its preceding phases.

Iterative waterfall model.

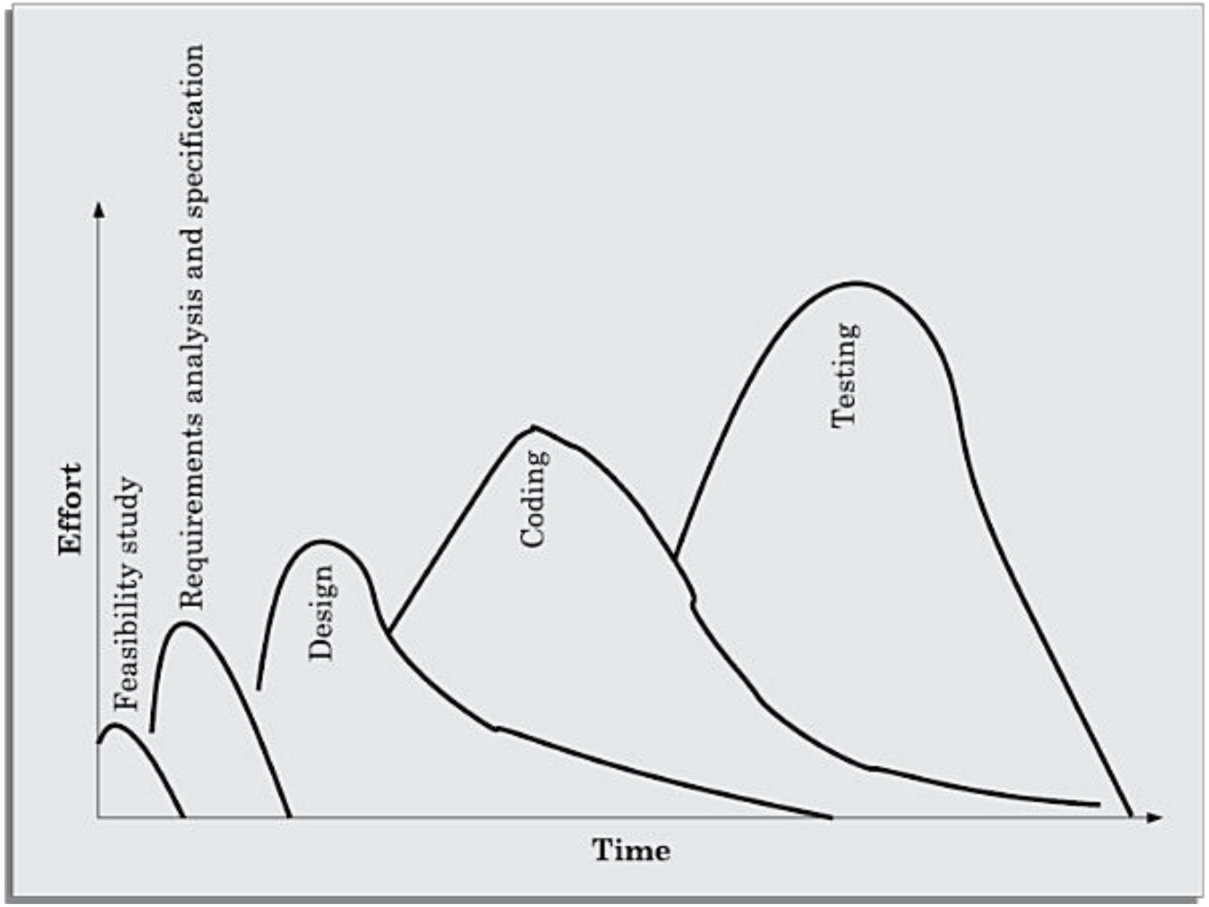


Feasibility study
Requirements analysis and specification
Design
Coding and unit testing
Integration and system testing
Maintenance

Phase containment of errors

The principle of detecting errors as close to their points of commitment as possible is
known as phase containment of errors.

Phase overlap

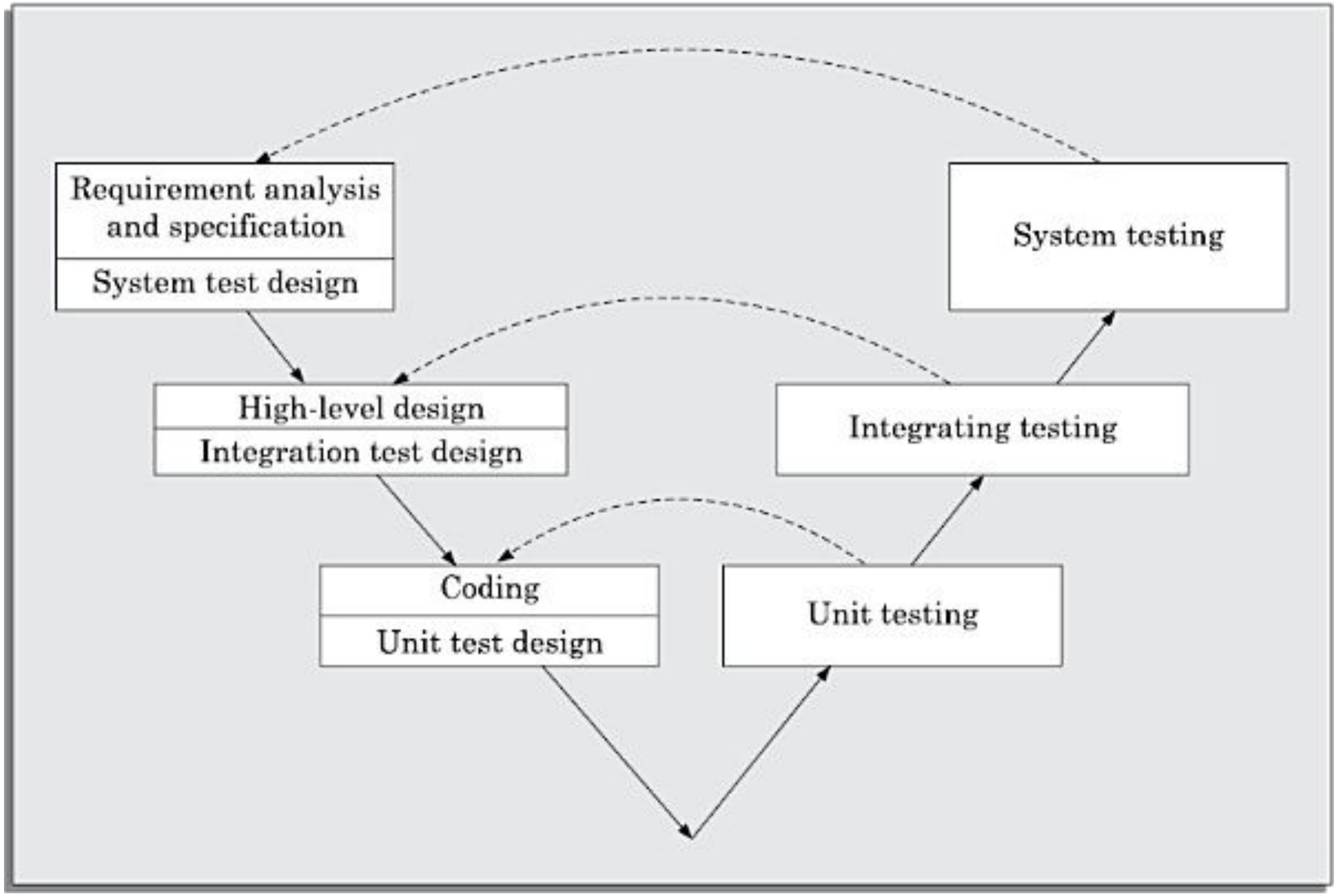


- In spite of the best effort to detect errors in the same phase in which they are committed, some errors escape detection and are detected in a later phase.
- These subsequently detected errors cause the activities of some already completed phases to be reworked.
- Because of this rework, phases overlap with other phases.
- If strict phase transitions are maintained, then the team members who complete their work early would idle waiting for the phase to be complete, and are said to be in a **blocking state**.(this is a cuase for wastage of resources)

Shortcomings of the iterative waterfall model

- Difficult to accommodate change requests:
- Incremental delivery not supported:
- Phase overlap not supported:
- Error correction unduly expensive:
- Limited customer interactions:
- Heavy weight:
- No support for risk handling and code reuse:

V-Model


Requirement analysis and specification
System test design
High-level design
Integration test design
Coding
Unit test design
Unit testing
Integrating testing
System testing

- V-model is a variant of the waterfall Model
- This model gets its name from its visual appearance
- verification and validation activities are carried out throughout the development life cycle.
- This model is considered with development of safety-critical software that are required to have high reliability
- In each development phase, along with the development of a work product, test case design and the plan for testing the work product are
- carried out.
- Actual testing is carried out in the validation phase.
- In the validation phase, testing is carried out in three steps:

  → unit

  → Integration

V-model versus waterfall model

- in the V-model testing activities are spread over the entire life cycle but in the water fall model they are restricted to testing phases only

Advantages of V-model

- much of the testing activities (test case design, test planning, etc.) are carried out in parallel with the development activities. That leads to shorter testing phase and an overall faster product development as compared to the iterative model
- Since test cases are designed when the schedule pressure has not built up, the quality of the test cases are usually better.
- The test team is reasonably kept occupied throughout the development cycle, This leads to more efficient manpower utilisation.
- As the test team involved in the project from the beginning, they build up a good understanding of the development artifacts, and this in turn, helps them to carry out effective testing of the software.

Disadvantages of V-model

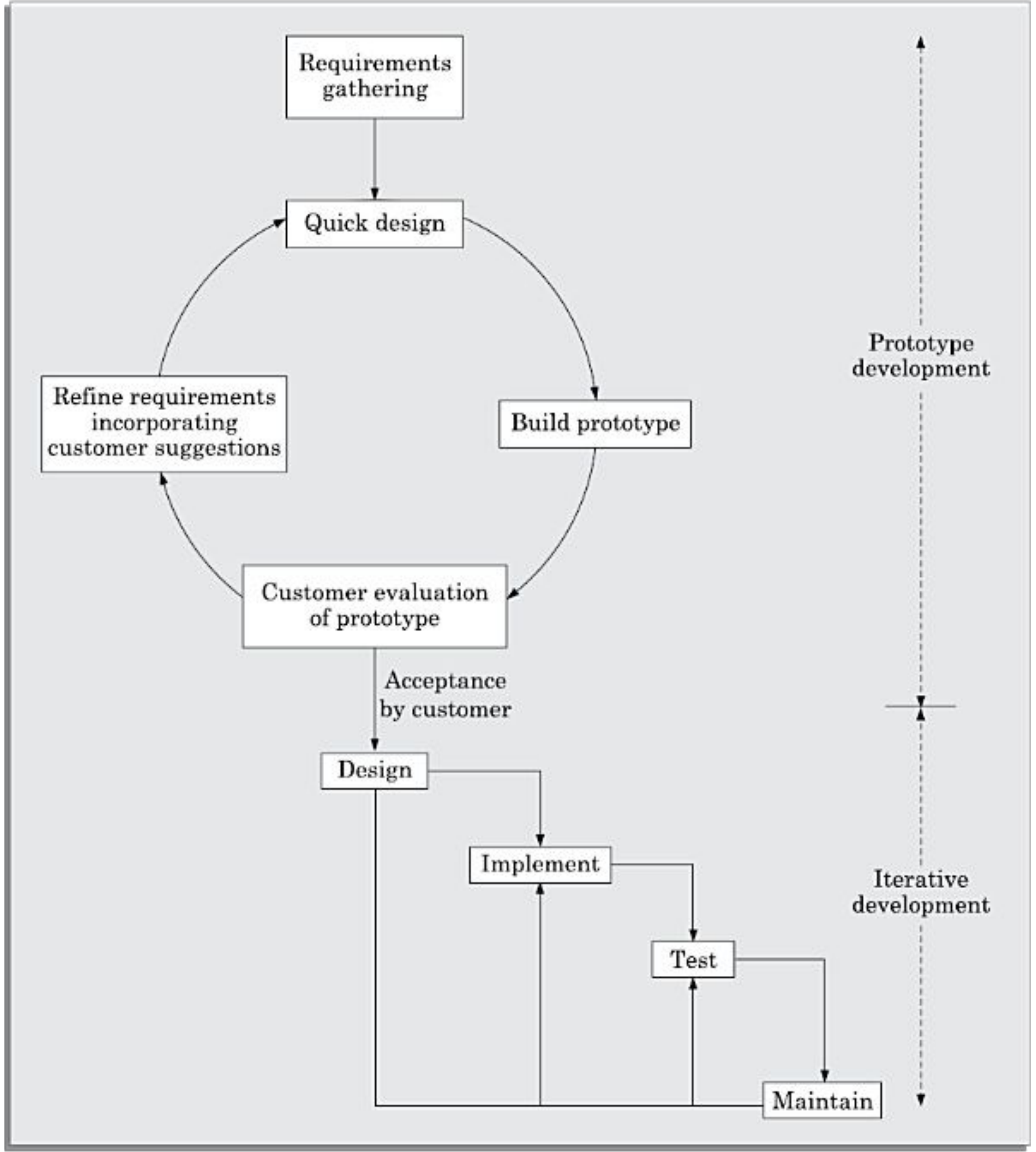
- Being a derivative of the classical waterfall model, this model inherits most of the weaknesses of the waterfall model.

Prototyping Model

- This model suggests building a working prototype of the system, before development of the actual software.
- A prototype can be built very quickly by using severalshortcuts (developing inefficient, inaccurate, or dummy functions).
- The term **rapid prototyping** is used when software tools are used for prototype construction

Necessity of the prototyping model

- The GUI part of a software system is almost always developed using the prototyping model.
- The prototyping model is especially useful when the exact technical solutions are unclear to the development team.
- An important reason for developing a prototype is that it is impossible to “get it right” the first time


Requirements gathering
Quick design
Build prototype
Customer evaluation of prototype
Refine requirements incorporating customer suggestions
Acceptance by customer
Design
Implement
Test
Maintain
Prototype development
Iterative development

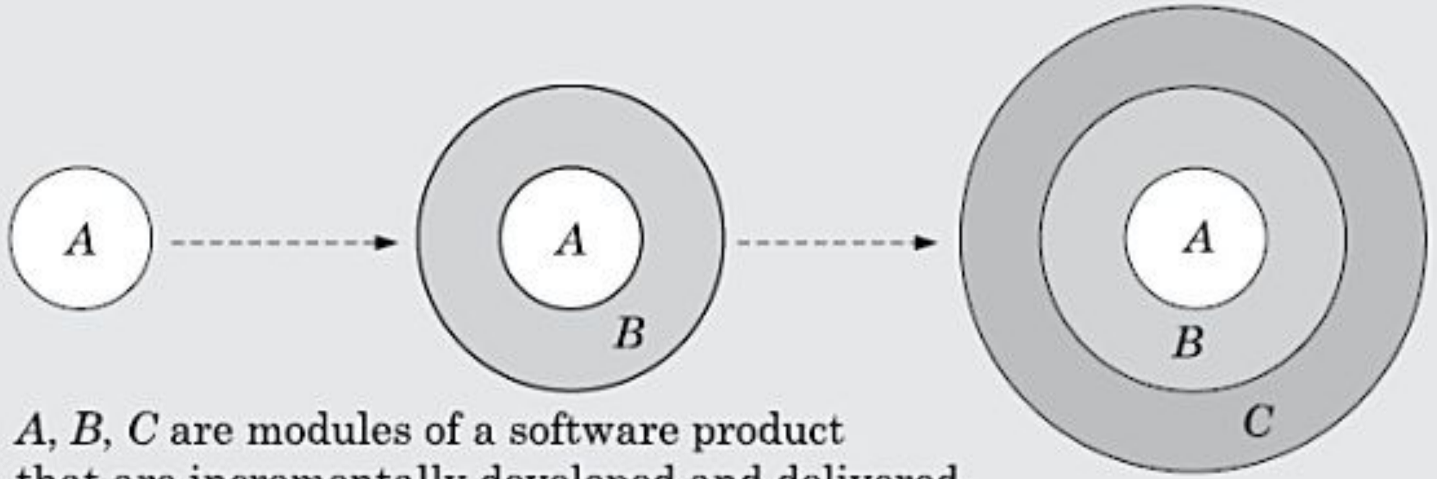
Life cycle activities of prototyping model

- Prototype development:
- Iterative development
- Even though the construction of a throwaway prototype might involve incurring additional cost, for systems with unclear customer requirements and for systems with unresolved technical issues, the overall development cost usually turns out to be lower compared to an equivalent system developed using the iterative waterfall model.

- **Strengths of the prototyping model:**
  - This model is the most appropriate for projects that suffer from technical and requirements risks.
- **Weaknesses of the prototyping model:**
  - The prototype model can increase the cost of development for projects that are routine development work and do not suffer from any significant risks.
  - prototyping model is effective only for those projects for which the risks can be identified upfront before the development starts.

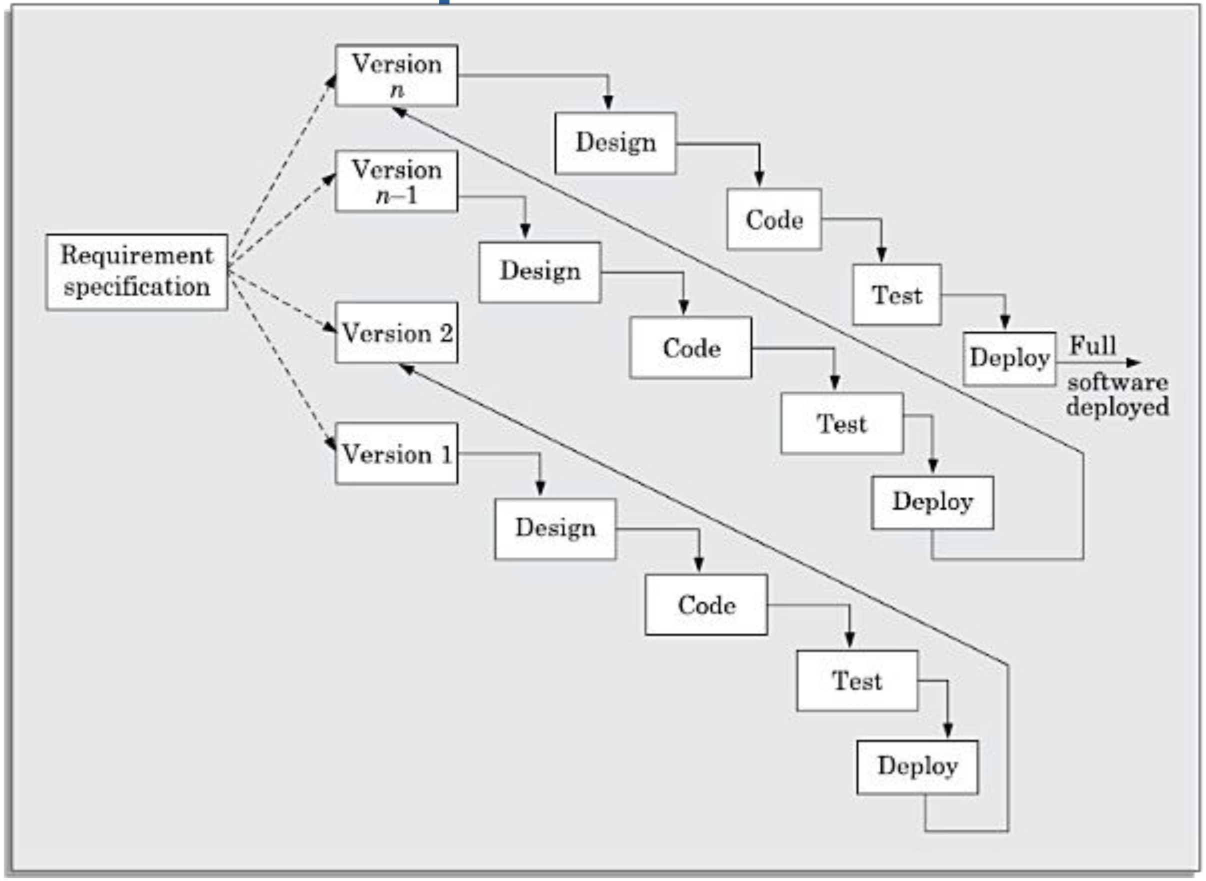
Incremental Development Model

- Also known as **successive versions model** and sometimes as the **incremental model.**
- In this life cycle model, first a simple working system implementing only a few basic features is built and delivered to the customer.
- Over many successive iterations successive versions are implemented and delivered to the customer until the desired system is realised.



A, *B*, *C* are modules of a software product that are incrementally developed and delivered.

Life cycle activities of incremental development model

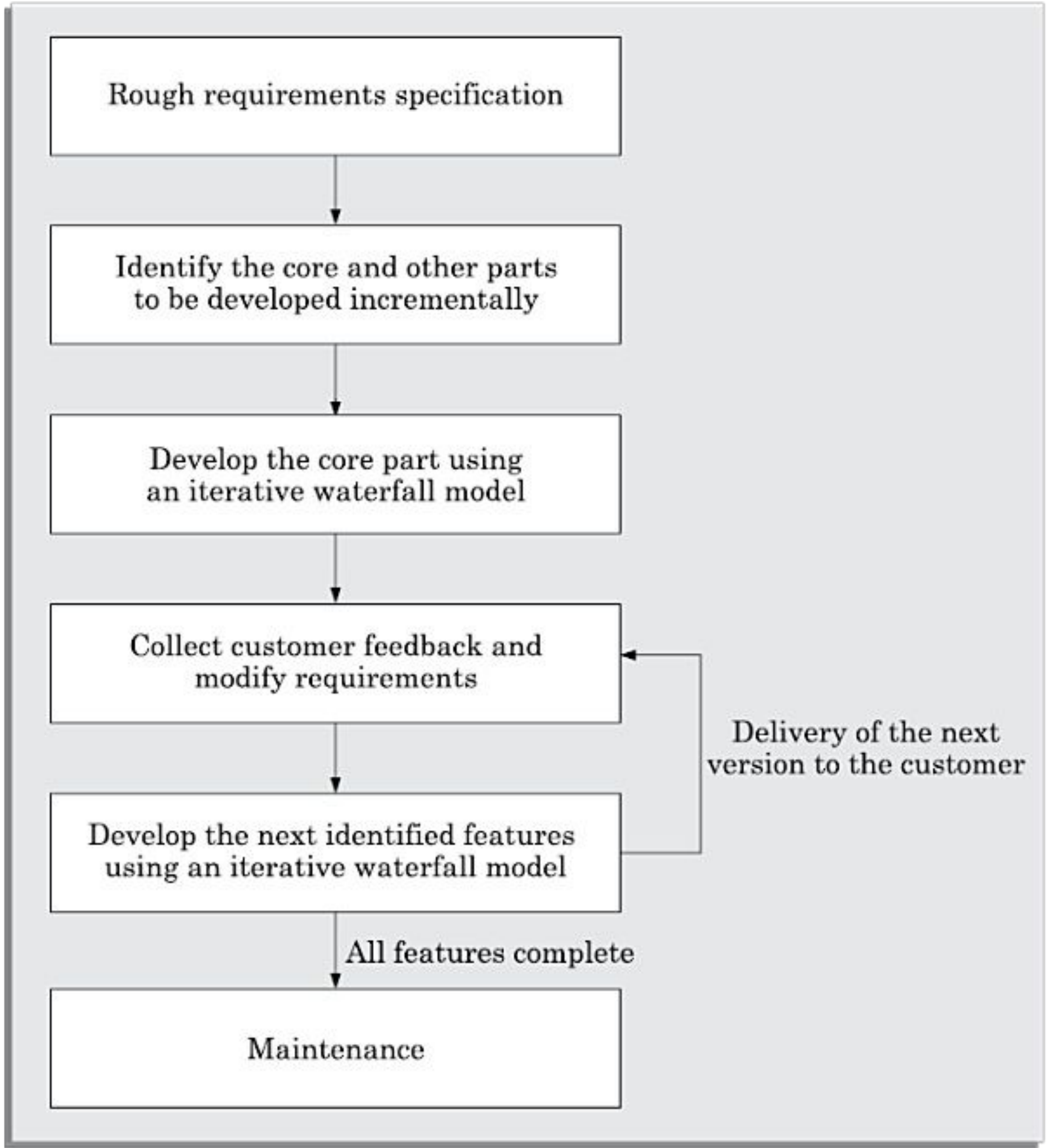


Advantages

- Error reduction:
- Incremental resource deployment:

Evolutionary Model

- In the incremental development model, complete requirements are first developed and the SRS document prepared.
- In contrast, in the evolutionary model, the requirements, plan, estimates, and solution evolve over the iterations, rather than fully defined and frozen in a major up-front specification effort before the development iterations begin.
- It is sometimes referred as design a little, build a little, test a little, deploy a little model.


Rough requirements specification
Identify the core and other parts to be developed incrementally
Develop the core part using an iterative waterfall model
Collect customer feedback and modify requirements
Develop the next identified features using an iterative waterfall model
Delivery of the next version to the customer
All features complete
Maintenance

Advantages

- Effective elicitation of actual customer requirements:
- Easy handling change requests:

Disadvantages

- Feature division into incremental parts can be non-trivial
- Ad hoc design

Applicability of the evolutionary model

The evolutionary model is well-suited to use in object-oriented software
development
projects.

RAPID APPLICATION DEVELOPMENT (RAD)

- In this model prototypes are constructed, and incrementally the features are developed and delivered to the customer.
- But unlike the prototyping model, the prototypes are not thrown away but are enhanced and used in the software construction
- The major goals of the RAD model are as follows:
  - → To decrease the time taken and the cost incurred to develop software systems.
  - → To limit the costs of accommodating change requests.
  - → To reduce the communication gap between the customer and the developers.

Working of RAD

- In the RAD model, development takes place in a series of short cycles or iterations.
- At any time, the development team focuses on the present iteration only, and therefore plans are made for one increment at a time.
- The time planned for each iteration is called a time box.
- Each iteration is planned to enhance the implemented functionality of the application by only a small amount.
- During each time box, a quick-and-dirty prototype-style software for some functionality is developed. The customer evaluates the prototype and gives feedback on the specific improvements that may be necessary.
- The prototype is refined based on the customer feedback. Please note that the prototype is not meant to be released to the customer for regular use though.

How does RAD facilitate accommodation of change requests?

- The customers usually suggest changes to a specific feature only after they have used it.
- Since the features are delivered in small increments, the customers are able to give their change requests pertaining to a feature already delivered.
- Incorporation of such change requests just after the delivery of an incremental feature saves cost as this is carried out before large investments have been made in development and testing of a large number of features.

How does RAD facilitate faster development?

- Visual style of development.
- Use of reusable components.

Applicability of RAD Model

- Customised software
- Non-critical software
- Highly constrained project schedule
- Large software

Application characteristics that render RAD unsuitable

- The RAD style of development is not advisable if a development project has one or more of the following characteristics:
  - ➔ Generic products (wide distribution)
  - ➔ Requirement of optimal performance and/or reliability
  - ➔ Lack of similar products
  - ➔ Monolithic entity

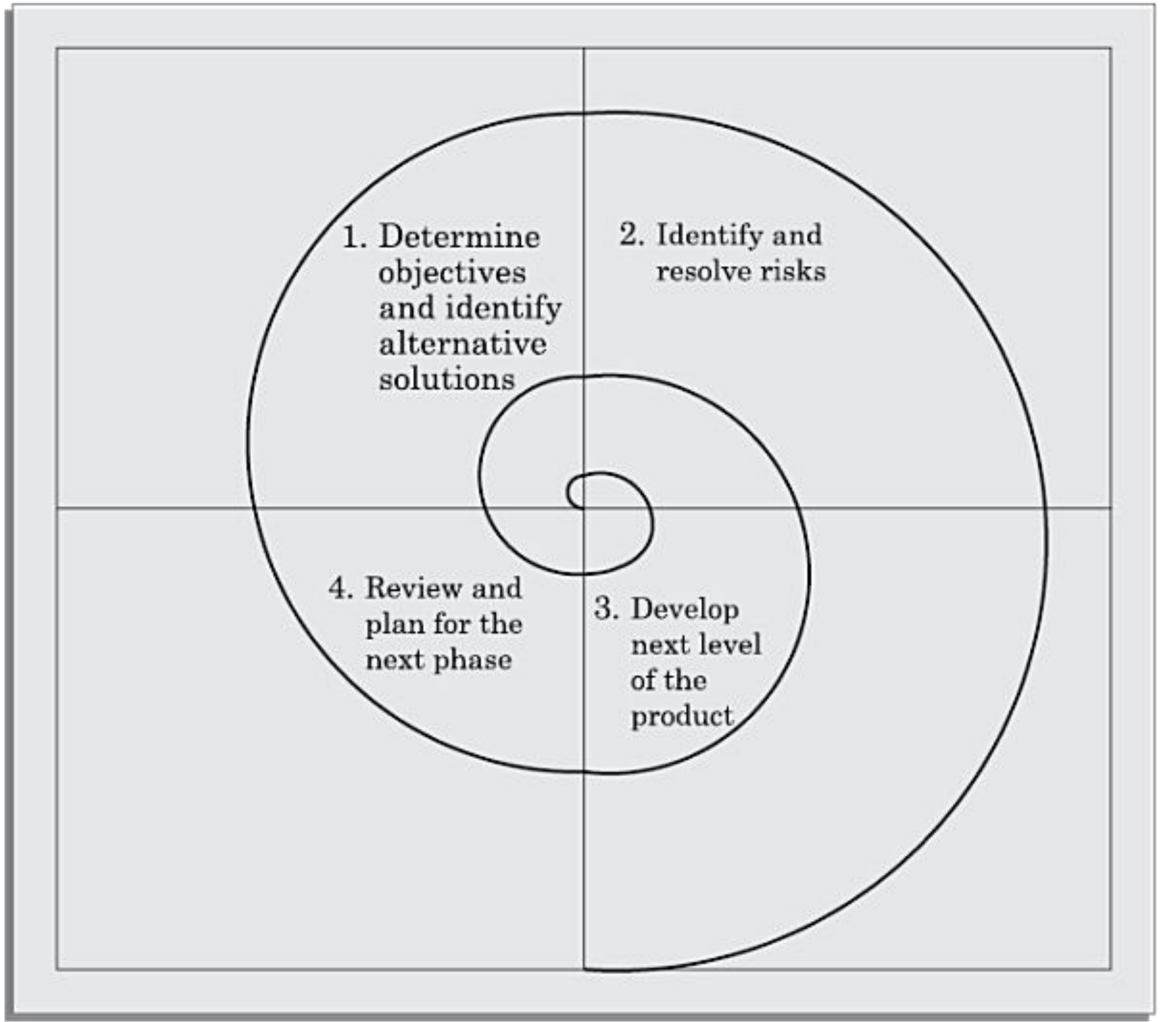
Comparison of RAD with Other Models

- RAD versus prototyping model
  - Though RAD is expected to lead to faster software development compared to the traditional models (such as the prototyping model), though the quality and reliability would be inferior.

- RAD versus iterative waterfall model
- RAD:
  - functionalities are developed incrementally through heavy code and design reuse.
  - customer feedback is obtained on the developed prototype after each iteration and based on this the prototype is refined. Thus, it becomes easy to accommodate any request for requirements changes.
- Iterative waterfall model:
  - all the functionalities of a software are developed together
  - the iterative waterfall model does not support any mechanism to accommodate any requirement change requests.
  - Use of the iterative waterfall model leads to production of good quality documentation which can help during software maintenance.
  - Also, the developed software usually has better quality and reliability than that developed using RAD.

- RAD versus evolutionary model
  - Both Supports Incremental Development
  - In RAD each increment results in essentially a quick and dirty prototype, whereas in the evolutionary model each increment is systematically developed using the iterative waterfall model.
  - In the RAD model, software is developed in much shorter increments compared to the evolutionary model

SPIRAL MODEL


1. Determine objectives and identify alternative solutions
2. Identify and resolve risks
3. Develop next level of the product
4. Review and plan for the next phase

- This model gets its name from the appearance of its diagrammatic representation that looks like a spiral with many loops
- Each loop of the spiral is called a phase of the software process.
- The exact number of phases through which the product is developed can be varied by the project manager depending upon the project risks.
- A prominent feature of the spiral model is handling unforeseen risks that can show up much after the project has started.
- in the spiral model prototypes are built at the start of every phase.
- Each phase of the model is represented as a loop in its diagrammatic representation.
- Over each loop, one or more features of the product are elaborated and analysed and the risks at that point of time are identified and are resolved through prototyping.

Based on this, the identified features are implemented

Risk handling in spiral model

- A risk is essentially any adverse circumstance that might hamper the successful completion of a software project.
- As an example, consider a project for which a risk can be that data access from a remote database might be too slow to be acceptable by the customer.
- This risk can be resolved by building a prototype of the data access subsystem and experimenting with the exact access rate. If the data access rate is too communication scheme can be deployed to overcome the slow dataaccess rate.

Phases of the Spiral Model

- Each phase in this model is split into four sectors (or quadrants)
- Quadrant 1: 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/pros of the spiral model

- For projects having many unknown risks that might show up as the development proceeds, the spiral model would be the most appropriate development model to follow.

Disadvantages/cons of the spiral model

- To the developers of a project, the spiral model usually appears as a complex model to follow, since it is risk-driven and is more complicated phase structure than the other models we discussed
- It is not very suitable for use in the development of outsourced projects, since the software risks need to be continually assessed as it is developed.

A COMPARISON OF DIFFERENT LIFE CYCLE MODELS

- The iterative waterfall model is probably the most widely used software development model so far. This model is simple to understand and use.
- However, this model is suitable only for well-understood problems, and is not
- suitable for development of very large projects and projects that suffer from large number of risks.

- The prototyping model is suitable for projects for which either the user requirements or the underlying technical aspects are not well understood, however all the risks can be identified before the project starts.
- This model is especially popular for development of the user interface part of projects.

- The evolutionary approach is suitable for large problems which can be decomposed into a set of modules for incremental development and delivery.
- This model is also used widely for object-oriented development projects.
- Of course, this model can only be used if incremental delivery of the system is acceptable to the customer.

- The spiral model is considered a meta model and encompasses all other life cycle models.
- Flexibility and risk handling are inherently built into this model.
- The spiral model is suitable for development of technically challenging and large software that are prone to several kinds of risks that are difficult to anticipate at the start of the project.
- However, this model is mu ch more complex than the other models—this is probably a factor deterring its use in ordinary projects.

Selecting an Appropriate Life Cycle Model for a Project

- Characteristics of the software to be developed
- Characteristics of the development team
- Characteristics of the customer