Software Engineering (SE)

Prepared by
Prof.Jigar dave
SoE
R.K.UNIVERSITY
8469766496

CHAPTER -1

INTRODUCTION TO SE

SOFTWARE

INTRODUCTION:

Software has become the key element in the **evolution** of computer-based systems and products.

Software is composed of **programs**, **data**, and **documents**. Each of these items comprises a configuration that is created as part of the software engineering process.

The intent of software engineering is to provide a **framework** for building software with higher quality.

Software

What is Software ?

Software is: (1) **instructions** (computer programs) that when executed provide **desired features, function, and performance**; (2) **data structures that enable** the programs to effectively **manipulate information**, and (3) **descriptive information in both** hard copy and virtual forms that describes the operation and use of the programs.

IEEE defines software as the collection of computer programs, procedures, rules, and associated documentation and data.

SOFTWARE

Software Characteristics :

1. Software is developed or engineered, it is not manufactured in the classical sense.
2. Software doesn’t “wear out”. (but it does deteriorate. (get worse))
3. Although the industry is moving toward component-based assembly, most software continues to be custom built.

Software

Software Applications:

Software may be applied in any situation for which a prespecified set of procedural steps has been defined.

Following are the areas which uses software applications:

 System Software
 Real time Software
 Business Software
 Engineering /Scientific Software
 Embedded Software
 Personal Computer Software
 Web Based Software
 AI software

SOFTWARE

1. System software

- System software is a collection of programs **written to service other programs**. e.g. compilers, editors etc.

2. Real time software

- Software that monitors/analyzes/controls real-world events as they occur is called real time.

- Elements of real-time software include a data **gathering information from an external environment** structures.

SOFTWARE

3. Business software

- Business information processing is the **largest single software application area**.

- Discrete systems like as payroll, accounts receivable/ payable, inventory

4. Engineering and Scientific software

- Engineering and scientific software have been **characterized by number crunching** algorithms.

- Applications range from **astronomy** to **volcanology**, from automotive stress analysis to space shuttle orbital dynamics, and from molecular biology to automated manufacturing.

SOFTWARE

5. Embedded software

- Intelligent products have become **commonplace in nearly every consumer** and **industrial market.**

- Embedded software resides in **read-only memory** and is used to control products and systems for the consumer and industrial mark etc.

6. Personal Computer Software

- The personal computer software market has thesaurus over the past two decades.

- Word processing, spreadsheets, computer graphics, multimedia, entertainment, database management, personal and business financial applications, external network, and database access are only a few of hundreds of applications.

SOFTWARE

7. Web Based Software

- The Web pages **retrieved by a browser are software** that incorporates executable instructions (e.g. CGI, HTML or Java), and data (e.g. hypertext and a variety of visual and audio formats).

8. Artificial intelligence software

- Artificial intelligence (AI) software **makes use of non-numerical algorithms to solve complex problems** that are not amenable (responsible) to computation or straightforward analysis.

- Expert systems, also called **knowledge- based systems**, pattern recognition (image and voice), artificial neural networks, theorem proving, and game playing are representative of applications within this category.

SOFTWARE

Software Myths:

1. Software is easy to change
2. Computers provide greater reliability than the devices they replace
3. Testing software or “providing” software correct can remove all the errors.

Software Engineering

- "Software Engineering is defined as the **systematic approach to the development**, **operation**, **maintenance**, and **retirement** of software".

- **Systematic approach** means that methodologies are used for developing software which are repeatable. If they are applied by different groups of people, similar software will be produced.

Software Engineering

The IEEE definition:

Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software. (2) The study of approaches as in .

Software Engineering

Software Engineering : A Layered Technology

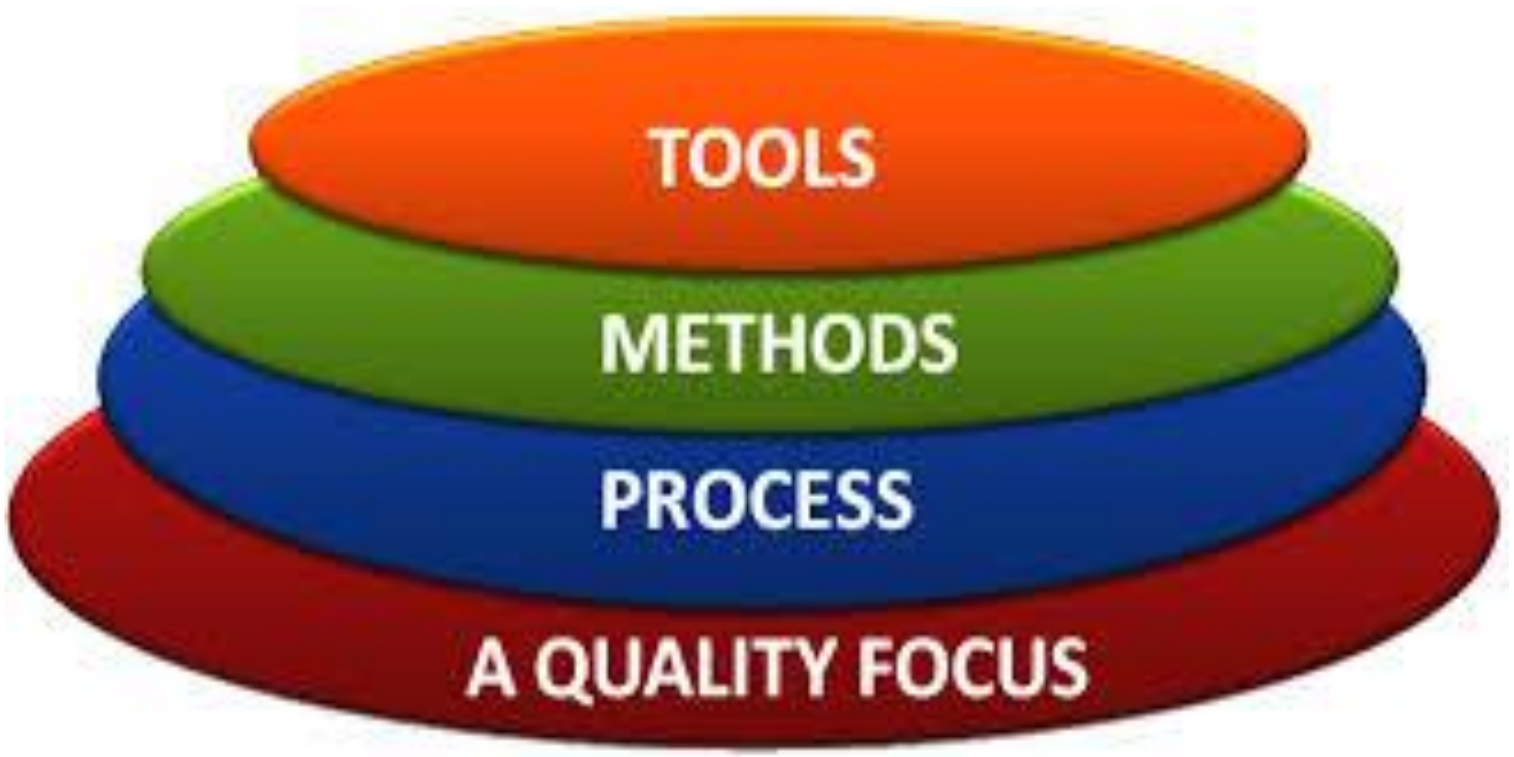
- To solve actual problems in an industry setting, a software engineer or a team of engineers must incorporate a development strategy that encompasses the process, methods, and tools layers.

- This strategy is often referred to as a process model or a software engineering paradigm.

- A process model for software engineering is chosen based on the nature of the project and application, the methods and tools to be used, and the controls and deliverables that are required.

Software Engineering

Software engineering is a layered technology as you can see in figure:



SOFTWARE ENGINEERING

Software Engineering is a layered technology whose bedrock(base) is a focus on **quality**.

The foundation for software engineering is the **process layer.** Process defines a framework for a set of key process areas (KPAs) that must be established for effective delivery of software engineering technology.

Software engineering **methods** provide the technical how-to use for building software including requirements analysis, design, program construction, testing, and support

Software Engineering

Software engineering **tools** provide automated or semi-automated support for the process and the methods. When tools are integrated so that information created by one tool can be used by another.

Software Process

Introduction

A process is a collection of **activities, actions, and tasks** that are performed when some work product is to be created.

- An **activity** strives to achieve a broad objective
- An **action** (e.g., architectural design) encompasses a set of tasks that produce a major work product.
- A **task** focuses on a small, but well-defined objective (e.g., conducting a unit test) that produces a real outcome.

Software Process

- "Software process **as a framework for the tasks** that are required to build high-quality software."

- The phases and related steps **described in our generic view** of software engineering are complemented by a number of **umbrella activities.**

- In addition, the process framework encompasses a set of umbrella activities that are applicable across the entire software process.

SOFTWARE PROCESS

A generic process framework for software engineering encompasses five activities:

1. Communication(project initiation, requirements gathering)
2. Planning (estimating, scheduling, tracking)
3. Modeling (analysis, design)
4. Construction (code, testing)
5. Deployment (delivery, support, feedback)

Software Process

1. **Communication**

   Before any technical work can begin, it is critically important to **communicate** and **collaborate** with customer.

2. **Planning**

   Any complicated journey can be simplified if map exist. Here software project is **complicated journey** and **planning activity** **create map** that helps guide the team as makes the journey.

Software Process

3. **Modeling**

   It does the same things by creating models to **better understand software requirements** and **the design** that will achieve those requirement.

4. **Construction**

   this activity combine **the code generation** and **testing** that is required to uncover errors in code.

5. **Deployment**

   these software is delivered to the customer who **evaluates the delivered product** and **provide feedback based** on the evaluation.

Prescriptive Process Model

- Prescriptive process models define a given set of process elements and a predictable process work flow.

- Called “prescriptive” because **they prescribe a set of process elements**—framework activities, software engineering actions, tasks, work products, quality assurance, and change control mechanisms for each project.

Prescriptive Process Model

1. **Water Fall Model :**

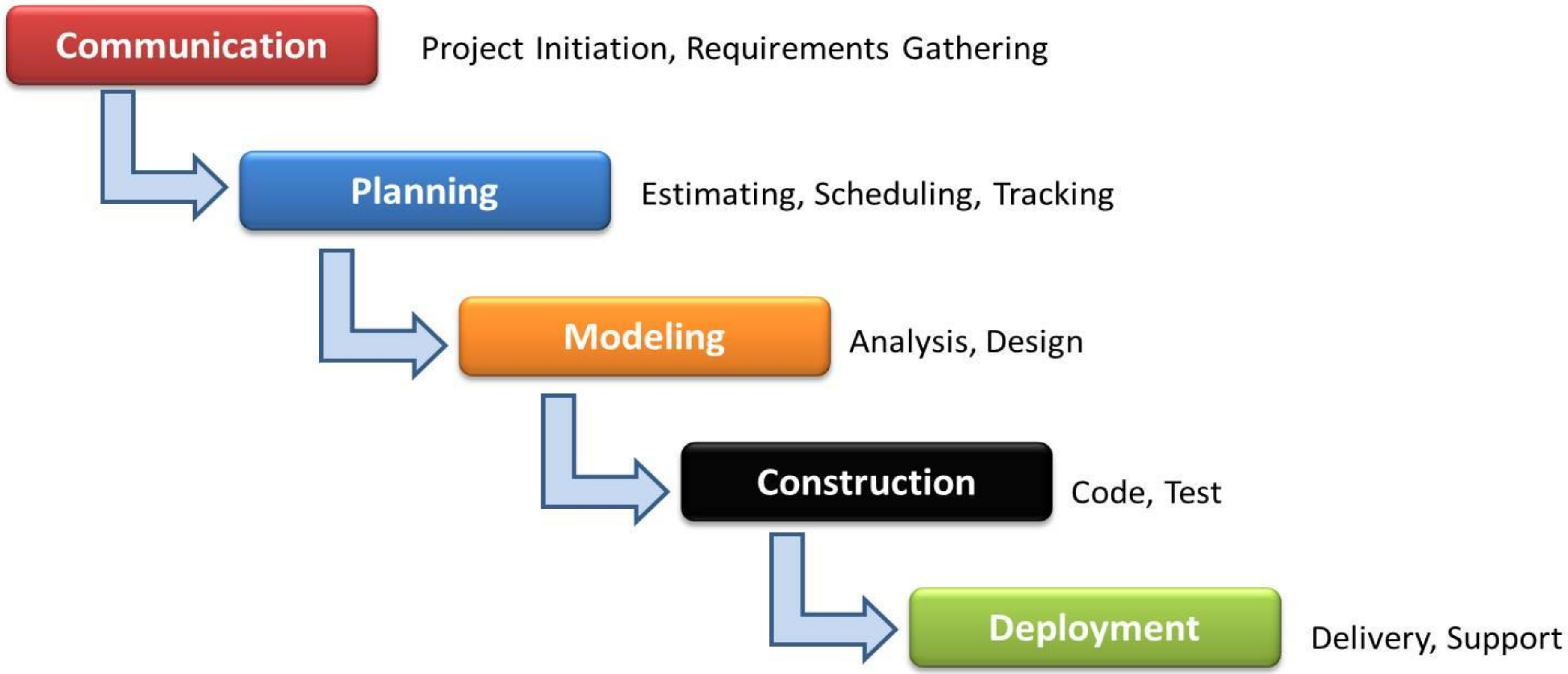
Sometimes called the **classic life cycle** or the waterfall model, the linear sequential model suggests a systematic sequential approach to **software development** that begins with **customer specification** of **requirements** and progresses through planning, modeling, construction, and deployment, culminating(excellence) in ongoing support of the completed software.

Prescriptive Process Model

There are times when the requirements for a **problem are well understood**—when work flows from communication through deployment in a reasonably linear fashion.

It may also occur in a limited number of new development efforts, but only when requirements are well defined and reasonably stable.

PRESCRIPTIVE PROCESS MODEL



The Waterfall Model: A Traditional Approach of SDLC

Prescriptive Process Model

When water fall model use?
What is Specialty of waterfall model?

Strength

1. The phase are clearly **separated** from each other.
2. This allow simple **planning** and **control**.
3. An exacta estimation of the required **efforts** and **costs** is possible

Problem

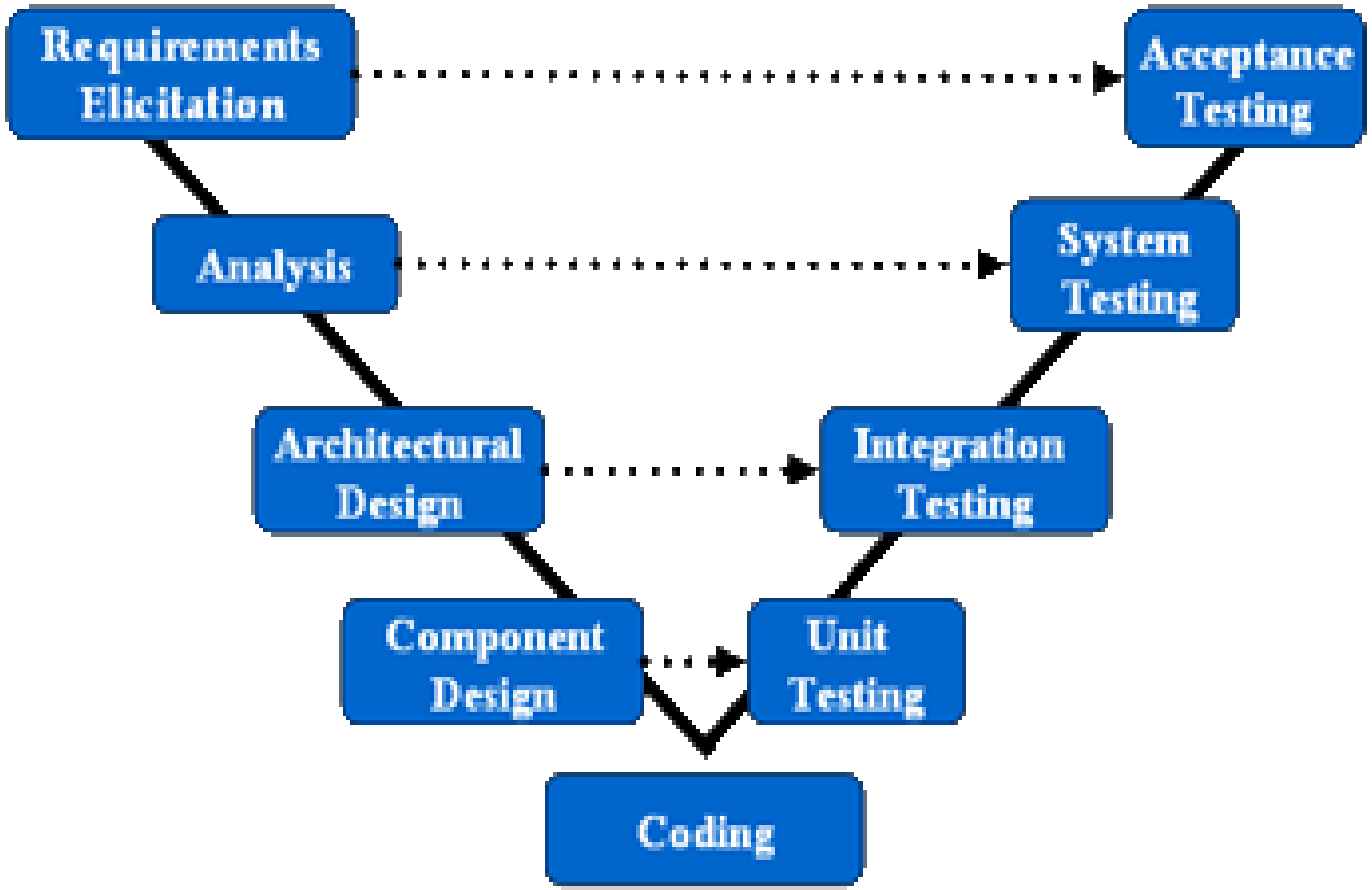
Specification is frozen early because:

1. It is costly and time consuming.
2. Less possibility to get success

Prescriptive Process Model

- A **variation** in the representation of the waterfall model is called the **Vmodel** , depicts the relationship of quality assurance actions to the actions associated with communication, modeling, and early construction activities.
- As a software team **moves down the left side** of the V, basic problem requirements are refined into progressively more detailed and technical representations of the problem and its solution.
- Once code has been generated, the team moves up the right side of the V, essentially performing a series of tests that validate each of the models created as the team moved down the left side.
- In reality, there is no fundamental difference between the **classic life cycle** and the **Vmodel**.

PRESCRIPTIVE PROCESS MODEL



PRESCRIPTIVE PROCESS MODEL

2. Incremental Process Model :

- The incremental model delivers a series of releases, called increments, that provide progressively more functionality for the customer as each increment is delivered.
- There are many situations in which **initial software requirements** are reasonably well defined, but the overall scope of the development effort precludes(prohibits) a purely linear process.

PRESCRIPTIVE PROCESS MODEL

- In addition, there may be a compelling need to provide a limited set of software functionality to users quickly and then refine and expand on that functionality in later software releases.

- In such cases, you can choose a process model that is designed to produce the software in increments.

Prescriptive Process Model

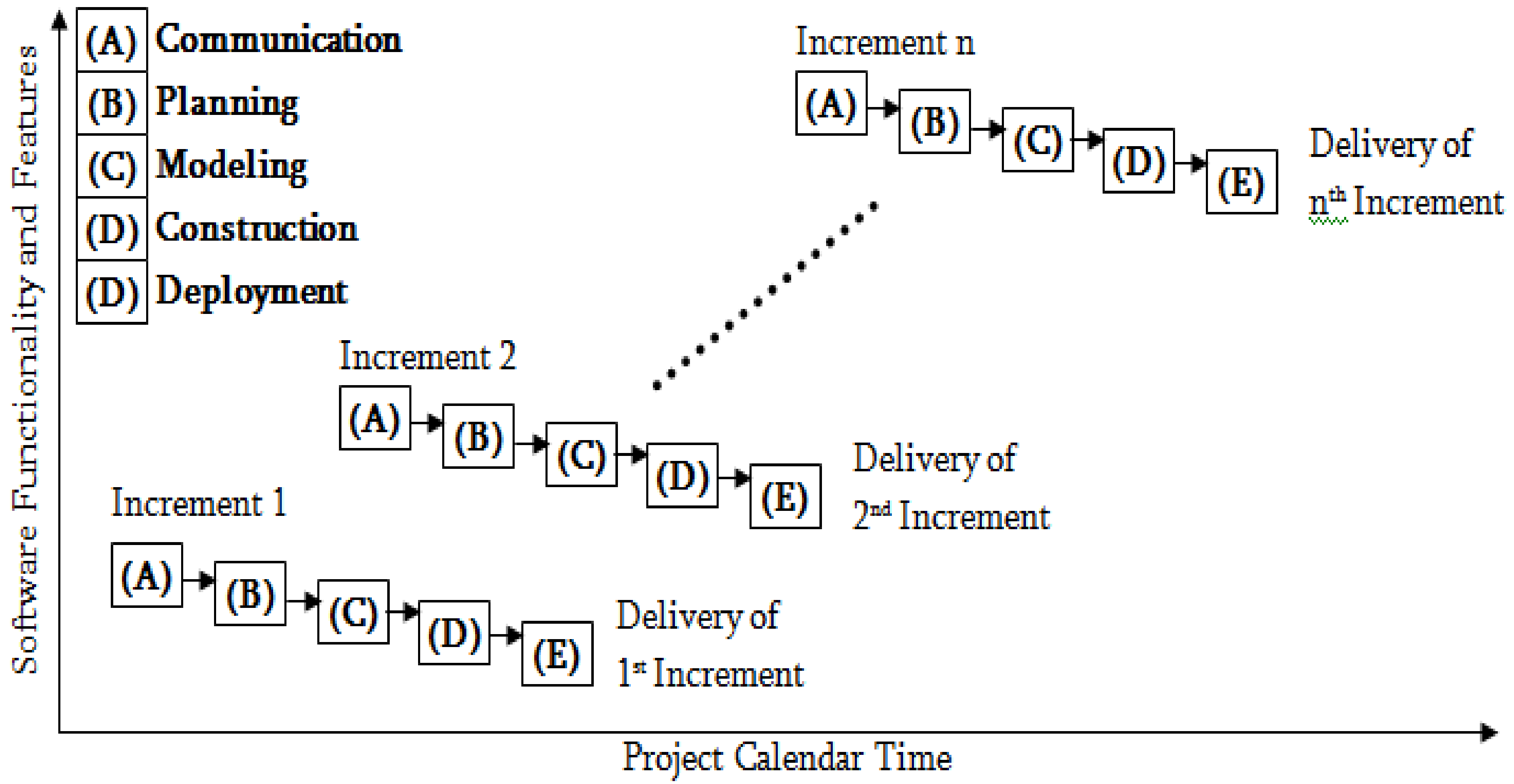


Figure : Flowchart of Incremental Model

Prescriptive Process Model

- When an incremental model is used, the first increment is often a **core product**. That is, basic requirements are addressed but many supplementary features (some known, others unknown) remain undelivered.
- The core product is used by the customer (or undergoes detailed evaluation). As a result of use and/or evaluation, a plan is developed for the next increment.
- The plan addresses the modification of the core product to better meet the needs of the customer and the delivery of additional features and functionality.
- This process is repeated following the delivery of each increment, until the complete product is produced.

Prescriptive Process Model

- Advantages
  - Possibility to use software **immediately**
  - **Highest priority requirements** tends to receive most testing
  - Early increment act as prototype to help draw requirements for later increment.

- Drawback
  - Hard to decide on size of each increments
  - Progress can be hard to judge and problems hard to find because there is no documentation to demonstrate what has been done.

Prescriptive Process Model

3. Evolutionary Process Models

- Nowadays modern computer software is characterized by continual change, by very tight time lines, and by an emphatic need for customer–user satisfaction.
- In many cases, time-to-market is the most important management requirement. If a market window is missed, the software project itself may be meaningless.
- Indeed, a software process that focuses on flexibility, extensibility, and speed of development over high quality does sound scary. And yet, this idea has been proposed by a number of well-respected software engineering experts.

Prescriptive Process Model

- The intent of evolutionary models is to develop high-quality software in an iterative or incremental manner. However, it is possible to use an evolutionary process to emphasize flexibility, extensibility, and speed of development.
- Evolutionary models are iterative. They are characterized in a manner that enables you to develop increasingly more complete versions of the software.

PRESCRIPTIVE PROCESS MODEL

- Two common evolutionary process models are:

1. Prototyping

2. The Spiral Model

Prescriptive Process Model

Prototype Model : (what is circumstances)

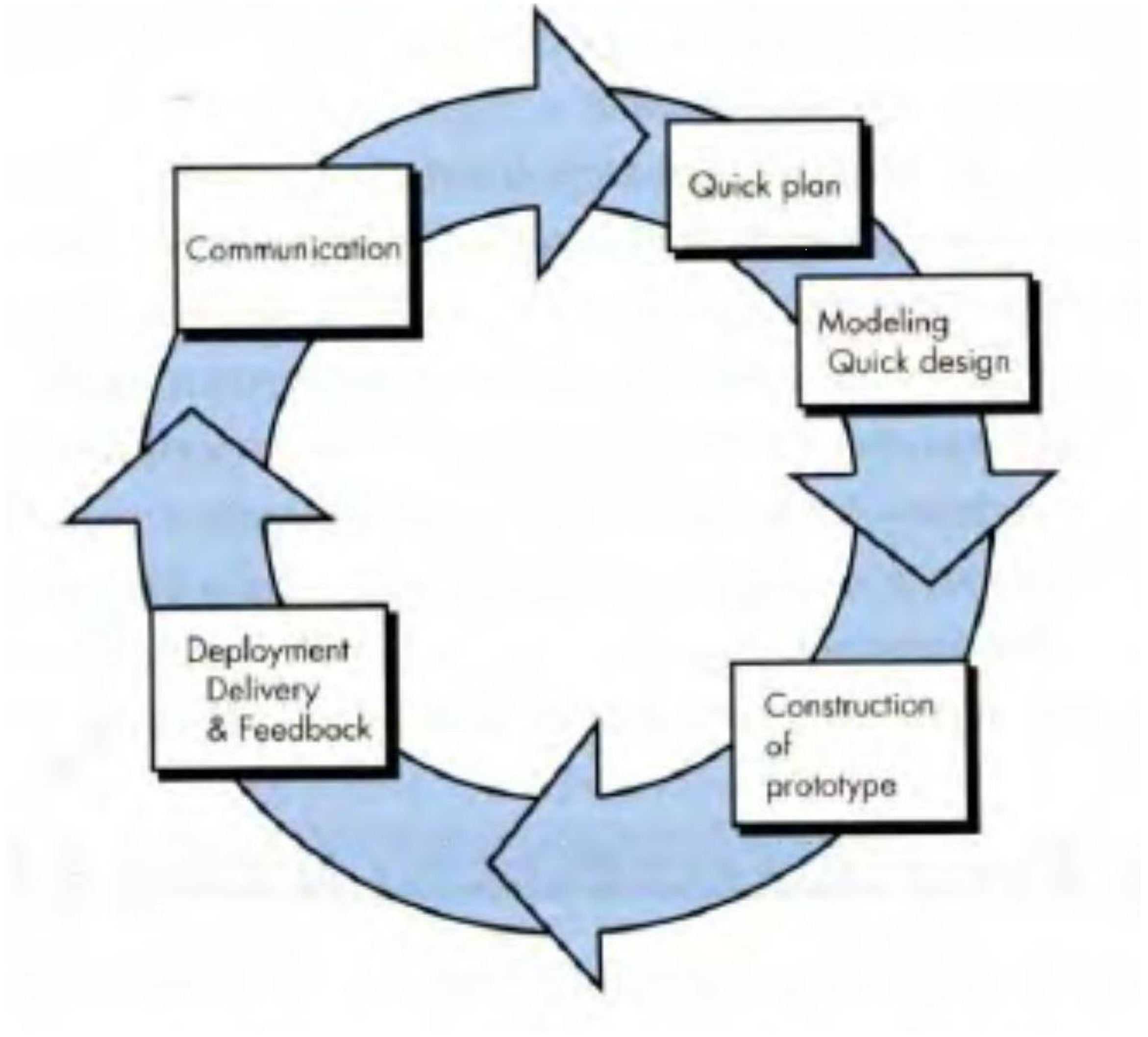
- It is not software.
- Developer build a prototype during the requirements phase.
- Prototype evaluated by end user.
- Users give corrective feedback.
- Developers further refine the prototype.
- When the user satisfied, the prototype code is brought up to the standards needed for a final product.

Prescriptive Process Model

Steps

- A preliminary project plan is developed
- An partial high-level paper model is created
- The model is source for a partial requirement specification
- A prototype is built with basic and critical attributes
- The design builds
  - The database
  - User interface
  - Algorithm function
- The designer demonstrates the prototype, the user evaluates for problems and suggests improvements
- This loop continues until the user is satisfied

PRESCRIPTIVE PROCESS MODEL



Prescriptive Process Model

Strength

- Customer can see the system requirements as they are being gathered
- Developer learn from customers
- A more accurate end product
- Unexpected requirements accommodated
- Allows for flexible design and development
- Steady, visible signs of progress produced
- Interaction with the prototype stimulates awareness of **additional needed functionality.**

Prescriptive Process Model

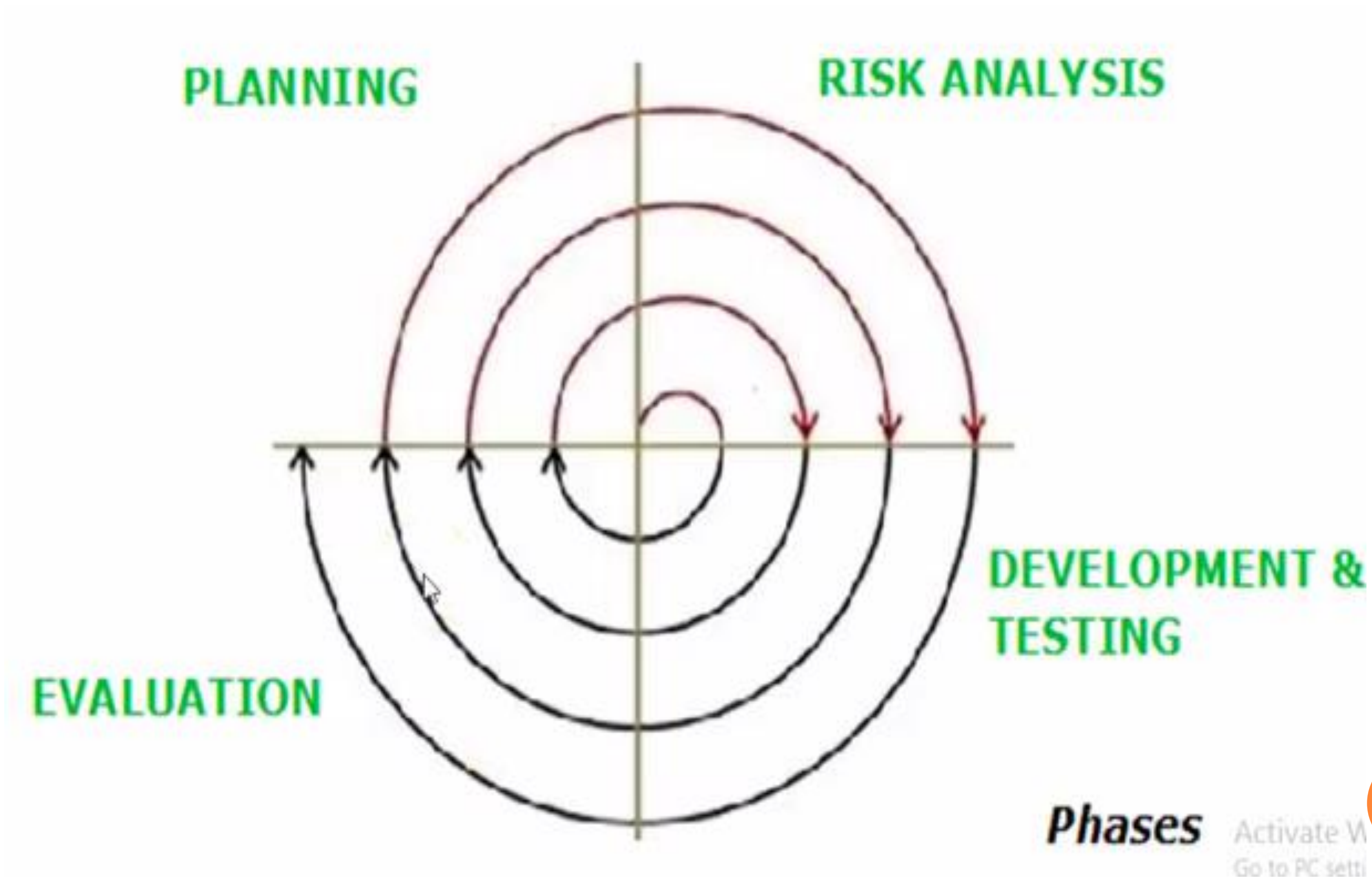
- **Weaknesses**
  - Tendency to abandon structured program development for “code and fix” development
  - Bad reputation for “quick and dirty” method
  - Overall maintainability may be overlooked

- **When we use**
  - Requirements are unstable
  - Requirements classification stage
  - Develop user interfaces
  - Short-lived demonstrations
  - New , original development

Prescriptive Process Model

- **The Spiral Model**
  - This model was first described by Barry Boehm in his 1986 paper "A Spiral Model of Software Development and Enhancement"
  - Spiral model is more an emphasis placed on risk analysis.
  - Each phase in spiral model begins with a design goal and ends with the client reviewing the progress.
  - Each phase in this model is split into four phases: **Planning, Risk analysis, Engineering, evaluation**

Prescriptive Process Model

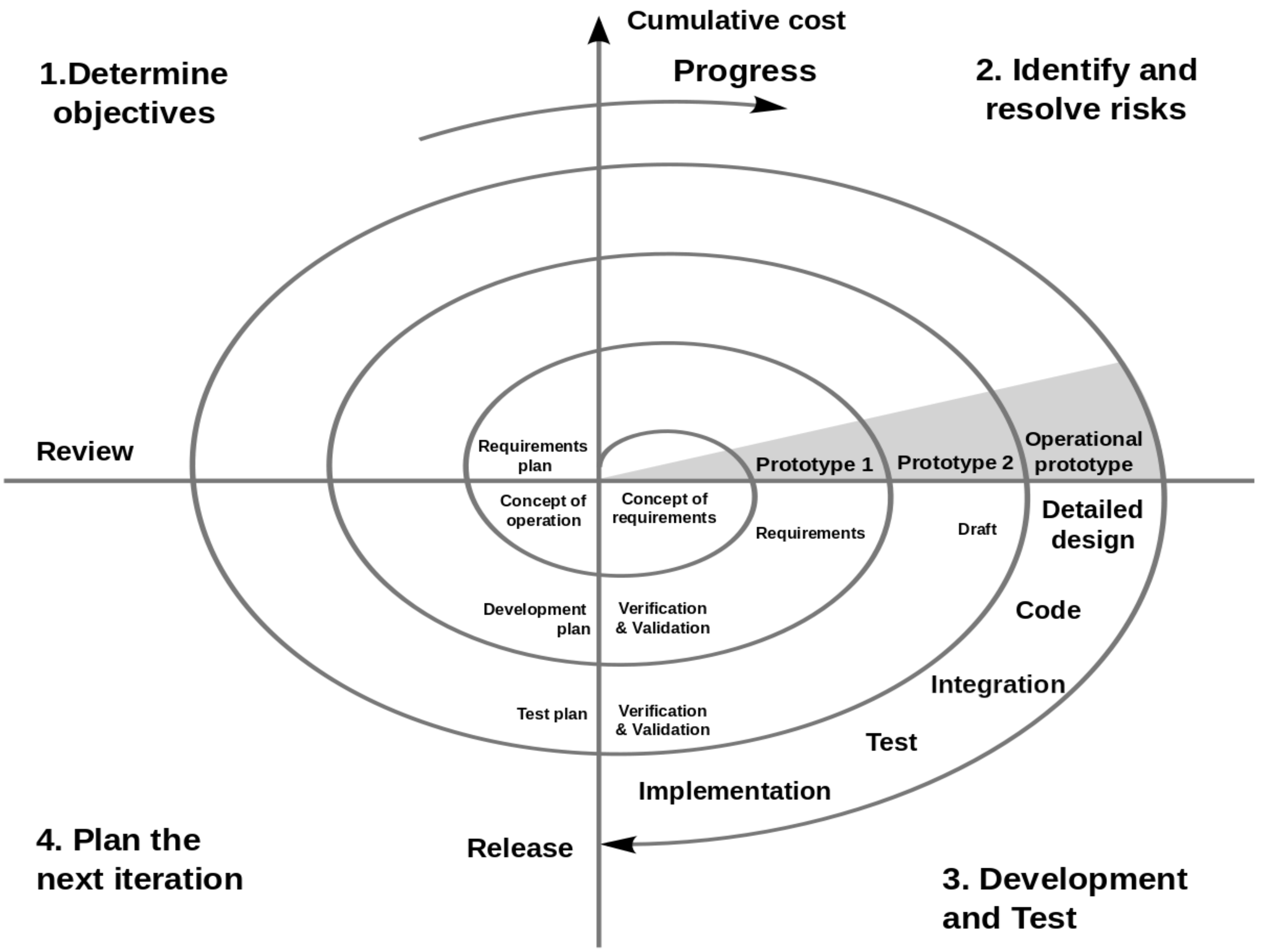


Prescriptive Process Model

Phase -1 (Planning)

- **Objective:** functionality, performance, unit requirement, critical success factors etc,
- **Alternative**: build, reuse, buy, sub-contract
- **Constraint :** cost, schedule, interface.
- Requirements are gathered during the planning phase.
- Requirements like 'BRS' that is Business Requirement Specifications and 'SRS' that System Requirement Specifications

Prescriptive Process Model



Prescriptive Process Model

- **Phase-2 (Risk analysis)**
  a) **Identify risk**: lack of experience, new technology, tight schedules, poor process

  b) **Resolve risk :** Evaluate if money could be lost by continuing system development)

  In the **risk analysis phase**, a process is undertaken to identify risk and alternate solutions. A prototype is produced at the end of the risk analysis phase. If any risk is found during the risk analysis then alternate solutions are suggested and implemented.

Prescriptive Process Model

- **Phase-3 (Engineering)**
  - In this phase software is **developed**, along with testing at the end of the phase. Hence in this phase the development and testing is done.

- **Phase-4 (evaluation)**
  - This phase **allows the customer to evaluate** the output of the project to date before the project continues to the next spiral.

PRESCRIPTIVE PROCESS MODEL

When to use Spiral-SDLC Model?

- When project is large.
- Where the software needs continuous risk evaluation.
- When releases are required to be frequent.
- When risk and costs evaluation is important.
- For medium to high-risk projects.
- When requirements are unclear and complex.
- When changes may require at any time.
- Long-term project commitment unwise because of potential changes to economic priorities
- Users are unsure of their needs
- Requirements are complex

Prescriptive Process Model

Strength

- High amount of risk analysis hence, avoidance of Risk is enhanced.
- Good for large and mission-critical projects.
- Strong approval and documentation control.
- Additional Functionality can be added at a later date.
- Software is produced early in the software life cycle.

Prescriptive Process Model

- **Weakness**

  - Time spent for small and low-risk project
  - The model is complex
  - Risk assessment expertise is required
  - Spiral may be continue indefinitely
  - Can be a costly model to use.
  - Project's success is highly dependent on the risk analysis phase.
  - Doesn't work well for smaller projects.

Waterfall model v/s spiral model

Waterfall Model	**Spiral Model**
1] Waterfall model is suitable for small projects.	1] Spiral model is not suitable for small projects.
2] High amount of risk and uncertainty.	2] Better risk management.
3] Easy to understand.	3] Process is complex.
4] Stages are clearly defined.	4] The process may go indefinitely.
5] This model is not suitable for long and ongoing projects.	5] This model is suitable for long and ongoing projects.
6] Sequence is followed	6] Iterations are followed
7] Requirements once fixed cannot be modified	7] Flexible with user requirements
8] Refinements are not so easy	8] Refinements are easily possible

Water Fall Model	**Spiral Model**
9] Phases are processed and completed one at a time.	9] Phases are repeated itself
10] In the software development life cycle, business requirements are frozen after the initial phase.	10] In the spiral model, requirements are not frozen by the end of the initial phase. It is kind of executed in a continuous mode.

Comparison of Various SDLC Models

Properties of Model	Water-Fall Model	Incremental Model	Spiral Model
Planning in early stage	Y	Y	Y
Returning to an earlier phase	N	Y	Y
Handle Large-Project	N	N	Y
Detailed Documentation	Compulsory	Yes but not much	Y
Cost	Low	Low	expensive
Requirement Specifications	Beginning	Beginning	Beginning
Flexibility to change	Difficult	Easy	easy
User Involvement	Only at beginning	Intermediate	High

Properties of Model	Water-Fall Model	Incremental Model	Spiral Model
Duration	Long	Very long	Long
Risk Involvement	High	Low	Medium to high risk
Framework Type	Linear	Linear + Iterative	Linear + Iterative
Testing (Developer Side)	After completion of coding phase	After every iteration	At the end of the engineering phase
Re-usability	Least possible	To some extent	To some extent
Time-Frame	Very Long	Long	Long
Working software availability	At the end of the life-cycle	At the end of every iteration	At the end of every iteration
Team size	Large Team	Not Large Team	Large Team
Customer control over administrator	Very Low	Yes	Yes

RAD MODEL.

- The **RAD (Rapid Application Development)** model is based on prototyping and iterative development with no specific planning involved.
- The process of writing the software itself involves the planning required for developing the product. The process of writing the software itself involves the planning required for developing the product.
- RAD focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using iterative concept, reuse of the existing prototypes (components), continuous integration and rapid delivery.

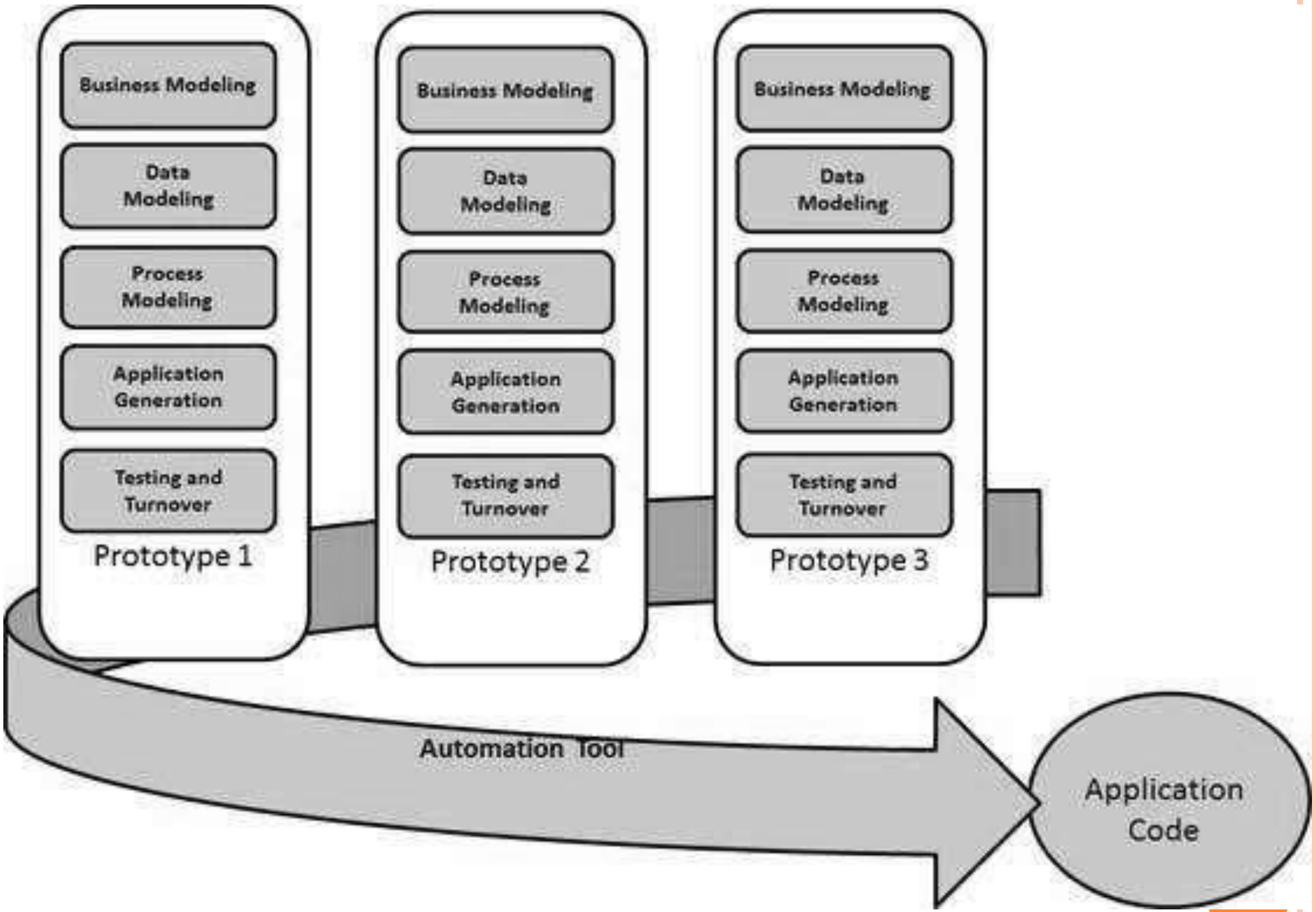
What is RAD?

- Rapid application development is a software development methodology that uses least planning in service of fast prototyping.
- RAD projects follow iterative and incremental model and have small teams comprising of developers, domain experts, customer representatives and other IT resources working progressively on their component or prototype.

RAD Model Design

- Following are the various phases of the RAD Model –
- Business Modeling
  - flow of **information** and the **distribution of information** between various business channels. A complete business analysis is performed to find the dynamic information for business.
  - In short req. gathering
- Data Modeling
  - Data object
  - Data attribute
  - Relationship
- Process Modeling
  - Process on data modeling & business modeling
  - In short Analysis & Designing

- Application Generation
  - The actual system is built and coding Done by by using automation tools to convert process and data models into actual prototypes.
- Testing and Turnover
  - Testing and Depolyment


Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing and Turnover
Prototype 1
Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing and Turnover
Prototype 2
Business Modeling
Data Modeling
Process Modeling
Application Generation
Testing and Turnover
Prototype 3
Automation Tool
Application Code

Assignment Questions

1. What is software? Define characteristics of software.
2. List out and explain Applications of Software.
3. What is software Engineering?
4. Write the importance of software engineering
5. Describe: **software engineering as layered technology.**
6. What is software process? Define a generic process framework activity of Software Engineering.
7. Explain all process models in brief.
8. Compare all incremental , spiral , waterfall models.
9. Differentiate : Waterfall v/s Spiral Model .
10. Justify : Software Can't be wear out.
11. Comment on software is custom built.
12. Explain SDLC in Depth.