Tribhuvan University
Institute of Engineering
Pulchowk Campus
Department of Electronics and Computer Engineering

Software Engineering

Chapter One

Software Process and requirements

by
Er. Bishwas Pokharel
Lecturer, Kathford Int'l College of Engg. and Mgmt.

Date: 29th Dec, 2017

Chapter One: **Software Process and requirements**

Course Outline: **12 hours, 20 Marks**

1.1. Software crisis

1.2. Software characteristics

1.3. Software quality attributes

1.4. Software process model

1.5. Process iteration

1.6. process activities

1.7. Computer-aided software engineering

1.8. Functional and non –functional requirements

1.9. User requirements

1.10. System requirement

1.11. Interface specification

1.12. The software requirements documents

1.13. Feasibility study

1.14. Requirements elicitation and analysis

1.15. Requirements validation and management

Since the **prime objective** of software engineering is to develop methods for large systems, which produce **high quality** software at **low cost** and in **reasonable time**, it is essential to perform software development **in phases**. This **phased development of software** is often referred to as software development life cycle **(SDLC)** or software life cycle. .

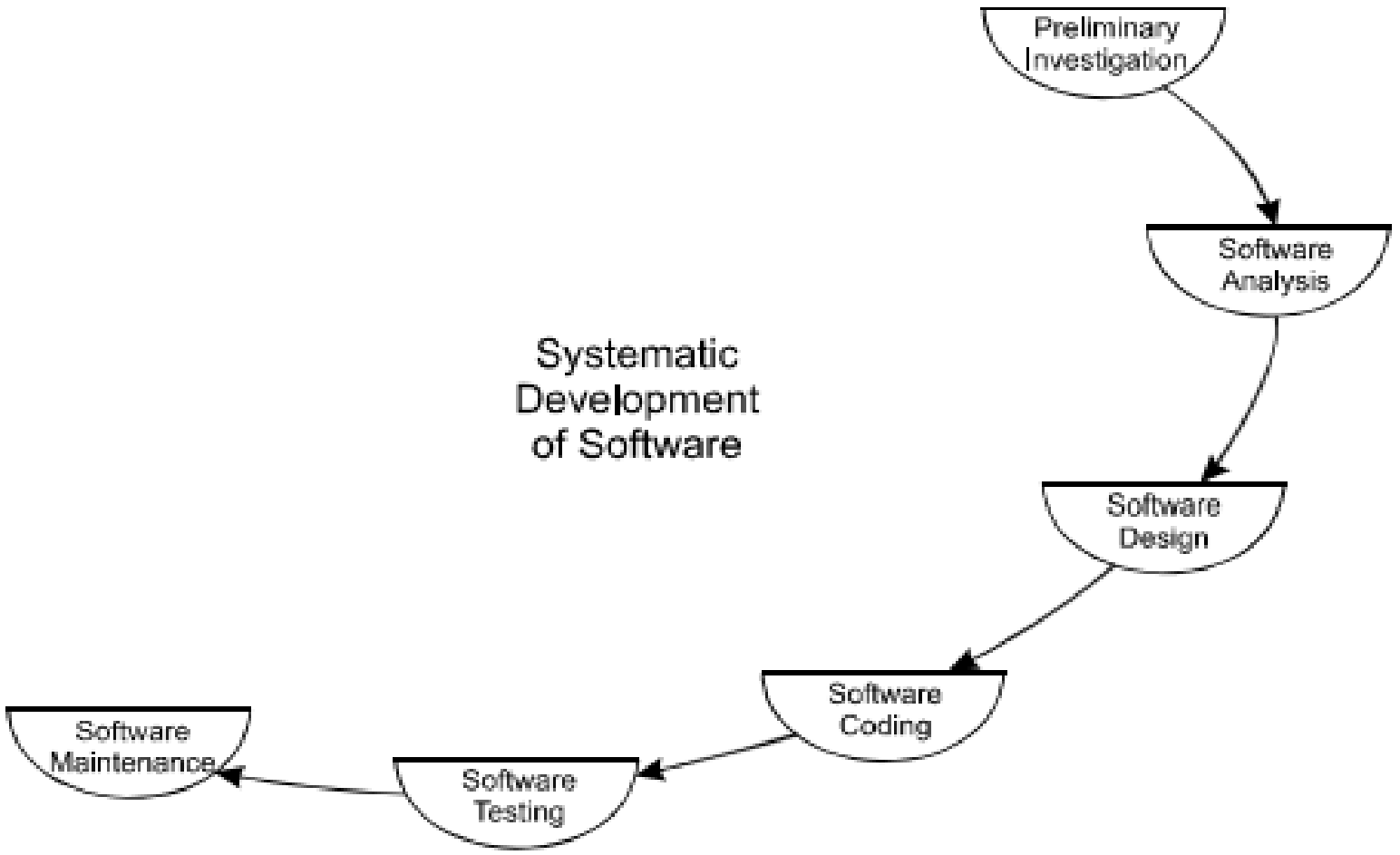


Fig 1 software development process

In fig 1,

- These phases **work in top to bottom** approach
- The phases take inputs from the previous phases, add features, and then produce outputs
- The outputs from different phases are referred to as **intermediate product, work product**, or derivable.

1. **Preliminary investigation:**

   The output of preliminary investigation decides whether the new system should be developed or not. There are three constraints, which decides the go or no-go decision.

   a. Technical: It determines whether technology needed for proposed system is available or not. If available then how can it be integrated within the organization. It also determines whether the existing system can be upgraded to use new technology and whether the organization has the expertise to use it or not.

b. Time: This evaluation determines the time needed to complete a project. Time is an important issue in software development as cost increases with an increase in the time period of a project.

c. Budgetary: This evaluation looks at the financial aspect of the project. Budgetary evaluation determines whether the investment needed to implement the system will be recovered at later stages or not.

2. **Software Analysis:** studies the problem or requirements of software in detail. After analyzing the requirements of the user, a requirement statement known as **software requirement specification** (SRS) is developed. After analyses, planning for the project begins. It includes developing plans that describes the activities to be performed during the project: software configuration management plans, project and scheduling, quality assurance plan and resources required.

3. **Software Design:** In this phase the requirements are given a ‘defined’ form. Software design serves as the **blueprint for the implementation of requirement** in the software system. Each element of the analysis model in the analysis phase provides information that is required to create design models. The requirement specification of software, together with data, functional, and behavioral models provides a platform to feed the design task to meet required functional and quality requirements of a system.

4. **Software Coding:** This phase can be defined as a process of translating the software requirements into a programming language using tools that are available. Writing a software code requires a thorough knowledge of programming language and its tools. Therefore, it is important to choose the appropriate programming language according to the user requirements. A program code is efficient if it makes optimal use of resources and contains minimum errors.

5. Software Testing: This testing is performed to assure that software is free from errors. Efficient testing improves the quality of software. To achieve this, software testing requires a thorough analysis of the software in a systematic manner. Test plan is created to test software in a planned and systematic manner. In addition, **software testing is performed to ensure that software produces the correct outputs.** This implies that outputs produced should be according to user requirements.

6. Software Maintenance: This phase comprises of a set of software engineering activities that occur after software is delivered to the user. After the software is developed and delivered, it may require changes. Sometimes, changes are made in software system when user requirements are not completely met. To make changes in software system, software maintenance process evaluates, controls, and implements changes. Note that changes can also be forced on the software system because of changes in government regulations or changes in policies of the organization.

Case Study

Q.

Mention different phases of Software Development life cycle(SDLC) if you are under the project of Bridge Development. (**class work**)

Case Study: your answer may differ but must be similar to following standardized answer,

Phase	Building Bridge	SDLC Phase
Formulate the problem by understanding the nature and general requirements of the problem.	Understand the load of the bridge it must carry, the approximate locations where it can be built, the height requirements, and so on.	Preliminary investigation.
Defining the problem precisely.	Specify the site for the bridge, its size, and a general outline of the type of bridge to be built.	Software requirement analysis and specifications.
Detailing the solution to the problem.	Determine exact configuration, size of the cables and beams, and developing blueprints for the bridge.	Software design.
Implementing.	Correspond to actual building of the bridge.	Software coding.
Checking.	Specify load, pressure, endurance, and robustness of the bridge.	Software testing.
Maintaining.	Specify repainting, repaving, and making any other repairs, which are necessary.	Software maintenance.

Process:

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

activities: achieve a broad objective (e.g., communication with stakeholders) and is applied regardless of the application domain, size of the project, complexity of the effort.

actions: action (e.g., architectural design) encompasses a set of tasks that produce major work product .Ex: Use models, visualizations as a communication and collaboration tool.

tasks: task focuses on a small, but well-defined objective(Ex: conducting a unit test) that produces a tangible outcome

What is Software process?

When you work to build a product or system, it's important to go through a series of predictable steps—a road map that helps you create a timely, high-quality result. The road ma that you follow is called a "software process."

Who does it? Software engineers and their managers adapt the process to their needs and then follow it. In addition, the people who have requested the software have in the process of defining, building, and testing it

Why is it important?

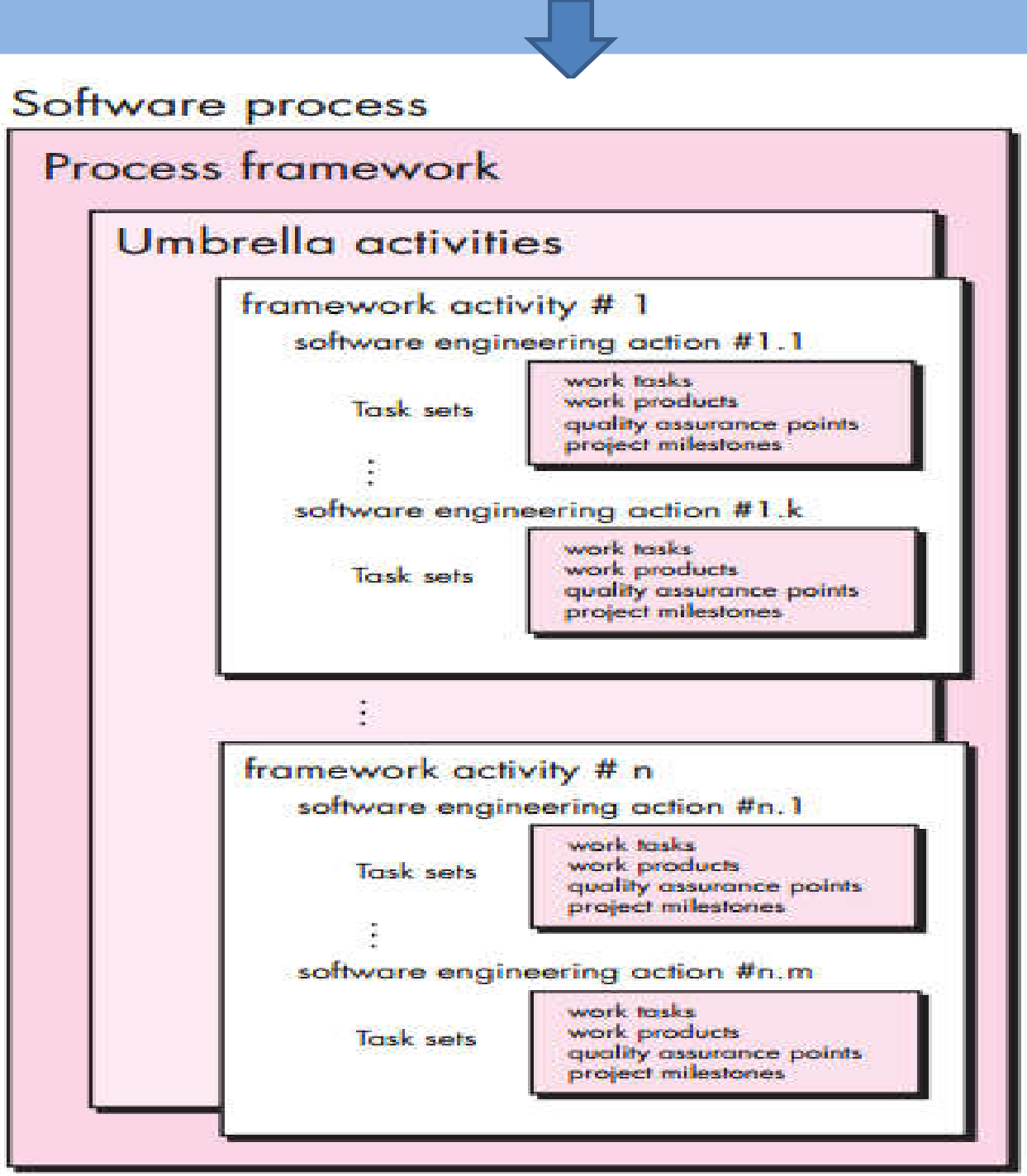
Because it provides stability, control.

What are the steps?

At a detailed level, the process that you adopt depends on the software that you're building. One process might be appropriate for creating software for an aircraft avionics system, while an entirely different process would be indicated for the creation of a website

what exactly is a software process from a technical point of view?

Within the context of pressman book, it is define a **software process** as a framework for the activities, actions, and tasks that are required to build high-quality software .


Software process
Process framework
Umbrella activities
framework activity # 1
software engineering action #1.1
Task sets
work tasks
work products
quality assurance points
project milestones
software engineering action #1.k
Task sets
work tasks
work products
quality assurance points
project milestones
framework activity # n
software engineering action #n.1
Task sets
work tasks
work products
quality assurance points
project milestones
software engineering action #n.m
Task sets
work tasks
work products
quality assurance points
project milestones

A generic process framework for software engineering encompasses five activities:

a. **Communication**. Before any technical work can commence, it is critically important to communicate and collaborate with the customer (and other stakeholders. The intent is to understand stakeholders' objectives for the project and to gather requirements that help define software features and functions.

A *stakeholder* is anyone who has a stake in the successful outcome of the project—business managers, end users, software engineers, support people, etc

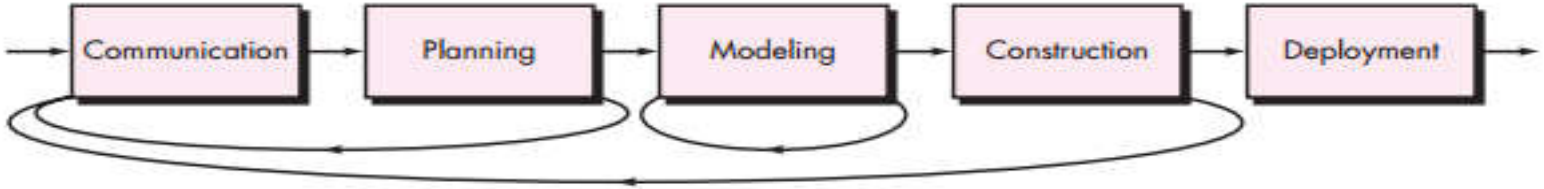
b. Planning. A software project is a complicated journey, and the planning activity creates a "map" that helps guide the team as it makes the journey. The map—called a software project plan—defines the software engineering work by describing the technical tasks to be conducted, the risks that are likely, the resources that will be required, the work products to be produced, and a work schedule.

c. Modeling. Whether you're a landscaper, a bridge builder, an aeronautical engineer, a carpenter, or an architect, you work with models every day. You create a "sketch" of the thing so that you'll understand the big picture—what it will look like architecturally, how the constituent parts fit together, and many other characteristics. If required, you refine the sketch into greater and greater detail in an effort to better understand the problem and how you're going to solve it. A software engineer does the same thing by creating models to better understand software requirements

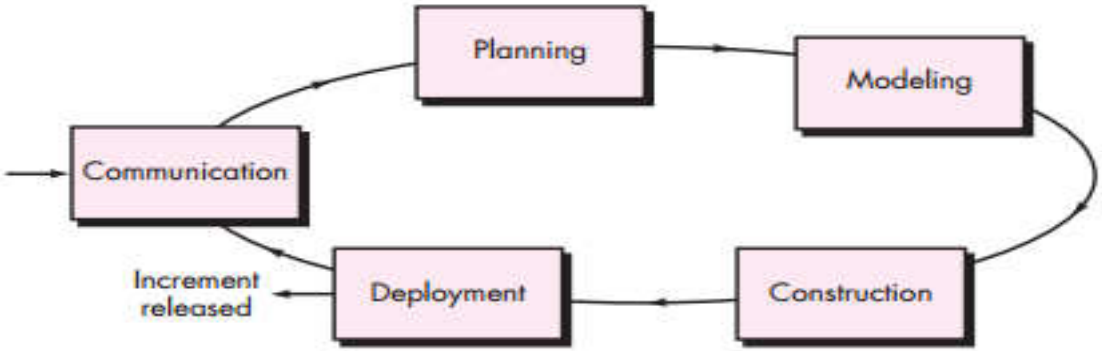
d. Construction. This activity combines code generation (either manual or automated) and the testing that is required to uncover errors in the code.

e. Deployment. The software (as a complete entity or as a partially completed increment) is delivered to the customer who evaluates the delivered product and provides feedback based on the evaluation.

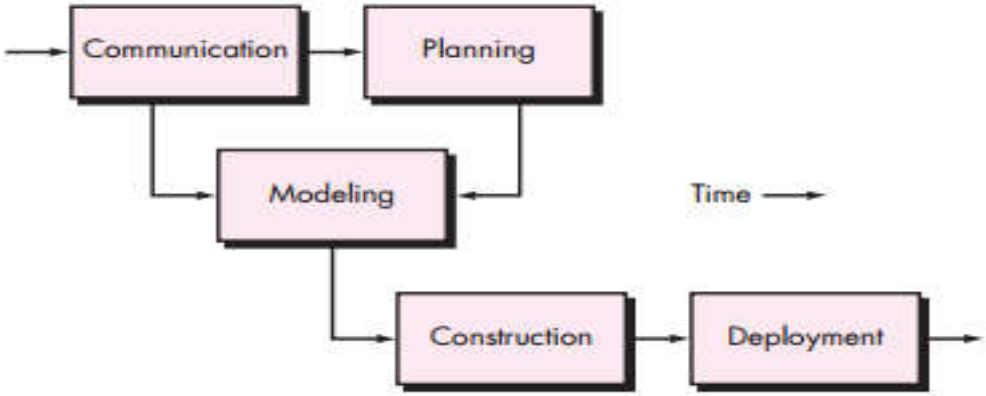
(a) Linear process flow



(b) Iterative process flow



(c) Evolutionary process flow



(d) Parallel process flow

Umbrella activities occur throughout the software process and focus primarily on project management, tracking, and control.

- **Software project tracking and control**—allows the software team to assess progress against the project plan and take any necessary action to maintain the schedule.

- **Risk management**—assesses risks that may affect the outcome of the project or the quality of the product.

Software quality assurance—defines and conducts the activities required to ensure software quality.

Technical reviews—assesses software engineering work products in an effort to uncover and remove errors before they are propagated to the next activity.

Software configuration management—manages the effects of change throughout the software process.

Work product preparation and production—encompasses the activities required to create work products:models,documents,logs,forms,lists .

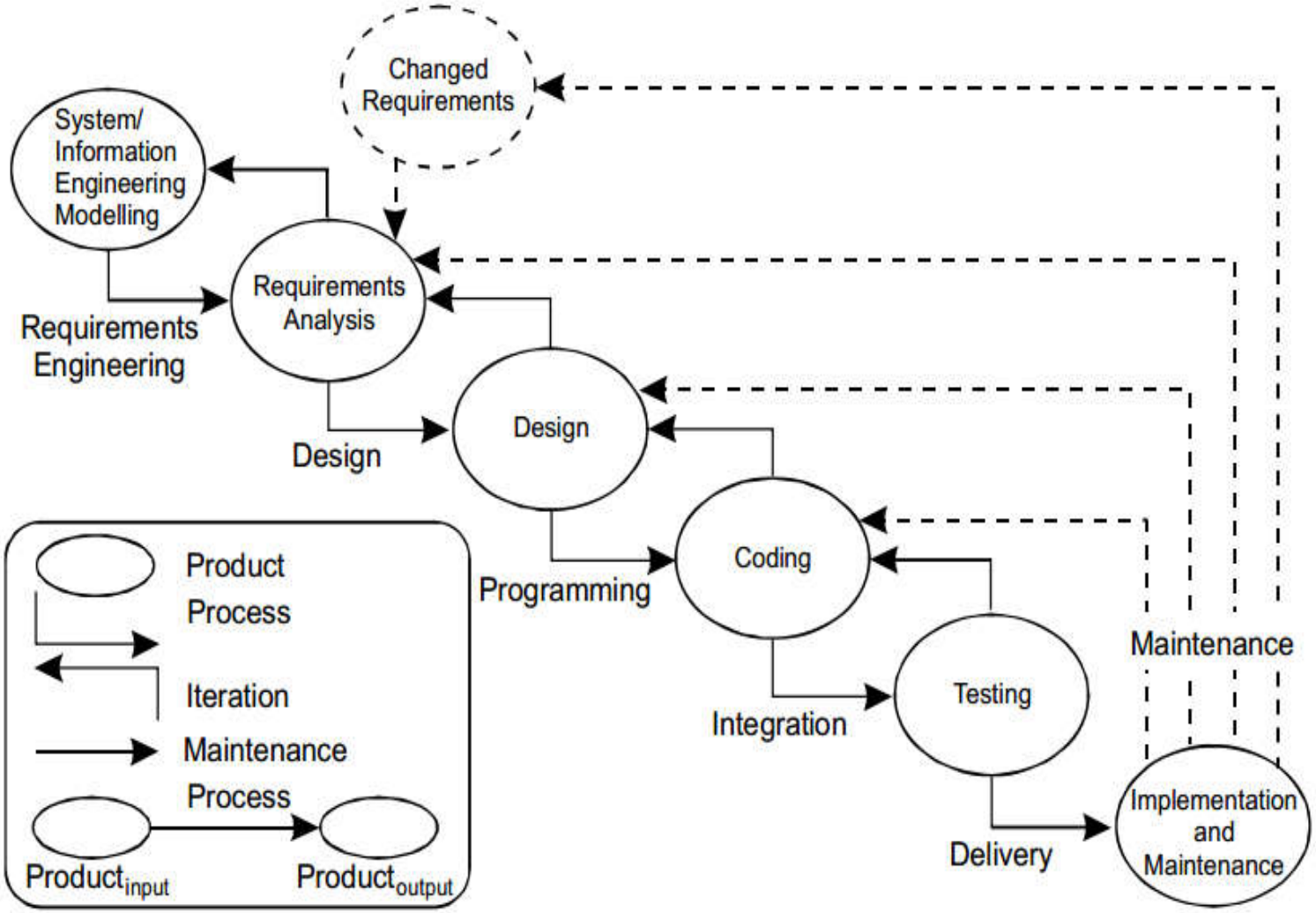
Software process model

:is a simplified representation of a software process.

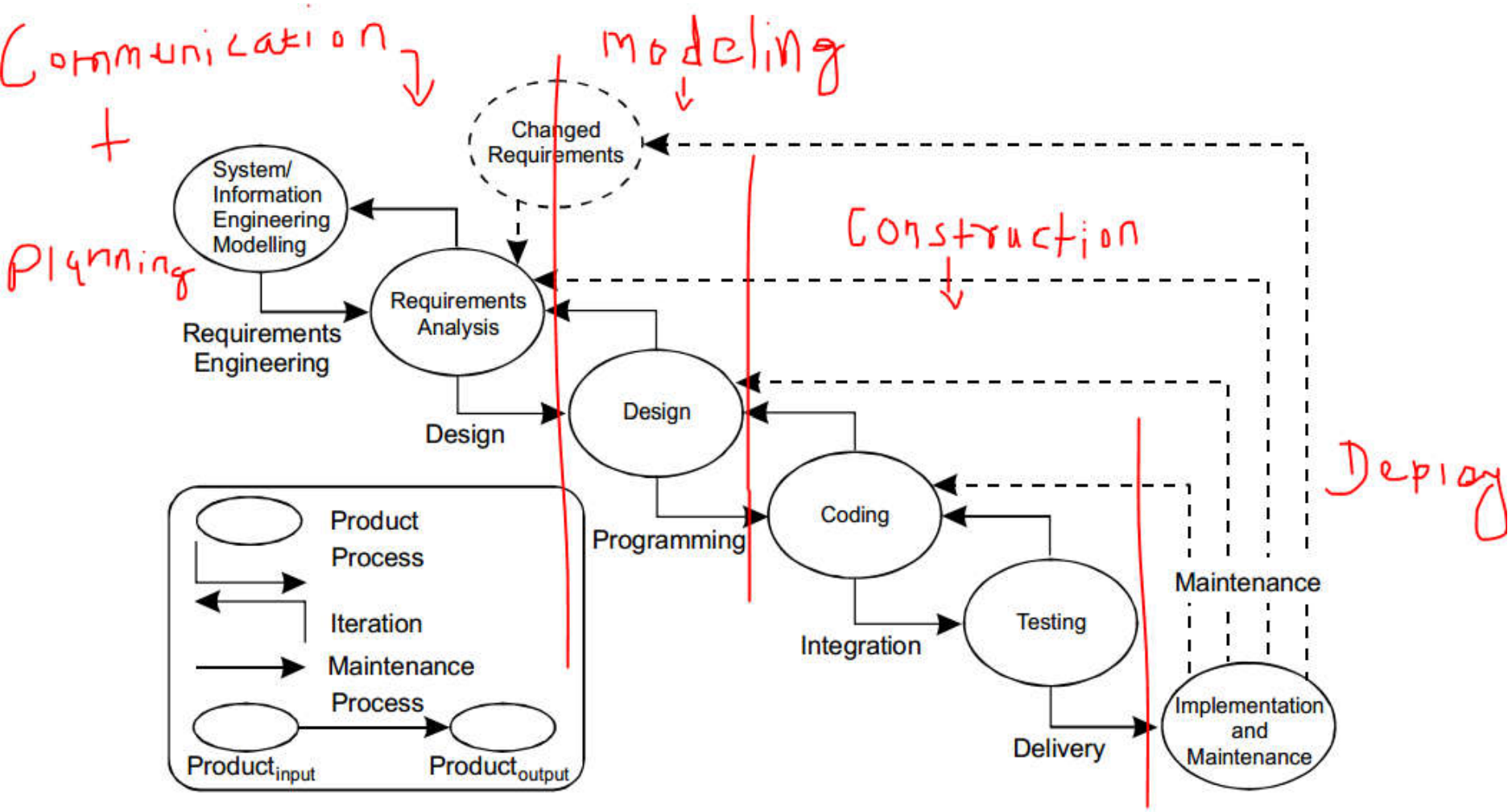
Some important models are as follows;

a. Waterfall model
b. Prototype model
c. Spiral model
d. Evolutionary model(Ex: Incremental model..)

a. Waterfall model



a. Waterfall model



a. Waterfall model

System/information engineering modeling:

System engineering includes collecting requirements at the system level like interaction of hardware with software.

i. **Requirement specification:** To specify the requirements' users specification should be clearly understood and the requirements should be analyzed. This phase involves interaction between user and software engineer, and produces a document known as software requirement specification (SRS).

a. Waterfall model

ii. Design: Determines the detailed process of developing software after the requirements are analyzed. It utilizes software requirements defined by the user and translates them into a software representation. In this phase, the emphasis is on finding a solution to the problems defined in the requirement analysis phase. The software engineer, in this phase is mainly concerned with the data structure, algorithmic detail, and interface representations.

a. Waterfall model

iii. Coding: Emphasizes on translation of design into a programming language using the coding style and guidelines. The programs created should be easy to read and understand. All the programs written are documented according to the specification.

iv. Testing: Ensures that the product is developed according to the requirements of the user. Testing is performed to verify that the product is functioning efficiently with minimum errors. It focuses on the internal logics and external functions of the software

a. Waterfall model

v. **Implementation and maintenance:** Delivers fully functioning operational software to the user. Once the software is accepted and deployed at the user's end, various changes occur due to changes in external environment (these include upgrading new operating system or addition of a new peripheral device). The changes also occur due to changing requirements of the user and the changes occurring in the field of technology. This phase focuses on modifying software, correcting errors, and improving the performance of the software.

a. Waterfall model

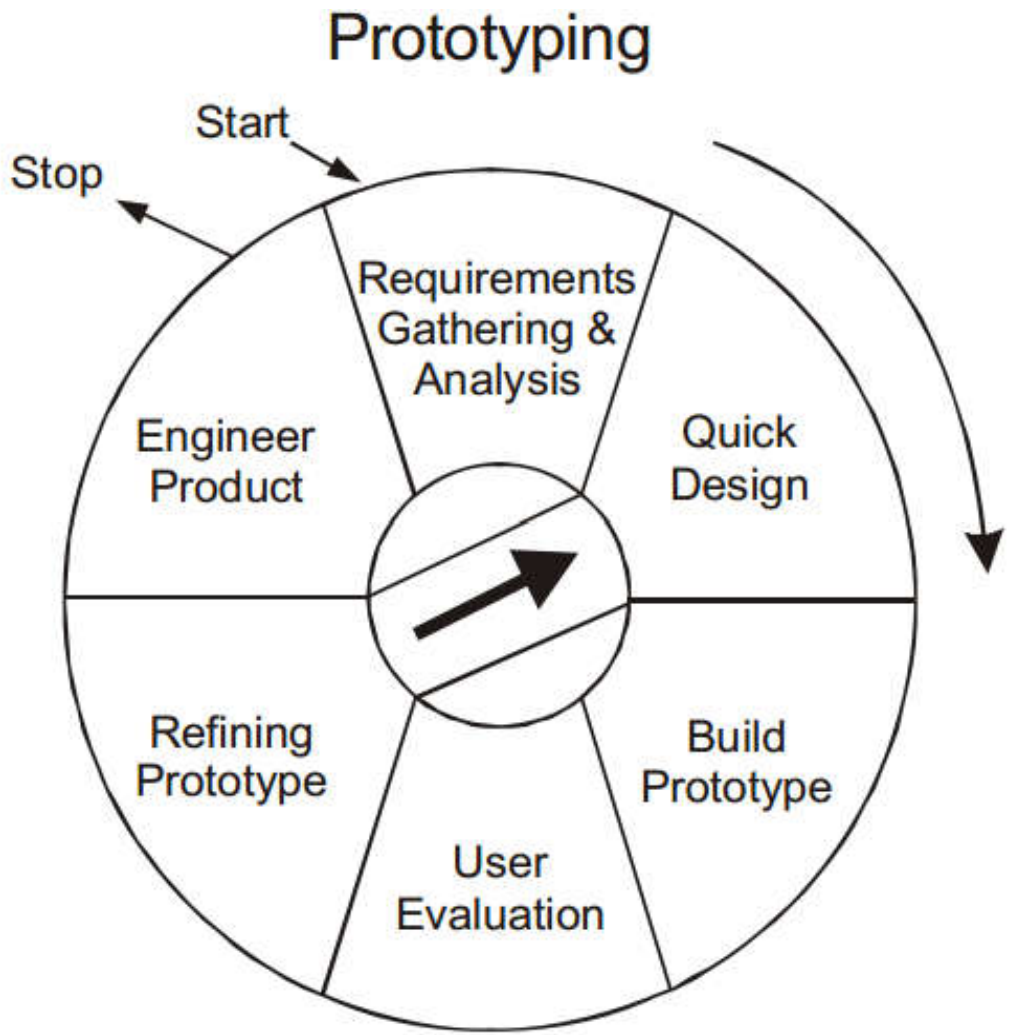
Input to the Phase	Process/Phase	Output of the Phase
Requirements defined through communication	Requirements analysis	Software requirements specification document
Software requirements specification document	Design	Design specification document
Design specification document	Coding	Executable software modules
Executable software modules	Testing	Integrated product
Integrated product	Implementation	Delivered software
Delivered software	Maintenance	Changed requirements

a. Waterfall model

Advantages	Disadvantages
▪ Relatively simple to understand. ▪ Each phase of development proceeds sequentially. ▪ Allows managerial control where a schedule with deadlines is set for each stage of development. ▪ Helps in controlling schedules, budgets, and documentation.	▪ Requirements need to be specified before the development proceeds. ▪ The changes of requirements in later phases of the waterfall model cannot be done. This implies that once an application is in the testing phase, it is difficult to incorporate changes at such a late phase. ▪ No user involvement and working version of the software is available when the software is developed. ▪ Does not involve risk management. ▪ Assumes that requirements are stable and are frozen across the project span.

b. prototype model

- The prototyping model is applied when there is an absence of detailed information regarding input and output requirements in the software. It is often difficult to know all the requirements at the beginning of a project.
- Prototyping model increases flexibility of the development process by allowing the user to interact and experiment with a working representation of the product known as **prototype**. A prototype gives the user an actual feel of the system.


Prototyping
Start
Stop
Requirements Gathering & Analysis
Quick Design
Build Prototype
User Evaluation
Refining Prototype
Engineer Product

i. **Requirements gathering and analysis:** Prototyping model begins with requirements analysis, and the requirements of the system are defined in detail. The user is interviewed to know the requirements of the system.

ii. **Quick design:** When requirements are known, a preliminary design or a quick design for the system is created. It is not a detailed design, however, it includes the important aspects of the system, which gives an idea of the system to the user. Quick design helps in developing the prototype.

iii.Build prototype: Information gathered from quick design is modified to form a prototype. The first prototype of the required system is developed from quick design. It represents a 'rough' design of the required system.

iv.User evaluation: Next, the proposed system is presented to the user for consideration as a part of development process. The users thoroughly evaluate the prototype and recognize its strengths and weaknesses, such as what is to be added or removed.Comments and suggestions are collected from the users and are provided to the developer.

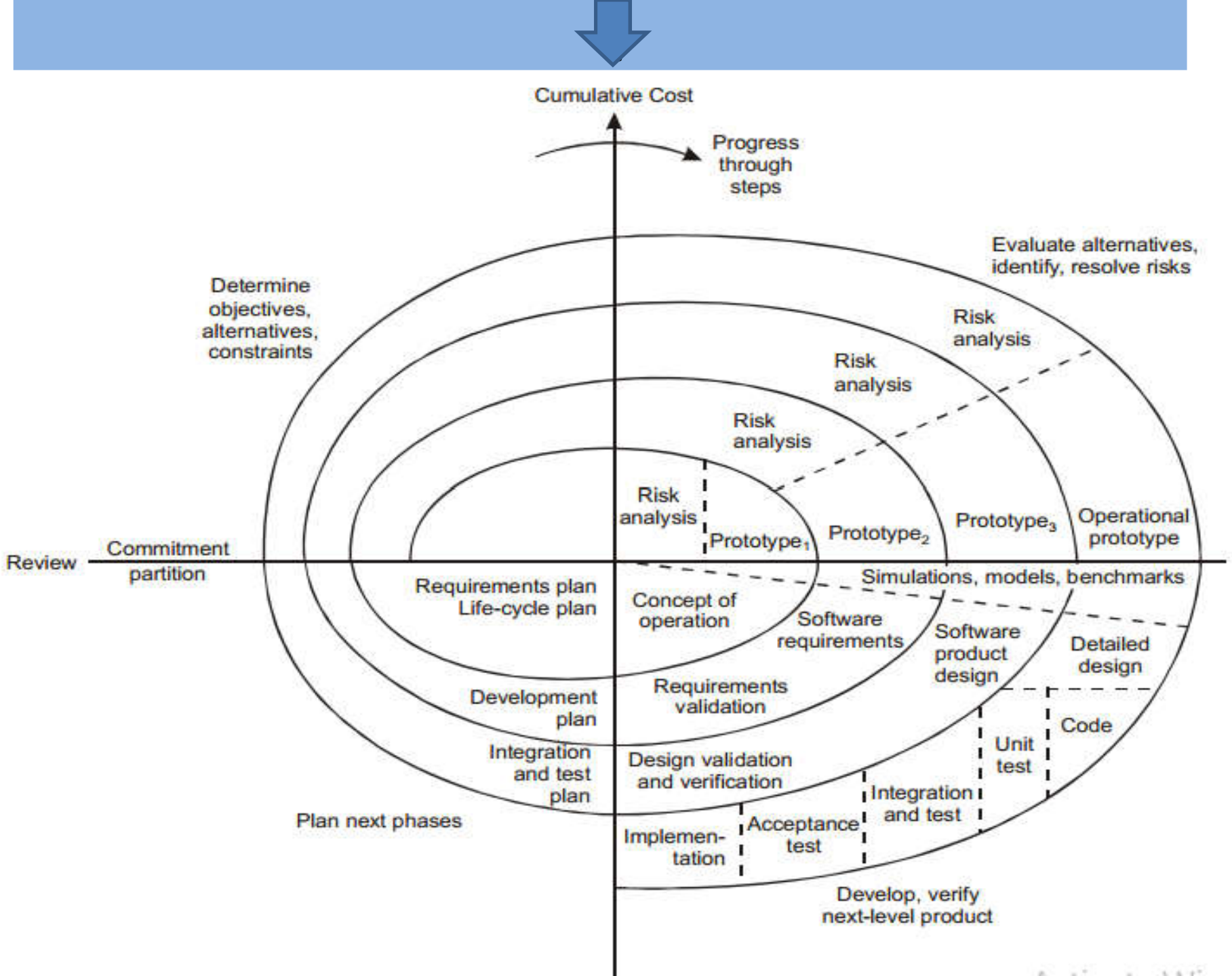
v. **Prototype refinement:** Once the user evaluates the prototype, it is refined according to the requirements. The developer revises the prototype to make it more effective and efficient according to the user requirement. If the user is not satisfied with the developed prototype, a new prototype is developed with the additional information provided by the user. The new prototype is evaluated in the same manner, as the previous prototype,process continues until all the requirements specified by the user are met. Once the user is satisfied a final system is developed.

vi. **Engineer product:** Once the requirements are completely known, user accepts the final prototype. The final system is thoroughly evaluated and tested followed by routine maintenance on continuing basis to prevent large-scale failures and to minimize downtime.

Advantages	Disadvantages
▪ Provides a working model to the user early in the process, enabling early assessment and increasing user confidence. ▪ Developer gains experience and insight by developing a prototype, thereby resulting in better implementation of requirements. ▪ Prototyping model serves to clarify requirements, which are not clear, hence reducing ambiguity and improving communication between developer and user. ▪ There is a great involvement of users in software development. Hence, the requirements of the users are met to the greatest extent. ▪ Helps in reducing risks associated with the project.	▪ If the user is not satisfied by the developed prototype, then a new prototype is developed. This process goes on until a perfect prototype is developed. Thus, this model is time consuming and expensive. ▪ Developer looses focus of the real purpose of prototype and compromise with the quality of the product. For example, they apply some of the inefficient algorithms or inappropriate programming languages used in developing the prototype. ▪ Prototyping can lead to false expectations. It often creates a situation where user believes that the development of the system is finished when it is not. ▪ The primary goal of prototyping is rapid development, thus, the design of system can suffer as it is built in a series of layers without considering integration of all the other components.

c. Spiral Model

In 1980's Boehm introduced a process model known as spiral model. The spiral model comprises of activities organized in a spiral, which has many cycles. This model combines the features of prototyping model and waterfall model and is advantageous for large, complex and expensive projects.


Cumulative Cost
Progress through steps
Evaluate alternatives, identify, resolve risks
Determine objectives, alternatives, constraints
Risk analysis
Risk analysis
Risk analysis
Risk analysis
Prototype1
Prototype2
Prototype3
Operational prototype
Review
Commitment partition
Requirements plan
Life-cycle plan
Concept of operation
Simulations, models, benchmarks
Software requirements
Software product design
Detailed design
Development plan
Requirements validation
Code
Unit test
Integration and test plan
Design validation and verification
Integration and test
Acceptance test
Implemen-tation
Plan next phases
Develop, verify next-level product

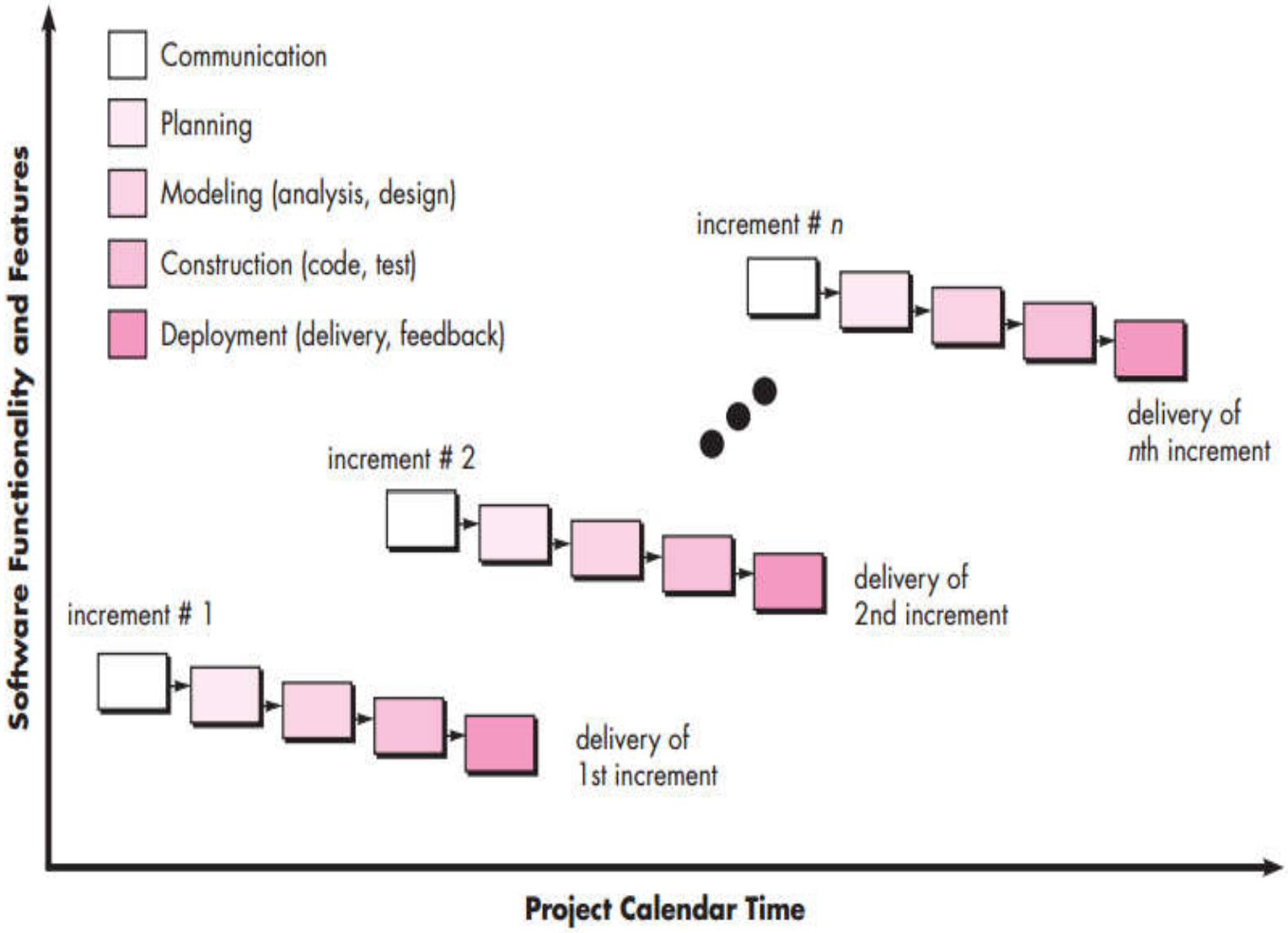
1. Each cycle of the **first quadrant** commences with identifying the goals for that cycle. In addition, it determines other alternatives, which are possible in accomplishing those goals.
2. Next step in the cycle evaluates alternatives based on objectives and constraints. This process identifies the project risks. Risk signifies that there is a possibility that the objectives of the project cannot be accomplished. If so the formulation of a cost effective strategy for resolving risks is followed. the strategy, which includes prototyping, simulation, benchmarking,.

3. The development of the software depends on remaining risks. The third quadrant develops the final software while considering the risks that can occur. Risk management considers the time and effort to be devoted to each project activity, such as planning, configuration management, quality assurance, verification, and testing.
4. The last quadrant plans the next step, and includes planning for the next prototype and thus,comprises of requirements plan, development plan, integration plan, and test plan

Advantages	Disadvantages
▪ Avoids the problems resulting in risk-driven approach in the software. ▪ Specifies a mechanism for software quality assurance activities. ▪ Spiral model is utilised by complex and dynamic projects. ▪ Re-evaluation after each step allows changes in user perspectives, technology advances or financial perspectives. ▪ Estimation of budget and schedule gets realistic as the work progresses.	▪ Assessment of project risks and its resolution is not an easy task. ▪ Difficult to estimate budget and schedule in the beginning, as some of the analysis is not done until the design of the software is developed.

d. Evolutionary model

- The evolution model **divides** the development cycle into **smaller, "Incremental Waterfall Model"** in which users are able to get access to the product at the end of each cycle.
- The users provide feedback on the product for planning stage of the next cycle and the development team responds, often by changing the product, plans or process.


Communication
Planning
Modeling (analysis, design)
Construction (code, test)
Deployment (delivery, feedback)
increment # n
delivery of
nth increment
increment # 2
delivery of
2nd increment
increment # 1
delivery of
1st increment
Software Functionality and Features
Project Calendar Time

Advantages of Evolutionary Model

- **Error reduction**: As the version is tested with customer which reduces the error thoroughly.
- **User satisfaction**: User gets satisfied and he gets the full chance of experimenting partially developed system.
- **Business benefit**: Successful use of this model can benefit not only business result but marketing and the internal operations as well.
- **High quality**: As you should get satisfied with every version, it produces the high quality product.

Disadvantages of Evolutionary Model

- **Several version release:** Developer has to make table version which increases their Efforts.
- **Dividing software:** It is difficult to "divide the software and the problems in several **versions** that would be acceptable to the customer which can be implemented and delivered incrementally.
- **Confusion by several version**: An user might get "confused by several versions of the software. It will affect on the final product.

- **Uncertain nature of customer needs:** A confused user has uncertainty over his requirements, so giving him several version may change his requirement Rapidly.

- **Time And Cost:** As this model reduces "Time And Cost" but requirement is not gathered correctly. It will subsequently time, cost and efforts.

Waterfall model: it can serve as a useful process model in situations where requirements are fixed and work is to proceed to completion in a linear manner.

Incremental model: our customer demands delivery by a date that is impossible to meet. Suggest delivering one or more increments by that date and the rest of the software(additional increment)later.

For example, word-processing software developed using the incremental paradigm might deliver basic file management, editing, and document production functions in the **first increment**; more sophisticated editing and document production capabilities in the **second increment**; spelling and grammar checking in the **third increment**; and advanced page layout capability in the **fourth increment.**

Prototype model: When your customer has a legitimate need, but is clueless about the details, develop a prototype as a first step.

Ex: Human interface layout or output display format.

Spiral model: If your management demands fixed-budge development (generally a bad idea), the spiral can be a problem. As each circuit is completed, project cost is revisited and revised.

Ex: Banking software, business software like complex project mostly use this model .

!! Attendance please !!

Thank You!!!