Software Engineering-BSCE-301L

Module 1:

Overview of Software Engineering

Dr . Saurabh Agrawal

Faculty Id: 20165

School of Computer Science and Engineering

VIT, Vellore-632014

Tamil Nadu, India

Outline

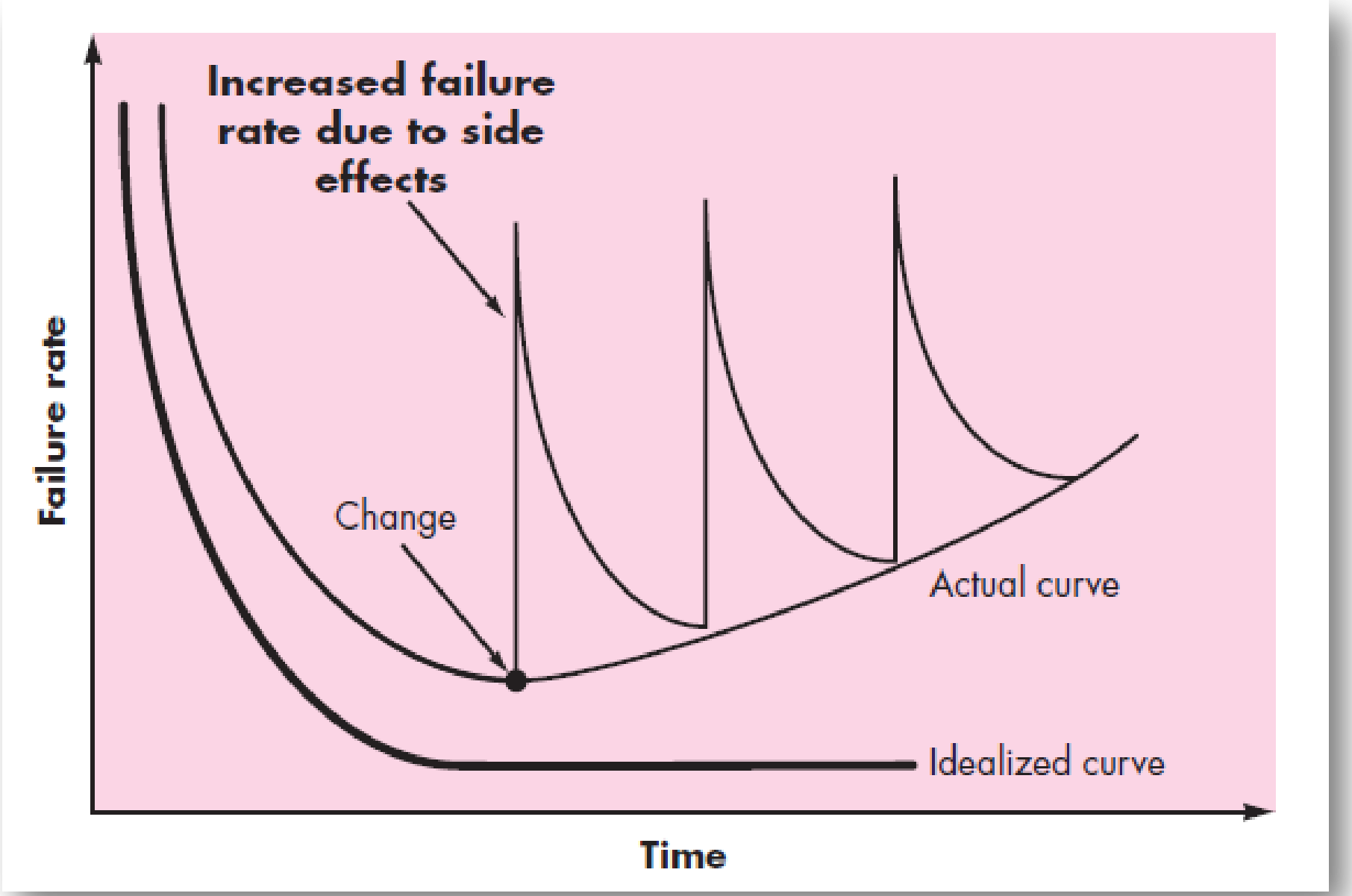
- Nature of Software (L2)
- Introduction to Software Engineering (L2)
- Software Process, Project, Product (L2)
- Process Models (Waterfall, Incremental, RAD) (L3)
- Classical Evolutionary Models (Prototype, Spiral, and Concurrent) (L4)
- Introduction to Agility and Agile Process (L5)
- Principles of Agile Software Development framework (L6)
- Extreme programming (L6)
- XP Process (L7)
- Overview of System Engineering (L7)

Nature of Software

- The software is instruction or computer program that when executed provide desired features, function, and performance.
- A data structure that enables the program to adequately manipulate information and document that describe the operation and use of the program.
- There are some **characteristics** of software as aforementioned.

1. **Functionality**
2. **Reliability**
3. **Usability**
4. **Efficiency**
5. **Maintainability**
6. **Portability**

Nature of Software



Nature of Software

Changing Nature of Software:

❑Following are the **Seven Broad Categories** of software are challenges for software engineers:

1. System Software: is a collection of programs written to service other programs. System software: such as compilers, editors, file management utilities.
2. Application Software: stand-alone programs for specific needs. This software are used to controls business needs. Ex: Transaction processing.
3. Embedded Software: This software resides within a system or product and it is used to implement and control features and functions from end user and system itself. This software performs limited function like keypad control for microwave oven.
4. Artificial Intelligence Software: makes use of nonnumeric algorithms to solve complex problems. Application within this area include robotics, pattern recognition, game playing.

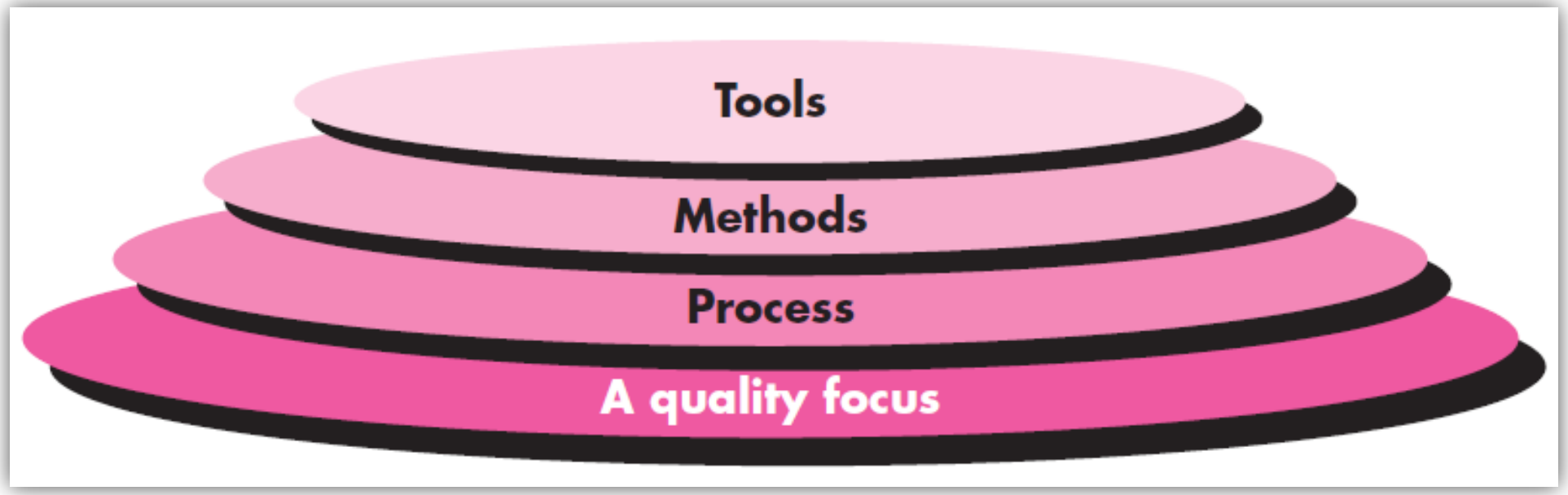
Changing Nature of Software:

5. Engineering and Scientific Software: Engineering and scientific software have been characterized by "number crunching" algorithm.
6. Product-line Software: focus on a limited marketplace to address mass consumer market. (word processing, graphics, database management) .
7. WebApps (Web applications) Network Centric Software: As web 2.0 emerges, more sophisticated computing environments is supported integrated with remote database and business applications.

What is Software Engineering?

- The term software engineering is the product of two words, software, and engineering.
- The software is a collection of integrated programs.
- Software subsists of carefully-organized instructions and code written by developers on any of various particular computer languages.
- Computer programs and related documentation such as requirements, design models and user manuals.
- Engineering is the application of scientific and practical knowledge to invent, design, build, maintain, and improve frameworks, processes, etc.


Deployment
Updates
Testing
Maintenance
Software Product
Code Design
System Design
System Analysis
Requirements


Tools
Methods
Process
A quality focus

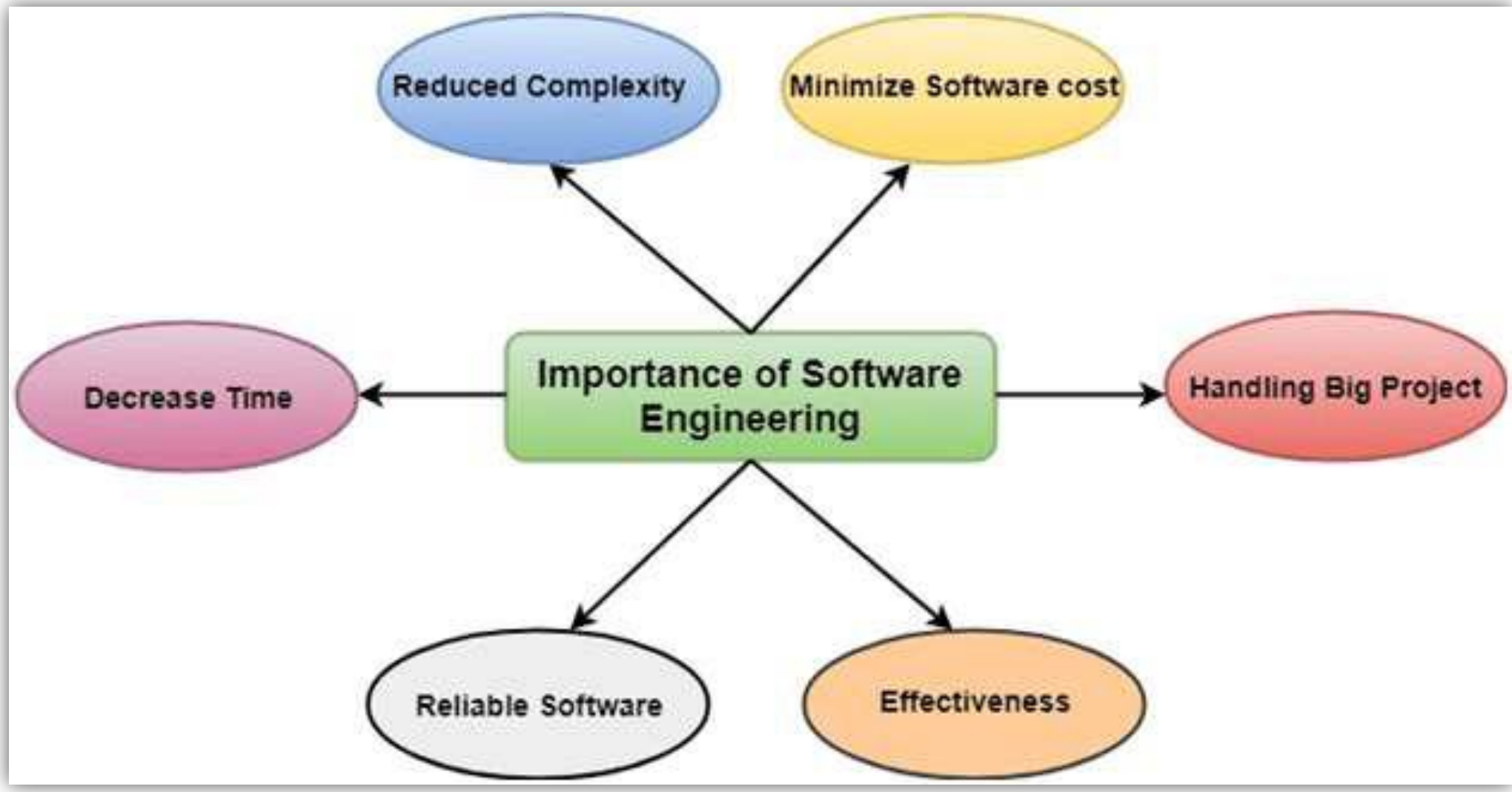
Why is Software Engineering required?

- Software Engineering is required due to the following reasons:
  - **To manage Large applications using software**
  - **For more Scalability**
  - **Cost Management**
  - **To manage the dynamic nature of software**
  - **For better quality Management**

Need of Software Engineering

- The necessity of software engineering appears because of a higher rate of progress in user requirements and the environment on which the program is working.
  - **Huge Programming:** It is simpler to manufacture a wall than to a house or building, similarly, as the measure of programming become extensive engineering has to step to give it a scientific process.
  - **Adaptability:** If the software procedure were not based on scientific and engineering ideas, it would be simpler to re-create new software than to scale an existing one.
  - **Cost:** As the hardware industry has demonstrated its skills and huge manufacturing has let down the cost of computer and electronic hardware, but cost of programming remains high if the proper process is not adapted.
  - **Dynamic Nature:** The continually growing and adapting nature of programming hugely depends upon the environment in which the client works. If the quality of the software is continually changing, new upgrades need to be done in the existing one.
  - **Quality Management:** Better procedure of software development provides a better and quality software product.

Importance of Software Engineering



Importance of Software Engineering

1.Reduces Complexity: Big software is always complicated and challenging to progress. Software engineering has a great solution to reduce the complication. Software engineering divides big problems into various small issues. And then start solving each small issue one by one. All these small problems are solved independently to each other.

2.To Minimize Software Cost: A lot of manpower is required to develop software with a large number of codes. But in software engineering, programmers project everything and decrease all those things that are not needed. In turn, the cost for software productions becomes less as compared to any software that does not use software engineering method.

3.To Decrease Time: Anything that is not made according to the project always wastes time. And if you are making great software, then you may need to run many codes to get the definitive running code. This is a very time-consuming procedure, and if it is not well handled, then this can take a lot of time. So if you are making your software according to the software engineering method, then it will decrease a lot of time.

Importance of Software Engineering

4.Handling Big Projects: Big projects are not done in a couple of days, and they need lots of patience, planning, and management. And to invest six and seven months of any company, it requires heaps of planning, direction, testing, and maintenance. No one can say that he has given four months of a company to the task, and the project is still in its first stage. Because the company has provided many resources to the plan and it should be completed. So to handle a big project without any problem, the company has to go for a software engineering method.

5.Reliable Software: Software should be secure, means if you have delivered the software, then it should work for at least its given time or subscription. And if any bugs come in the software, the company is responsible for solving all these bugs. Because in software engineering, testing and maintenance are given, so there is no worry of its reliability.

6.Effectiveness: Effectiveness comes if anything has made according to the standards. Software standards are the big target of companies to make it more effective.

Question: What is the difference between “Program” and “Software”?

Software Process

- 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 (communication with stakeholders) and is applied regardless of the application domain, size of the project, complexity of the effort, or degree of rigor with which software engineering is to be applied.
- An **action** (architectural design) encompasses a set of tasks that produce a major work product (an architectural design model).
- A **task** focuses on a small, but well-defined objective that produces a tangible outcome.
- In the context of software engineering, a process is *not a rigid prescription for how* to build computer software.
- Rather, it is an adaptable approach that enables the people doing the work (the software team) to pick and choose the appropriate set of work actions and tasks.
- The intent is always to deliver software in a timely manner and with sufficient quality to satisfy those who have sponsored its creation and those who will use it.

Software Process

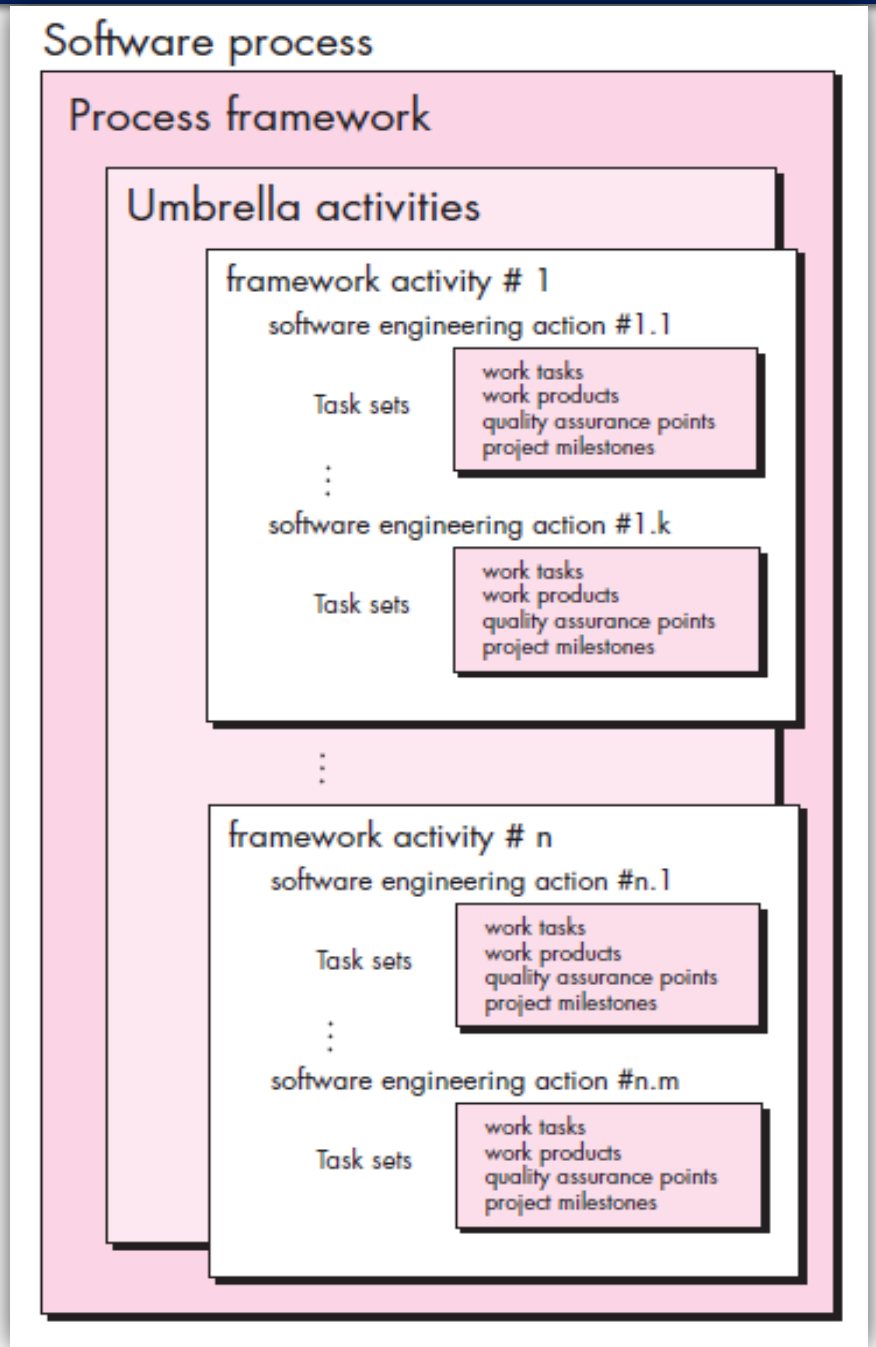
❑A **process framework** establishes the foundation for a complete software engineering process by identifying a small number of framework activities that are applicable to all software projects, regardless of their size or complexity.

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

❑A generic process framework for software engineering encompasses five activities:

1. **Communication**
2. **Planning**
3. **Modeling**
4. **Construction**
5. **Deployment**

Software Process



Software Process

1. **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.
2. **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.
3. **Modeling:** 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 and the design that will achieve those requirements.

4. **Construction:** This activity combines code generation (either manual or automated) and the testing that is required to uncover errors in the code.
5. **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.

Software Process

- Software engineering process framework activities are complemented by a number of umbrella activities.
- In general, umbrella activities are applied throughout a software project and help a software team manage and control progress, quality, change, and risk.
- Typical umbrella activities include:

1. **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.
2. **Risk Management:** Assesses risks that may affect the outcome of the project or the quality of the product.
3. **Software Quality Assurance:** Defines and conducts the activities required to ensure software quality.
4. **Technical reviews:** Assesses software engineering work products in an effort to uncover and remove errors before they are propagated to the next activity.

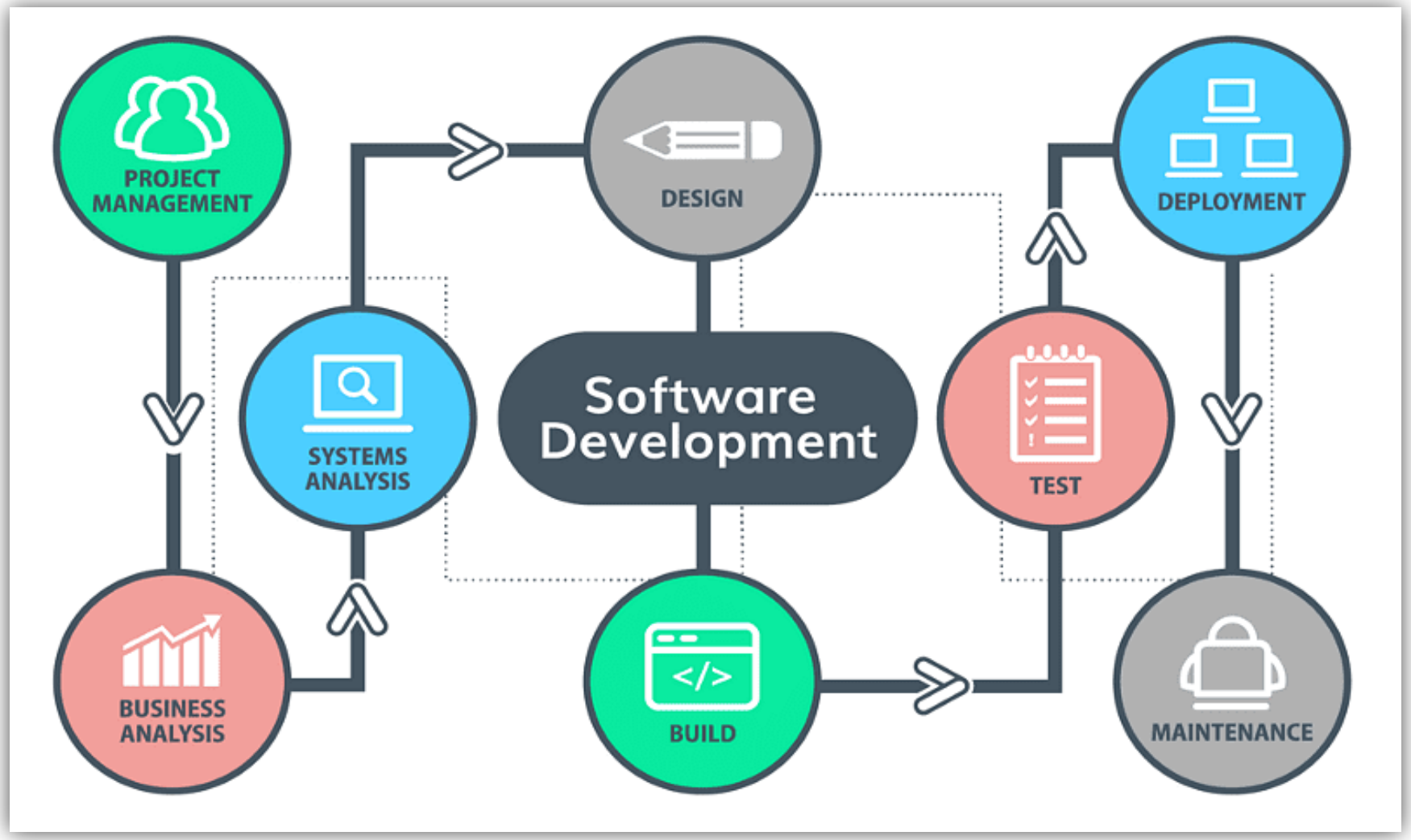
Software Process

5. **Measurement:** Defines and collects process, project, and product measures that assist the team in delivering software that meets stakeholders' needs; can be used in conjunction with all other framework and umbrella activities.
6. **Software Configuration Management:** Manages the effects of change throughout the software process.
7. **Reusability Management:** Defines criteria for work product reuse (including software components) and establishes mechanisms to achieve reusable components.
8. **Work Product Preparation and Production:** Encompasses the activities required to create work products such as models, documents, logs, forms, and lists.

Software Project

- A Software Project is the complete procedure of software development from requirement gathering to testing and maintenance, carried out according to the execution methodologies, in a specified period of time to achieve intended software product.
- Also known as a software project comprises the steps involved in making a product before it is actually available to the market.
- The project can be handled by people which are as less as one person to the involvement of a lot of people.
- These are usually assigned by an enterprise and are undertaken to form a new product that has not already been made.

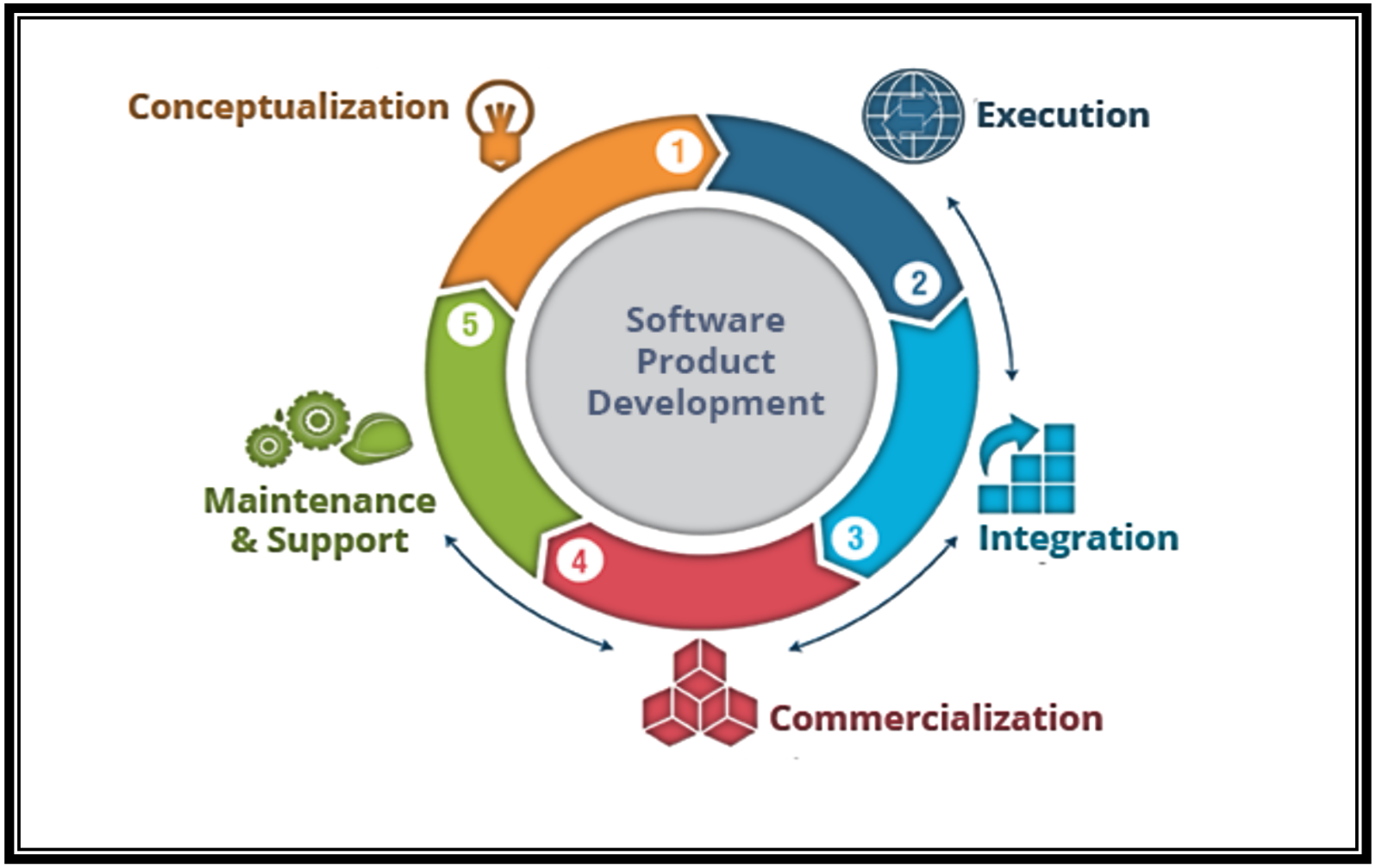
Software Project



Software Product

- The product is the final outcome of the software development process.
- It is the software application or system that is being built or maintained.
- A product is built on the customer's requirements/requests.
- In the context of software development, a product consists of several components such as requirement specifications, design and test documents, user manuals, etc.
- To develop a desired software product, the software development team and the customer should define the purpose and scope of the product.
- The "purpose" gives the information about the objective of the product, and the "scope" gives the details about fundamental data, functions, and behavior of the product.

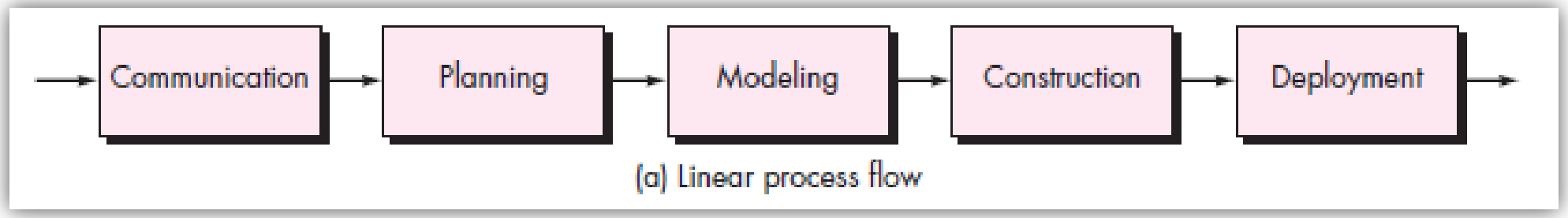
Software Product



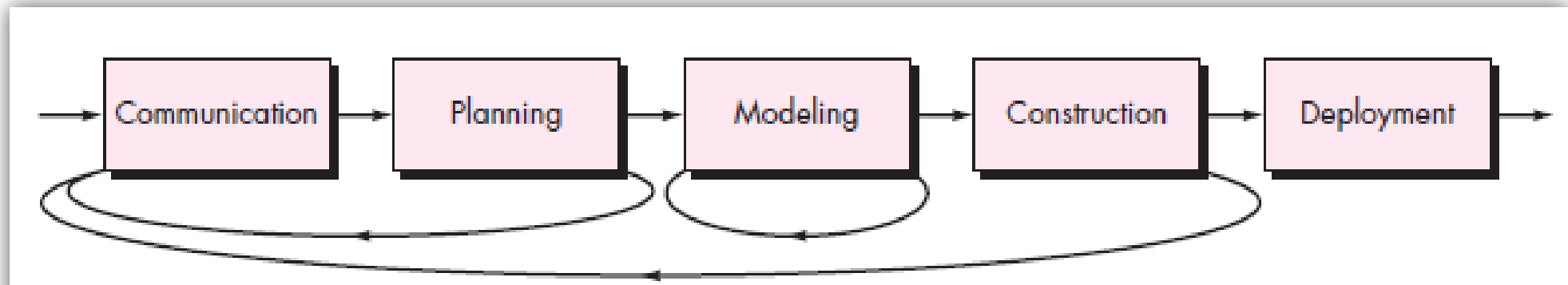
Process Models

- A generic process framework for software engineering defines five framework activities—**communication, planning, modeling, construction, and deployment.**
- Important aspect of the software process has been process flow.
- Process flow describes how the framework activities and the actions and tasks that occur within each framework activity are organized with respect to sequence and time.
- A **linear process** flow executes each of the five framework activities in sequence, beginning with communication and culminating with deployment.
- An **iterative process** flow repeats one or more of the activities before proceeding to the next.
- An **evolutionary process** flow executes the activities in a “circular” manner. Each circuit through the five activities leads to a more complete version of the software.
- A **parallel process** flow executes one or more activities in parallel with other activities (e.g., modeling for one aspect of the software might be executed in parallel with construction of another aspect of the software).

Process Models

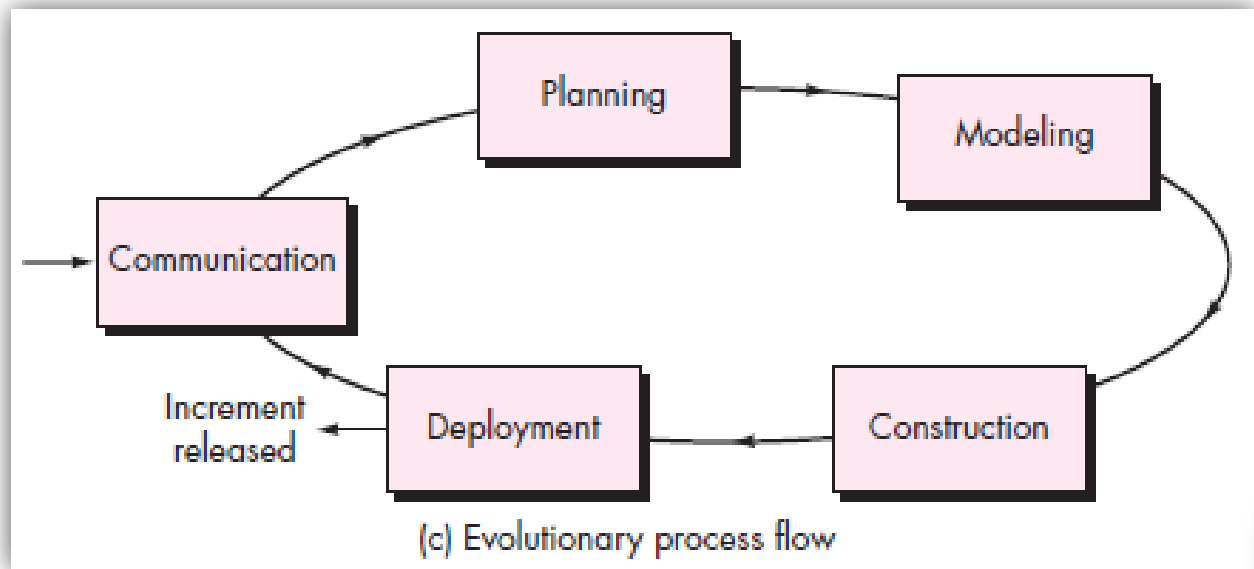


(a) Linear process flow

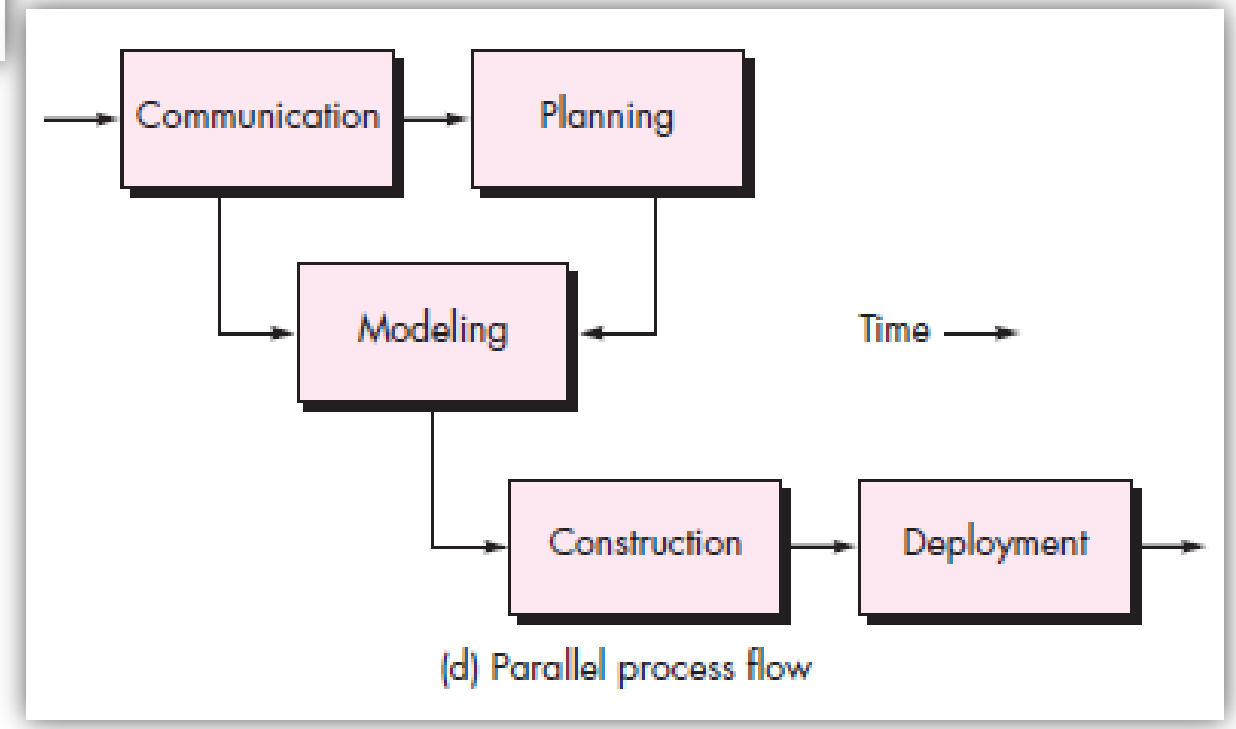


(b) Iterative process flow

Process Models



(c) Evolutionary process flow



(d) Parallel process flow

Perspective Process Models (Waterfall, Incremental, RAD)

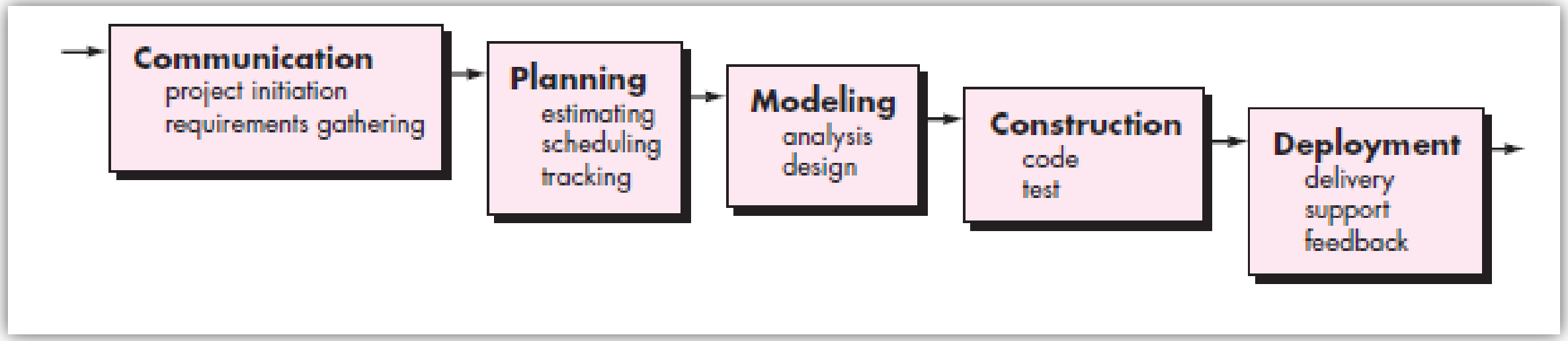
- Prescriptive process models were originally proposed to bring order to the chaos of software development.
- History has indicated that these traditional models have brought a certain amount of useful structure to software engineering work and have provided a reasonably effective road map for software teams.
- However, software engineering work and the product that it produces remain on “the edge of chaos.”
- All software process models can accommodate the generic framework activities.
- But each applies a different emphasis to these activities and defines a process flow that invokes each framework activity (as well as software engineering actions and tasks) in a different manner.
- Following are some popular perspective process models:

1. **Waterfall**
2. **Incremental**
3. **Rapid Application Development (RAD)**

Perspective Process Models (Waterfall-V1)

- There are times when the requirements for a problem are well understood when work flows from **communication through deployment in a reasonably linear fashion.**
- This situation is sometimes encountered when well-defined adaptations or enhancements to an existing system must be made.
- It may also occur in a limited number of new development efforts, but only when requirements are well defined and reasonably stable.
- The waterfall model, sometimes called the classic life cycle, suggests a systematic, sequential approach to software development that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment, culminating in ongoing support of the completed software.

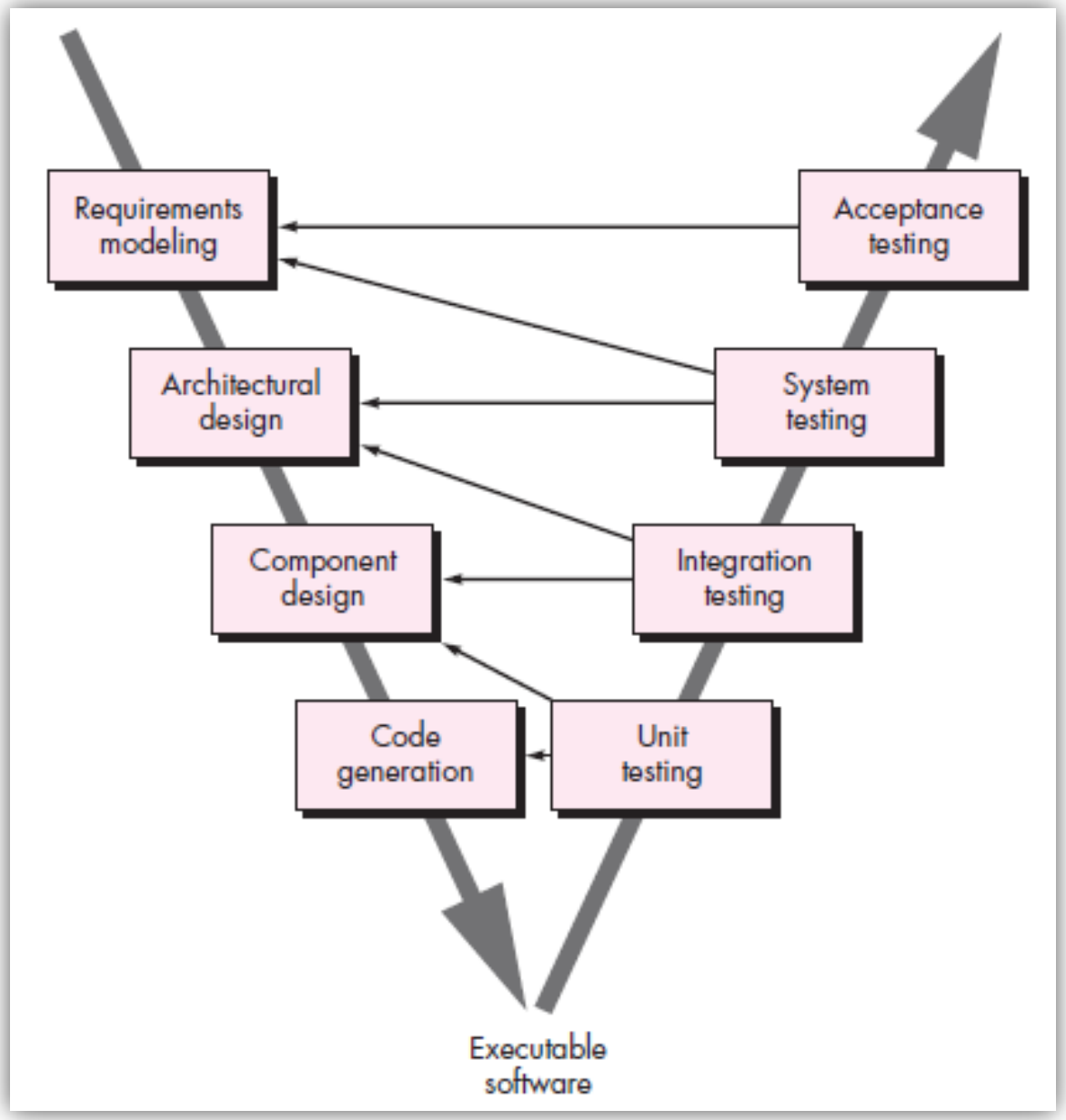
Perspective Process Models (Waterfall-V1)



Perspective Process Models (Waterfall-V1)

- A variation in the representation of the waterfall model is called the *V-model.*
- V-model 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 (quality assurance actions) 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 V-model.
- The V-model provides a way of visualizing how verification and validation actions are applied to earlier engineering work.

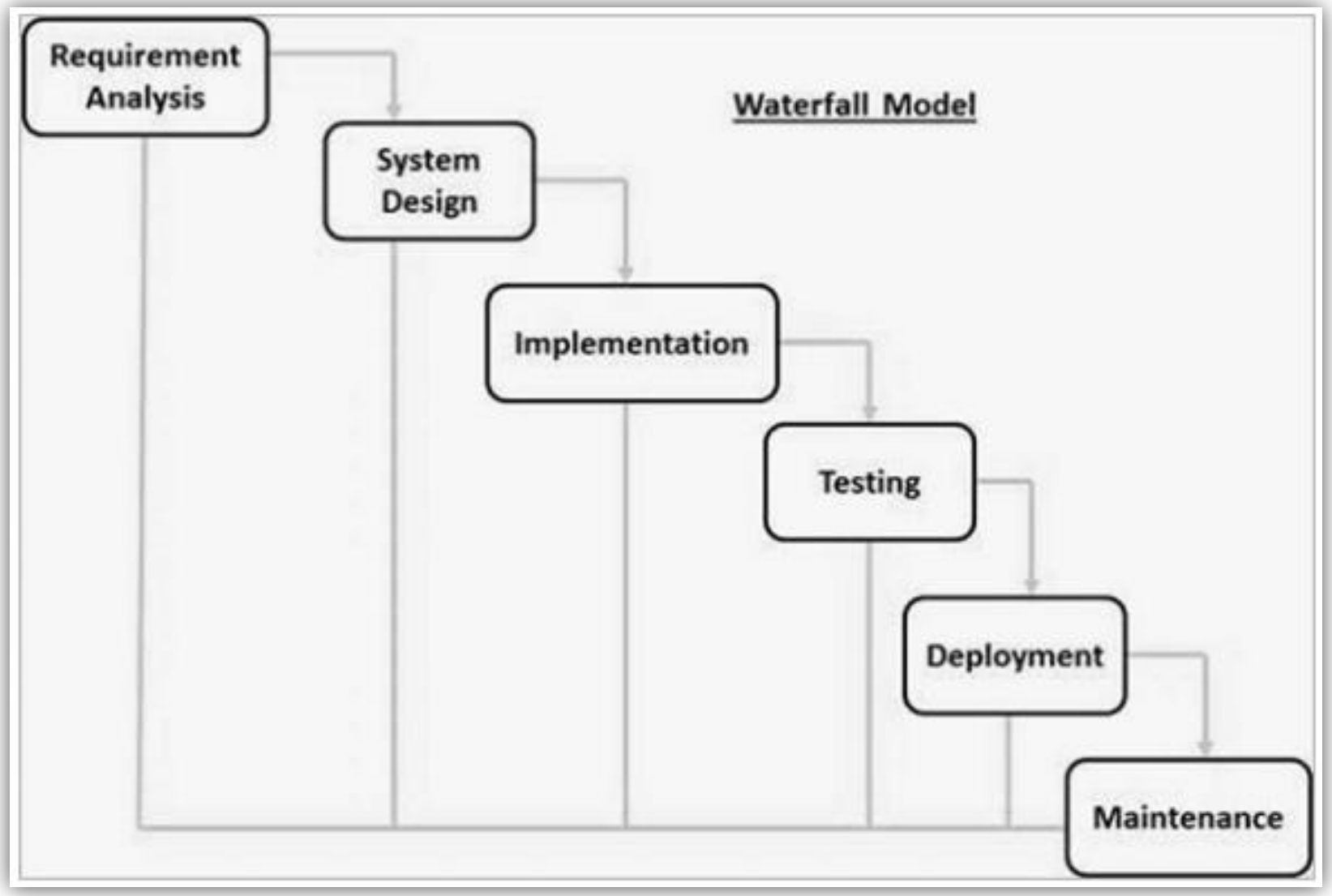
Perspective Process Models (Waterfall-V1)



Perspective Process Models (Waterfall-V2)

- The Waterfall Model was the first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**.
- It is very simple to understand and use.
- In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.
- The Waterfall model is the earliest SDLC approach that was used for software development.
- The waterfall Model illustrates the software development process in a linear sequential flow.
- This means that any phase in the development process begins only if the previous phase is complete.
- In this waterfall model, the phases do not overlap.

Perspective Process Models (Waterfall-V2)



Perspective Process Models (Waterfall-V2)

1. **Requirement Gathering and Analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
2. **System Design:** The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
3. **Implementation:** With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
4. **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

Perspective Process Models (Waterfall-V2)

5. **Deployment of system:** Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
6. **Maintenance:** There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Waterfall Model: Application

- Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors.
- Some situations where the use of Waterfall model is most appropriate are:

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Perspective Process Models (Waterfall-V2)

❑ Advantages

- Easy to manage due to the rigidity of the model.
- Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.
- Simple and easy to understand and use

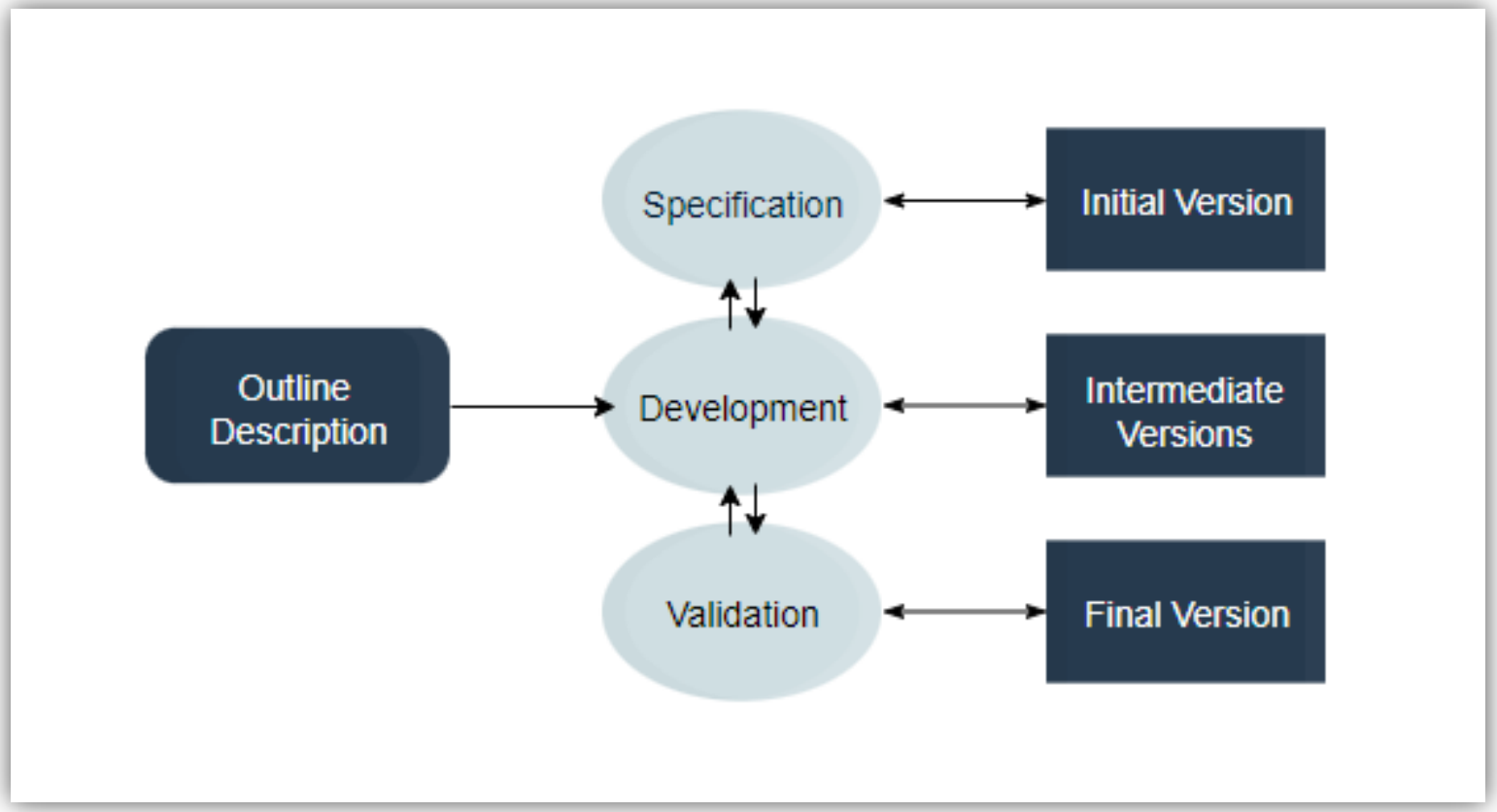
❑ Disadvantages

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.

Perspective Process Models (Incremental)

- The incremental model divides the system's functionality into **small increments** that are delivered one after the other in quick succession.
- The most important functionality is implemented in the initial increments.
- The subsequent increments expand on the previous ones until everything has been updated and implemented.
- Incremental development is based on developing an initial implementation, exposing it to user feedback, and evolving it through new versions.
- The process' activities are interwoven by feedback.
- Each iteration passes through the requirements, design, coding, and testing stages.
- The incremental model lets stakeholders and developers see results with the first increment. If the stakeholders don't like anything, everyone finds out a lot sooner.
- It is efficient as the developers only focus on what is important and bugs are fixed as they arise, but you need a **clear and complete definition** of the whole system before you start.
- The incremental model is great for projects that have loosely-coupled parts and projects with complete and clear requirements.

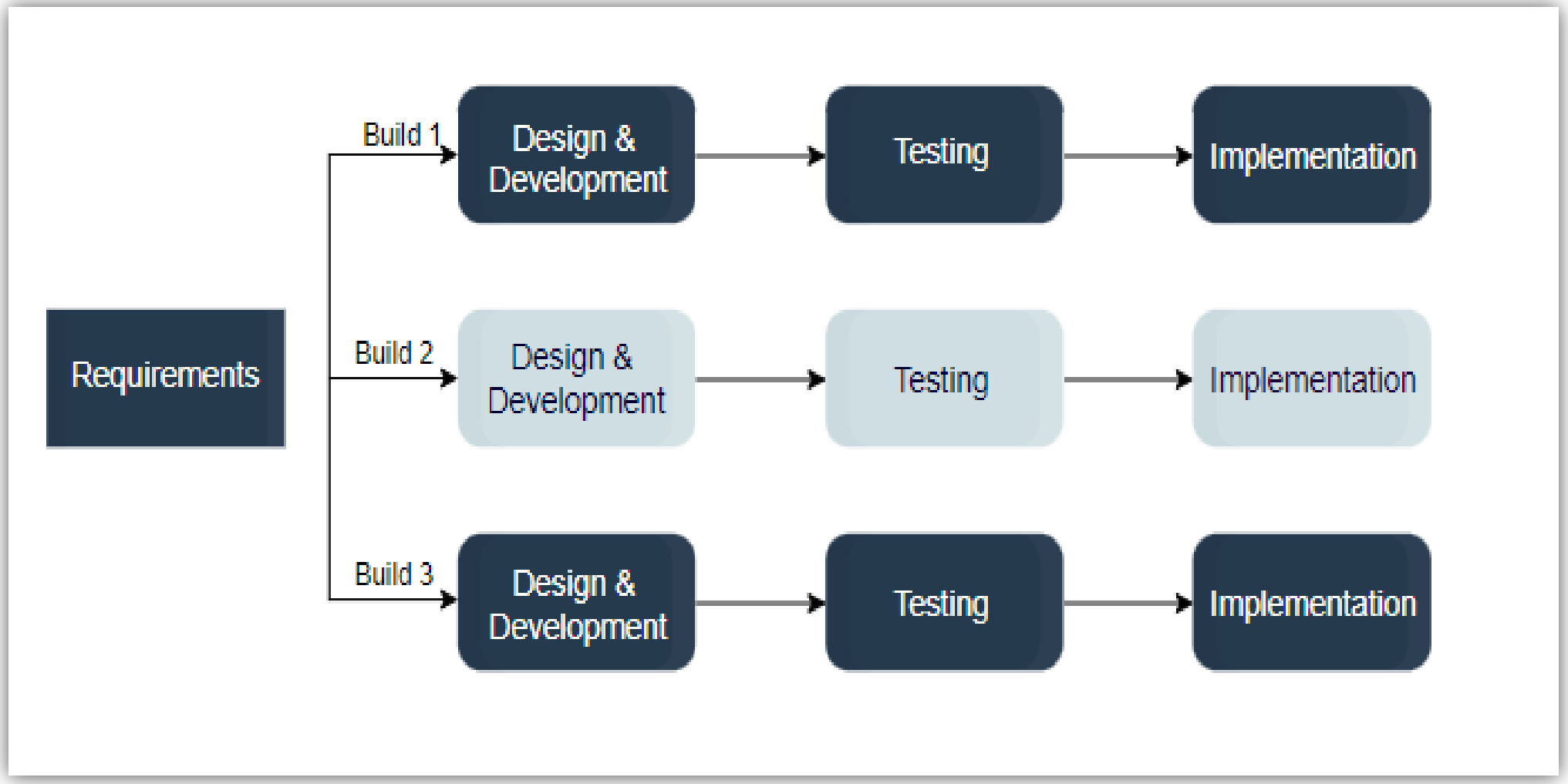
Perspective Process Models (Incremental)



Perspective Process Models (Incremental-Iterative Model-Version Control)

- The iterative development model develops a system through **building small portions** of all the features.
- This helps to meet initial scope quickly and release it for feedback.
- In the iterative model, you start off by implementing a small set of the software requirements.
- These are then **enhanced iteratively** in the evolving versions until the system is completed.
- This process model starts with part of the software, which is then implemented and reviewed to identify further requirements.
- Like the incremental model, the iterative model allows you to see the results at the early stages of development.
- This makes it easy to identify and **fix any functional or design flaws**. It also makes it easier to manage risk and change requirements.
- The deadline and budget may change throughout the development process, especially for large complex projects.
- The iterative model is a good choice for large software that can be **easily broken down into modules**.

Perspective Process Models (Incremental-Iterative Model-Version Control)



Perspective Process Models (Incremental-Iterative Model-Version Control)

- **Various phases of incremental model are as follows:**

1. Requirement analysis: In this phase, product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team.

2. Design & Development: In this phase, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

3. Testing: In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task.

4. Implementation: Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

Perspective Process Models (Incremental-Iterative Model-Version Control)

❑ When to use Incremental Model:

- When the requirements are superior.
- A project has a lengthy development schedule.
- When Software team are not very well skilled or trained.
- When the customer demands a quick release of the product.
- You can develop prioritized requirements first.

Perspective Process Models (Incremental-Iterative Model-Version Control)

❑Advantages:

- Errors are easy to be recognized.
- Easier to test and debug
- More flexible.
- Simple to manage risk because it handled during its iteration.
- The Client gets important functionality early.

❑Disadvantages:

- Need for good planning
- Total Cost is high.
- Well defined module interfaces are needed.

Perspective Process Models (RAD → Rapid Application Development)

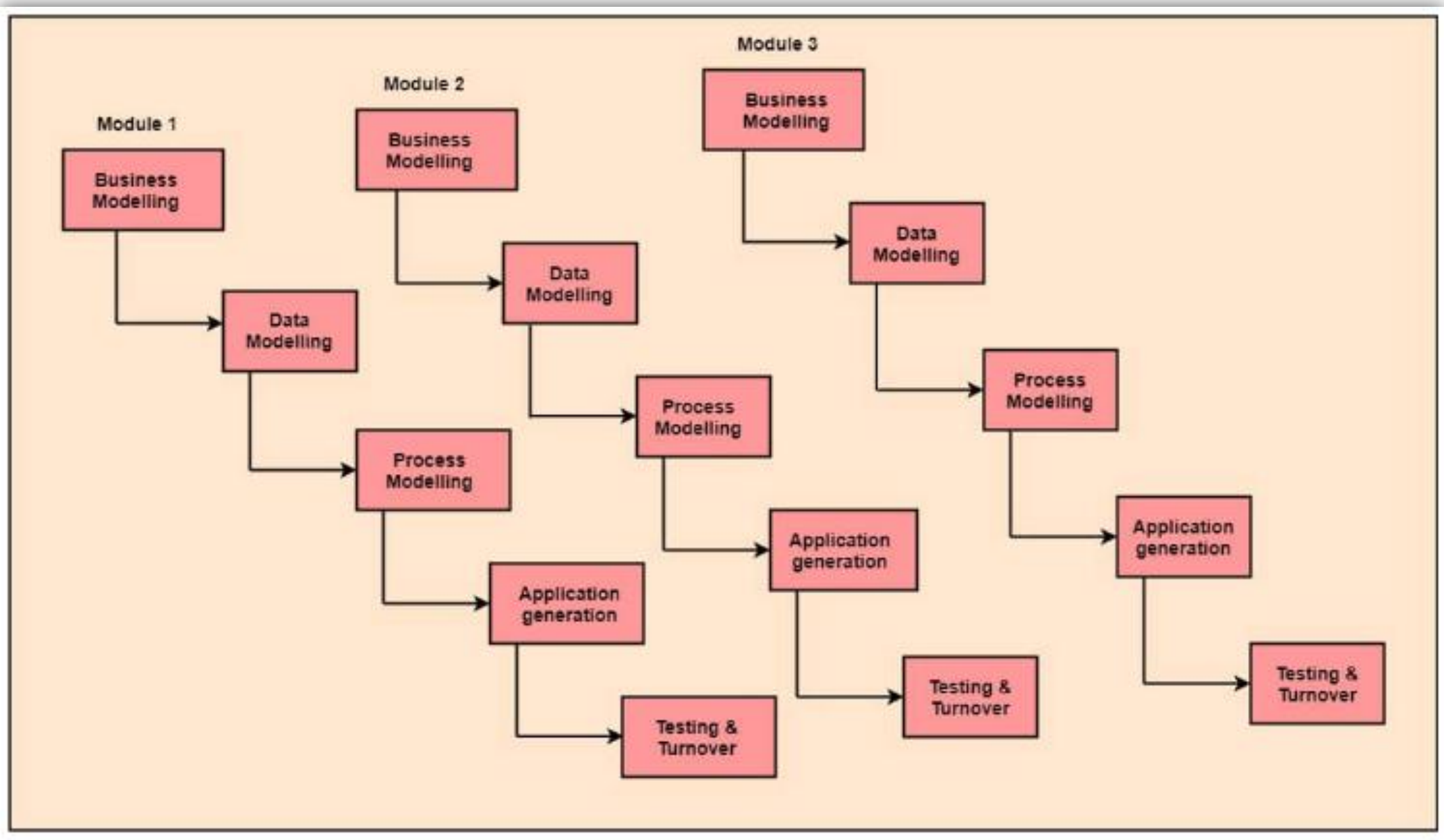
❑RAD is a linear sequential software development process model that emphasizes a concise development cycle using an element based construction approach.

❑If the requirements are well understood and described, and the project scope is a constraint, the RAD process enables a development team to create a fully functional system within a concise time period.

❑RAD (Rapid Application Development) is a concept that products can be developed faster and of higher quality through:

- Gathering requirements using workshops or focus groups
- Prototyping and early, reiterative user testing of designs
- The re-use of software components
- A rigidly paced schedule that refers design improvements to the next product version
- Less formality in reviews and other team communication

Perspective Process Models (RAD → Rapid Application Development)



Perspective Process Models (RAD → Rapid Application Development)

❑Various phases of RAD are as follows:

1.Business Modelling: Information flow among business functions is defined by answering questions like what data drives the business process, what data is generated, who generates it, where does the information go, who process.

2. Data Modelling: Data collected from business modeling is refined into a set of data objects that are needed to support the business. The attributes are identified, and the relation between these data objects is defined.

3. Process Modelling: The information object defined in the data modeling phase are transformed to achieve the data flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.

4. Application Generation: Automated tools are used to facilitate construction of the software; even they use the 4th GL techniques.

5.Testing & Turnover: Many of the programming components have already been tested since RAD emphasis reuse. This reduces the overall testing time. But the new part must be tested, and all interfaces must be fully exercised.

Perspective Process Models (RAD → Rapid Application Development)

❑When to use RAD Model:

- When the system should need to create the project that modularizes in a short span time (2-3 months).
- When the requirements are well-known.
- When the technical risk is limited.
- When there's a necessity to make a system, which modularized in 2-3 months of period.
- It should be used only if the budget allows the use of automatic code generating tools.

Perspective Process Models (RAD → Rapid Application Development)

❑Advantages:

- This model is flexible for change.
- In this model, changes are adoptable.
- Each phase in RAD brings highest priority functionality to the customer.
- It reduced development time.
- It increases the reusability of features.

❑Disadvantages:

- It required highly skilled designers.
- All application is not compatible with RAD.
- For smaller projects, we cannot use the RAD model.
- On the high technical risk, it's not suitable.
- Required user involvement.

Note for Students

- This power point presentation is for lecture, therefore it is suggested that also utilize the text books and lecture notes.