Unit – 1
Software Development Process

- **1.1 Software**
- Definition
- Characteristics
- **1.2 Software Myths**
- **1.3 Software Engineering**
- A layered Technology approach
- Definition Need
- **1.4 Software development**
- **1.5 Generic Framework activities, Umbrella**
- **activities**

1.6 Software Development Models

- Waterfall Model
- Incremental Model
- RAD Model
- Prototyping Model
- Spiral Model

1.1 Software

- Software is the “collection of computer programs, procedures, rules, associated documents and concerned data with the operation of data processing system”.
- It also includes representation of pictorial, video and audio information.
- Software are of two types.
  - System Software
  - Application Software

- **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 etc.

Software Characteristics

- The characteristics of software decide whether the software is good or bad.
- **Understandability**
  - Software should be easy to understand
  - It should be efficient to use.
- **Cost**
  - Software should be cost effective as per its usage.
- **Maintainability**
  - Software should be easily maintainable and modifiable in future.

- **Modularity**
  - Software should have modular approach so it can be handled easily for testing.
- **Functionality**
  - Software should be functionally capable to meet user requirements.
- **Reliability**
  - It should have the capability to provide failure-free service.
- **Portability**
  - Software should have the capability to be adapted for different environments.

- **Correctness**
  - Software should be correct as per its requirements.
- **Documentation**
  - Software should be properly documented so that we can re-refer it in future.
- **Reusability**
  - It should be reusable, or its code or logic should be reusable in future.
- **Interoperability**
  - Software should be able to communicate with various devices using standard bus structure and protocol.

SOFTWARE DOESN'T WEAR OUT

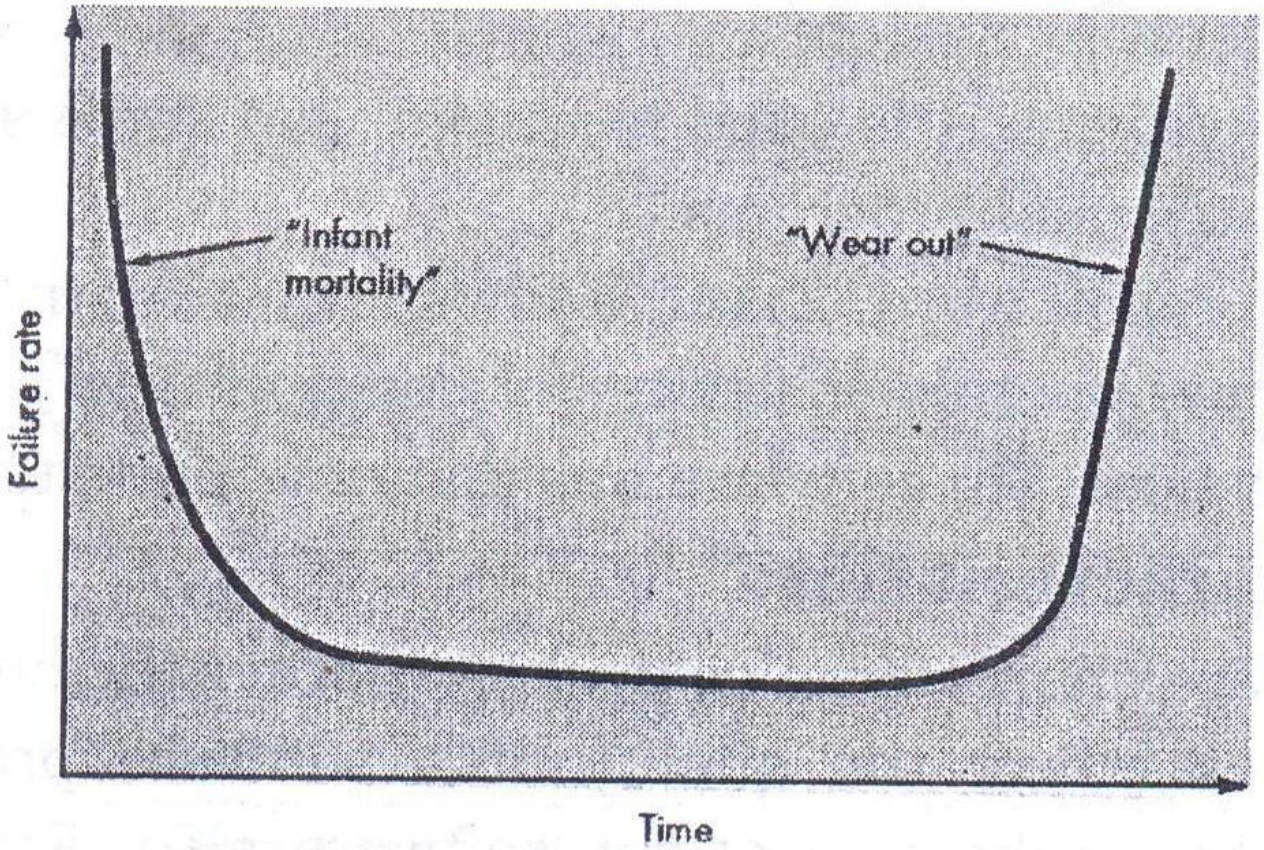


Figure 1.1 hardware failure curve

- In figure, the relationship between time and failure called "**bath-tub curve**".
- It indicates that hardware exhibits relatively high failure rates early in its life, then defects are corrected and the failure rate drops to a steady-state level for some period of time.
- As time passes, however, the failure rate rises again as hardware components suffer from the affects of dust, vibration, temperature extremes, and many other environmental factors.
- So, simply, we can say hardware begins to wear out.

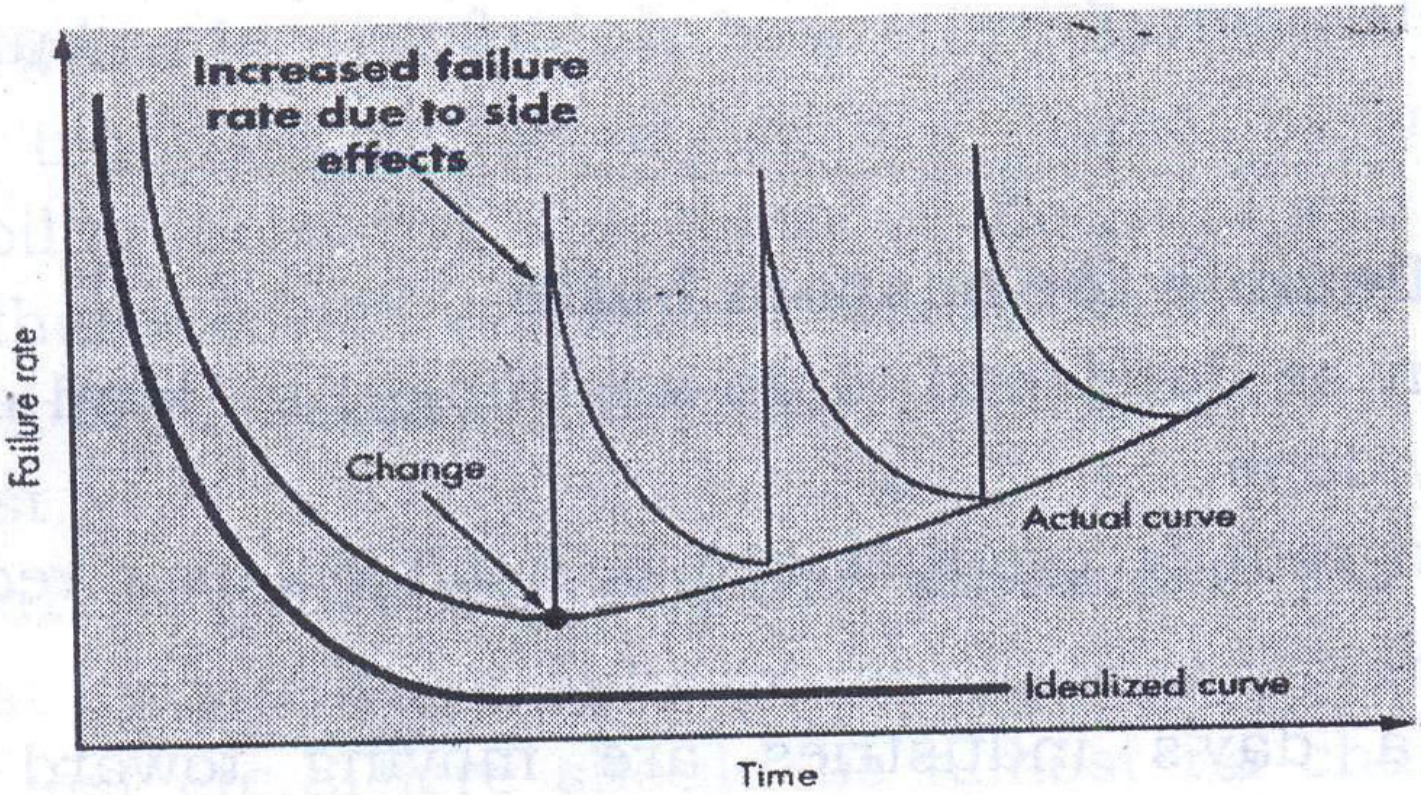


Figure 1.2 Software Failure curve

- The above figure shows the software failure rate.
- Software is not highly affected by environmental effects.
- In the early stage, due to many errors, software could have high failure.
- But it becomes reliable as time passes instead of wearing out. Once software is made it has a longer life span.
- In actual curve, we can see that software may have increased failure rate as it may become obsolete as the environment in which it was developed, changes.
- Spike in the curve if due to chance of maintenance and side effects.
- Software may be retired due to new requirement.
- So, software doesn't wear out, but it may be deteriorate.

SOFTWARE IS ENGINEERED, NOT MANUFACTURED LIKE HARDWARE.

- Once a product is manufactured, it is not easy to modify it.
- While in case of software we can easily change or modify or change it for later use.
- Evan making multiple copies of software is a very easy task rather it is much more tough in case of hardware.
- In hardware, costing is due to assembly of raw material and other processing expenses while in software development no assembly needed like hardware.
- So, software is not manufactured as it is developed or it is engineered.

1.2 Software Myths

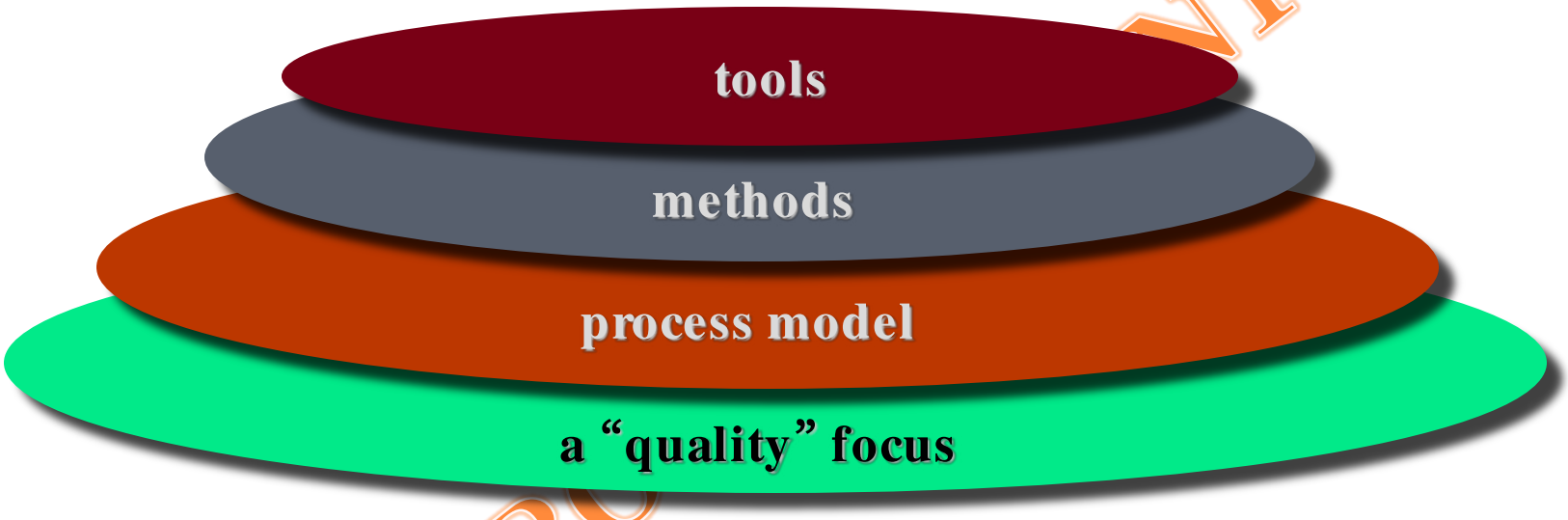
- Software is easy to change
- Outsourcing of software to a third party can relax the customers.
- Software can work right from the first time.
- Increasing of software reliability will increase software safety.
- Reusing software increase safety.
- Best software is one which has more features.
- Testing of software will remove all errors.
- Once the project is working, job is done.

1.3Software Engineering

- Software engineering is an engineering discipline that covers all aspects of software from specification to maintenance.
- It is an engineering discipline that delivers high quality software at agreed cost & in planed schedule.
- It also provide framework that guides the software engineers to develop the software.
- It tell how the software will work with machines.
- It covers technical and management issues.

- The Main aspect of software engineering are
  - Provide quality product
  - Expected cost
  - Complete work on agreed schedule
- **Definition:**
  - Software engineering is the application of a systematic, disciplined and quantifiable approach to the development, operation and maintenance of software.

Software Engineering – A Layered Approach



- Software engineering can be viewed as a layered technology.
- It contains process, methods, and tools that enables software product to be built in a timely manner.

- **A Quality Focus Layer**
  - Software engineering mainly focuses on quality product.
  - It checks whether the output meets with its requirement specification or not.
  - Every organization should maintain its total quality management.
- **Process Layer**
  - It is the heart of software engineering.
  - It is also work as foundation layer.
  - Software process is a set of activities together if ordered and performed properly, then the desired result would be produced.
  - It defines framework activities.
  - The main objective of this layer is to deliver software in time.

- **Method Layer**
  - It describes ‘how-to’ build software product.
  - It creates software engineering environment to software product using CASE tools
- **Tools Layer**
  - It provides support to below layers.
  - Due to this layer, process is executed 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.

1.4 Software Development

- Software development is the process of developing software through successive phases in an orderly way.
- Software development is the computer programming, documenting, testing and bug fixing involved in creating and maintaining applications and framework involved in software 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.

- Software development process is a set of steps that a software program goes through when developed.
- The phases of software development process are
  - Requirement gathering & Analysis
  - Design
  - Implementation (Coding)
  - Testing
  - Documentation
  - Maintenance.

- In **requirement phase**, the goals of what the program will be capable of doing is decided.
- The **design phase** covers how the program is going to be created, who will be doing what etc.
- The **implementation phase** is where the programmers and other designers start work on the program.
- **Testing and verification phase** can begin to help verify the program has no error. During this phase, problems are fixed, until the program meets the requirement.
- The **documentation phase** tells how to use the program or project.
- Finally, maintaining the program must continue for several years after the initial release.

1.5 Generic framwork Activities of software engineering

- The work associated with software engineering can be categorized into three generic phases.
  - Definition phase
  - Development phase
  - Support phase

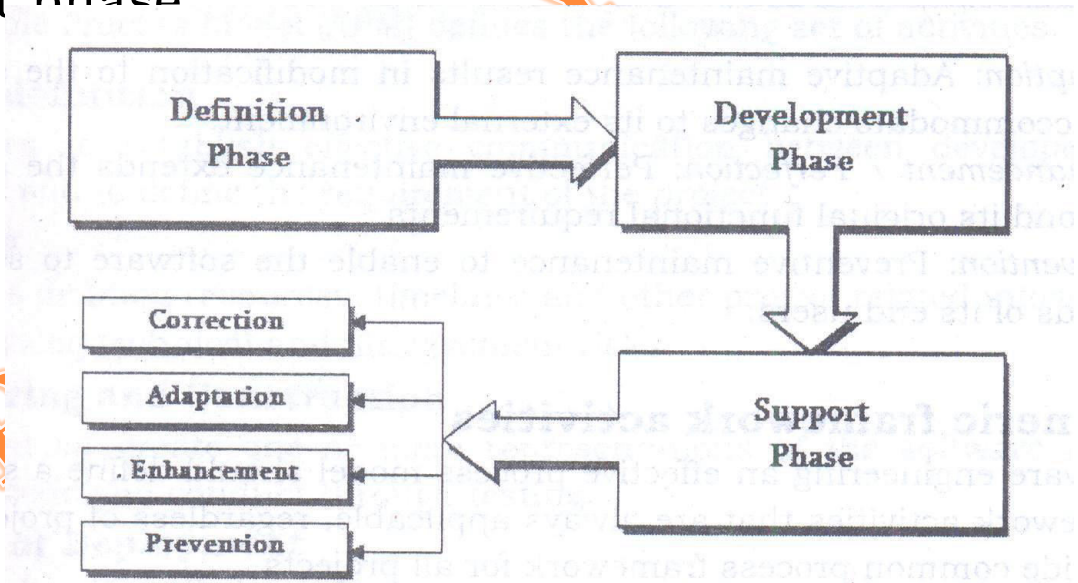


Figure 1.4 Software generic view

- **Definition phase**
  - It focuses on what part.
  - During definition phase, the software engineer attempts to identify
    - What information is to be processed?
    - What function and performance are desired?
    - What system behavior can be expected?
    - What interfaces are to be established?
    - What design constraints exist?
    - What validation criteria are required to define a successful system?
  - Thee main activities performed during this phase:
    - System or information engineering
    - Software project planning
    - Requirement analysis

- **Development Phase**
  - It focuses on ***how*** part of the development.
  - During development a software engineer attempts to define
    - How data are to be structured?
    - How function is to be implemented within a software architecture?
    - How interfaces are to be characterized?
    - How design will be translated into a programming language?
    - How testing will be performed?
  - Main activities are performed under this phase are:
    - Software design
    - Code generation
    - Software testing

- **Support Phase**
  - The support phase focuses on change associated with error correction.
  - Four types of change are encountered during the support phase.
  - **Correction**: corrective maintenance changes the software to correct defects.
  - **Adaption**: Adaptive maintenance results in modification to the software to accommodate changes to its external environment.
  - **Enhancement / perfection**: Perfective maintenance extends the software beyond its original functional requirements.
  - **Prevention:** Preventive maintenance to enable the software to serve the needs of its end users.

Generic Framework Activities

- **Project Definition**:
  - It requires to establish effective communication between developer and customer and to define the requirement of the project.
- **Planning**:
  - It requires defining resources, timelines and other project related information and assessing technical and management risks.
- **Engineering & Construction:**
  - It required for create one or more representations of the software and to generate code and conduct through testing.
- **Release or Deployment:**
  - It required to install the software in its target environment and to provide customer support.
- **Customer Use:**
  - It required for obtaining customer feedback based on use and evaluation of project delivered during release.

UMBRELLA ACTIVITIES

- Umbrella activities are performed throughout the process.
- These activities are independent of any framework activity.
- The umbrella activities are given below:
- **Software project tracking and control:**
  - It assess progress against the plan and take actions to maintain the schedule.
- **Formal Technical Review:**
  - This includes reviewing the techniques that has been used in the project.
- **Software Quality Assurance:**
  - This is very important to ensure the quality management of each part to ensure them.

- Document Preparation and production
  - All the project planning and other activities should be hardly copied and the production gets started here.
- Reusability Management
  - This includes the backing up of each part of the software project they can be corrected or any kind of support can be given to them later to update or upgrade the software at user/time demand.
- Measurement
  - This will include all the measurement of every aspects of the software project.
- Risk Management
  - Risk management is a series of steps that help a software team to understand and manage uncertainty.

Software Development Life Cycle

- 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. So, it died and new system or replacement of it taken place.
- SDLC is a framework that describes the activities performed at each stage of a software development project.


Planning Stage
Design Stage
Software Development Life Cycles
Implementation Stage
Maintenance Stage

1.6 Software Development Models

- Different Software life cycle models
  - Classical Waterfall model
  - Iterative waterfall model
  - Incremental Model
  - RAD Model
  - Spiral Model
  - Prototype Model

Waterfall Model

- Waterfall model was proposed by Royce in 1970.
- It is also called as “traditional waterfall model” or “conventional waterfall model”.
- This model break down the life cycle into set of phases like.
  - Feasibility study
  - Requirements analysis and specification
  - Design
  - Coding and unit testing
  - Integration and system testing
  - Maintenance.

- During each phase of life cycle, a set of well defined activities are carried out. And each phase required different amount of efforts.
- The phases between feasibility study and testing known as development phases.
- Among all life cycle phases maintenance phase consumes maximum effort.

- Figure

Classical Waterfall model – figure

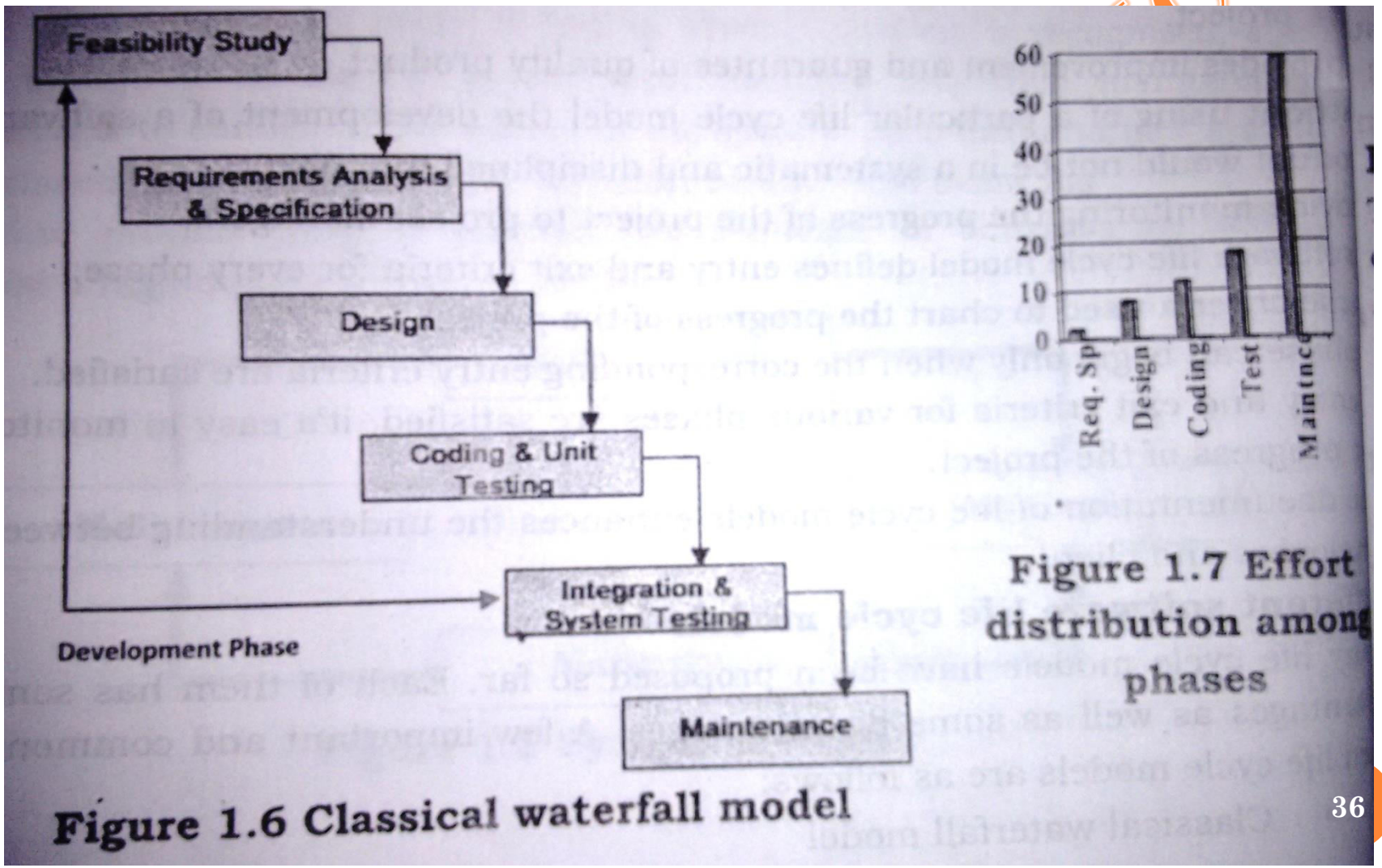


Figure 1.7 Effort distribution among phases

Figure 1.6 Classical waterfall model

- Feasibility Study:
  - Aim of this phase is to determine whether the system would be financially and technically feasible to develop the product.

- Requirement Analysis and Specification:
  - Aim of this phase is to understand the exact requirements of the customer and to document them properly.

- Design:
  - The goal of design phase is to transform the requirements specified in SRS document into a structure that is suitable for implementation in some programming language.

- Coding and Unit Testing
  - It is also called as implementation phase.
  - Aim of this phase is to translate the software design into source code and unit testing is done module wise.

- Integration and System Testing
  - Once all the modules are coded and tested individually, integration of different modules is undertaken.
  - Goal of this phase is to ensure that the developed system works well to its requirements described in the SRS document.

- Maintenance
  - It requires maximum efforts to develop software product.
  - This phase is needed to keep system operational.
  - General maintenance is needed due to change in the environment or the requirement of the system.

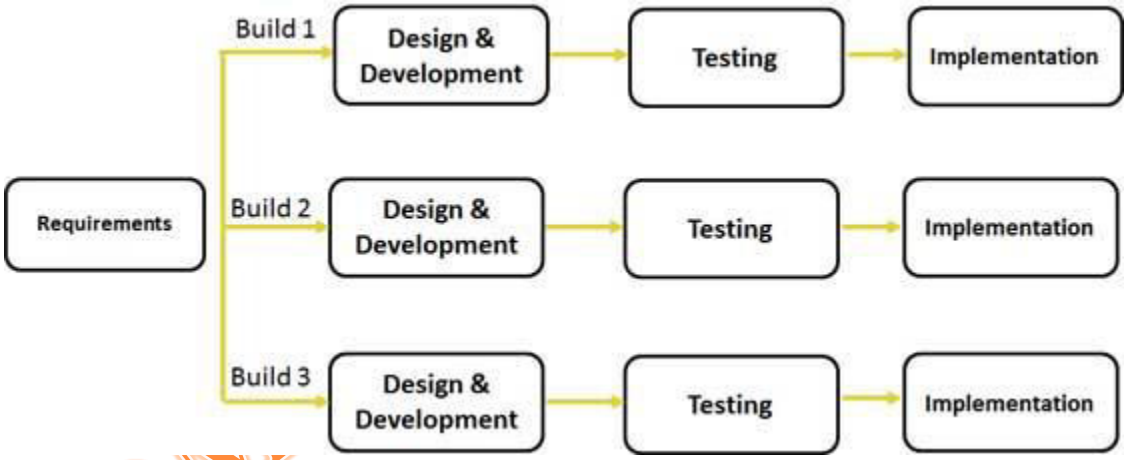
ITERATIVE/INCREMENTAL WATERFALL MODEL

- Classical waterfall model is idealistic: it assumes that no defect is introduced during any development activity.
- But in practice: defects do get introduced in almost every phase of the life cycle.
- Even defects may get at much later of the life cycle.
- So, the solution of this problem is iterative waterfall model.

INCREMENTAL MODEL

- The incremental model is also referred as the successive version of waterfall model using incremental approach.
- 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 functionality in successive version.

INCREMENTAL MODEL - FIGURE



INCREMENTAL MODEL

- Each successive version performing more useful work than previous version.
- The core modules get tested thoroughly, there by reducing chance of error in final product.
- The model is more flexible and less costly to change the scope and requirements.
- User gets a chance to experiment with partially developed software.
- Feedback providing at each increment is useful for determining the better final product.

RAD (Rapid Application Development) Model

- The RAD model is proposed by IBM in 1980.
- Rapid Application Development model is an incremental software development process model that emphasizes an extremely short development cycle.
- It emphasize on reuse.
- If requirements are well understood and project scope is constrained, the RAD process enables a development team to create a "fully functional system" within short time periods.
- In this model, user involvement is essential from requirement analysis to delivery.

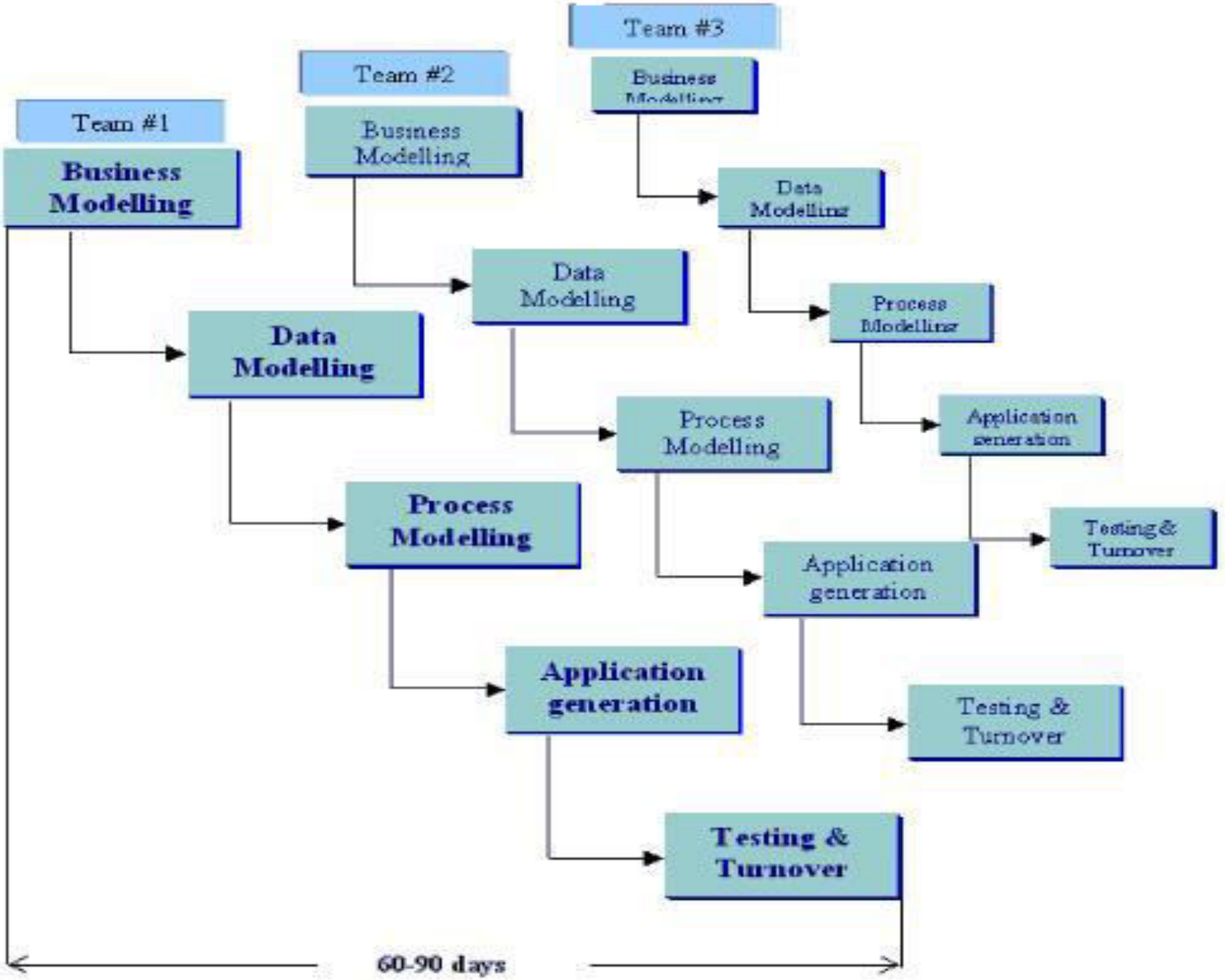
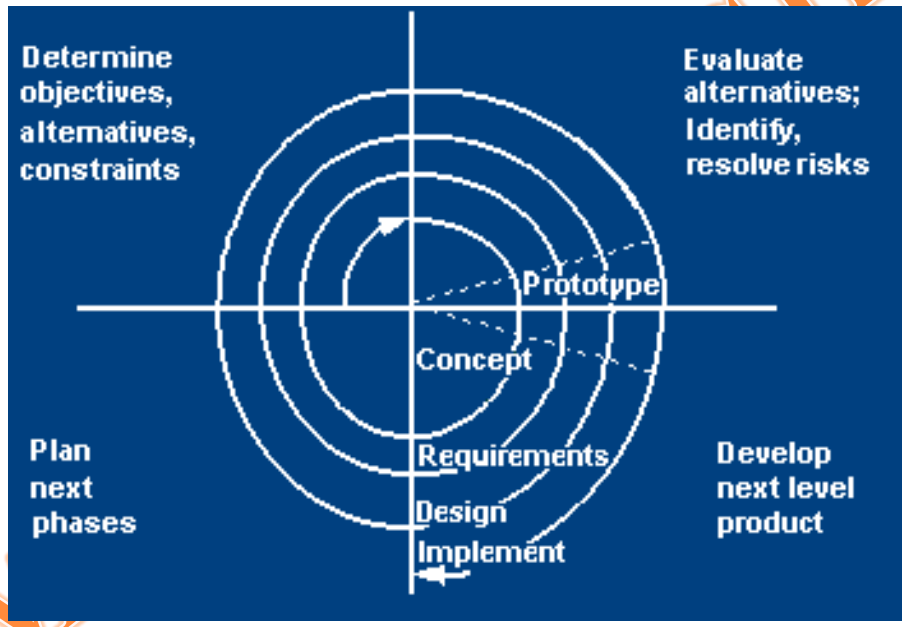


Figure 1.5 – RAD Model

Spiral Model

- This model is proposed by Boehm in 1986.
- In application development, spiral model uses fourth generation (4GL) languages and development tools.
- In pictorial view, this model appears like a spiral with many loops.
- 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 requirement definition.
  - The next one with system design and so on.
- Each loop in the spiral split into four sectors. (quadrants)

Spiral Model - Figure



- 1st Quadrant: Determine Objectives
- 2nd Quadrant: Identify and resolve risks
- 3rd Quadrant: Develop next level product
- 4th Quadrant: Review and planning

- In spiral model, at any point, **Radius represents: cost** and **Angular dimension represent : progress** of the current phase.

Prototype Model

- Prototype is a working physical system or subsystem.
- Prototype is nothing but a tip implementation of a system.
- In this model, before starting actual development, a working prototype of the system should be built first.
- A prototype is actually a partial developed product.
- Compared to the actual software, a prototype usually have,
  - Limited functional capabilities
  - Low reliability
  - Inefficient performance

Prototype Model - figure