Course Name: Software Engineering

Course Code: KCS601

UNIT 1: INTRODUCTION

Faculty Name: Mr. Ashish Jain

Faculty Email: ashish.jain@glbitm.ac.in

Vision and Mission

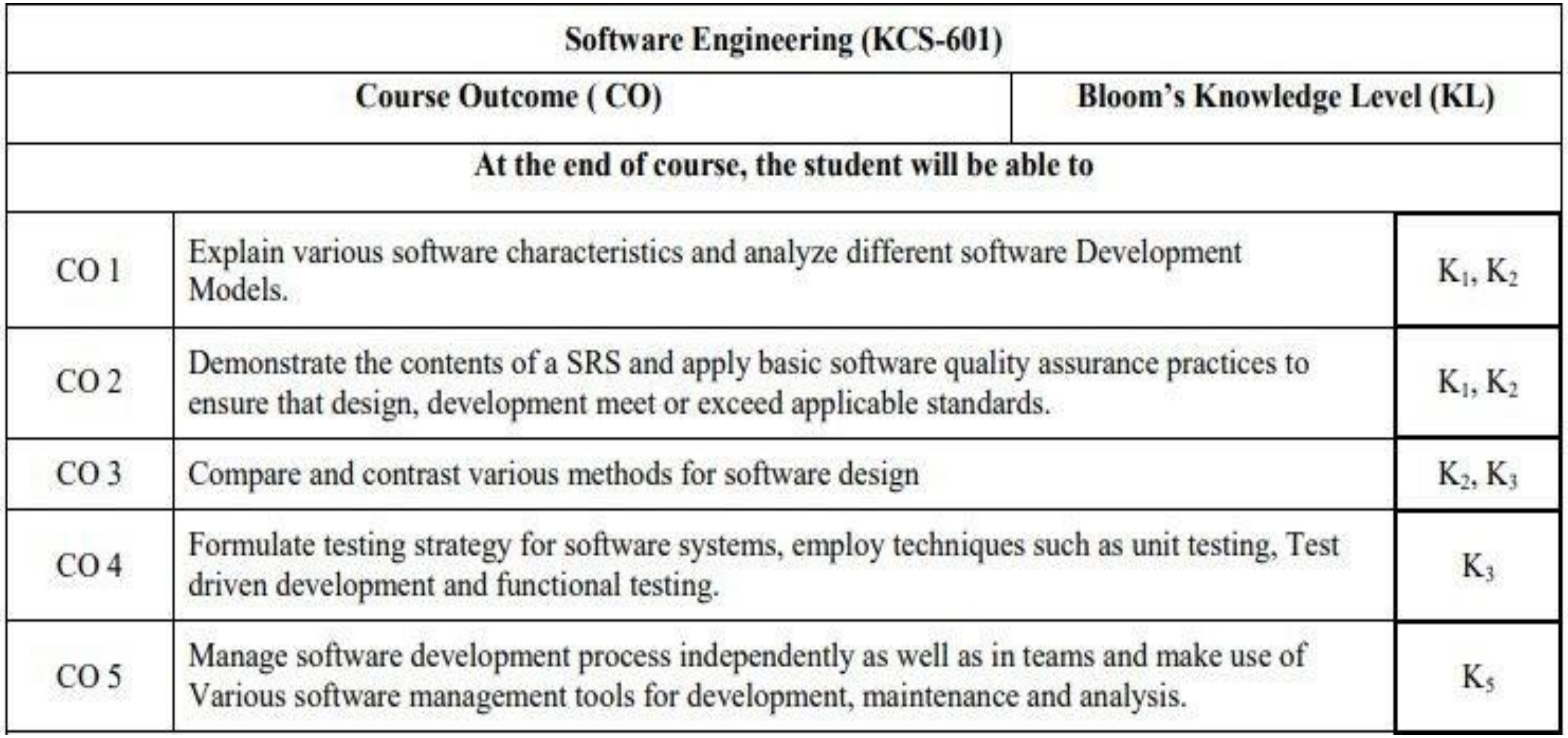
Vision

To build strong teaching environment that responds to the needs of industry and challenges of the society

Mission

- **M1 :** Developing strong mathematical & computing skill set among the students.
- **M2 :** Extending the role of computer science and engineering in diverse areas like Internet of Things (IoT), Artificial Intelligence & Machine Learning and Data Analytics.
- **M3 :** Imbibing the students with a deep understanding of professional ethics and high integrity to serve the Nation.
- **M4 :** Providing an environment to the students for their growth both as individuals and as globally competent Computer Science professional with encouragement for innovation & start-up culture.

Course Outcome



Software Engineering (KCS-601)		
Course Outcome (CO)		**Bloom's Knowledge Level (KL)**
At the end of course, the student will be able to		
CO 1	Explain various software characteristics and analyze different software Development Models.	K_1, K_2
CO 2	Demonstrate the contents of a SRS and apply basic software quality assurance practices to ensure that design, development meet or exceed applicable standards.	K_1, K_2
CO 3	Compare and contrast various methods for software design	K_2, K_3
CO 4	Formulate testing strategy for software systems, employ techniques such as unit testing, Test driven development and functional testing.	K_3
CO 5	Manage software development process independently as well as in teams and make use of Various software management tools for development, maintenance and analysis.	K_5

Syllabu

DETAILED SYLLABUS		3-1-0
Unit	**Topic**	**Proposed Lecture**
I	**Introduction:** Introduction to Software Engineering, Software Components, Software Characteristics, Software Crisis, Software Engineering Processes, Similarity and Differences from Conventional Engineering Processes, Software Quality Attributes. Software Development Life Cycle (SDLC) Models: Water Fall Model, Prototype Model, Spiral Model, Evolutionary Development Models, Iterative Enhancement Models.	**08**
II	**Software Requirement Specifications (SRS):** Requirement Engineering Process: Elicitation, Analysis, Documentation, Review and Management of User Needs, Feasibility Study, Information Modelling, Data Flow Diagrams, Entity Relationship Diagrams, Decision Tables, SRS Document, IEEE Standards for SRS. Software Quality Assurance (SQA): Verification and Validation, SQA Plans, Software Quality Frameworks, ISO 9000 Models, SEI-CMM Model.	**08**
III	**Software Design:** Basic Concept of Software Design, Architectural Design, Low Level Design: Modularization, Design Structure Charts, Pseudo Codes, Flow Charts, Coupling and Cohesion Measures, Design Strategies: Function Oriented Design, Object Oriented Design, Top-Down and Bottom-Up Design. Software Measurement and Metrics: Various Size Oriented Measures: Halestead's Software Science, Function Point (FP) Based Measures, Cyclomatic Complexity Measures: Control Flow Graphs.	**08**
IV	**Software Testing:** Testing Objectives, Unit Testing, Integration Testing, Acceptance Testing, Regression Testing, Testing for Functionality and Testing for Performance, TopDown and Bottom- Up Testing Strategies: Test Drivers and Test Stubs, Structural Testing (White Box Testing), Functional Testing (Black Box Testing), Test Data Suit Preparation, Alpha and Beta Testing of Products. Static Testing Strategies: Formal Technical Reviews (Peer Reviews), Walk Through, Code Inspection, Compliance with Design and Coding Standards.	**08**
V	**Software Maintenance and Software Project Management:** Software as an Evolutionary Entity, Need for Maintenance, Categories of Maintenance: Preventive, Corrective and Perfective Maintenance, Cost of Maintenance, Software Re- Engineering, Reverse Engineering. Software Configuration Management Activities, Change Control Process, Software Version Control, An Overview of CASE Tools. Estimation of Various Parameters such as Cost, Efforts, Schedule/Duration, Constructive Cost Models (COCOMO), Resource Allocation Models, Software Risk Analysis an	**08**

Text Book

Text books:

- RS **Pressman,** Software Engineering: A Practitioners Approach, McGraw Hill.
- Pankaj Jalote, Software Engineering, Wiley
- **Rajib Mall**, Fundamentals of Software Engineering, PHI Publication.
- **KK Aggarwal** and Yogesh Singh, Software Engineering, New Age International Publishers.
- Ghezzi, M. Jarayeri, D. Manodrioli, Fundamentals of Software Engineering, PHI Publication.
- Ian Sommerville, Software Engineering, Addison Wesley.
- Kassem Saleh, “Software Engineering”, Cengage Learning.
- P fleeger, Software Engineering, Macmillan Publication

LECTURE -1

Topic: Introduction to software Engineering

DATE 19/02/24

Presented By: Mr. Ashish Jain

Covered the following sub topics

- ◻ Process and Project
- ◻ Program and Software
- ◻ Chronological evolution of software
- ◻ Software Components
- ◻ Software Characteristics
- ◻ Difference in Program and Software

GL BAJAJ Institute of Technology & Management FIND YOUR SPARK

Process and Project

- A **process** is a sequence of steps performed for a given purpose to develop software to satisfy the needs of some users or clients.
- A **project** is one instance of the problem, and the development process is what is used to achieve this purpose.

Program and Software

- □ **Program** is a set of sequential logical instructions for a computer to perform .
- □ **Software** is a set of items or objects that form a “configuration” that includes
  - • Programs • Documents • Data

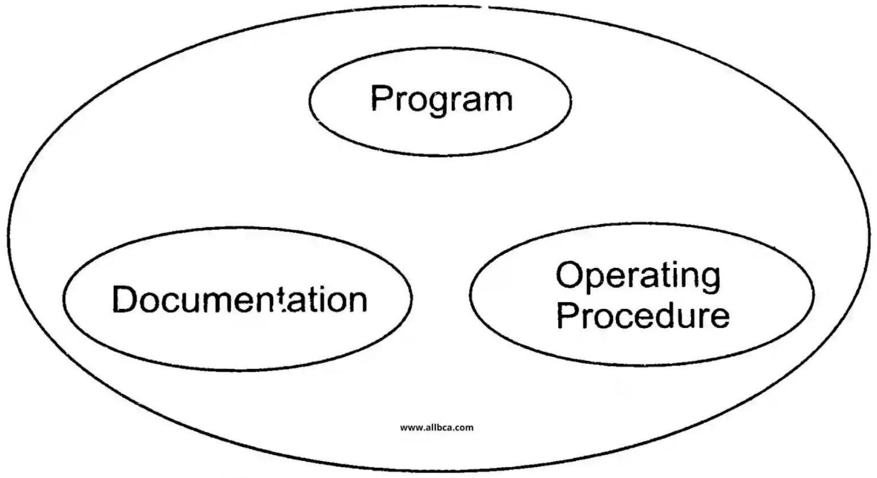
Software consists of (1) Instructions (computer programs) that when executed provided desired function and performance, (2) Data structures that enable the programs to adequately manipulate information, and (3) Documents that describe the operation and use of the programs.

Program and Software

Software is more than one programs

Software = Programs + Documentation + Operating Procedures.

- Any program is a subset of software. It becomes software only if documentation and operating procedure manuals are prepared.



by Mayank Rana

Fig. Software components.

Program and Software

Software products may be ·

Generic - developed to be sold to a range of different customers in general market or open market e.g. PC software such as Excel or Word. ·

Bespoke (custom) - developed for a single or Particular customer according to their specification.

Program and Software

EXAMPLES :

-<u>System software</u> - compilers, editors.
-<u>Real time software-</u> software that monitor/control real world event - air traffic control, navigation

-<u>Embedded software</u> - keypad of microwave oven, mp3 players, cell phones
-<u>Personal computer software</u> – operating system and other application software

-<u>Web based software</u> – Google calendar, web-based systems for different organizations

LECTURE -2

Topic: Introduction to software Engineering

DATE /02/24

Presented By: Mr. Ashish Jain

Chronological evolution of software

- **Chronological evolution of software**

1. 1950-1965 –Early Years

   General Purpose Hardware, Batch Orientation, Limited Distribution, Custom

Software

2. 1968-1973- Second Era

   Multi User, Real Time, Data Base, Product Software, Concept of S/W Maintenance

Chronological evolution of software

- **Chronological evolution of software**

3. 1974-1986-Third Era
Distributed Systems, Embedded Systems, Low cost HW, Consumer Impact

4. 1987-2000-Fourth Era

Powerful Desktop, Object Orient Technology, Expert Systems ,Artificial Neural

Network ,Parallel Computing ,Network Computers

GL BAJAJ Institute of Technology & Management FIND YOUR SPARK

Software Component

- **Software Components**

Lowest Level- It mirrors the instruction set of the h/w. makes efficient use of

memory and optimize programmed execution speed.

Mid Level
Machine language- ·

High Level

Assembly language-·

Machine independents used to create procedural description of the program.

<u>More efficient then machine language C, C++ ·</u>

LECTURE -3

Topic: Introduction to software Engineering

DATE 27/02/23

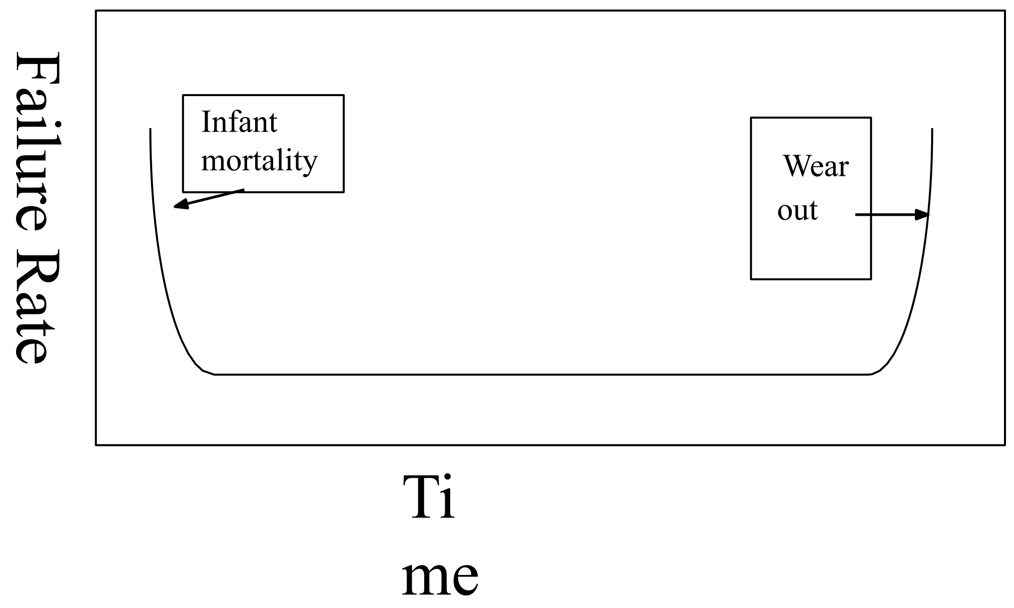
Presented By: Mr. Ashish Jain

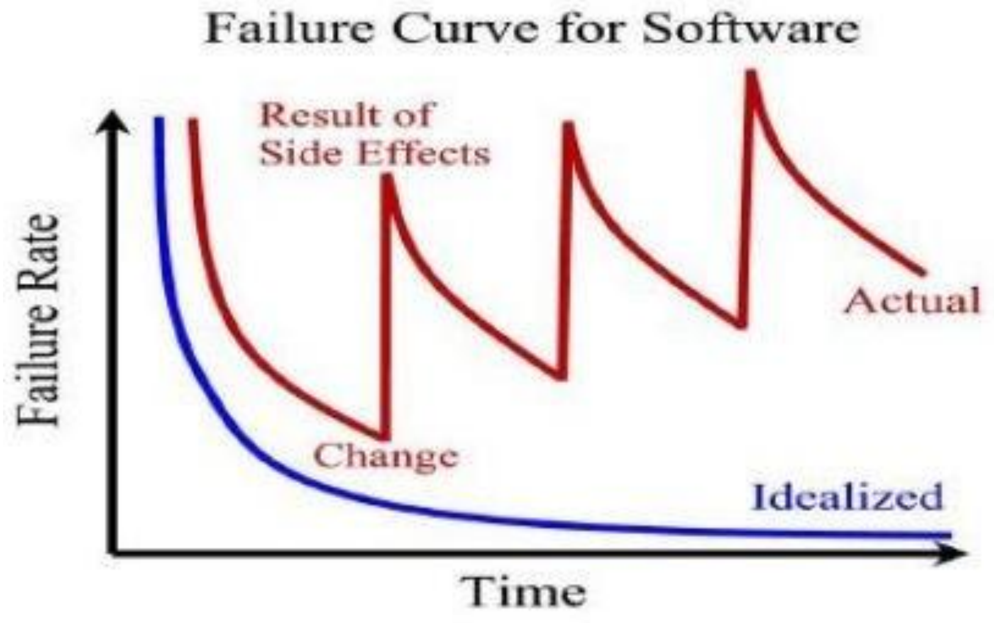
Software Characteristics

- **Software Characteristics**

1) Software does not wear out.

Failure ("Bathtub") Curve for Hardware





Software Characteristics

- ## Software Characteristics

2)Software is not manufactured, but it is developed through the life cycle concept.

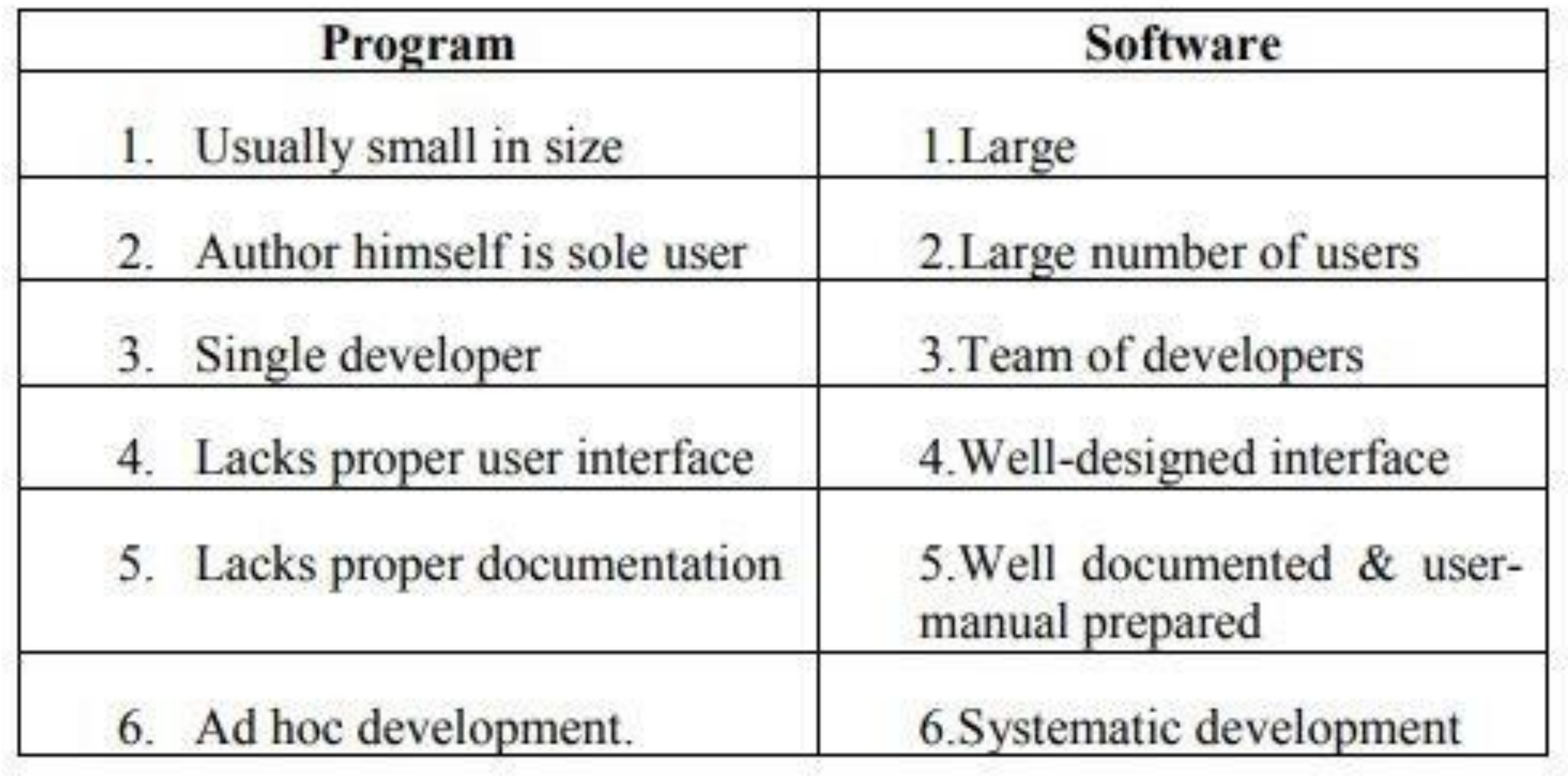
3) Reusability of components

4) Software is flexible and can be amended to meet new requirements

5) Cost of its change at if not carried out initially is very high at later stage .

Difference in Program and Software

- **Difference in Program and Software**



Program	Software
1. Usually small in size	1.Large
2. Author himself is sole user	2.Large number of users
3. Single developer	3.Team of developers
4. Lacks proper user interface	4.Well-designed interface
5. Lacks proper documentation	5.Well documented & user-manual prepared
6. Ad hoc development.	6.Systematic development

LECTURE -4

Topic: Introduction to software Engineering

DATE 28/2/23

Presented By: Mr. Ashish Jain

Covered the following sub topics

- Software Crisis
- Software Engineering process
- Definitions of software engineering
- Difference between software engineering and computer science
- Difference between software engineering and traditional or conventional engineering
- Software Myths

Software Crisis

□ **Software Crisis (Why we need Software Engineering?)**

The idea of software engineering was first proposed in 1968 at a conference held to discuss **‘software crisis**’

Software fails because it :

· crash frequently

·expensive

·difficult to alter, debug, enhance

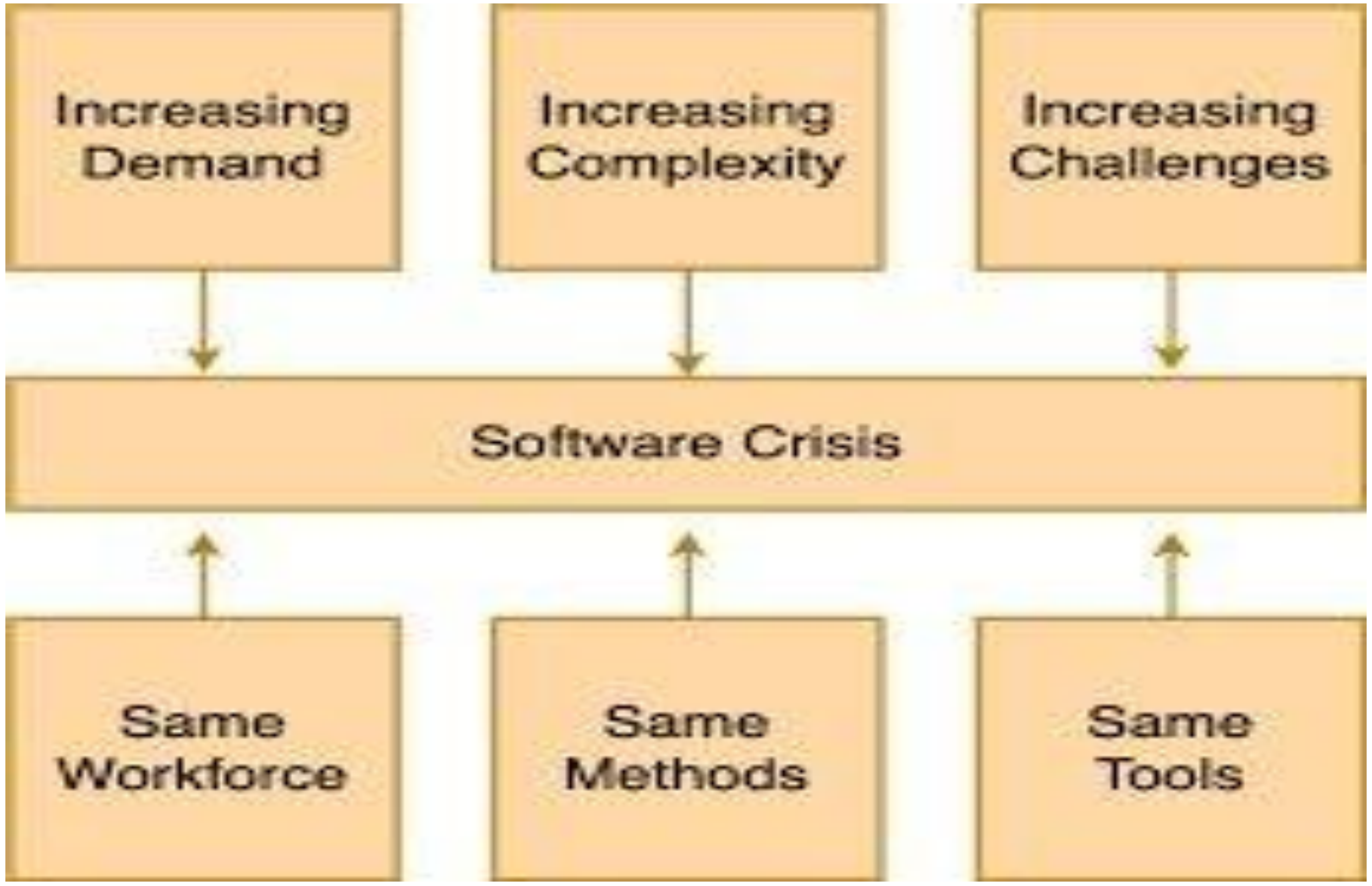
·often delivered late

·use resources non-optimally

·Software professionals lack engineering training

Software Crisis

□ **Software Crisis (Why we need Software Engineering?)**



LECTURE -5

Topic: Introduction to software Engineering

DATE 1/3/23

Presented By: Mr. Ashish Jain

Software Engineering Process

- **Software Engineering Process**

  Concept of the Software Engineering has evolved in 1986 by BOHEM to reduce the effect of his software crisis because **software engineering defined** as "scientific and systematic approach to develop , operate and maintain software project to meet a given specific object." It beautifully **conceive** the concept of life cycle of any software project through following five phase:- ·

  Requirement analysis · Design · Coding · Testing · Maintenance

Software Engineering

- ☐ **Definitions of Software Engineering**

1. Software engineering is an engineering discipline which is concerned with all aspects of software production

<u>Software engineers should adopt:</u>

- a systematic and organized approach to their work
- use appropriate tools and techniques depending on the problem to be solved
- Engineers make thing work, they apply theories, methods and tools where these are appropriate.

2. Software engineering is: the systematic use of many disciplines, tools, and resources for the practical application.

Software Engineering

- **Definitions of Software Engineering**

3. Engineering is the art and science of managing engines for practical application.

To *manage these engines*, we have to consider all aspects of the intended applications, such as <u>operational concepts</u>, <u>requirements</u>, <u>design</u>, <u>development</u>, and <u>maintenance</u>.

LECTURE -6

Topic: Introduction to software Engineering

DATE 13/3/23

Presented By: Mr. Ashish Jain

Difference between software engineering and computer science

- **Difference between software 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 foundation or basis for software engineering (unlike e.g. physics and electrical engineering).

Difference between software engineering and traditional or conventional engineering

□ **Difference between software engineering and traditional or conventional engineering**

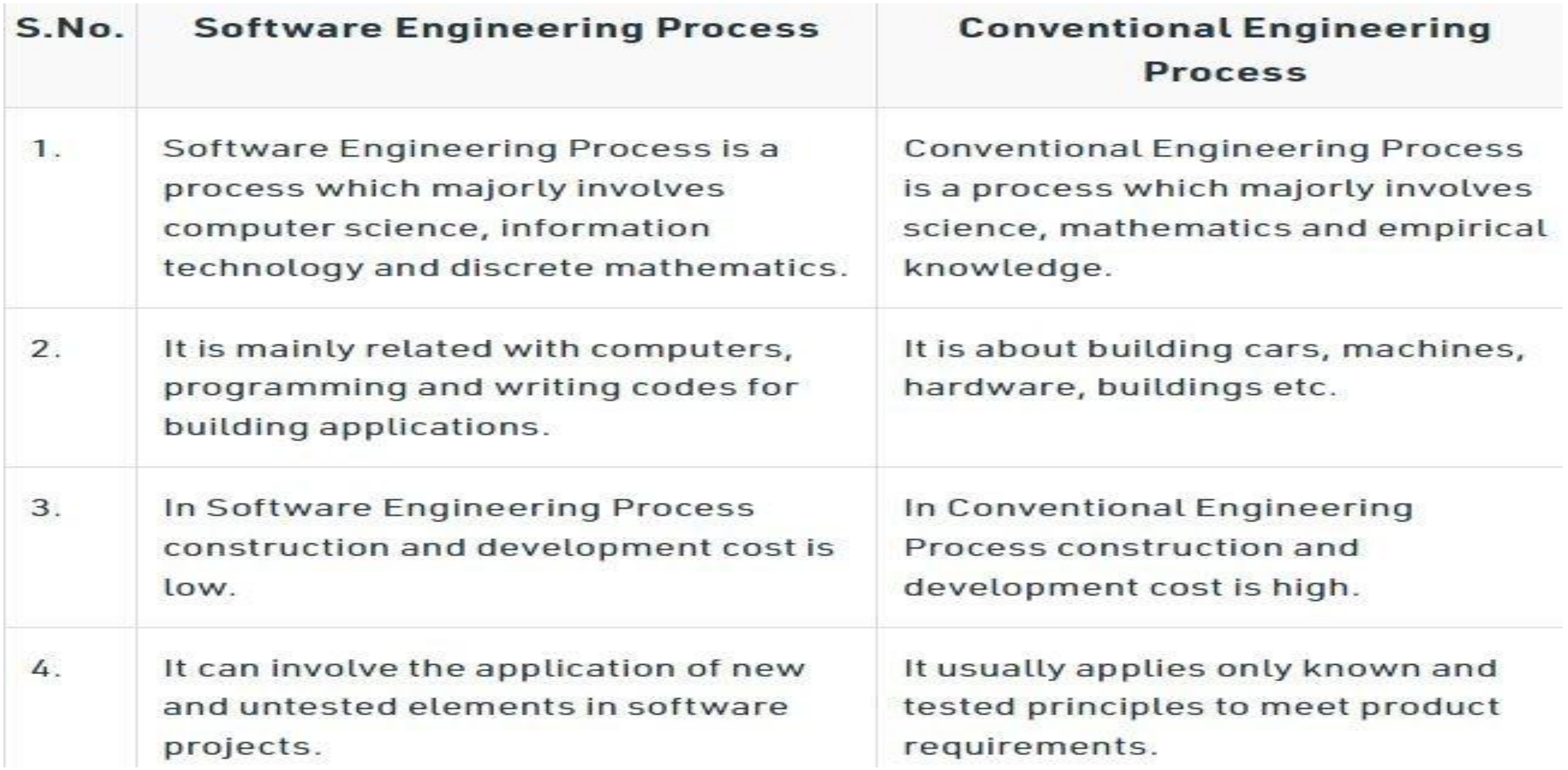
1. **Software Engineering Process :**

   It is a engineering process which is mainly related to computers and programming and developing different kinds of applications through the use of information technology.

2. **Conventional Engineering Process :**

   It is a engineering process which is highly based on empirical(based on experiments and practical experience, not on ideas) knowledge and is about building cars, machines and hardware.

Difference between software engineering and traditional or conventional engineering



S.No.	Software Engineering Process	Conventional Engineering Process
1.	Software Engineering Process is a process which majorly involves computer science, information technology and discrete mathematics.	Conventional Engineering Process is a process which majorly involves science, mathematics and empirical knowledge.
2.	It is mainly related with computers, programming and writing codes for building applications.	It is about building cars, machines, hardware, buildings etc.
3.	In Software Engineering Process construction and development cost is low.	In Conventional Engineering Process construction and development cost is high.
4.	It can involve the application of new and untested elements in software projects.	It usually applies only known and tested principles to meet product requirements.

LECTURE -7

Topic: Introduction to software Engineering

DATE 14/3/23

Presented By: Mr. Ashish Jain

Software Myths

□ **Software Myths**

- Management

1. We have all the standards and procedures available for software development.

Fact:

a) Software experts do not know all the requirements for the software development.

b)And all **existing processes are incomplete** as new software development is based on new and different problem.

2. The addition of the latest hardware programs will improve the software development

Fact:

a)The role of the latest hardware is not very high on standard software development; instead (CASE) Engineering tools help the computer, they are more important than hardware to produce quality and productivity.

b) Hence, the hardware resources are misused.

Software Myths

- ☐ **Software Myths**

3. With the addition of more people and program planners to Software development can help meet **project deadlines** (If lagging behind).

Fact:

- If software is late, adding more people will merely make the problem worse. This is because the people already working on the project now need to spend time educating the newcomers, and are thus taken away from their work. The newcomers are also far less productive than the existing software engineers, and so the work put into training them to work on the software does not immediately meet with an appropriate reduction in work.

- Customer

1. A general statement of intent is enough to start writing plans (software development) and details of objectives can be done over time.

Fact:

Software Myths

- ☐ **Software Myths**

Fact:

a) Official and detailed description of the database function, ethical performance, communication, structural issues and the verification process are important.

b) Unambiguous requirements (usually **derived iteratively**) are developed only through effective and continuous communication between customer and developer.

2. Software requirements continually change, but change can be easily accommodated because software is flexible

Fact:

- It is true that software requirements change, but the impact of change varies with the time at which it is introduced. When requirements changes are requested early (before design or code has been started), the cost impact is relatively small. However, as time passes, the cost impact grows rapidly—resources have been committed, a design framework has been established, and change can cause disruption that requires additional resources and major design modification.

Software Myths

- **Software Myths**

• Professionals

1. believe that their work has been completed with the writing of the plan.

Fact:

- It is true that every 60-80% effort goes into the maintenance phase (as of the latter software release). Efforts are required, where the product is available first delivered to customers.

2. There is no other way to achieve system quality, until it is "running".

Fact:

- Systematic review of project technology is the quality of effective software verification method. These updates are quality filters and more accessible than test

LECTURE -8

Topic: Software Development Life Cycle(SDLC)

DATE 20/3/2023

Presented By: Mr. Ashish Jain

Covered the following sub topics

□ **Software Development Life Cycle (SDLC)Models**

1. Classical Waterfall Model
2. Iterative Waterfall Model
3. Prototyping Model
4. Evolutionary Model
5. Spiral Model
6. Rapid Development Application model (RAD)

Software Development Life Cycle (SDLC)Models

- **Software Development Life Cycle (SDLC)**

- Software-development life-cycle is used to facilitate the development of a large software product in a systematic, well-defined, and cost-effective way.
- An information system goes through a series of phases from conception to implementation. This process is called the Software-Development Life-Cycle.
- Various reasons for using a life-cycle model include:
  - Helps to understand the entire process
  - Enforces a structured approach to development
  - Enables planning of resources in advance
  - Enables subsequent controls of them
  - Aids management to track progress of the system

Software Development Life Cycle (SDLC)Models

- **Software Development Life Cycle (SDLC) Phases:**

Phase 1: **<u>Activities undertaken during feasibility study</u>**: - The main aim of feasibility study is to determine whether it would be financially and technically feasible to develop the product.

Phase **<u>2:Activities undertaken during requirements analysis and specification</u>**: - The aim of the requirements analysis and specification phase is to understand the exact requirements of the customer and to document them properly. This phase consists of two distinct activities, namely Requirements gathering and analysis, this phase **ends** with the preparation of **Software requirement Specification (SRS)**

Phase 3. Activities undertaken during design: - The goal of the design phase is to transform the requirements specified in the SRS document into a structure that is suitable for implementation in some programming language . **Design specification Document is outcome of this phase**.

Phase 4: **<u>Activities undertaken during coding and unit testing</u>**:-The purpose of the coding and unit testing phase (sometimes called the implementation phase) of software development is to translate the software design into source code.

GL BAJAJ Institute of Technology & Management FIND YOUR SPARK

Software Development Life Cycle (SDLC)Models

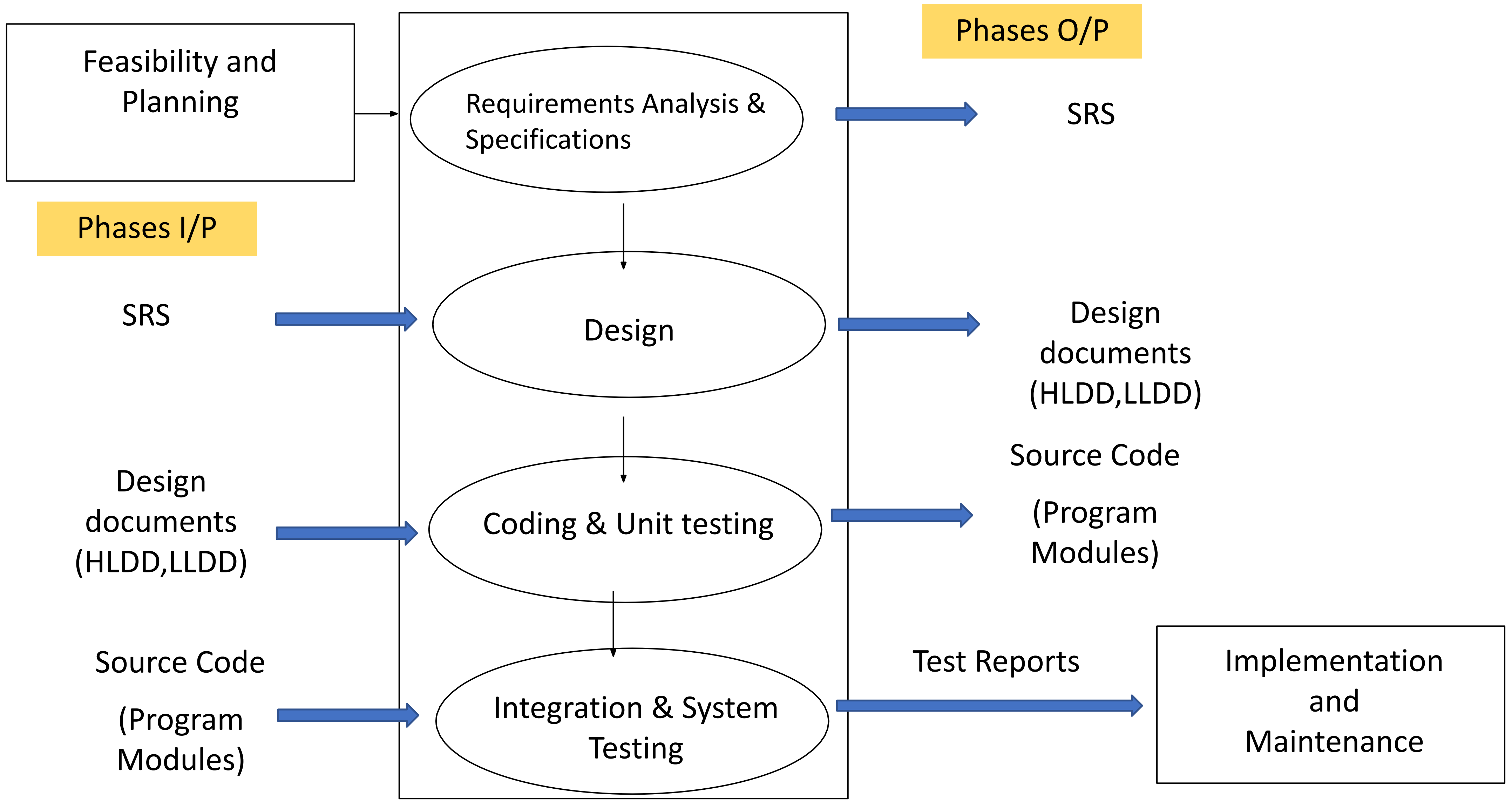
- **Software Development Life Cycle (SDLC) Phases:**

Each component of the design is implemented as a program module. **The end-product of this phase is a set of program modules that have been individually tested**. **Code Listings are generated after this phase**.

Phase 5 : Activities undertaken during integration and system testing: - Integration of different modules is undertaken once they have been coded and unit tested During each integration step, the partially integrated system is tested and a set of previously planned modules are added to it. Finally, when all the modules have been successfully integrated and tested, system testing is carried out. The goal of system testing is to ensure that the developed system conforms to its requirements laid out in the SRS document. **Test Reports are generated after this phase**.

Phase 6: <u>Activities undertaken during maintenance</u>: - Maintenance of a typical software product requires much more than the effort necessary to develop the product itself. Many studies carried out in the past confirm this and indicate that the relative effort of development of a typical software product to its maintenance effort is roughly in the 40:60 ratio. This phase continues till the software is in use.

Software Development Life Cycle (SDLC)Models


Feasibility and Planning
Phases I/P
Phases O/P
Requirements Analysis & Specifications
SRS
SRS
Design
Design documents (HLDD,LLDD)
Design documents (HLDD,LLDD)
Coding & Unit testing
Source Code (Program Modules)
Source Code (Program Modules)
Integration & System Testing
Test Reports
Implementation and Maintenance

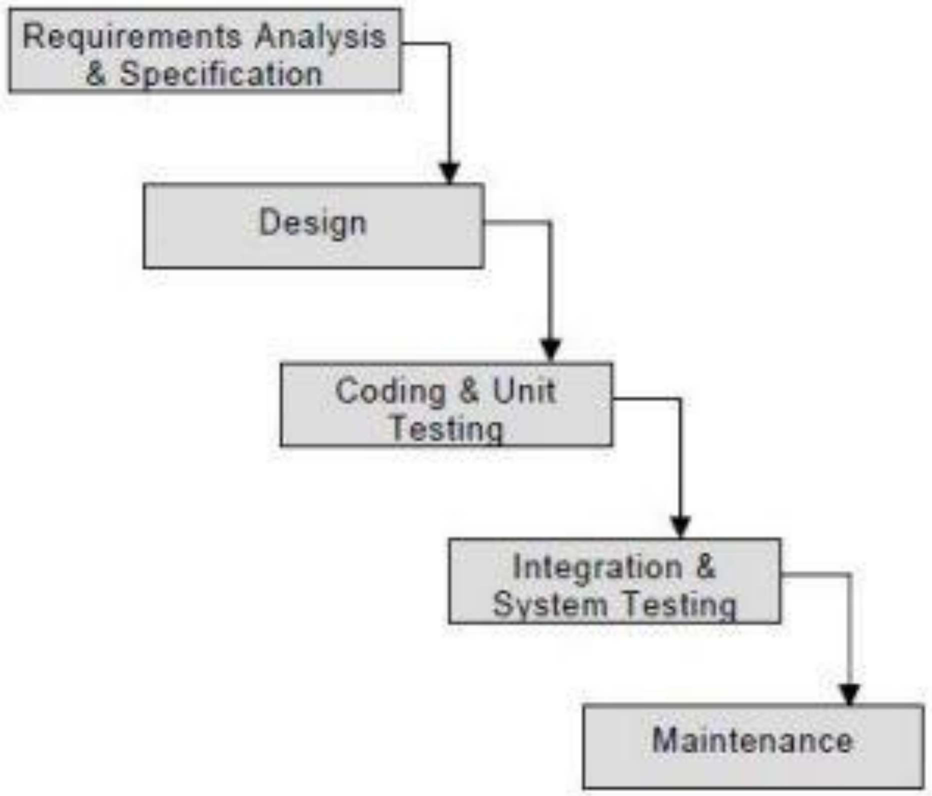
1. Classical Waterfall Model

1. Classical Waterfall Model

- Non – Iterative approach . All the requirements should be well collected before starting the SDLC during SDLC process any new requirement can not considered.
- It is not a practical model in the sense that it cannot be used in actual software development projects.
- Classical waterfall model divides the life cycle into the following phases:

1.Feasibility Study 2. Requirements Analysis and Specification 3. Design 4. Coding and Unit Testing 5. Integration and System Testing 6. Maintenance

1. Classical Waterfall Model


Requirements Analysis & Specification
Design
Coding & Unit Testing
Integration & System Testing
Maintenance

Classical Waterfall Model

LECTURE -9

Topic: Software Development Life Cycle(SDLC)

DATE 21/3/2023

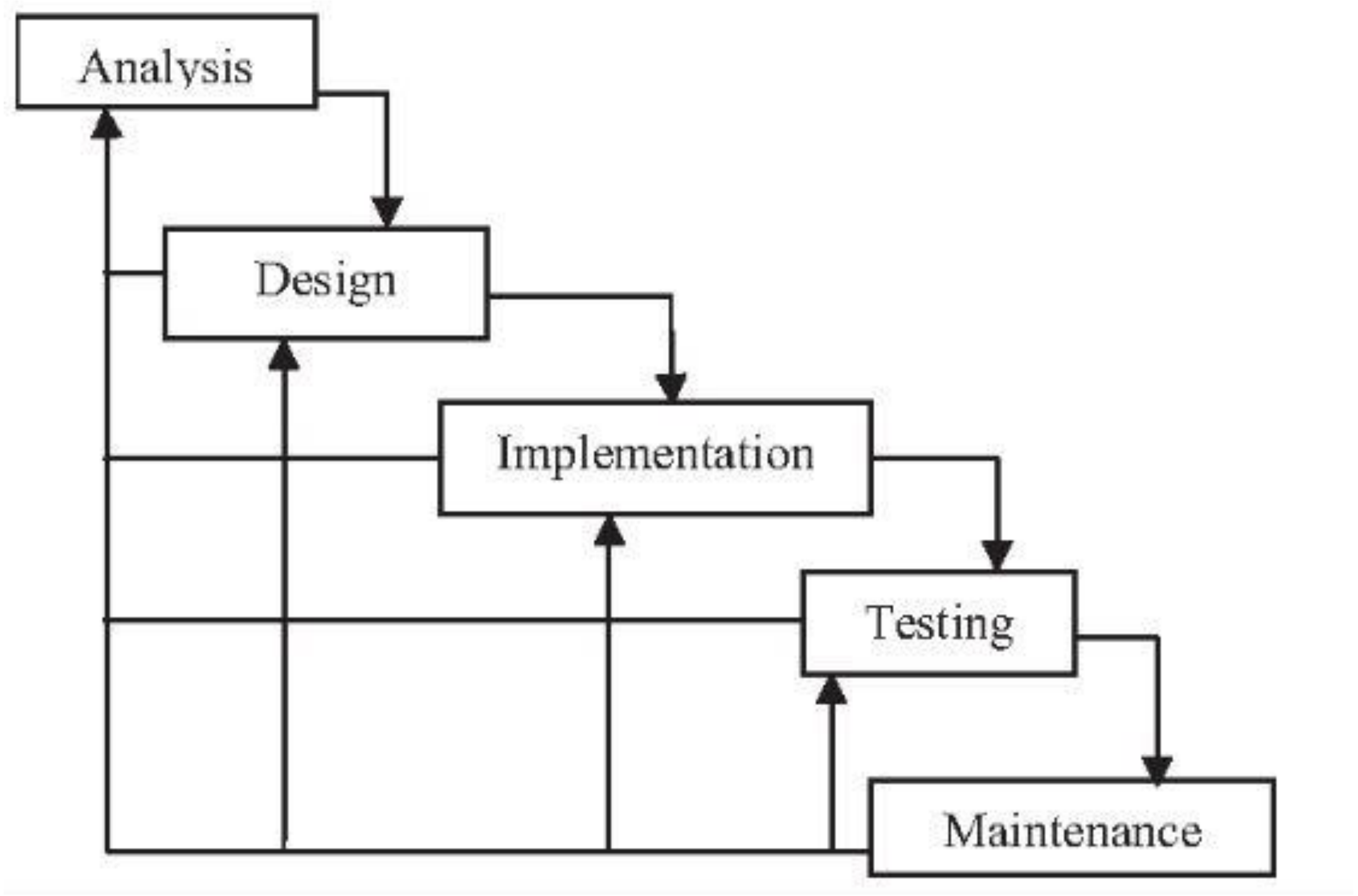
Presented By: Mr. Ashish Jain

2. Iterative Waterfall Model

2. Iterative Waterfall Model

- Waterfall model assumes in its design that no error will occur during the design phase
- Iterative waterfall model introduces feedback paths to the previous phases for each process phase
- It is still preferred to detect the errors in the same phase they occur • Conduct reviews after each milestone

2. Iterative Waterfall Model



Waterfall Model

Advantages of Waterfall Model

- It is a linear model.
- It is a segmental model.
- It is systematic and sequential.
- It is a simple one and best for small projects.
- It has proper documentation

Disadvantages of Waterfall Model

- It is difficult to define all requirements at the beginning of the project.
- Model is not suitable for accommodating any change.
- It does not scale up well to large projects
- This model is only appropriate when the requirements are well understood and changes will be fairly limited during the design process.

LECTURE -10

Topic: Software Development Life Cycle(SDLC)

DATE 22/3/2023

Presented By: Mr. Ashish Jain

3. Prototype Model

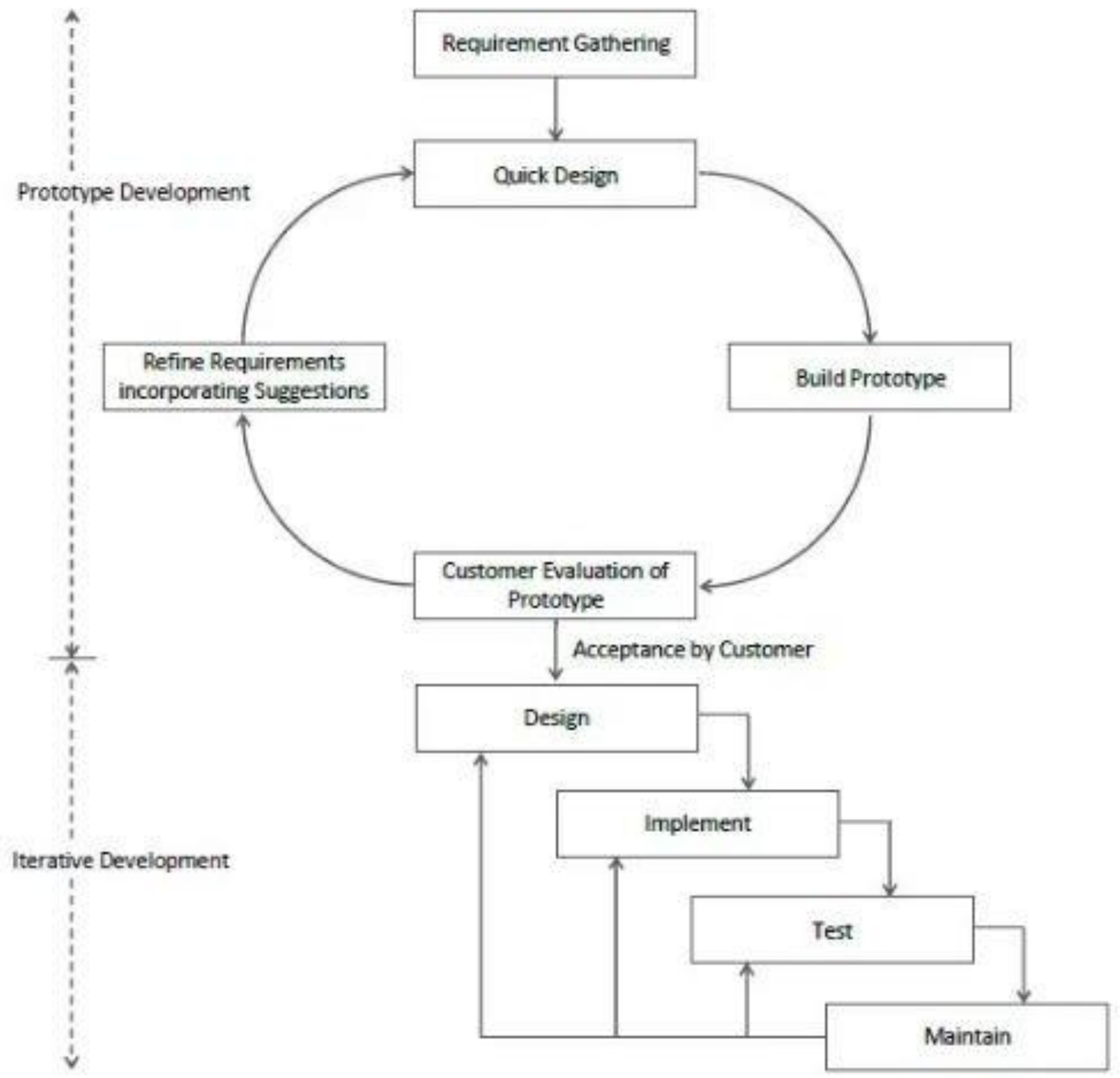
3. Prototype Model

- The Prototyping Model is a systems development method (SDM) in which a prototype is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed prototyping paradigm begins with requirements gathering.
- Developer and customer meet and define the overall objectives for the software, identify whatever requirements are known, and outline areas where further definition is mandatory.

3. Prototype Model

- A "quick design" then occurs. The quick design focuses on a representation of those aspects of the software that will be visible to the customer/user (e.g., input approaches and output formats).
- There are several steps in the Prototyping Model as shown in the diagram:-

3. Prototype Model



3. Prototype Model

Types of Prototyping Models

1.Rapid Throwaway prototypes

2.Evolutionary prototype

3. Prototype Model

Advantage of Prototype model

- Suitable for large systems for which there is no manual process to define there requirements.
- User training to use the system.
- User services determination.
- System training.
- Quality of software is good.
- Requirements are not freezed.

Disadvantage of Prototype model

- It is difficult to find all the requirements of the software initially.
- It is very difficult to predict how the system will work after development

LECTURE -11

Topic: Software Development Life Cycle(SDLC)

DATE 23/3/2023

Presented By: Mr. Ashish Jain

4. Evolutionary Process Model

4. Evolutionary Process Model

- In EP model development engineering effort is made first to establish correct, precise requirement definitions and system scope, as agreed by all the users across the organization.
- This is achieved through application of **iterative processes** to evolve a system most suited to the given circumstances.
- The process is iterative as the software engineer goes through a repetitive process of requirement until all users and stakeholders are satisfied.
- This model differs from the iterative enhancement model in the sense that this does not require a useable product at the end of each cycle.
- In evolutionary development, requirements are **implemented by category as per priority.**

4. Evolutionary Process Model

Main characteristics:

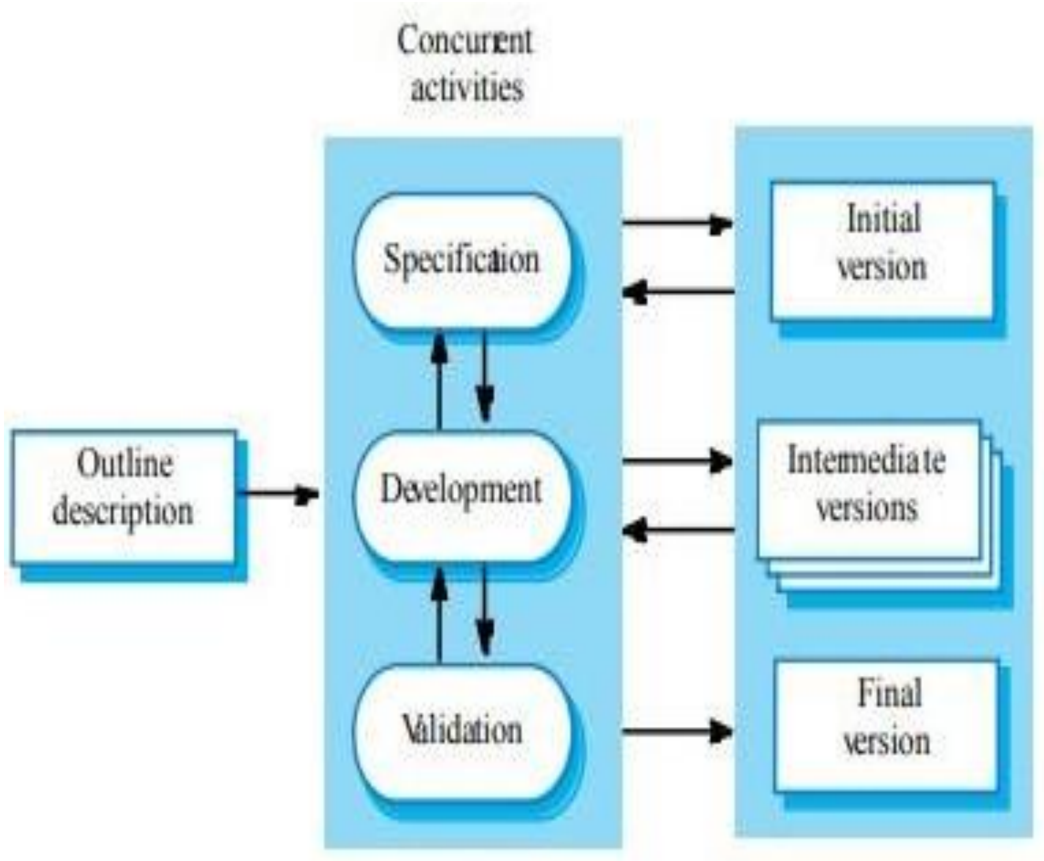
– The phases of the software construction are interleaved

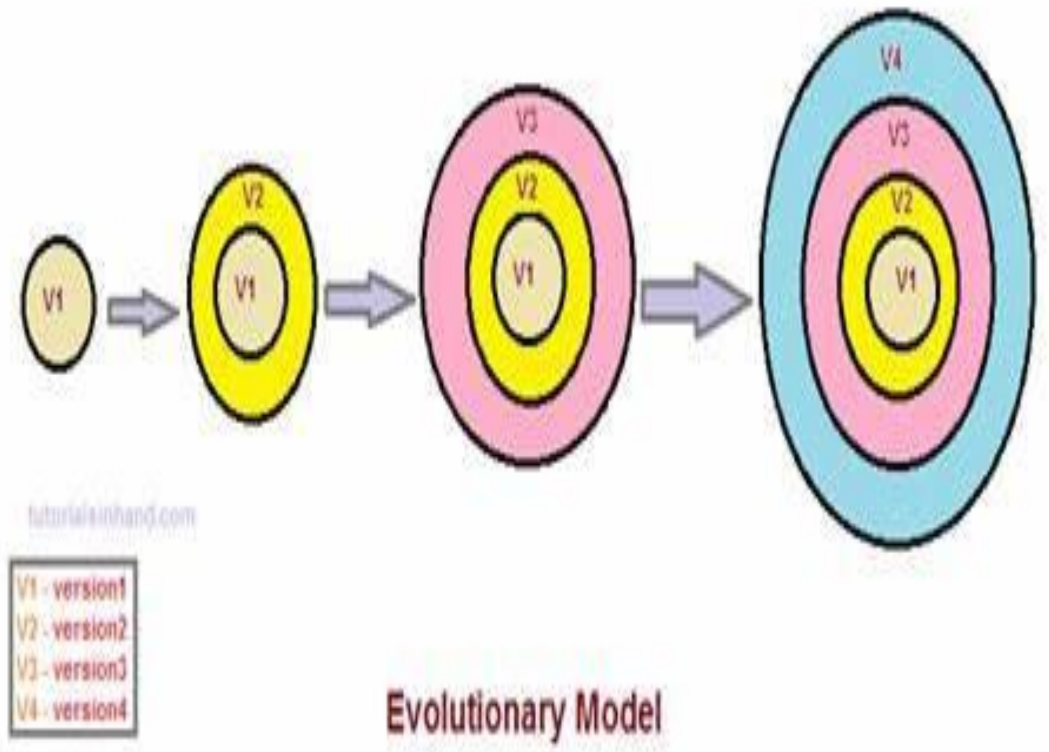
– Feedback from the user is used throughout the entire process

– The software product is refined through many versions

- **Types of evolutionary development**:
  - – Exploratory development
  - – Throw-away prototyping

4. Evolutionary Process Model





4. Evolutionary Process Model

Advantages:

·Deals constantly with changes

·Provides quickly an initial version of the system

·Involves all development teams

Disadvantages:

·Quick fixes may be involved

·"Invisible" process, not well-supported by documentation

·The system's structure can be corrupted by continuous change

LECTURE -12

Topic: Software Development Life Cycle(SDLC)

DATE 27/3/2023

Presented By: Mr. Ashish Jain

5. Spiral Model

5. Spiral Model

- The diagrammatic representation of this model appears like a spiral with many loops.
- The exact number of loops in the spiral is not fixed. Each loop of the spiral represents a phase of the software process. For example, the innermost loop might be concerned with feasibility study. The next loop with requirements specification, the next one with design, and so on.

Each phase in this model is split into four sectors (or quadrants):

5. Spiral Model

The following activities are carried out during each phase of a spiral model.

First quadrant (Objective Setting) · During the first quadrant, it is n eeded to identify the objectives of the phase. · Examine the risks associated with these objectives.

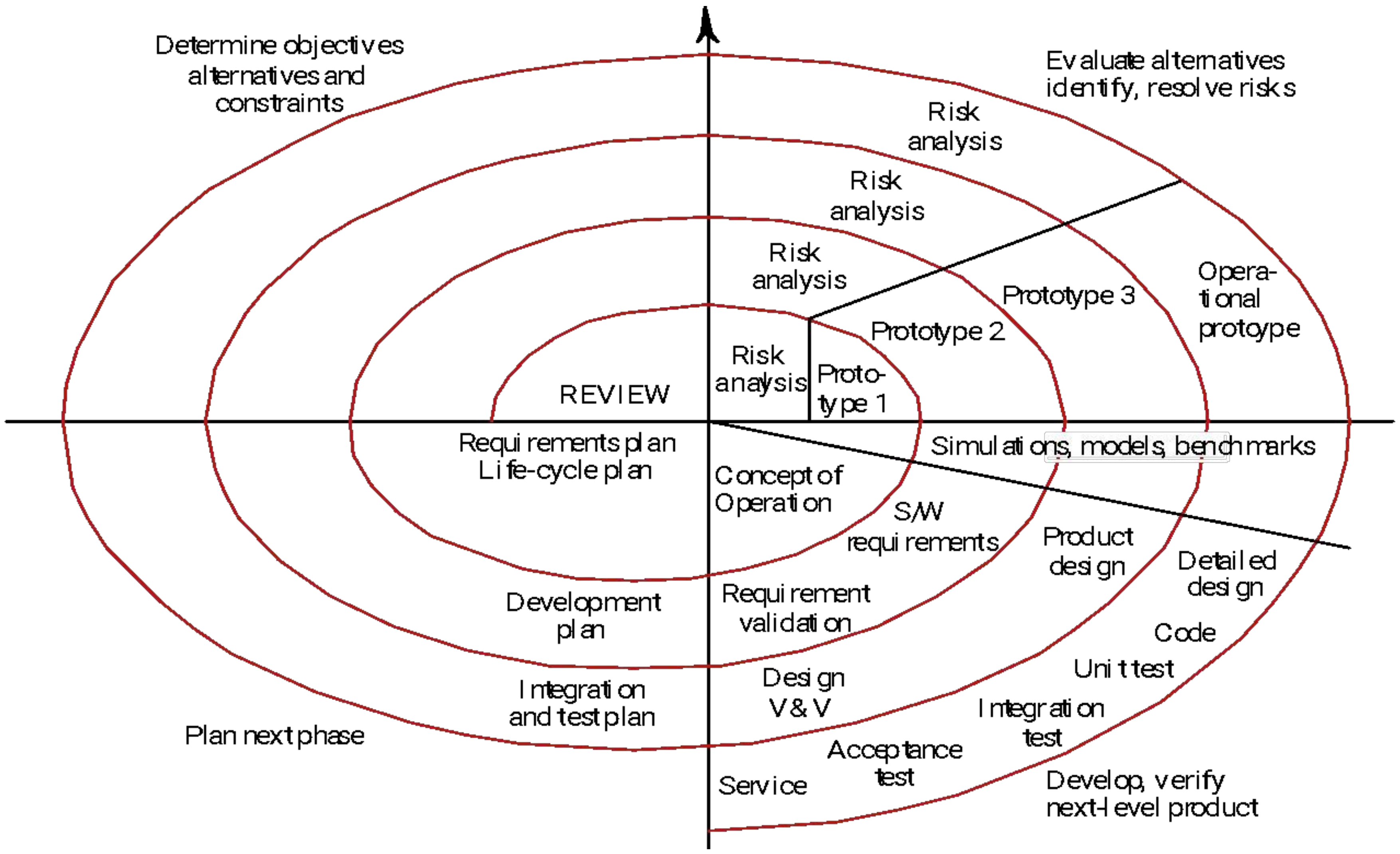
Second Quadrant (Risk Assessment and Reduction) • A detailed analysis is carried out for each identified project risk. • Steps are taken to reduce the risks. For example, if there is a risk that the requirements are inappropriate, a prototype system may be developed.

Third Quadrant (Development and Validation) • Develop and validate the next level of the product after resolving the identified risks.

Fourth Quadrant (Review and Planning) • Review the results achieved so far with the customer and plan the next iteration around the spiral. 15 • Progressively more complete version of the software gets built with each iteration around the spiral

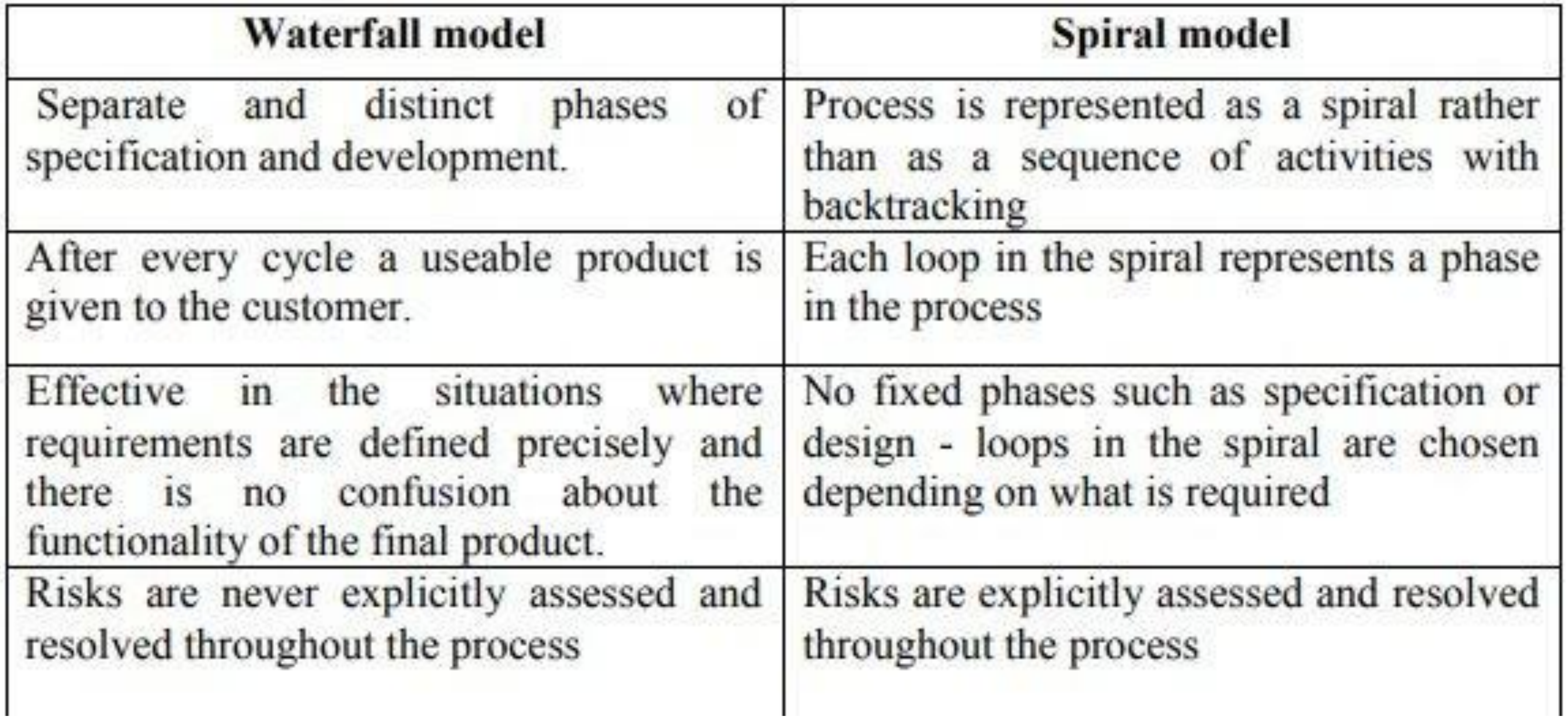
GL BAJAJ Institute of Technology & Management FIND YOUR SPARK

5. Spiral Model



Difference between spiral and waterfall model

Difference between spiral and waterfall model



Waterfall model	Spiral model
Separate and distinct phases of specification and development.	Process is represented as a spiral rather than as a sequence of activities with backtracking
After every cycle a useable product is given to the customer.	Each loop in the spiral represents a phase in the process
Effective in the situations where requirements are defined precisely and there is no confusion about the functionality of the final product.	No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required
Risks are never explicitly assessed and resolved throughout the process	Risks are explicitly assessed and resolved throughout the process

LECTURE -13

Topic: Software Development Life Cycle(SDLC)

DATE 28/3/2023

Presented By: Mr. Ashish Jain

6. Rapid Application Development model (RAD)

6. Rapid Application Development model (RAD)

- RAD is proposed when requirements and solutions can be modularized as independent system or software components, each of which can be developed by different teams.
- User involvement is essential from requirement phase to deliver of the product. The process is stared with rapid prototype and is given to user for evolution.
- In that model user feedback is obtained and prototype is refined. SRS and design document are prepared with the association of users.
- RAD becomes faster if the software engineer uses the component's technology (CASE Tools) such that the components are really available for reuse. Since the development is distributed into component-development teams, the teams work in tandem and total development is completed in a short period (i.e., 60 to 90 days).

6. Rapid Application Development model (RAD)

6. Rapid Application Development model (RAD) : Involves

- Quick Prototyping
- Iterative Development
- Incremental Releases
- User Involvement
- **Time-Boxing**
- Parallel Development

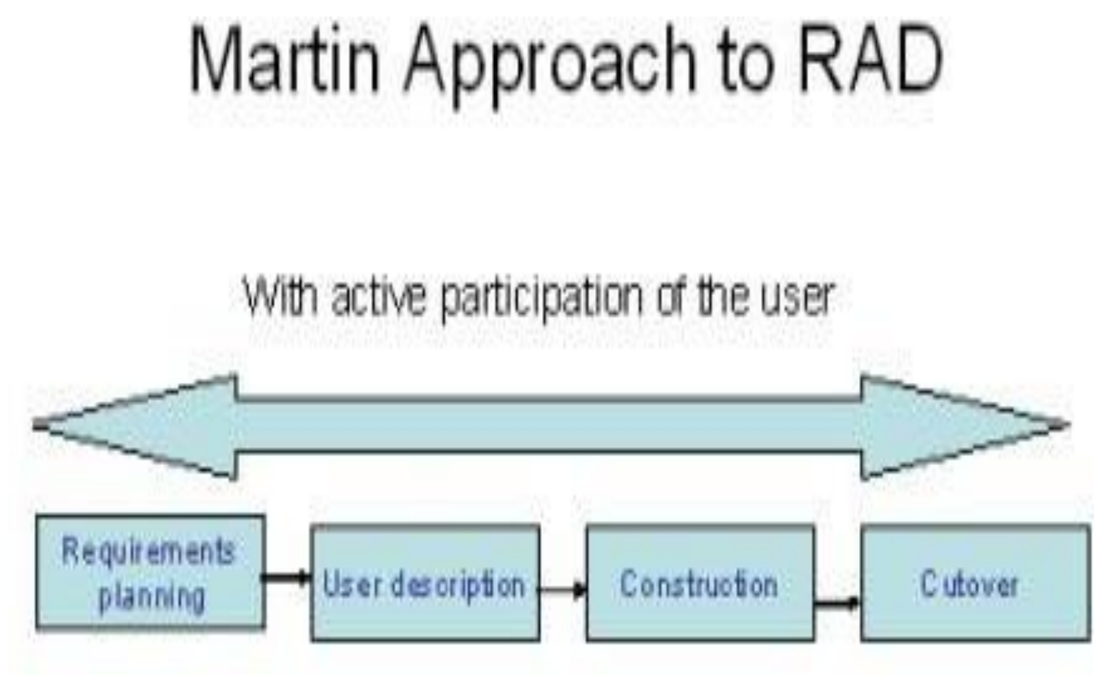
6. Rapid Application Development model (RAD)

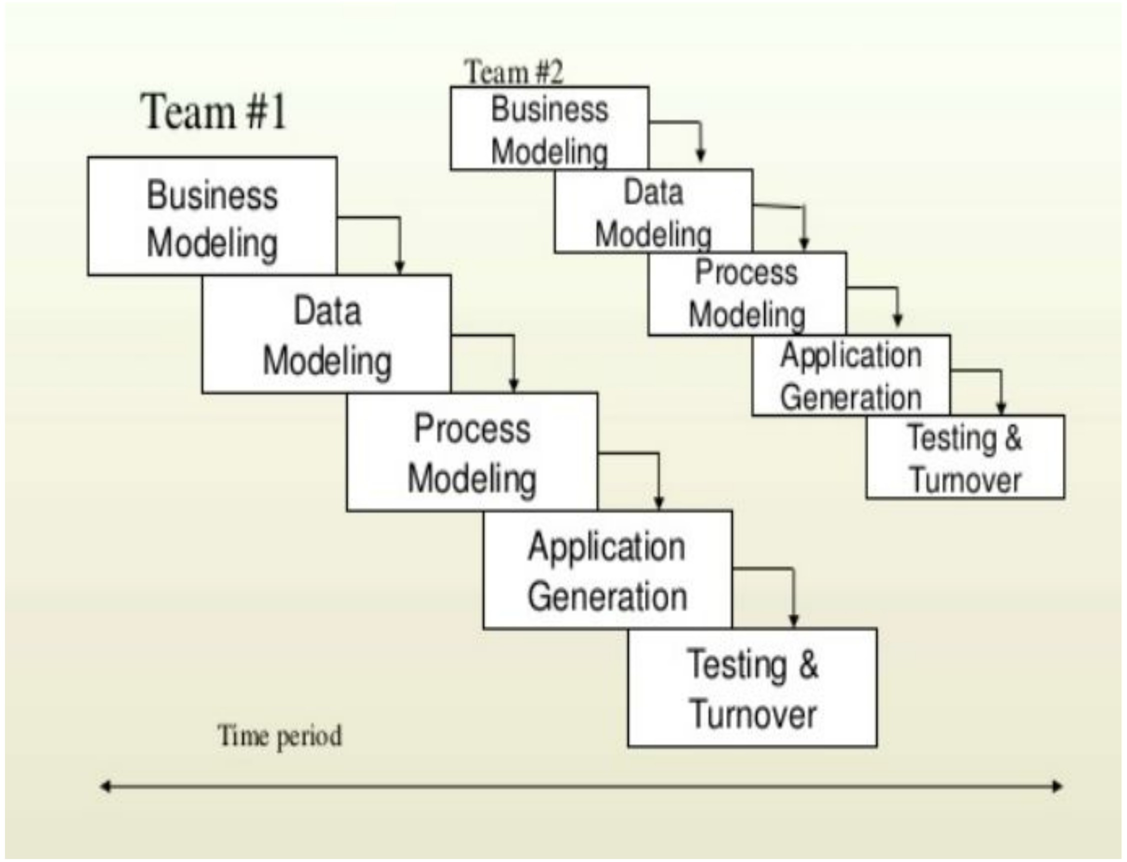
RAD Phases :

- Requirements planning phase (a workshop utilizing structured discussion of business problems)
- User description phase – automated tools capture information from users
- Construction phase – productivity tools, such as code generators, screen generators, etc. inside a time-box. (“Do until done”)
- Cutover phase -- installation of the system, user acceptance testing and user training

6. Rapid Development Application model (RAD)

•





6. Rapid Development Application model (RAD)

Advantage of RAD

- Dramatic time savings the systems development effort
- Can save time, money and human effort
- Tighter fit between user requirements and system specifications
- Works especially well where speed of development is important

Disadvantage of RAD

- More speed and lower cost may lead to lower overall system quality
- Danger of misalignment of system developed via RAD with the business due to missing information
- May have inconsistent internal designs within and across systems
- Possible violation of programming standards related to inconsistent naming conventions and inconsistent documentation