



Object Oriented Software Engineering

MCA 104

Pre-requisite based Study Material

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.1



Pre- Requisite Modules

- **Object Oriented Analysis and Design, NPTEL**
Instructor– Prof. Partha Pratim Das, Department of Computer Science and Engineering, IIT Kharagpur
Duration – 08 Weeks
Link – <https://nptel.ac.in/courses/106105153/>
- **Lecture Slides**
Pre-requisite based Study Material, Object Oriented Software Engineering, by Dr. Ritika Wason

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.2



What is Software?


- Software is a set of items or objects that form a “configuration” that includes
 - programs
 - documents
 - data ...

Documents

Consist of different type of manuals

- Documentation manuals
- Operating procedure manuals


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.3



Documentation Manuals

- Analysis Specifications
 - Formal specification
 - Context diagram
 - Data flow diagrams
- Design
 - Flow charts
 - Entity Relationship Diagrams
- Implementation
 - Source code listing
 - Cross reference listing
- Testing
 - Test data
 - Test results


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.4



Operating Procedural Manuals

- Consist of instructions to setup and use the software system and instructions on how to react to system failures
- User manuals
 - System overview
 - Beginner's Guide Tutorial
 - Reference Guide
- Operational manuals
 - Installation Guide
 - System Administration Guide

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.5



The Nature of Software

Software is intangible

- Hard to understand development effort

Software is easy to reproduce

- Cost is in its *development*
 - ✓ in other engineering products, manufacturing is the costly stage

The industry is labor-intensive

- Hard to automate

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.6

The Nature of Software...

Chances of Hacking

- Quality problems are hard to notice

Software is easy to modify

- People make changes without fully understanding it

Software does not 'wear out'

- ✓ **in ways that were not anticipated, thus making it complex**

- Software doesn't wear out
- Software is not manufactured

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.7

Software Characteristics

- Reusability of components
- Software is flexible

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.8

Types of Software

Custom

- For a specific customer
- **System Software**
- **Application Software**


Generic

- Sold on open market
- Often called
 - ✓ **COTS (Commercial Off The Shelf)**
 - ✓ **Shrink-wrapped**

Embedded

- Built into hardware
- Hard to change

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.9



Application Software

Real time software


- E.g. control and monitoring systems
- Must react immediately
- Safety often a concern

Data processing software

- Used to run businesses
- Accuracy and security of data are key

Some software has both aspects


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-10



Application Software

- Embedded software
- Business software
- Personal computer software
- Artificial Intelligence software
- Web based software
- Engineering and scientific software


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-11



Attributes of Good Software

- **Functionality**
 - to meet stated and implied need
- **Usability**
 - to be understood, learned and used
- **Reliability**
 - To maintain a specified level of performance
- **Portability**
 - To be adapted for different specified environment
- **Maintainability**
 - To be modified for the purposes of making corrections, improvements, or adaptations
- **Efficiency**
 - To provide appropriate performance relative to the amount of resources used


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-12



Software Crisis

- **Failure to master the complexity of software results in projects that are late, over budget and deficient in their stated requirements**
- **Software crisis arise because:**
 - Informal methods to specify what software should do
 - Software tools are complicated and unreliable


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-13



To Avoid Software Crises

- need to design software properly
 - To ease the verification
- need to maintain and upgrade software at a lower cost
 - Require Proper Documentation
- need to re-use components.
 - needs to precisely document what the software does
- important to have precise languages and tools
 - enable good documentation and communication of ideas at all stages
- standardized notations used to express specifications and designs
 - workers on a large project can collaborate without misunderstanding

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-14



What is Software Engineering?

- The process of solving customers' problems by the systematic development and evolution of large, **high-quality software** systems within **cost, time** and **other constraints**
- Solving customers' problems
 - Goal of software engineering
 - Sometimes the solution is to buy, not build
 - Adding unnecessary features does not help solve the problem
 - Software engineers must communicate effectively to identify and understand the problem

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-15

What S/W Engineering is and is not..

- Software engineering is concerned with "engineering" software systems, that is, building and modifying software systems:
 - on time,
 - within budget,
 - meeting quality and performance standards,
 - delivering the features desired/expected by the customer.
- Software engineering is not...
 - Just building small or new systems.
 - Hacking or debugging until it works.
 - Easy, simple, boring or even pointless!

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-16

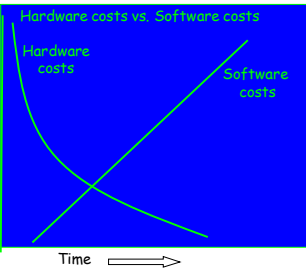
S/W Engineering and Computer Science

- Computer science is concerned with theory and fundamentals;
- Software engineering is concerned with the practicalities of developing and delivering useful software
- Computer science theories are currently insufficient to act as a complete underpinning for software engineering

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-17


Software Development Costs

What can you infer from the graph? **Software costs are increasing as hardware costs continue to decline**



- Hardware technology has made great advances
- Simple and well understood tasks are encoded in hardware
- Least understood tasks are encoded in software
- Demands of software are growing
- Size of the software applications is also increasing
- Hence "the software crisis"

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-18



Why are Software Projects Late? ...

Does effort necessarily == progress?


Is one man working six months equal to six men working one month?

Unit of man-month implies that men and month are interchangeable.

- True only when a task can be partitioned among many workers
- with no communication between them.
- For sequential tasks, more effort has no effect on the schedule.
- Many tasks in software engineering have sequential constraints.

Our estimating techniques fallaciously confuse effort with progress, hiding the assumption that men and months are interchangeable.
- Fred Brooks, *The Mythical Man-Month*

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-19



Why Software Projects are Late?...


Managers do not monitor progress effectively

Schedule slips day-by-day.

Day-by-day slips are harder to recognize,
harder to prevent and harder to make up.

How does a software project get to be a year late?
One day at a time!
Fred Brooks, *The Mythical Man-Month*

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-20




Why are Software Projects Late ?...

When we recognize slippage, should we add more people?

- Most tasks require communication among workers.
- Communication consists of:
 - Training.
 - Sharing information (intercommunication).
- Training affects effort at worst linearly, with the number of people.
- For n workers, intercommunication adds $n(n-1)/2$ to effort.
If each worker must communicate with every other worker.
- Adding more people to an already late project is usually like "Adding gasoline to fire!"

Brooks' Law:
Adding manpower to a late software project makes it later.
Fred Brooks, *The Mythical Man-Month*

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-21




Software Myths...

Myth : Sufficient literature full of standards and procedures for building the software

- Reality
 - Is the literature is really used?
 - Are software practitioners aware of its existence?
 - Does it reflects modern software engineering practice?
 - Is it complete?
 - Is it streamline to improve time to delivery while still maintaining the focus on quality?


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.22



Software Myths...

- Myth : Software is easy to change
- Myth : Computers provide greater reliability than the devices they replace
- Myth : Testing software or "proving" software correct can remove all the errors
- Myth : Reusing software increases safety
- Myth : Software can work right the first time


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.23



Software Myths cont..

- Myth : Software with more features is better software
- Myth : If we get behind schedule, we can add more programmers and catch up Propagated misinformation and confusion
- Myth :According to customer A general statement of objective is sufficient to begin writing programs- we can fill in the details later
- Myth : Once we write the program and get it work, our job is done
- Myth : Until I get the program running I have no way of assessing its quality
- Myth : The only deliverable work product for a successful project is the working program
- Myth : Software engineering will make us create voluminous and unnecessary documentation and will invariably slow us down


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.24



Software Process

- **Process** :Anything that operates for a period of time, normally consuming resources during that time and using them to create a useful result
- A set of activities whose goal is the development or evolution of software
- Generic activities in all software processes are:
 - **Specification** - what the system should do and its development constraints
 - **Development** - production of the software system
 - **Validation** - checking that the software is what the customer wants


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.25



Software Process Model

- A simplified representation of a software process, presented from a specific perspective
- Examples of process perspectives are
 - Workflow perspective - sequence of activities
 - Data-flow perspective - information flow
 - Role/action perspective - who does what
- Generic process models
 - Waterfall
 - Evolutionary development
 - Prototyping
 - Rapid Application development
 - Integration from reusable components

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.26



Difficulty in S/W Process Improvement

- Not enough time
- Lack of knowledge
- Wrong Motivations
- Insufficient Commitment

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.27



Software Life Cycle Model

Set of Processes that Results in Software Products

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.28



Generic Software Process Models

- Build-and-fix model
- Waterfall model
- Prototyping model
- Incremental model
- V Model
- Spiral model
- RAD

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.29



Inherent Problems With S/W Development

Requirements are complex

- The client usually does not know all the functional requirements in advance


Requirements may be changing

- Technology enablers introduce new possibilities to deal with nonfunctional requirements

Frequent changes are difficult to manage

- Identifying milestones and cost estimation is difficult

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.30




Inherent Problems with S/W Development..

There is more than one software system

- New system must often be backward compatible with existing system ("legacy system")
- Phased development: Need to distinguish between the system under development and already released systems


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-31



What Do Programmers Do?

- The Software Crisis led to a push to improve the way we develop software.
- Before we could do this, it was necessary to understand *how* software is developed.
- People soon recognized that what was commonly known as "programming" actually consisted of many more activities than just "programming".


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-32



Definitions

- Software lifecycle modeling
 - Attempt to deal with complexity and change
- Software lifecycle
 - Set of activities and their relationships to each other to support the development of a software system
- Software development methodology
 - A collection of techniques for building models - applied across the software lifecycle


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-33



Definitions..

- Series of identifiable stages that a software product undergoes during its lifetime and these stages is called a **life cycle phase**
- Breaking software development down into a number of phases like these led to the idea of the Software Lifecycle
 - cf. butterfly's lifecycle (caterpillar, pupae, ...)

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-34



S/W Development Sub Processes

Generating Request

- **System conception**

Discovering and documenting *what* the software system should do

- **Requirements Specification**


Deciding *how* the system is going to do it

- **Software Design**

Creating the software which implements it

- **Coding/Implementation/System Construction**
- **System Integration**

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-35



Software Development Activities cont..

Making sure that the software actually does what it is supposed to

- **Testing**
- **Software Quality Assurance**

Installing the software in the live environment

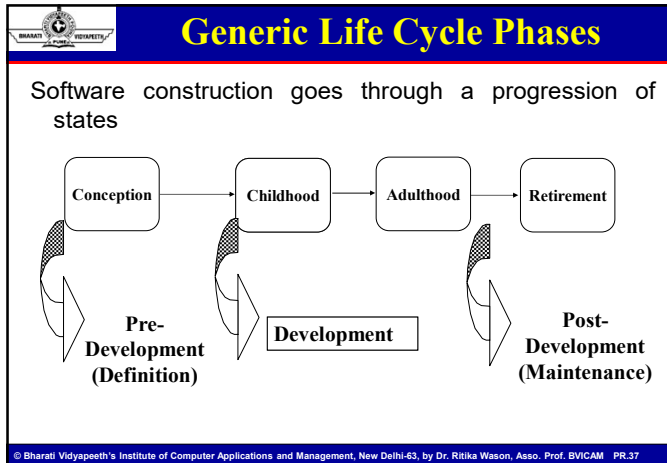
- **System Installation/System Conversion**

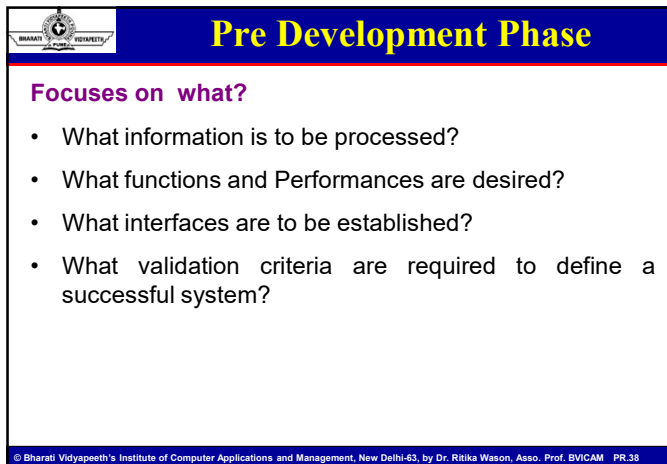
Keeping the software doing what it should

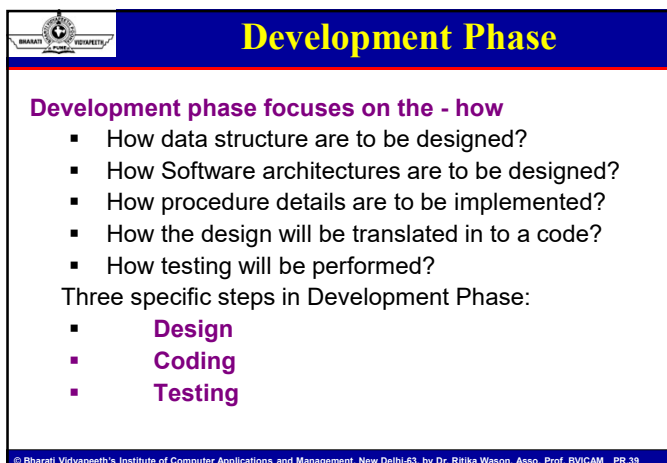
- **Software Maintenance/Evolution**

Phasing out the software when it is no longer of use

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-36







Post Development Phase

The Maintenance phase focuses on change that is associated with

- Corrective
- Adaptive
- Perfective
- Preventive

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.40

S/W Development Activities

Requirements Analysis	What is the problem?	Problem Domain
System Design	What is the solution?	
Program Design	What are the mechanisms that best implement the solution?	Implementation Domain
Program Implementation	How is the solution constructed?	
Testing	Is the problem solved?	
Delivery	Can the customer use the solution?	
Maintenance	Are enhancements needed?	

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.41

Processes, Activities and Tasks

Process Group:
 ▪ Consists of Set of Processes


Process:
 ▪ Consists of Activities

Activity:
 ▪ Consists of sub activities and tasks

```

  graph LR
    PG[Process Group] --> Dev[Development]
    P[Process] --> Design[Design]
    A[Activity] --> DD[Design Database]
    T[Task] --> MPR[Make a Purchase Recommendation]
  
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.42



Example

The Design Process is part of Development


The Design Process consists of the following Activities

- Perform Architectural Design
- Design Database (If Applicable)
- Design Interfaces
- Select or Develop Algorithms (If Applicable)
- Perform Detailed Design (= Object Design)

The Design Database Activity has the following Tasks

- Review Relational Databases
- Review Object-Oriented Databases
- Make a Purchase recommendation
-


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.43



Generic Software Process Models

- Build-and-fix model
- Waterfall model
- Rapid prototyping model
- Incremental model
- Spiral model
- RAD

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.44



Software Engineering Life Cycle Models

The life cycle model is selected based on:

- The nature of the Project and application
- The methods and tools to be used
- The deliverables that are required

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.45

Build and Fix Model

- Problems
 - No specifications
 - No design
- Cost is high
- Maintenance is extremely difficult
- Totally unsatisfactory
- Need life-cycle model
 - “Game plan”
 - Phases
 - Milestones
- Work for small Projects

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.46


Water Fall Model

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.47

Typical Characteristics

- Sometimes called classic life cycle or the linear sequential model
- Each phase is considered to be completed with the production of certain *deliverables*
- For development of a large-scale system, each phase will typically be undertaken by a different set of people
 - different skills are required for each activity
- Communication between phases is principally by means of the deliverables


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.48



Advantages

- Follows the usual engineering life cycle
- The Waterfall Model is simple to understand
 - even for non-technical managers!
- Its simplicity means that planning for a Waterfall development is relatively easy


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.49



Disadvantages

- Unfortunately, real projects are rarely so straightforward and sequential
- It is generally not possible to completely define (and freeze) all the requirements at the start of the project
- It is not until late in the process that something that can be demonstrated to the user is created
- In practice, "blocking states" occur, causing delays for some members of the team

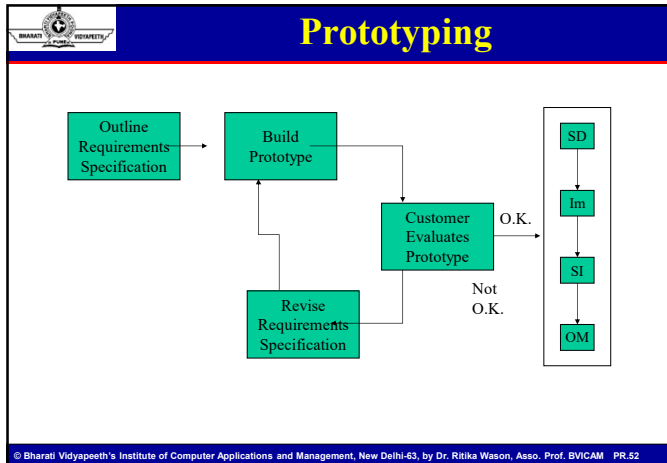
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.50

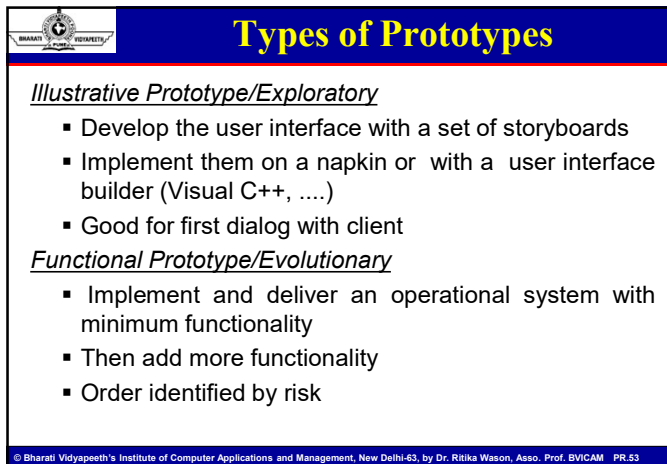


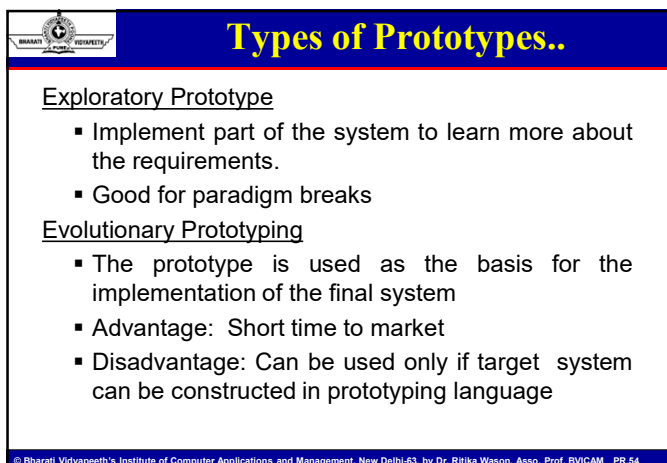
However ...

- ... there is no question that even a poor model like the Waterfall model is significantly better than no model at all
- Variants of this sequential model are still widely used today, covering more or less of the activities that surround software development

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.51







Advantages

- Prototype system is much easier to evaluate than a dry SRS document
- Customers and users can give immediate feedback on the requirements specification
- The implications of requirements can be judged within the “live” environment
- Construction of the prototype can help developers to make better decisions when implementing the full system

Quarter	East	West	North
1st Qtr	20	30	45
2nd Qtr	25	35	45
3rd Qtr	85	35	45
4th Qtr	20	30	45

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.55

Disadvantages

- The customer may think that the prototype is nearly the finished product
- As a result, the customer may not be prepared to wait another 6 months (or whatever) while the system is “rebuilt”
- Requires extensive participation and involvement of the customer, which is not always possible

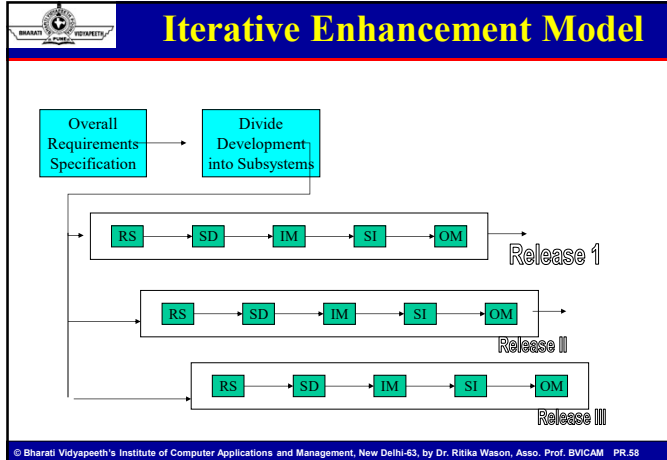
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.56

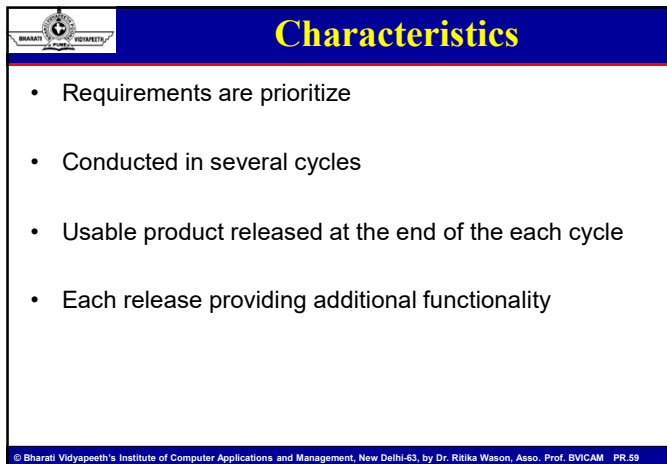
The Incremental Model

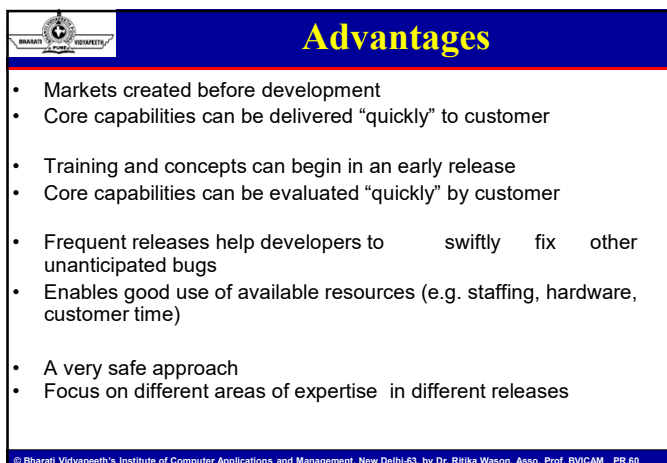
```

graph TD
    A[Overall Requirements Specification] --> B[Divide Development into Subsystems]
    B --> C1[RS]
    B --> C2[RS]
    B --> C3[RS]
    C1 --> D1[SD]
    C2 --> D2[SD]
    C3 --> D3[SD]
    D1 --> E1[Im]
    D2 --> E2[Im]
    D3 --> E3[Im]
    E1 --> F1[SI]
    E2 --> F2[SI]
    E3 --> F3[SI]
    F1 --> G1[OM]
    F2 --> G2[OM]
    F3 --> G3[OM]
    
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.57







Disadvantages

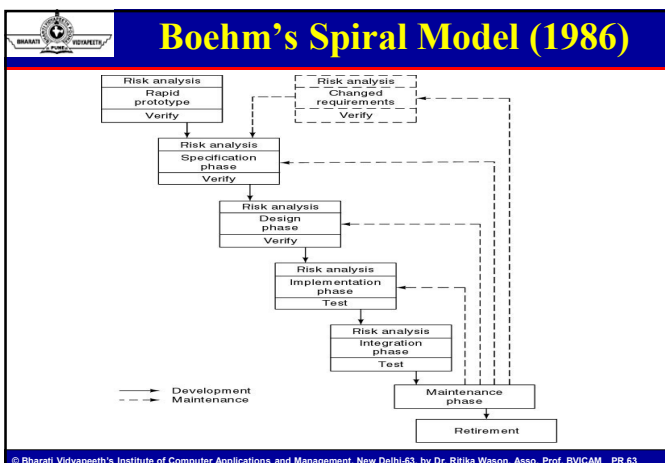
- Every increment must be developed through to production standard
- Extra time spent on testing, documenting and maintaining a "temporary" product
- Can be difficult to split the problem up into appropriate increments
- Expensive

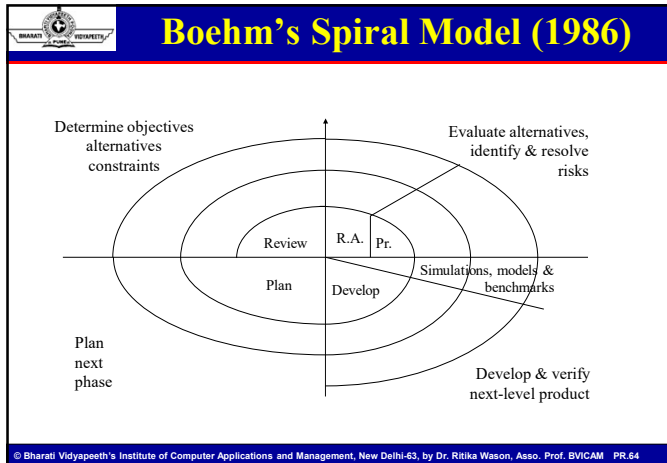
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.61

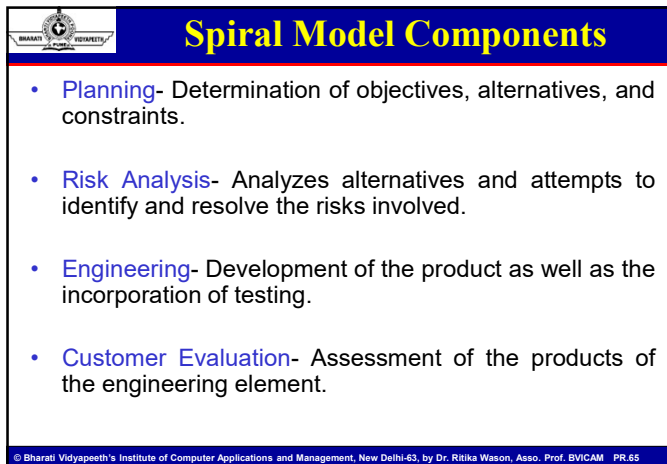
Evolutionary Development Model

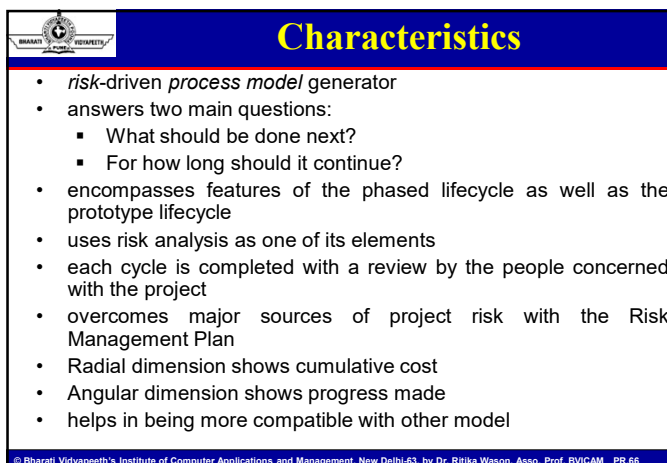
- Resembles iterative enhancement model
- Does not require a useable product at the end of each cycle
- Requirements are implemented by category rather than by priority
- Used when it is not necessary to provide a minimal version of the system quickly
- Useful for projects using new technology or complex projects

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.62







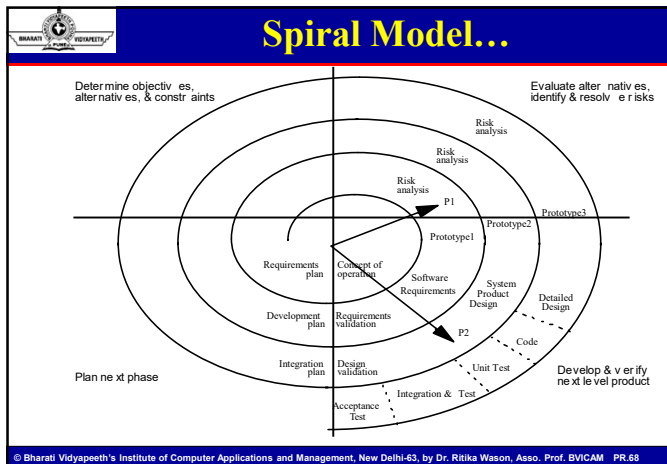


Activities ("Rounds")

- ◆ Concept of Operations
- ◆ Software Requirements
- ◆ Software Product Design
- ◆ Detailed Design
- ◆ Code
- ◆ Unit Test
- ◆ Integration and Test
- ◆ Acceptance Test
- ◆ Implementation

- ◆ For each cycle go through these steps
 - Define objectives, alternatives, constraints
 - Evaluate alternative, identify and resolve risks
 - Develop, verify prototype
 - Plan next "cycle"


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.67



Strengths

- has a wide range of options to accommodate the good features of other lifecycle models.
- the risk-avoidance approach keeps from having additional difficulties.
- prepares for lifecycle evolution, growth, and changes of the software product.
- incorporates software quality objectives into software product development. Emphasis is placed on identifying all objectives and constraints during each round.
- The risk analysis and validation steps eliminate errors early on.
- Great amounts of detail are not needed except in the case where this lack of detail jeopardizes the project.


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.69



Weaknesses

- Lack of explicit process guidance in determining objectives, constraints, alternatives
- The risk-driven model is dependent on the developers' ability to identify project risk
- Provides more flexibility than required for many applications


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.70



The Limitations of the Waterfall and Spiral Models

- Neither of these model deals well with frequent change
 - The Waterfall model assume that once you are done with a phase, all issues covered in that phase are closed and cannot be reopened
 - The Spiral model can deal with change between phases, but once inside a phase, no change is allowed
- What do you do if change is happening more frequently? ("The only constant is the change")

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.71



Rapid Application Development (RAD)

- Topical in 1990's after
- Book Rapid Application Development by Martin, J (1991)
- a 'tool kit' methodology
- can utilize a wide range of techniques and tools
- Incremental, plus reliance on many standard modules
- usually very small team.
- emphasis on user involvement and responsibility throughout whole development
- Quality definition in a RAD environment put by James Martin
- ***"meeting the true business (or user) requirements as effectively as possible at the time the system comes into operation"***

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.72

RAD Goals

Radically changes way systems are developed with goals of.

- High quality systems
- fast development and delivery
- low costs

should go hand in hand when the right tools and techniques are used

A Venn diagram with three overlapping circles. The top-left circle is labeled 'High Speed', the top-right circle is labeled 'High Quality', and the bottom circle is labeled 'Low Cost'. The circles overlap in the center, and each pair of circles also overlaps.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.73

RAD Properties

- Must be delivered in 2 - 6 months
- split into increments if too large to enable this
- each increment is implemented separately with frequent delivery of working parts of system.


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.74

Rapid Development

A flowchart with 'Rapid Development' at the top. It branches into four boxes: 'Small Teams', 'Reusable parts', 'Automated tools', and 'User Involvement'. 'Small Teams' leads to 'Lower Cost'. 'Reusable parts' leads to 'Higher Quality'. 'Automated tools' leads to 'Meets business needs better'. 'User Involvement' leads to 'Meets business needs better'. Both 'Higher Quality' and 'Meets business needs better' lead to 'Lower maintenance costs'.

Rapid development, high quality and lower costs go hand in hand if an appropriate development methodology is used, Martin 1991

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.75



RAD - Essentials

Tools

- Code generators, CASE tools, prototyping tools and 4GLs

Methodology

- to use tools as effectively as possible

People

- right skills and talents. Well selected and motivated. End users


Management

- not place obstacles, facilitate fast development

Infrastructure

- In which fast development can take place


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.76



Advantages

- Quick initial view is possible
- Less development time
- User involvement increases the acceptability


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.77



Disadvantages

- User involvement is difficult through out the life cycle
- Difficult to reduce the development time significantly
- Reusable components may not be available
- Availability of highly skilled personnel is difficult
- Not effective if system is not modularized properly


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.78



Selection of a Life Cycle Model

- Requirement
- Development Team
- Users
- Project type & Associated Risk


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.79



Based on Requirements

Requirements	Waterfall	Prototype	Iterative Enhance	Evolut. Develop.	Spiral	RAD
Easily understandable and defined	Yes	No	No	No	No	Yes
Change requir.	No	Yes	No	No	Yes	No
Define requir. early	Yes	No	Yes	Yes	No	Yes
Indicating a complex system to be built	No	Yes	Yes	Yes	Yes	No

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.80



Based on Development Team

Development Team	Waterfall	Prototype	Iterative Enhance	Evolut. Develop.	Spiral	RAD
Less experience on similar projects	No	Yes	No	No	Yes	No
Less domain knowledge (new to technology)	Yes	No	Yes	Yes	Yes	No
Less Experience on tools	Yes	No	No	No	Yes	No
Availability of training, if required	No	No	Yes	Yes	No	Yes

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.81

Based on User Involvement						
User Involvement	Waterfall	Prototype	Iterative Enhance	Evolut. Develop.	Spiral	RAD
In all phases	No	Yes	No	No	No	Yes
Limited participation	Yes	No	Yes	Yes	Yes	No
No previous experience of participation in similar projects	No	Yes	Yes	Yes	Yes	No
Expert in problem domain	No	Yes	Yes	Yes	No	Yes

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.82

Based on Project Type & Risk						
Project type & Risk	Waterfall	Prototype	Iterative Enhance	Evolut. Develop.	Spiral	RAD
Enhancement of existing system	No	No	Yes	Yes	No	Yes
Funding is stable for the project	Yes	Yes	No	No	No	Yes
High reliability requirements	No	No	Yes	Yes	Yes	No
Tight project schedule	No	Yes	Yes	Yes	Yes	Yes
Use of reusable components	No	Yes	No	No	Yes	Yes
Resources scarce	No	Yes	No	No	Yes	No

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.83

What we learnt
<ul style="list-style-type: none"> ✓ Inherent Problems with Software Development ✓ Generic life cycle phases ✓ Processes, Activities and Tasks ✓ Modeling Dependencies in a Software Lifecycle ✓ Generic software process models <ul style="list-style-type: none"> ✓ Build-and-fix model ✓ Waterfall model ✓ Prototyping model ✓ Incremental model ✓ V Model ✓ Spiral model ✓ RAD ✓ Selection of Life Cycle Model

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.84

Evolution of Object Orientation

- The idea of object-oriented programming gained **momentum** in the 1970s and in the early 1980s.
- Bjorn Stroustrup** integrated object-oriented programming into the C language. The resulting language was called **C++** and it became the **first object-oriented language** to be widely used commercially.
- In the early 1990s a group at Sun led by **James Gosling** developed a simpler version of C++ called **Java** that was meant to be a programming language for video-on-demand applications.
- This project was going nowhere until the group **re-oriented** its focus and marketed Java as a language for programming Internet applications.
- The language has gained **widespread popularity** as the Internet has boomed, although its market penetration has been limited by its inefficiency.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.85

Evolution of Object Orientation

- Monolithic Programming Approach:** In this approach, the program consists of **sequence of statements** that **modify data**.

- All the **statements** of the program are **Global** throughout the whole program. The **program control** is achieved through the use of **jumps** i.e. **goto statements**.
- In this approach, **code is duplicated** each time because there is no support for the function. **Data is not fully protected** as it can be accessed from any portion of the program.
- So this approach is useful for designing **small and simple** programs. The programming languages like **ASSEMBLY** and **BASIC** follow this approach.


Machine Language	Monolithic Approach Assembly and BASIC	Procedural Approach FORTRAN and COBOL	Structured Prog. App C and PASCAL	OOP C++ and JAVA
------------------	---	--	--------------------------------------	---------------------

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.86

Evolution of Object Orientation

Program in monolithic programming

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.87




Evolution of Object Orientation

2. **Procedural Programming Approach:** This approach is **top down approach**. In this approach, a program is divided into **functions** that perform a **specific task**.

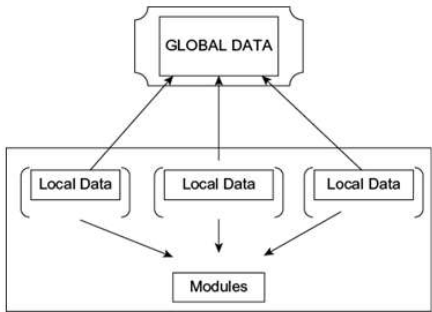
- This approach **avoids repetition of code** which is the main drawback of **Monolithic Approach**.
- The basic **drawback** of Procedural Programming Approach is that **data is not secured** because data is **global** and can be accessed by any function.
- This approach is mainly used for **medium sized applications**. The programming languages: **FORTTRAN and COBOL** follow this approach.

•3. **Structured Programming Approach:** The basic principal of **structured programming approach** is to divide a program in **functions and modules**.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.88




Evolution of Object Orientation



Program in procedural/structured programming

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.89




Evolution of Object Orientation

- The use of modules and functions makes the program more **comprehensible** (understandable). It helps to write **cleaner code** and helps to **maintain control** over each function. This approach gives importance to **functions** rather than **data**.
- It focuses on the development of large software applications. The programming languages: **PASCAL and C** follow this approach.

4. **Object Oriented Programming Approach:** The basic principal of the OOP approach is to **combine** both **data** and **functions** so that both can operate into a **single unit**. Such a unit is called an **Object**.

- This approach **secures data** also. Now a days this approach is used mostly in applications. The programming languages: **C++ and JAVA** follow this approach. Using this approach we can write any lengthy code.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.90




Object Orientation Paradigm

- An approach to the solution of problems in which all **computations** are performed in **context of objects**.
- The objects are instances of **programming constructs**, normally called as **classes** which are **data abstractions** with **procedural abstractions** that operate on objects.
- A software system is a set of mechanism for performing certain **action** on **certain data**

$$\text{Algorithm} + \text{Data structure} = \text{Program}$$
- **Data Abstraction + Procedural Abstraction**


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.91



Trade-offs of a Programming

- Ease-of-use versus power
- Safety versus efficiency
- Rigidity versus extensibility


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.92



Object & Classes

- **Class** is at **core** of Object Orientation
- Any concept implemented in any object-oriented programming language is **encapsulated** within a class
- Class defines **new data type** which is used to create object of that type.


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.93



Classes – The Blueprint !!

- A **class** is a *blueprint of an object*.
- A class is a **group of objects** that share **common properties & behavior/ relationships**.
- In fact, **objects** are the **variables** of the **type class**.
- Classes are **user defined data types** and behaves like the built-in types of a programming language.
- **Class** are a **concept**, and the **object** is the **embodiment** of that **concept**.
- Each class should be designed and programmed to accomplish **one, and only one, thing**, in accordance to **single responsibility principle** of **SOLID design principles**.
- In the OOPs concept the variables declared inside a class are known as "**Data Members**" and the functions are known as "**Member Functions**".


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.94



Class Members

- A class has different **members**, and developers in Microsoft suggest to program them in the following order:
- **Namespace**: The namespace is a keyword that defines a **distinctive name** or last name for the class. A namespace categorizes and organizes the library (assembly) where the class belongs and avoids **collisions** with classes that share the same name.
- **Class declaration**: Line of code where the class name and type are **defined**.
- **Fields**: Set of **variables** declared in a class block.
- **Constants**: Set of constants declared in a **class block**.
- **Constructors**: A method or group of methods that contains code to **initialize** the class.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.95



Class Members

- **Properties**: The set of **descriptive data** of an object.
- **Events**: Program **responses** that get fired after a user or application action.
- **Methods**: Set of **functions** of the class.
- **Destructor**: A method that is called when the class is **destroyed**. In managed code, the Garbage Collector is in charge of destroying objects; however, in some cases developers need to take extra actions when objects are being released, such as freeing handles or deallocating unmanaged objects.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.96

Classes – A Classification

A **Class** “is a set of objects that share a common structure and a common behavior.” [Booch 1994].

Abstract Classes cannot be instantiated directly.

- The main purpose of an abstract class is to define a common interface for its subclasses.

Concrete Classes are not abstract and can have instances.

```

classDiagram
    class AbstractClass {
        operation()
    }
    class ConcreteClass {
        operation()
    }
    AbstractClass <|-- ConcreteClass
            
```

8

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.97

Defining Classes..

- Initializing data fields
 - By setting a value in a constructor
 - By assigning a value in the declaration
 - An initialization block
- When constructor is called
 - All dat fields are initialized to their default values
 - All fields initializers and initialization blocks are executed, in the order in which they occur in the class declaration
 - If the first line of the constructor calls a second constructor, then the body of the second constructor is executed
 - The body of the constructor is executed

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.98

Defining Classes..

- Object Destruction & the finalize Method
 - Java doesn't support destructors
 - finalize method can be added to any class
 - Called before the garbage collector deprecated alternative is Runtime.addShutdownHook

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.99

Object- The CRUX of the matter!!

- o “An object is an **entity** which has a **state** and a defined set of **operations** which **operate** on that state.”
- o The **state** is represented as a set of **object attributes**. The operations associated with the object **provide services** to other objects (clients) which request these services when some **computation** is required
- o Objects are **created** according to some **object class definition**. An object class definition serves as a **template** for objects. It includes **declarations** of all the attributes and services which should be associated with an object of that class.
- o An Object is anything, **real** or **abstract**, about which we **store data** and those **methods** that **manipulate** the **data**.
- o An **object** is a component of a program that knows how to perform certain **actions** and how to **interact** with other elements of the program.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-100

Object- The CRUX of the matter!!

- Each **object** is an **instance** of a particular **class** or **subclass** with the class's own **methods** or procedures and **data variables**. An object is what **actually runs** in the computer.
- Objects are the basic **run time entities** in an **object oriented system**.
- They **match** closely with **real time objects**.
- Objects take up **space in memory** and have an associated **address** like a Record in Pascal and a Structure in C.
- Objects interact by **sending Message** to one other. E.g. If “Customer” and “Account” are two objects in a program then the customer object may send a message to the account object requesting for bank balance without divulging the details of each other's data or code.
- Code in object-oriented programming is **organized** around **objects**.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-101

Object- A representation

The diagram illustrates two objects as containers. The first container, labeled 'An Object', has a section for 'Data Members' and a section for 'Functions'. The second container, labeled 'A Car', has a section for 'Model', 'Year of Mfg', and 'Colour' (representing data members), and a section for 'Start', 'Move', and 'Stop' (representing functions).

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-101

Object- Attributes and Methods

Object's Attributes

- Attributes represented by **data type**.
- They describe objects **states**.
- In the Car example the car's attributes are: color, manufacturer, cost, owner, model, etc.

Object's Methods

- Methods define objects **behavior** and specify the way in which an Object's data are **manipulated**.
- In the Car example the car's methods are: drive it, lock it, carry passenger in it.

Objects- blueprints of classes

- The role of a class is to define the **state** and **behavior** of its instances.
- The class car, for example, defines the property color.
- Each individual car will have property such as "maroon," "yellow"

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-103

Object Orientated Features


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-104

Object Orientated Features

Object orientation adapts to the following criteria's-

1. Changing requirements
2. Easier to maintain
3. More robust
4. Promote greater design
5. Code reuse
6. Higher level of abstraction
7. Encouragement of good programming techniques
8. Promotion of reusability


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-105



Object Orientated Features

1. **OBJECT** - Object is a **collection** of number of **entities**. Objects take up space in the memory. Objects are **instances of classes**. When a program is executed, the objects interact by sending **messages** to one another. Each object contains **data** and **code** to manipulate the data. Objects can interact without knowing details of each other's data or code. **Each instance** of an object can hold its own **relevant data**.
2. **CLASS** - Class is a **collection** of **objects** of **similar type**. Objects are **variables** of the **type class**. Once a class has been defined, we can create any number of objects belonging to that class. Classes are **user define data types**. A class is a **blueprint** for any **functional entity** which defines its **properties** and its **functions**.


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-106



Object Orientated Features

3. **DATA ENCAPSULATION** – Combining data and functions into a **single unit** called **class** and the process is known as **Encapsulation**. **Class variables** are used for storing data and functions to specify various operations that can be performed on data. This process of **wrapping up** of data and functions that operate on data as a **single unit** is called as data encapsulation. Data is **not accessible** from the outside world and only those functions which are present in the class can access the data.
4. **DATA ABSTRACTION**- Abstraction (from the Latin *abs* means *away from* and *trahere* means *draw*) is the **process** of taking away or **removing characteristics** from something in order to reduce it to a **set of essential characteristics**. Advantage of data abstraction is **security**.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-107



Object Orientated Features

5. **INHERITANCE**- It is the process by which object of one class **acquire the properties** or features of objects of **another class**. The concept of inheritance provides the idea of reusability means we can add **additional features** to an existing class **without modifying it**. This is possible by deriving a new class from the existing one. **Advantage** of inheritance is **reusability** of the **code**.
6. **MESSAGE PASSING** - The process by which **one object** can **interact** with **other object** is called **message passing**.
7. **POLYMORPHISM** - A Greek term means **ability to take more than one form**. An operation may exhibit **different behaviours** in different instances. The behaviour depends upon the **types of data** used in the operation.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-108

Object Orientated Features

8. PERSISTENCE - The process that allows the **state** of an **object** to be saved to **non-volatile storage** such as a file or a database and later **restored** even though the original creator of the object no longer exists.

```

graph TD
    Root[Pillars of Object Oriented Programming] --> Major[Major Pillars]
    Root --> Minor[Minor Pillars]
    Major --> Abstraction
    Major --> Encapsulation
    Major --> Modularity
    Major --> Hierarchy
    Minor --> Concurrency
    Minor --> Persistence
  
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-109

What are Requirements?

- Defined, during the early stages of a system development as a **specification** of what should be implemented or as a constraint of some kind on the system.
- Can be defined as
 - a user-level facility description,
 - a detailed specification of expected system behavior,
 - a general system property,
 - a specific constraint on the system,
 - information on how to carry out some computation,
 - a constraint on the development of the system.
- inevitable as requirements may serve a dual function
 - the basis for a bid for a contract - therefore must be open to interpretation
 - the basis for the contract itself - therefore must be defined in detail

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-110

What happens if the Requirements are Wrong?

- The system may be delivered late and cost more than originally expected.
- The customer and end-users are not satisfied with the system. They may not use its facilities or may even decide to scrap it altogether.
- The system may be unreliable in use with regular system errors and crashes disrupting normal operation.
- If the system continues in use, the costs of maintaining and evolving the system are very high.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-111

Why is Requirements Engineering Difficult?

- Businesses operate in a rapidly changing environment so their requirements for system support are constantly changing.
- Multiple stakeholders with different goals and priorities are involved in the requirements engineering process.
- System stakeholders do not have clear ideas about the system support that they need.
- Requirements are often influenced by political and organizational factors that stakeholders will not admit to publicly.
- Over-reliance on CASE tools
- Tight project schedule
- Communication barriers
- Lack of resources

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-112

Requirement Engineering Process Steps

```

graph TD
    subgraph RD [Requirements Definition]
        RE[Requirements Elicitation]
        RA[Requirements Analysis]
        RDoc[Requirements Documentation]
        RR[Requirements Review]
    end
    RD --> SRS[Software Requirements Specification]
    subgraph RM [Requirements Management]
        RCM[Requirements Change Management]
        RT[Requirements Traceability]
    end
  
```

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-113


Definitions and Specifications

Requirement definition
The software must provide a means of representing and accessing external files created by other tool

Requirement Specifications

1. The user should be provided with facilities to define the type of external files.
2. Each external file type may have an associated tool which may be applied to the file.
3. Each external file type may be represented as a specific icon on the user display.
4. Facilities should be provided for the icon representing an external file to be defined by the user
5. When a user selects an icon representing an external file, the effect of that selection is to apply the tool associated with the type of the external file to the file represented by the selected

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-114



Type of Requirements-I

Functional requirements

- Statements of services the system should provide,
- how the system should react to particular inputs
- how the system should behave in particular situations.


Non-functional requirements

- constraints on the services or functions offered by the system such as
- timing constraints, constraints on the development process, standards, etc.

Domain requirements

- Requirements that come from the application domain of the system
- reflect characteristics of that domain


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.115



Examples of Functional Requirements

- The user shall be able to search either all of the initial set of databases or select a subset from it.
- The system shall provide appropriate viewers for the user to read documents in the document store.
- Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copy to the account's permanent storage area.

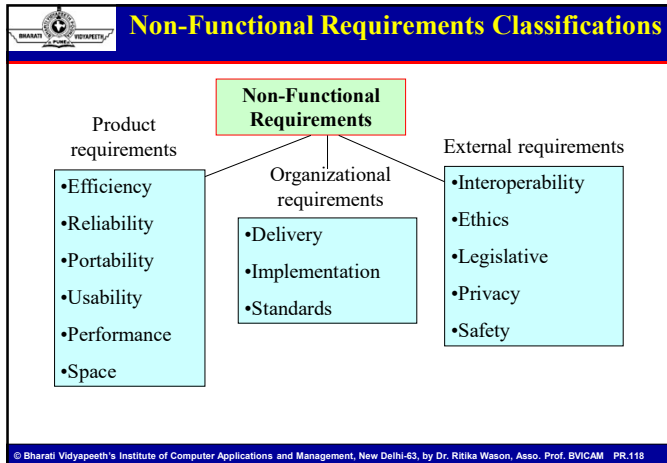
© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.116



Non-functional Requirements

- Define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.117



Non-functional Requirements Examples

Product requirement

- 4.C.8 It shall be possible for all necessary communication between the APSE and the user to be expressed in the standard Ada character set

Organisational requirement

- 9.3.2 The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-95

External requirement


- 7.6.5 The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.119

Non-functional Requirements Measures

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	K Bytes Number of RAM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR.120



Type of Requirements-II

Known requirements

- Something a stakeholder believes to be implemented

Unknown requirements


- Forgotten by the stakeholder because they are not needed right now or needed only by another stakeholder

Undreamt requirements

- Stakeholder may not be able to think of new requirements due to limited domain knowledge

Known, Unknown and Undreamt requirement may be functional or nonfunctional


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-121



Type of Requirements-III

- User requirements
- System requirements


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-122



User Requirements

- Should describe functional and non-functional requirements so that they are understandable by system users who don't have detailed technical knowledge
- User requirements are defined using natural language, tables, and diagrams
- Problems with natural language
 - Precision vs. understand ability
 - Functional vs. non-functional requirements confusion
 - Requirements amalgamation


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-123



System Requirements

- More detailed specifications of user requirements
- Serve as a basis for designing the system
- May be used as part of the system contract


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-124



Requirement Document

- Specify external system behaviour
- Easy to change
- Serve as reference tool for maintenance
- Record forethought about the life cycle of the system
 - i.e. predict changes
- Characterise responses to unexpected events

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-125



Requirements Elicitation and Analysis


Requirements Elicitation:

- Definition of the system in terms understood by the client ("Problem Description")
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc.
- These are called *stakeholders*.

Requirements Analysis:

- Technical specification of the system in terms understood by the developer ("Problem Specification")


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-126



Requirement Elicitation Methods

- Onsite Observation
- Questionnaire
- Interviews
- Brainstorming Sessions
- Facilitated Application Specification Technique (FAST)
- Quality Function Deployment (QFD)
- Viewpoint-oriented elicitation
- Use Case Approach
- Prototyping

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-127



Interviews

- Face-to-face interpersonal meeting designed to identify relations or verify information and capture information as it exists


Advantages

- Flexible tool
- Offering better opportunity than questionnaire
- Effective for complex subjects
- People enjoy being interviewed

Drawbacks

- Needs preparation time and money to conduct

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-128



Interviews cont..

The art of interviewing

- Creating permissive situation in which the answers offered are reliable

Arranging the interview


- Physical location, time of the interview and order of interviewing assures privacy and minimal interruption

Guides to a successful interview

- Set the stage for the interview
- Establish rapport; put the interviewee at ease
- Phrase questions clearly and succinctly
- Be a good listener; avoid arguments
- Evaluate the outcome of the interview

Interviews may be open-ended or structured


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-129



Selection of Stakeholder

- Must be selected based on their technical expertise, domain knowledge, credibility and accessibility
- Several groups to be considered for interview
 - Entry Level personnel
 - Mid level stakeholders
 - Managers or other Stakeholders
 - Users of the Software


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-130



Brainstorming Sessions

- A group technique to understand the requirements
- Requirements in the long list can be categorized, prioritized and pruned
- The facilitator required to handle group bias and group conflicts


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-131



Basic Guidelines

- Arrange a meeting at a neutral site for developers and customers
- Establishment of rules for preparation and participation
- Prepare an informal agenda that encourages free flow of ideas
- Appoint a facilitator
- Prepare a definition mechanism
- Participants should not criticize or debate

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-132




FAST Session Preparations

Each FAST attendee is asked to make a list of objects that are:

- Part of the environment that surrounds the system
- Produced by the system
- Used by the system
- List of services that interact or manipulate the objects
- List of constraint
- Performance criteria


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-133



Activities of FAST session

- Participants presents the list of objects, services, constraints and performance for discussion
- Prepare the combine list for each topic
- Discuss the consensus combined list and finalized by facilitator
- Team are divided in subteams, each works for mini specifications
- Each subteam presents the mini specifications to all FAST attendee
- Prepare the issue list
- Each attendee prepares a list of validation criteria to finalize the consensus validation criteria list
- Subteam write the complete draft specifications using all inputs from the FAST meeting


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-134



QFD steps

- Identify all the stakeholders and any initial constraints identified by customer that affect requirement development
- List out customer requirements
- Assign a value to each requirement indicating the degree of importance
- Final list of requirements may be reviewed by requirement engineers and categorize like
 - It is possible to achieve
 - It should be deferred and the reason thereof
 - It is impossible and should be dropped from consideration


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-135



Nature of the SRS

- Functionality
- External Interfaces
- Performance
- Attributes
- Design Constraints imposed on an implementation


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-136



Characteristics of a Good SRS

- Correct
- Unambiguous
- Complete
- Consistent
- Ranked for importance and/or stability
- Verifiable
- Modifiable
- Traceable

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-137



Review Question Objective

Q1. Which is the oldest Profession? Why?
a) Physician b) Civil Engineer c) Computer Scientist

Q2. What is Software?

Q3. What are different types of Documents?

Q4. Define Documentation Manual?

Q5. Define Procedure Manuals?

Q6. What are Documentation Manuals?

Q7. What are Operating Procedure Manuals?


Q8. Why Software is Intangible?

Q9. Why Software is easy to reproduce?

Q10. Why Software industry is Labor Intensive?

Q11. Why Software is easy to modify?

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-138

 **Review Question Objective..**

Q12. Software doesn't wear out. Give reason.

Q13. What are different Software Characteristics?

Q14. Define Failure Intensity.

Q15. Compare Failure intensity of Hardware and Software.

Q16. What are different types of software?


Q17. Define Custom, Generic and Embedded Software.

Q18. For which type of software maximum number of copies are used
a) Custom b) Generic c) Embedded

Q19. For which type of software lowest processing time is devoted
a) Custom b) Generic c) Embedded

Q20. For which type of software annual development efforts are highest
a) Custom b) Generic c) Embedded

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-139

 **Review Question Objective**

Q21. How you will decide the Software is Good or Bad?

Q22. What are the different types of maintenance?

Q23. Differentiate Corrective and perfective maintenance?

Q24. What type of maintenance is least in practice?

Q25. Define Following Software Attributes
i) Maintainability ii) Dependability iii) Efficiency iv) Usability

Q26. Define Software Crises?


Q27. Give three suggestions to avoid the situation of Software Crises.

Q28. What Are Quality Issues in Software?

Q29 Why Software Cost are increasing as Hardware Costs Continue to decline?

Q30. What are the primary drivers of Software Cost?

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-140

 **Review Question Objective..**

Q31. Why Cost to fix an error increases as it is found later and later in the software lifecycle?

Q32. Why Software Project late? Give any two reasons.

Q33. When we recognize slippage in the IT Project, should we add more people? Give Reason.

Q34. Define Quality, Quality Control, Quality Assurance and Quality Management System.

Q29. Q35. Differentiate Quality Control and Quality Assurance.

Q36. Define CASE


Q37. Differentiate Lower CASE and Upper CASE

Q38. Write at least two software myths.

Q39. Define Process

Q40. What are generic activities in all Software Process?


© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-141

 **Review Question Objective..**

Q41. Why Software process improvement is difficult?
 Q42. Draw and label Process Improvement Learning Curve.
 Q43. Define the terms with example


- I. Deliverables & Milestones
- II. Product
- III. Process
- IV. Measure, Metric, Measurement
- V. Software Process & Product Metric
- VI. Productivity
- VII. Effort
- VIII. Software Component
- IX. Verification & Validation

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-142

 **Review Question Objective..**


Q44. Define Software Lifecycle Modeling, Software Lifecycle, Software Development Methodology, Software Life Cycle Phase
 Q45. What must be focus in Pre Development Phase of any Software?
 Q46. What must be focus in Development Phase of any Software?
 Q47. What are the three specific steps in Development Phase of any Software?
 Q48. What must be focus in Post Development Phase of any Software?
 Q49. Define Process, Activity and Task with example.
 Q50. Project risk factor is consider in (i) Waterfall model (ii) Spiral model (iii) Quick & Fix model (iv) (ii) and (iii).

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-143

 **Review Question Short Type**

Q1. Describe the characteristics of software contrasting it with the characteristics of hardware?
 Q2. Why researchers identify the necessity to engineer software?
 Q3. Why we are in perpetual "software crisis"?
 Q4. Discuss the history of software crises with examples. Why are the case tools not normally able to control it?
 Q5. What is software engineering? Is it an art, craft or a science? Discuss.
 Q6. What is more important : product or process? Justify your answer.
 Q7. What are the key challenges facing software engineering?
 Q8. Why is the primary goal of software development now shifted from producing good quality software to good quality maintainable software?

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-144

 **Review Question Short Type**

Q10. List down the documents after each activity of Software Development Activity.


Q11. What do you understand by the expression “life cycle model of software development”? Why is it important to adhere to a life cycle model during the development of a larger software product?

Q12. What problems will a software development organization face if it does not adequately document its software process?

Q13. Q14. Compare iterative enhancement model and evolutionary development model.

Q15. What do you understand by the expression “life cycle model of software development”? Why is it important to adhere to a life cycle model during the development of a larger software product?

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-146

 **Review Question Long Type**

Q1. Discuss different types of Software Documents.

Q2. Discuss the nature of software


Q3. What are various type of Software?

Q4. How are the software myths affecting software process? Explain with the help of example.

Q5. What do you mean by a software process? What is the difference between a methodology and a process? What problems will a software development house face if it does not follow any systematic process in its software development efforts?


Q6. Discuss the selection process parameters for a life cycle model.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-146

 **References**

1. K. K. Aggarwal & Yogesh Singh, “Software Engineering”, 2nd Ed., New Age International, 2005.
2. R. S. Pressman, “Software Engineering – A practitioner's approach”, 5th Ed., McGraw Hill Int. Ed., 2001.
3. Pankaj Jalote, “An Integrated Approach to Software Engineering”, Narosa, 3rd Ed., 2005.
4. Stephen R. Schach, “Classical & Object Oriented Software Engineering”, IRWIN, 1996.
5. James Peter, W. Pedrycz, “Software Engineering: An Engineering Approach”, John Wiley & Sons.
6. Sommerville, “Software Engineering”, Addison Wesley, 8th Ed., 2009.

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-147

	<h2 style="text-align: center;">References</h2>
7.	Frank Tsui and Orlando Karan, "Essentials of Software Engineering", Joes and Bartlett, 2nd Ed., 2010.
8.	Rajib Mall, "Fundamrntal of Software Engineering", PHI, 3rd Ed., 2009.
9.	Carlo Ghizzi , Mehdi Jazayeri and Dino Mandrioli, " Fundamental of Software Engineering", PHI, 2nd Ed.,2003.
10.	Carol L. Hoover, Mel Rosso-Llopart and Gil Taran, "Evaluating Project Decision Case Studies in Software Engineering", Pearson, 2010.
11.	Sommerville, "Software Engineering", Addison Wesley, 2002 http://www.cse.iitk.ac.in/JaloteSEbook/
12.	http://www.csse.monash.edu.au/~ionmc/CSE2305/Topics/07.14

© Bharati Vidyapeeth's Institute of Computer Applications and Management, New Delhi-63, by Dr. Ritika Wason, Asso. Prof. BVICAM PR-148
