Unit : 1
Software Development Process

1.1 Software

❖ **Definition:**

- Anything that can be stored electronically is called software.
- **(IEEE Definition):** Software is the "Collection of computer programs, procedures, rules, associated documents and concerned data with the operation of data processing system."
- It also includes representations of pictorial, video, and audio information.
- Software is of two types:
  - ***System software:*** it is responsible for controlling, integrating the hardware components of a system so the software and the users can work with them.

    Example: Operating system
  - ***Application software:*** it is used to accomplish some specific task. It should be collection of small programs.

    Example: Microsoft word, Excel, Railway reservation system.
- Software is logical rather than physical.

Software Characteristics

- These are the attributes reflect the quality of a software product.
- Following are the characteristics of good software. (Qualities for good software).
    - ➥ *Understandability*
    - ➥ *Cost*
    - ➥ *Maintainability*
    - ➥ *Modularity*
    - ➥ *Functionality*
    - ➥ *Reliability*
    - ➥ *Portability*
    - ➥ *Correctness*
    - ➥ *Documentation*
    - ➥ *Reusability*
    - ➥ *interoperability*

Software Characteristics

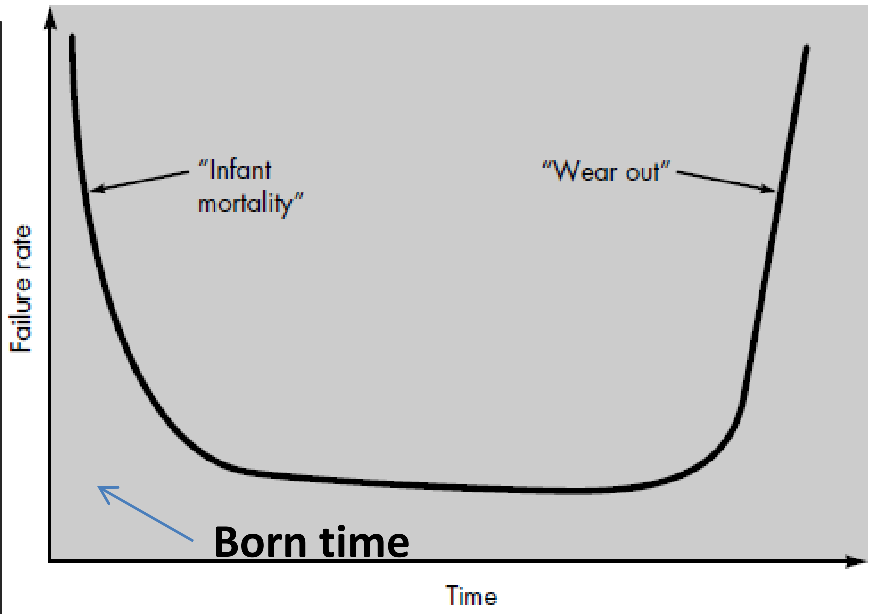
❖ **Some special characteristics of s/w:**

❑ **Software doesn't wear out (નકામુ થવું).**

– Hardware can damage after running time. It can be affected by environmental effects. So the failure rate rises.

-“**bathtub curve**” shows hardware failure
- there are three phases in h/w life

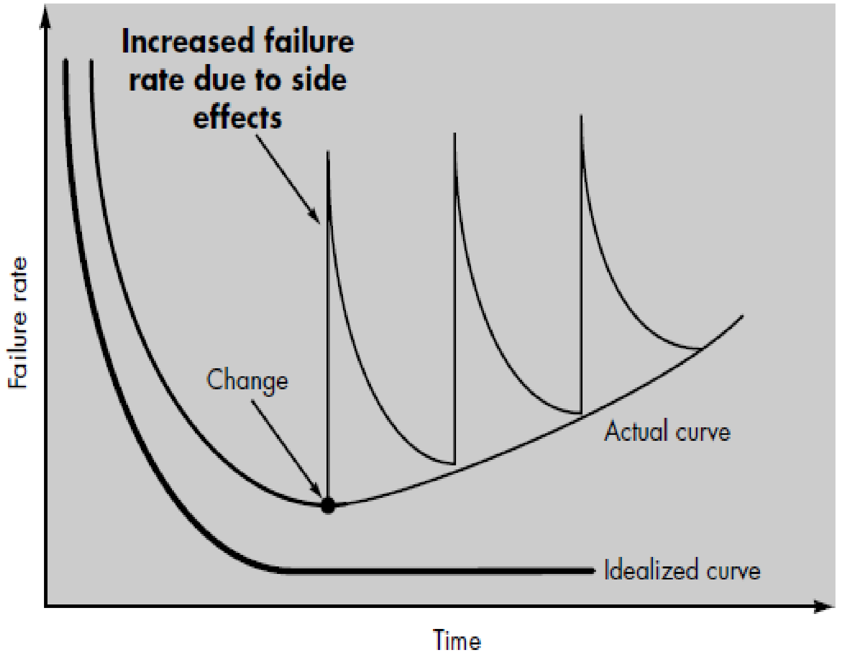
- initially failure rate is much more. But after testing and defects are corrected, failure rate come down.
- In it, h/w is much more useful and chance of failure is quite low.
- As time passes, however, the failure rate rises again as hardware components suffer from the affects of dust, vibration, abuse, temperature extremes, and many other environmental factors.
- So simply, hardware does wear out.



H/W failure curve

Software Characteristics

- Software is not highly affected by environmental effects. The "idealized curve" shows software failure.
- In the early stage, due to lot many errors, software could have high failure. But it becomes reliable as time passes instead of wearing out. Software become reliable.
- Software may be retired due to new requirements, new expectations etc.
- Hence, software doesn't wear out, but it may be deteriorate.



S/W failure curve

❑ S/W is engineered, not manufactured.

Once a product is manufactured, it is not easy to modify it, change. While in case of software we can easily change or modify or change it for later use.

Even making multiple copies of software is a very easy.

Software Characteristics

In hardware, costing is due to assembly of raw material and other processing expenses while in software development no assembly needed like hardware. Hence, software is not manufactured as it is developed or it is engineered.

- **Reusability of components.**

  Self description.

- **Software is flexible for custom built.**

  A program can be developed to do anything. Any kind of change needed in software easily done.

  A software program or product can be built on user requirements basis or custom built.

1.2 Software Myths (ખોટી માન્યતા)

- Software myths propagated misinformation and confusion.
- There are many myths of software and software engineering in software development community.
- **Myth: Software is easy to change.**
- Reality: ----
- **Myth: Outsourcing of s/w to a third party can relax the customers.**
- Reality: ---
- **Myth: Software can work right the first time.**
- Reality: ---
- **Myth: Increasing of software reliability will increase software safety.**
- Reality: ---

1.2 Software Myths

- **Myth: Reusing software increase safety.**
- Reality: ---
- **Myth: Best software is one which has more features.**
- Reality: ---
- **Myth: Testing of software will remove all errors.**
- Reality: ---
- **Myth: once the project is working, job is done.**
- Reality: ---

1.3 Software Engineering

❖ Various Definition:

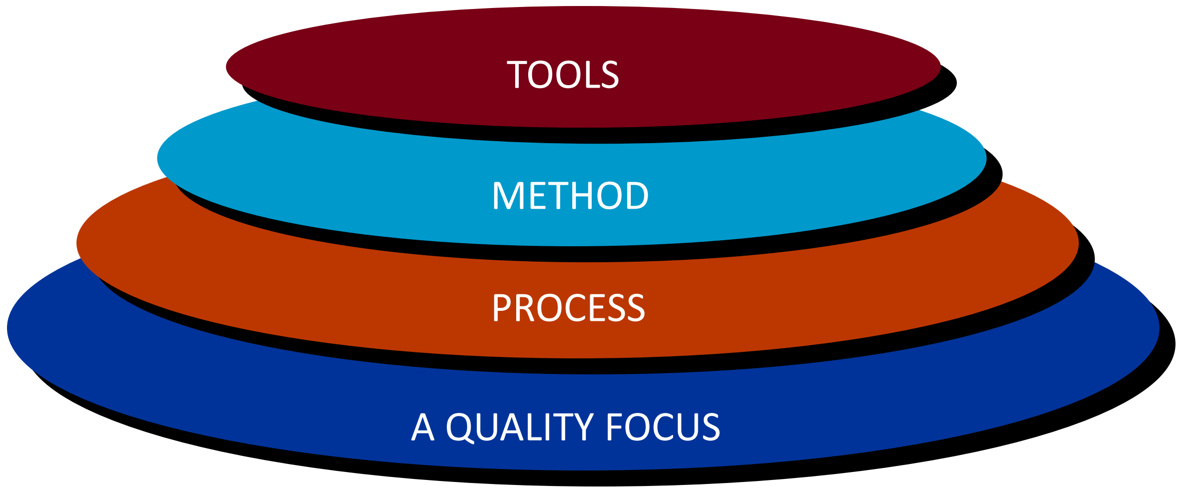
- **SE is an engineering discipline that covers all aspects of s/w from specification to maintenance.**
- **SE is an engineering discipline that delivers high quality s/w at agreed cost & in planed schedule.**
- SE provide framework that guides the s/w engineers to develop the software.
- SE tells how s/w will work with machines.
- SE covers technical and management issues.
- Three main aspects of SE is → **(Quality S/W at agreed cost in schedule time)**
  - Provide quality product
  - Expected cost
  - Complete work on agreed schedule
- SE discipline began since 5 decade and provide solution to s/w crisis.

Software Engineering (Cont…)

- SE is the establishment and use of sound engineering principles in order to obtain economically s/w that is reliable and work efficiently on real machines.
- **(IEEE Definition)** → "Software engineering is the application of a symmetric , disciplined and quantifiable approach to the development, operation and maintenance of software."
- **(Sommerville)**: Software Engineering is concerned with the theories, methods and tools to develop the software products in a cost effective way.

1.3 Software Engineering Layered approach

- Software engineering can be viewed as a layered technology. Actually software engineering is totally a layered technology.
- It encompasses process, methods, tools that enables a s/w product to be built in a timely manner.
- Four layers are there.
  - Quality
  - Process
  - Method
  - Tools



SE Layers

Software Engineering Layered approach

- **A Quality focus Layer**
  - SE mainly focuses on quality product.
  - It checks whether the output meets with its requirement specifications or not.
  - Every organization should maintain its total quality management.
  - This layer supports software engineering.
- **Process Layer**
  - It is the heart of the SE.
  - It is a foundation layer for development.
  - s/w process is a set of activities together if ordered and performed properly, the desired result would be produced.
  - Define framework activities.
  - Main idea is → ***to deliver s/w in a timely manner.***
- **Method Layer**
  - It describes 'how-to' build software product.
  - It creates SE environment to software product using CASE tools.
- **Tools layer**
  - It provides support to below layers.
  - Execute process in proper manner.

Need of Software Engineering?

- To help developers ***to obtain high quality software product***.
- To develop the product ***in appropriate manner*** using life cycle models.
- To acquire skills ***to develop large programs***.
- To acquire skills ***to be a better programmer.***
- To provide a software product ***in a timely manner.***
- To provide ***a quality software product.***
- To provide a software product ***at a agreed cost.***
- To develop ability ***to solve complex programming problems.***
- Also learn techniques of: specification, design, user interface development, testing, project management, etc.

1.4 Software Development

- Software development is the process of developing software through **successive phases** in an orderly way.
- This process includes **not only the actual writing of code** but also the preparation of requirements and objectives, the design of what is to be coded, and confirmation that what is developed has met objectives.
- In other words, Software development is the computer programming, documenting, testing and bug fixing involved in creating and maintaining application and frameworks involved in a software release life cycle and resulting in a software product.

- **Three most common being for software development.**
  - To meet specific needs of **specific clients**.
  - To meet a perceived need of **some set of potential users**.
  - To develop for **personal use.**

1.4 Software Development

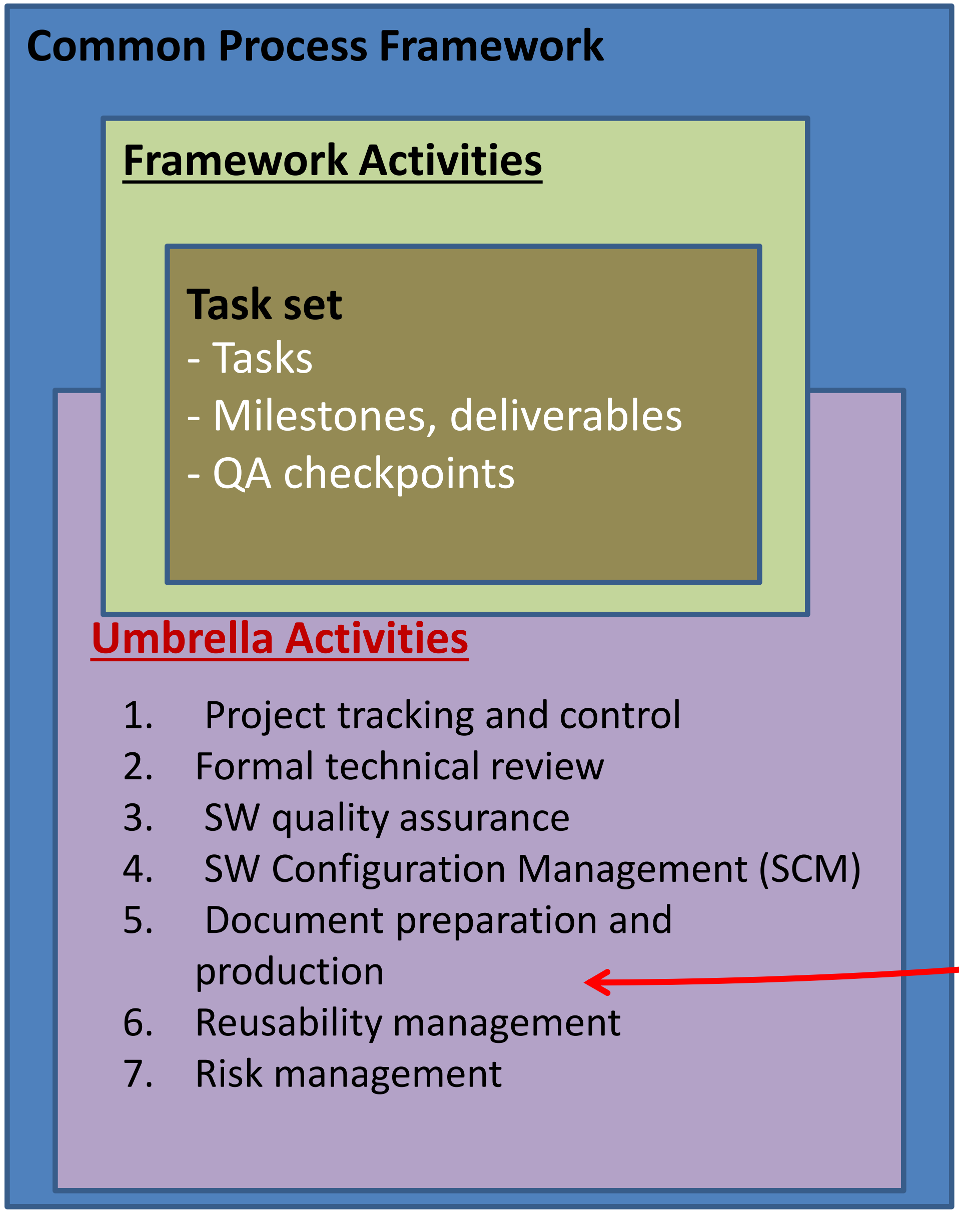
- *Software development process* is a set of steps that a software program goes through when developed.
- General phases of software development are:

  Requirements → Design → Implementation → Testing / Verification → Documentation → Maintenance.

➥ **Importance of software development.**

- Software is important to **make the *hardware working.***
- Software is important to ***build up security*** where it needs to be done, such as, in banks, money transaction etc.
- Software is important to ***make a task easier,*** such as, distribution of products, products information etc.
- Software is important to use the computers power or working efficiency to perform those tasks which cannot be done or controlled by human.
- ***Shortly, it can be said that software is important to make things easier, faster, more reliable and safer.***

1.5 Generic Framework and Umbrella activities



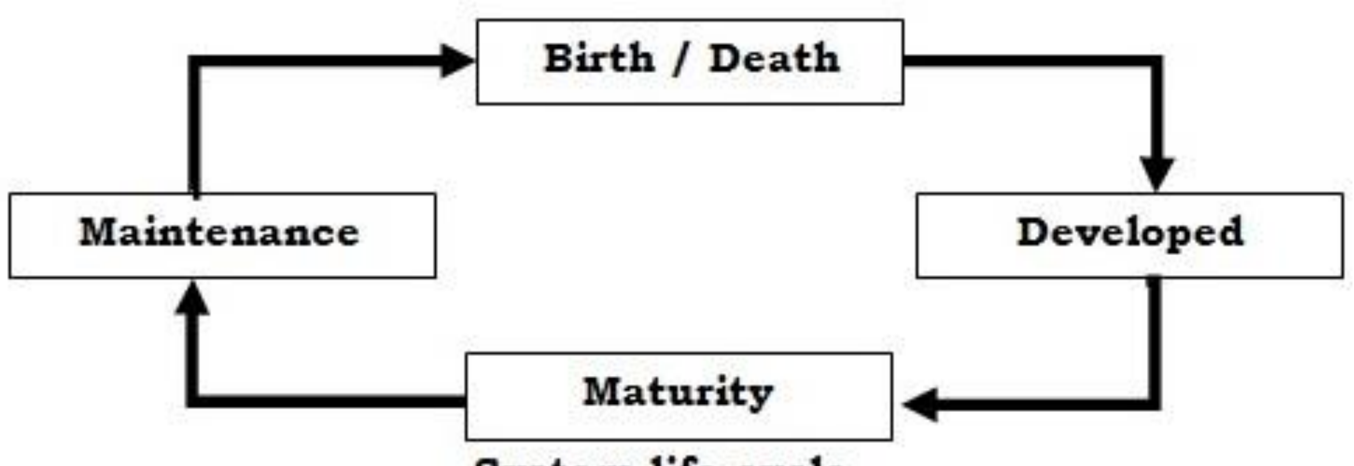
1. Project tracking and control
2. Formal technical review
3. SW quality assurance
4. SW Configuration Management (SCM)
5. Document preparation and production
6. Reusability management
7. Risk management

- Each framework activity is populated by set of task, milestones and quality assurance.
- Umbrella activities are performed through out the process.
- these are independent of any framework activity.
- the list of umbrella activities are given in the figure.

નોંધ : દરેક ટોપીકમાં થોડુ ડીટેલમાં લખવું.

1.6 Software Development Models Or Life cycle Models

- Every system has a life cycle. It begins when a problem is recognized, after then system is developed, grows until maturity and then maintenance needed due to change in the nature of the system.



System life cycle

- Goal → is to produce high quality software product.
- **As per IEEE Standards, software life cycle is:** “the period of time that starts when software product is conceived and ends when the product is no longer available for use.”
- **A software life cycle model is also called a Software Development Life Cycle (SDLC).**

1.6 Software Development Models Or Life cycle Models

- Software life cycle is the ***series of identifiable stages*** that a s/w product undergoes during its lifetime.
- Software life cycle model (process model)→ is a ***descriptive and diagrammatic representation*** of the software life cycle.
- A life cycle model represents all the activities required to make a software product transit through its life cycle phases.
- ***General stages*** → feasibility study, requirement analysis and specification, design, coding, testing and maintenance.
- Every software development process model includes system requirements as ***input*** and deliverable product as ***output***.

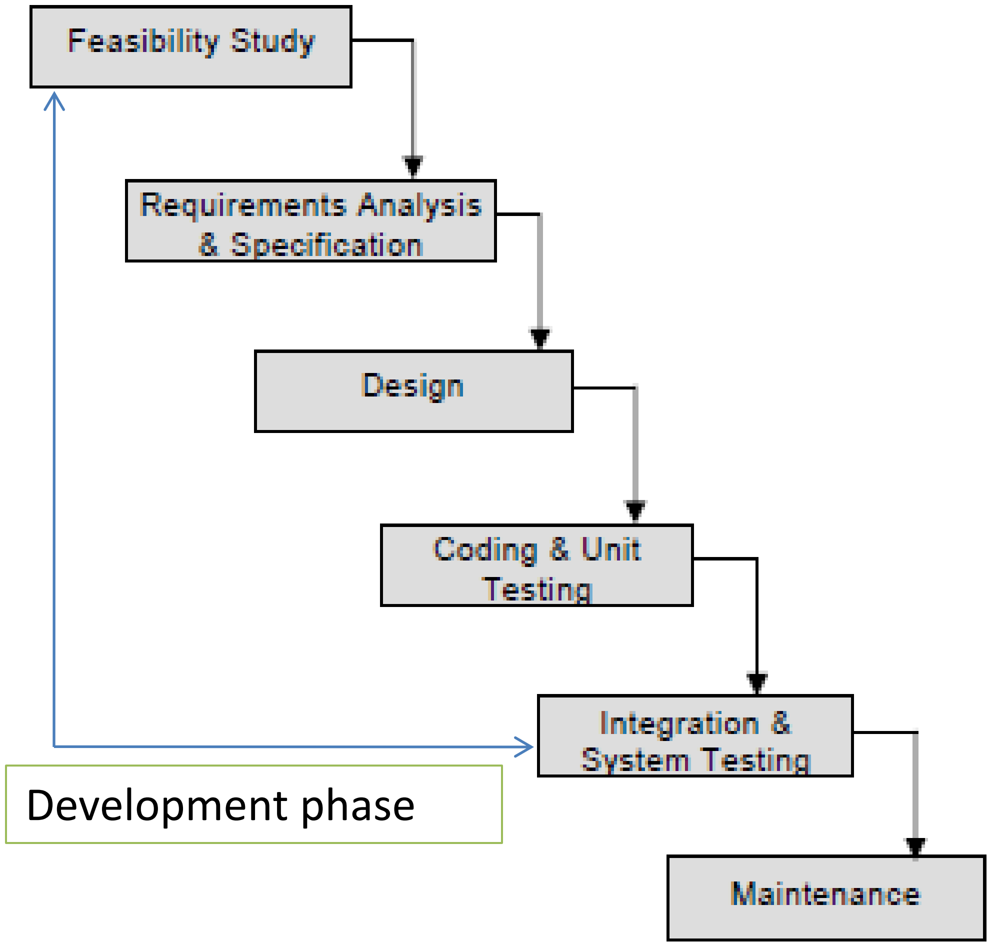
1.6 Software Development Models Or Life cycle Models

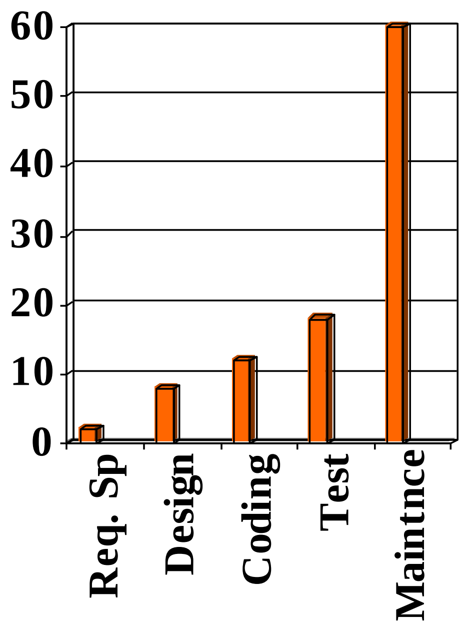
- **Need of life cycle models.**
- Provide ***generic guidelines*** for developing a suitable process for a project.
- It provides improvement and guarantee of ***quality product***.
- Without using of a particular life cycle model the development of a software product would not be in a ***systematic and disciplined manner***.
- Provide ***monitoring the progress***.
- Defines ***entry and exit criteria***.
- The ***documentation*** of life cycle models enhances the understanding between developers and client.
- **Different software life cycle models.**

1.6.1 Classical Waterfall model

- This model was originally proposed by Royce (1970).
- It is also called ‘traditional waterfall model’ or ‘conventional waterfall model.
- It is the most obvious way to develop the software.
- It’s just a theoretical way of developing s/w. All other life cycle models are essentially derived from it.
- **Phases of this model are:**
  - Feasibility study
  - Requirements analysis and specification,
  - Design
  - Coding and unit testing
  - Integration and system testing
  - Maintenance

1.6.1 Classical waterfall model





Phases between feasibility study and testing

known as development phases.

Among all life cycle phases

maintenance phase consumes maximum effort.

Among development phases,

testing phase consumes the maximum effort.

1.6.1 Classical Waterfall model

◈Feasibility Study.

- **Aim** → To determine whether the system would be financially and technically feasible to develop the product.
- Includes the analysis of the problem and collection of relevant information of **i/p, processing and o/p data.**
- Collected data are analyzed :
  - For an abstract definition
  - Formulation of different solutions
  - Analysis of alternative solutions
- **Cost / benefit analysis** is performed.
- **Three main issues** are concerned with feasibility:
  - *Technical feasibility*
  - *Economical feasibility*
  - *Operational feasibility*

1.6.1 Classical Waterfall model

◈ Requirement Analysis and Specification

- **Aim** → is to understand the exact requirements of the customer and to document them properly.
- Also reduces communication gap between developers and customers.
- **Two different activities are performed during this phase:**
  - ***Requirements gathering and analysis.***
  - ***Requirements specification***

➥ Requirement specification:

- In it, user requirements are systematically organized into a Software Requirements Specification (SRS) document.
- The important components of SRS are – the functional requirement, the non functional requirement and the goals of implementations.
- Output of this phase is → SRS document, which is also called *Black box specification* of the problem.

This phase concentrates on "what" part not "how".

1.6.1 Classical Waterfall model

❖ Design

- The goal of the design phase is → to transform the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language.
- This phase affecting the quality of the product.
- Two main approaches are concerned with this phase.
  - I. *Traditional design approach*
  - II. *Object oriented design approach*
- *Traditional design* consists of two different activities :
  - I. Structural Analysis:
    - Where the **detailed structure** of the problem is examined.
    - **Identify the processes and data flow** among these processes.
    - **DFD is used** to perform the structure analysis and to produce result.
    - In structure analysis – functional requirement specified in SRS are decomposed and analysis of data flow is represented diagrammatically by DFD.

1.6.1 Classical Waterfall model

II. Structure design:

- During structured design, the results of structured analysis are transformed into the software design.
- Two main activities are associated with it :

1. Architectural design (High-level design)

- decomposing the system into modules and build relationship among them.

2. Detailed design (Low-level design)

- identified individual modules are design with data structure and algorithms.

1.6.1 Classical Waterfall model

- In ***object oriented approach***, objects available in the system and relationships are identified during this approach.
- Several tools and techniques are used in designing like:
  - Flow chart
  - DFD
  - Data Dictionary
  - Decision Table
  - Decision Tree
  - Structured English

1.6.1 Classical Waterfall model

◈ Coding and unit testing

- It is also called the implementation phase.
- **The aim of this phase is →** to translate the software design into source code and unit testing is done module wise.
- Each component of the design is implemented as a program module, and each unit is tested to determine the correct working of the system.
- System is being operational in this phase.
- This phase affects testing and maintenance.
- Simplicity and clarity should be maintained.
- Output of this phase is → set of program modules that have been individually tested.

1.6.1 Classical Waterfall model

◈ Integration and system testing.

- Once all the modules are coded and tested individually, integration of different modules is undertaken.
- This phase is carried out incrementally over a number of steps and during each integration step, the partially integrated system is tested and a set of previously planned modules are added to it.
- Finally, when all the modules have been successfully integrated and tested, system testing is carried out.
- **Goal of this phase is →** to ensure that the developed system works well to its requirements described in the SRS document.
- Testing procedure is carried out using test data: Program test and System test. Program test is done on test data and system test is done actual data.
- It consists of three different kinds of testing activities.

α – testing	β – testing	Acceptance testing

- Output of this phase is → system test plan and test report (error report).

1.6.1 Classical Waterfall model

❖ Maintenance.

- It requires max efforts to develop s/w product.
- This phase is needed to keep system operational.
- Generally maintenance is needed due to change in the environment or the requirement of the system.
- Maintenance involves performing following three kinds of activities:
  - **I. Corrective maintenance**
  - **II. Perfective maintenance**
  - **III. Adaptive maintenance**
- In this phase the system is reviewing for full capabilities of the system.

1.6.1 Classical Waterfall model

❖ **Advantages:**

- It is simple and easy to understand and use.
- Each phase has well defined input and outputs.
- It helps project personnel in planning.
- Waterfall model works well for smaller projects where requirements are very well understood.
- It divides complex tasks into smaller, more manageable works.

❖ **Disadvantages:**

- It is a theoretical model.
- It may happen that the error may be generated at any phase and encountered in later phase. So it’s not possible to go back and solve the error in this model.
- High amounts of risk and uncertainty.
- Formal documents are needed at each phase.

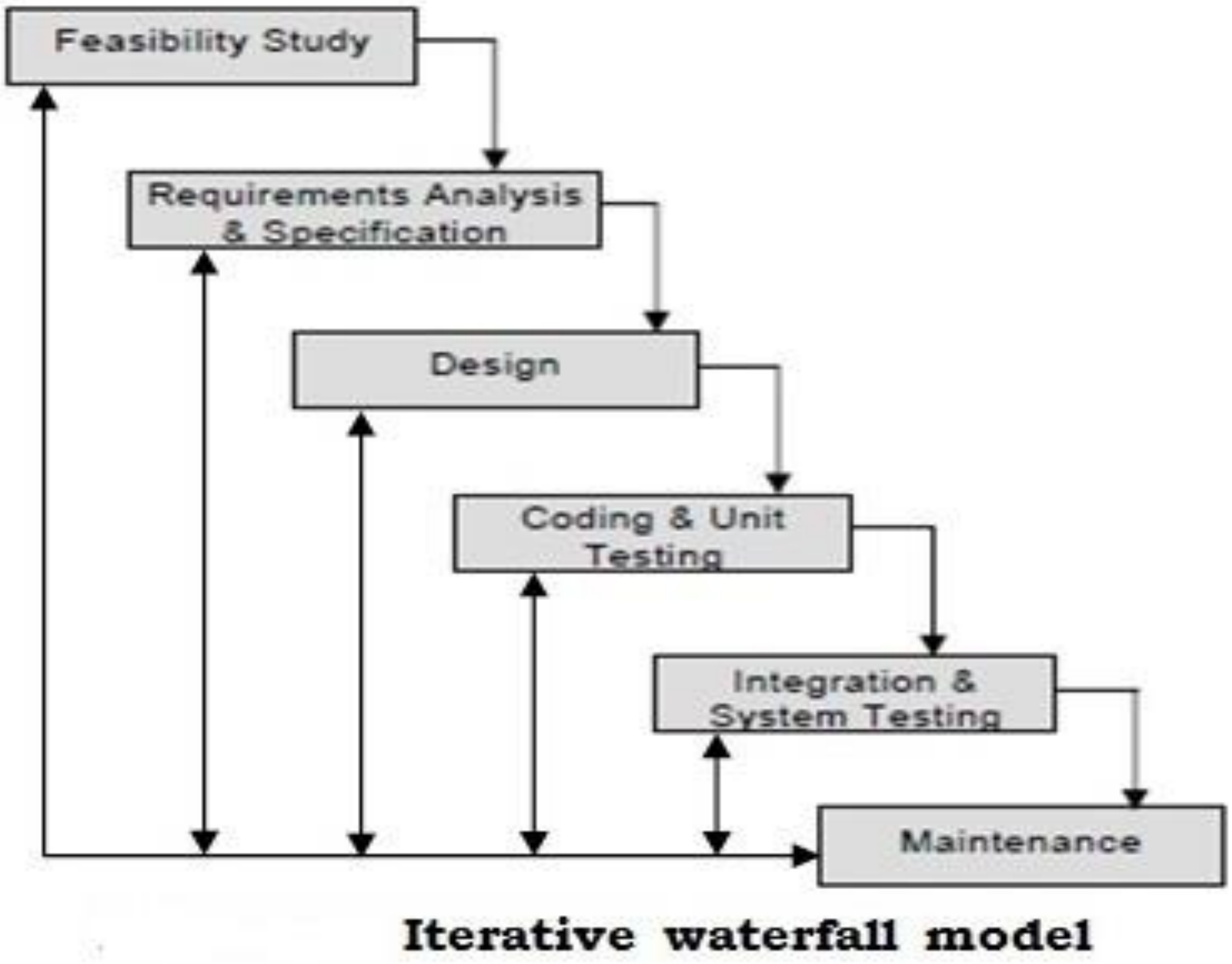
1.6.1 Classical Waterfall model

◈ Application:

- This model is used only when the requirements are very well known, clear and fixed.
- When the product definition is stable.
- When the technology is understood.
- When the project is short.

1.6.1 Iterative Waterfall model

- Classical waterfall model is idealistic: it assumes that no de
- Bu t every ph ch later sta
- So, del.
- Ite y used mc m the wa
- Th oint of int *nent of err*
- Ph viewing aft


Feasibility Study
Requirements Analysis & Specification
Design
Coding & Unit Testing
Integration & System Testing
Maintenance
Iterative waterfall model

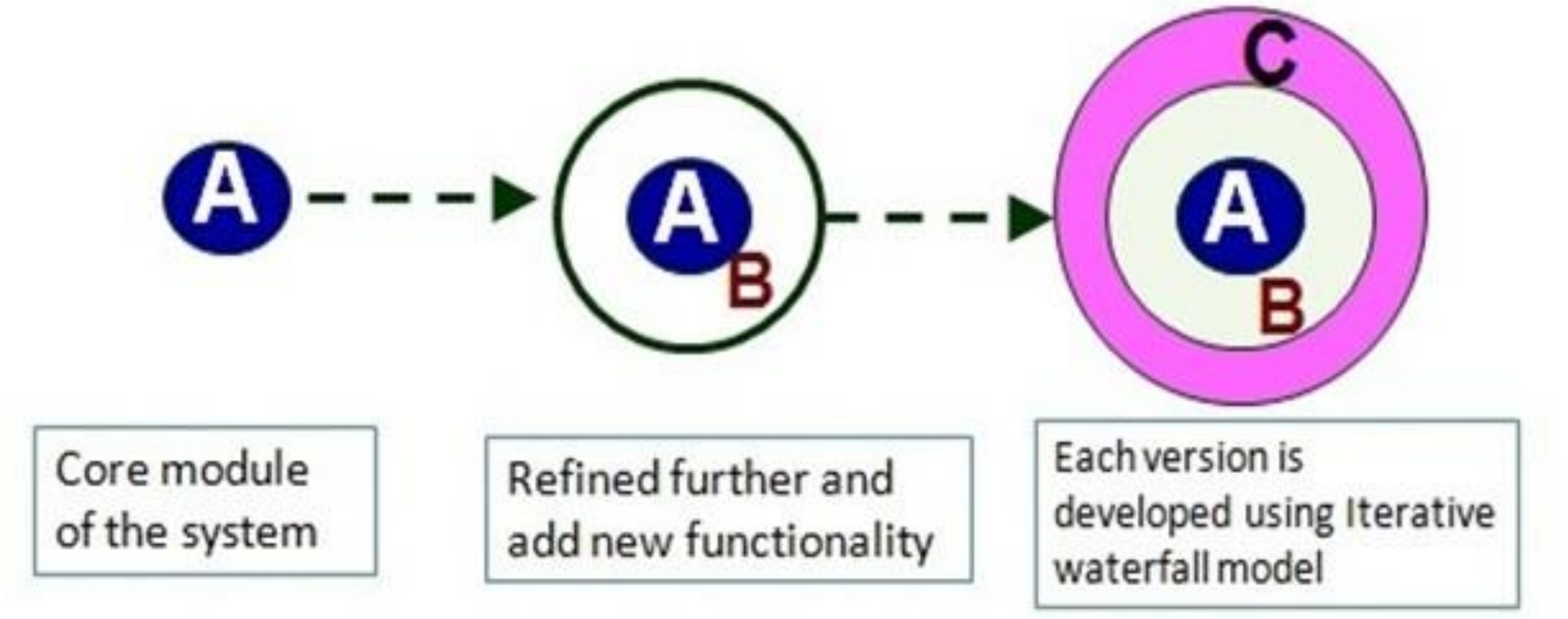
1.6.2 Incremental model

- It is also referred as the ***successive version*** *of waterfall model* using incremental approach and *evolutionary model.*
- In this model, the system is **broken down into several modules** which can be incrementally implemented and delivered.
- First develop the core product of the system. The core product is used by customers to evaluate the system.
- The initial product skeleton is refined into increasing levels of capability: by adding new functionalities in successive versions.

1.6.2 Incremental model

➥ **Advantages:**

- Each successive version more useful work than previous


A
A
B
A
B
C
Core module of the system
Refined further and add new functionality
Each version is developed using Iterative waterfall model

Incremental model

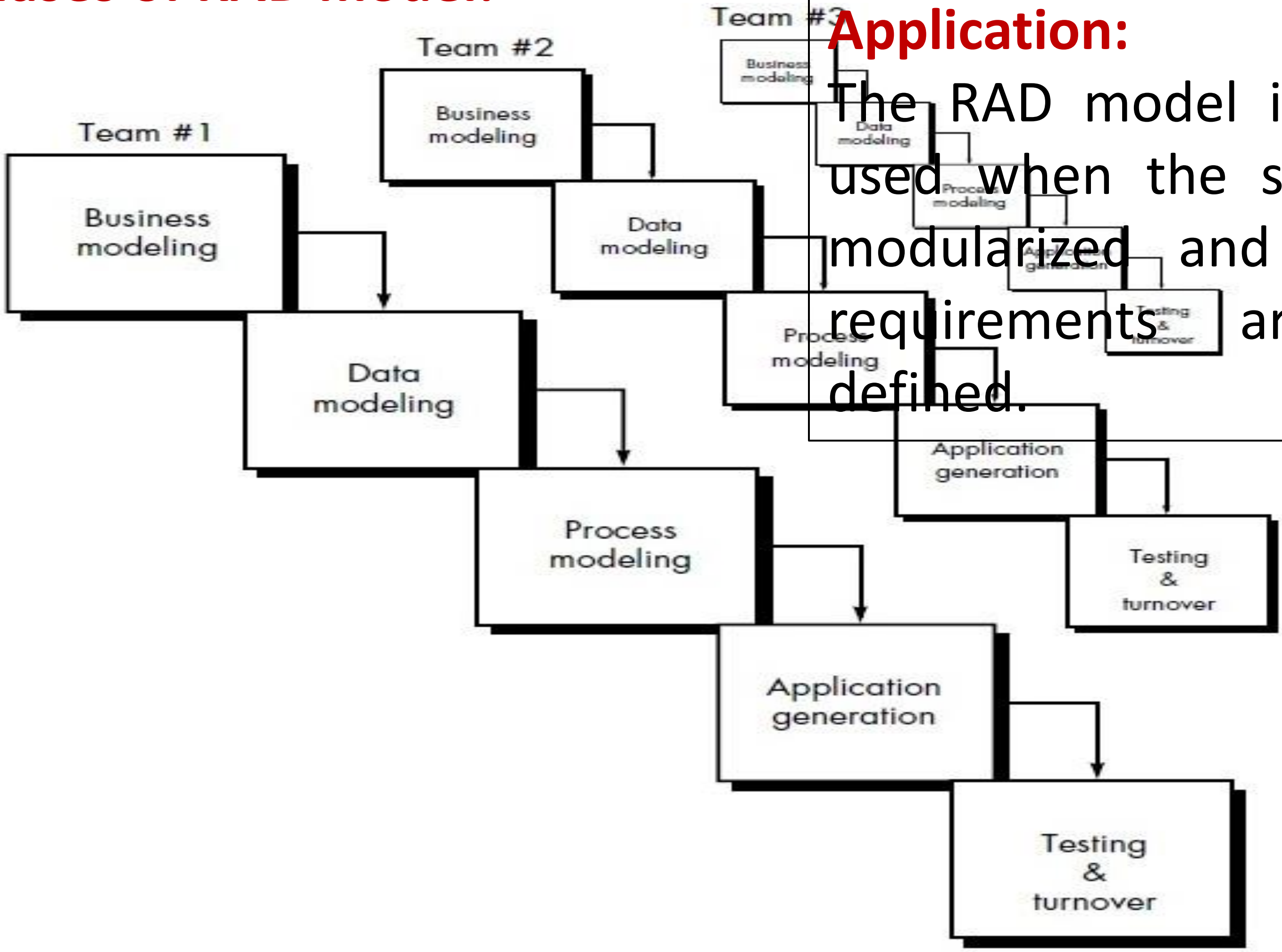
- It needs good planning and design.
- **Application** → when large problems and user req^n are not well specified

1.6.3 RAD model

- It is proposed by IBM in 1980.
- It emphasizes an extremely short development cycle.
- It emphasize on *reuse*.
- In it rapid development is achieved by using component-based construction.
- If requirements are well understood and project scope is constrained, the RAD process enables a development team to create a "fully functional system" within very short time periods (e.g., 60 to 90 days).
- User involvement is essential from req^n analysis to delivery.
- For this model, requirements must be cleared and well understood initially.
- Many development teams are working parallel to complete the task.

1.6.3 RAD model

Phases of RAD model:



Application:

The RAD model is mostly used when the system is modularized and all the requirements are well defined.

A

- A
- Tl
- R
- C

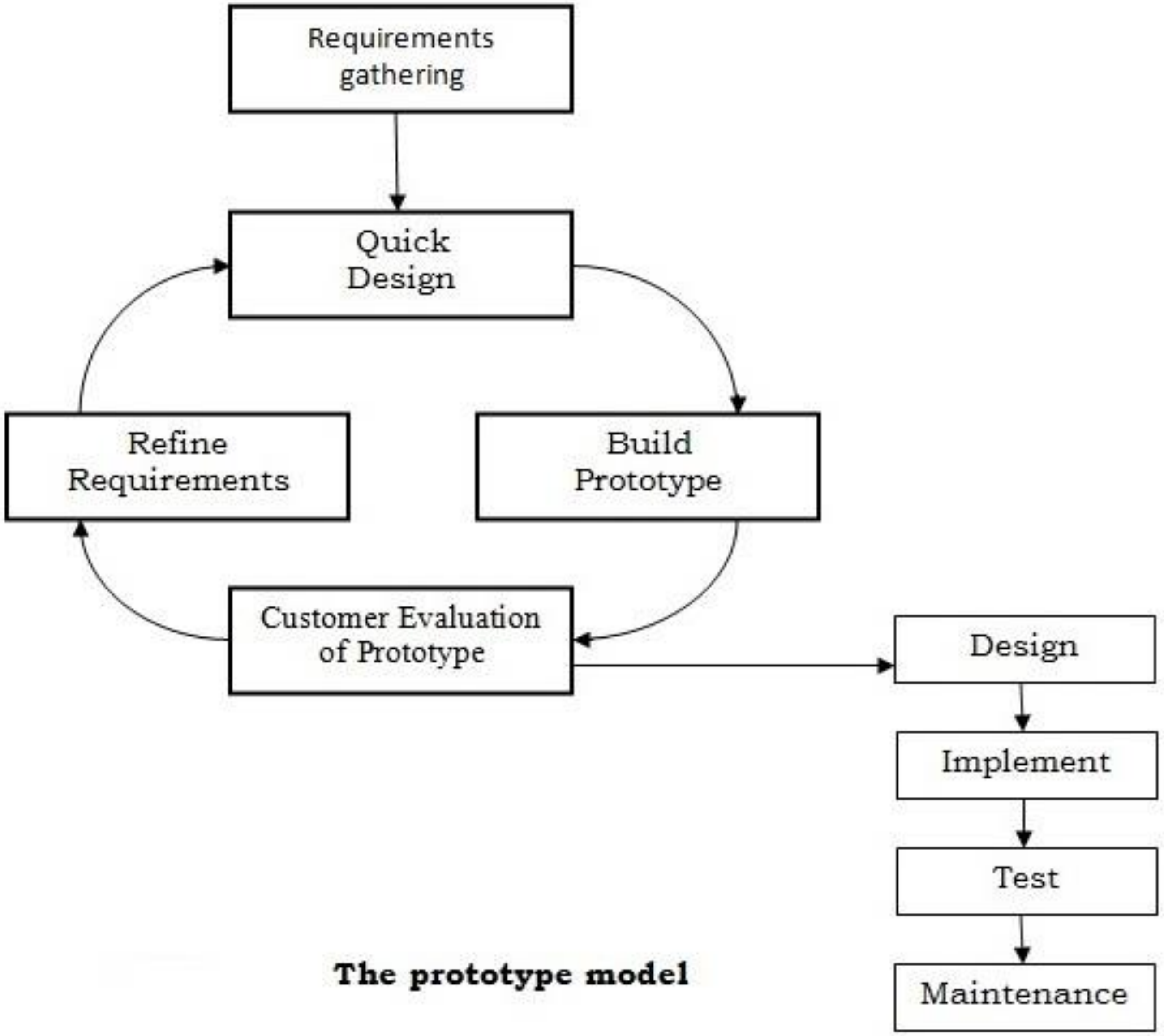
D

- R
- It
- In it, highly skilled and expert developers are needed.

1.6.4 Prototype model

- Prototype is a working physical system or sub system. Prototype is nothing but a **toy implementation** of a system.
- In this model, before starting actual development, a **working prototype of the system should first be built**.
- A prototype is actually a **partial developed product**.
- Compared to the actual software, a prototype usually have
  - **limited functional capabilities**
  - **low reliability**
  - **inefficient performance**
- It is built using **several shortcuts**.
- It is very **useful in developing GUI** part of system.
- It turns out to be a very **crude version** of the actual system.

1.6.4 Prototype model



The prototype model

1.6.4 Prototype model

- In working of the prototype model, product development starts with initial requirements gathering phase.
- Then, quick design is carried out and prototype is built.
- The developed prototype is then submitted to the customer for his evaluation.
- Based on customer feedback, the requirements are refined and prototype is modified.
- This cycle of obtaining customer feedback and modifying the prototype continues till the customers approve the prototype.
- The actual system is developed using the different phases of iterative waterfall model.
- There are two types of prototype model.
  - ***Exploratory development prototype model***:
  - ***Throw away prototype model***:

1.6.4 Prototype model

Advantages:

- Customers can get a chance to have a look of the product.
- New requirements can be accommodate easily.
- Missing functionalities identified quickly.
- It provides better flexibility in design and development.
- More chance of user satisfaction.

Disadvantages:

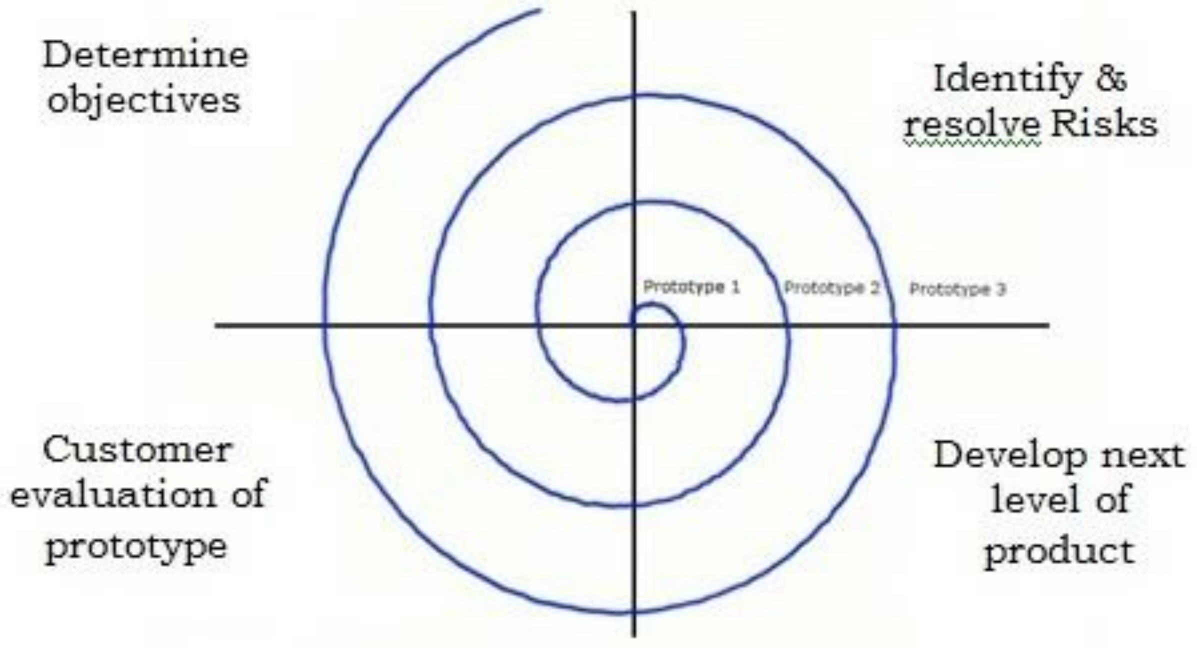
- wasting of time is there because core product is thrown away.
- The construction cost is very high.
- This model requires extensive participation and involvement of the customers that is not possible every time.
- If end user is not satisfied with the initial prototype, he may lose interest in the final product.

Application:

- This model used when desired system needs to have a lot of interactions with end users.
- Generally used in GUI based application.

1.6.5 Spiral model

- This model is proposed by Boehm in 1986.
- In application development, spiral model **uses fourth generation (4GL) languages** and developments tools.
- The diagrammatic representation of this model appears like a spiral with many loops.



Spiral model

1.6.5 Spiral model

- Each loop of the spiral represents a phase of the software process:
  - the innermost loop might be concerned with system feasibility,
  - the next loop with system requirements definition,
  - the next one with system design, and so on.
- **No fixed number of phases** is there in this model.
- This model is **more flexible** compared with other models.
- Each loop in the spiral is split into **four sectors (quadrants)**

1st Quadrant: Determine objectives

- Identifying the **objectives**, **their relationships** and find the possible alternative solutions.
- Also **examine the risks** associated with these objectives.

1.6.5 Spiral model

- *2nd Quadrant: Identify and resolve risks*
- **Detailed analysis** is carried out of each identified risk.
- **Alternate solutions** are evaluated and risks are reduces at this quadrant.
- *3rd Quadrant: Develop next level of product*
- Develop and validate the next level of the product.
- **Activities** like design, code development, code inspection, testing and packaging are performed.
- **Resolution of critical operations and technical issues** of next level product are performed.
- *4th Quadrant: Customer evaluation (Review and Planning)*
- In this part, **review the results achieved** so far.
- **Plan the next iteration** around the spiral. Different plans like development plan, test plan, installation plan are performed.

1.6.5 Spiral model

- With each iteration around the spiral: progressively more complete version of the software gets built.
- In spiral model → at any point, ***Radius*** represents: cost and ***Angular dimension*** represents: progress of the current phase.
- At the end, **all risks are resolved and s/w is ready to use**.

➥ ***Advantages:***

- It is more flexible, as we can easily deal with changes.
- Due to user involvement, user satisfaction is improved.
- It provides cohesion between different stages.
- Risks are analyzed and resolved so final product will be more reliable.
- New idea and additional functionalities can be easily added at later stage.

1.6.5 Spiral model

➥ ***Disadvantages:***

- It is more complex to understand.
- It is applicable for large problem only.
- It can be more costly to use.
- More number of documents are needed as more number of spirals.
- Risk analysis phase requires much higher expertise.

➥ ***Application:***

- Used when medium to high risk projects.
- When users are unsure for their needs.
- When requirements are complex.

1.6.5 Spiral model

☛ **Spiral model can be viewed as a Meta model.**

- Spiral model subsumes almost all the life cycle models.
- Single loop of spiral represents ***Waterfall Model.***
- Spiral model uses a ***prototyping approach*** by first building the prototype before developing actual product.
- The iterations along the spiral model can be considered as supporting the ***Evolutionary Model.***
- The spiral model retains the step-wise approach of the ***waterfall model.***

Comparison of different life cycle models

The Classical waterfall model

- Can be considered as a basic model.
- But, it is not used in practical development.

The Iterative waterfall model

- Most widely used model.
- But, suitable only for well-understood problems.

The Prototype model

- Suitable for projects in which user requirements and technical aspects are not well understood.
- Especially popular for user interface part of the project.

Comparison of different life cycle models

The Evolutionary model

- It is suitable for large problems.
- used for object oriented development projects.
- But can be used only if the incremental delivery of the system is acceptable by the customer.

The Spiral model

- Used to develop the projects those have several kinds of risks.
- But, it is much complex than other models.

Programs versus Software Products

Program	Software product
- It is usually small in size.	- It is large in size.
- It has single developer.	- Here, team of developers.
- Author himself is sole user.	- Large number of users.
- It lacks proper user interface.	- Here, well designed interface.
- It lacks proper documentation.	- Here, well documented and user manual prepared.
- It is ad-hoc development.	- It is systematic development.
- No need of systematic methodologies.	- Require proper systematic methodologies.

Thank you…