UNIT-1 INTRODUCTION TO SOFTWARE AND SOFTWARE ENGINEERING

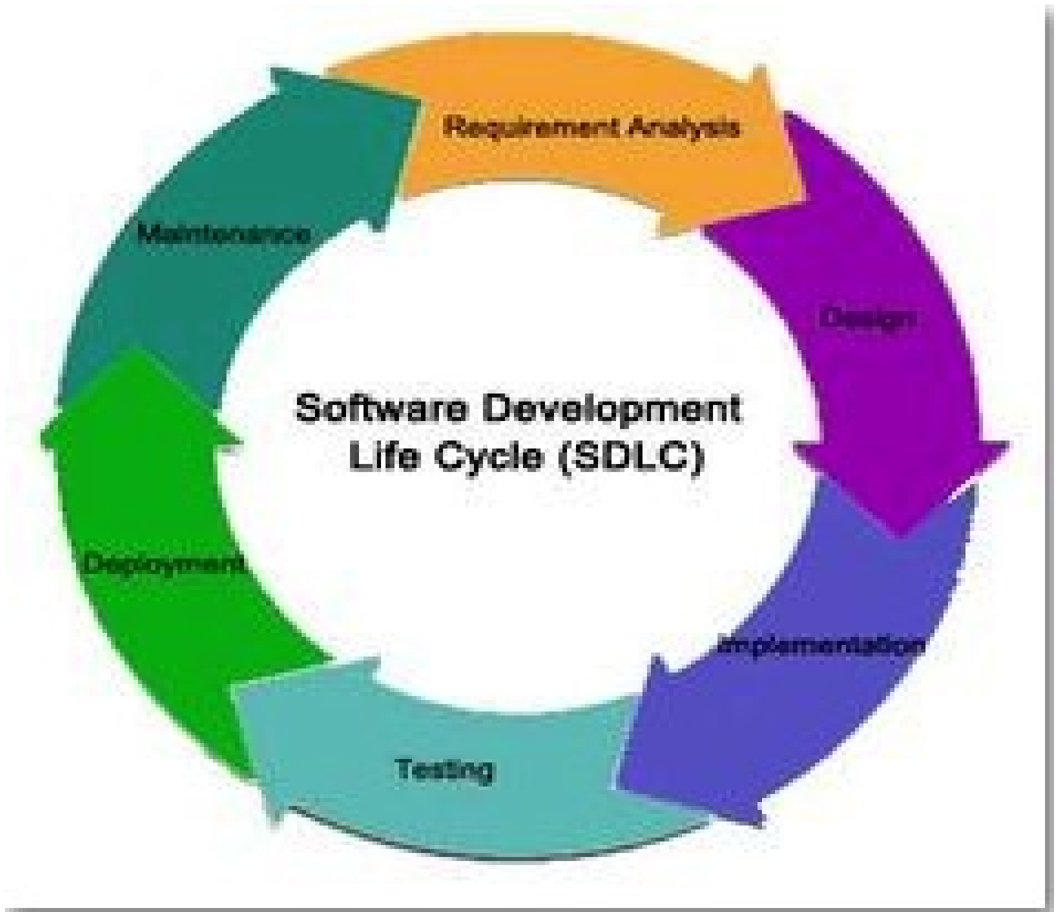
Part-2

Software Process Model

- Software Process Models are simplified and abstract description of a software process that presents one view of that process.
- The Software Process is the set of activities and associated results that produce a software product.

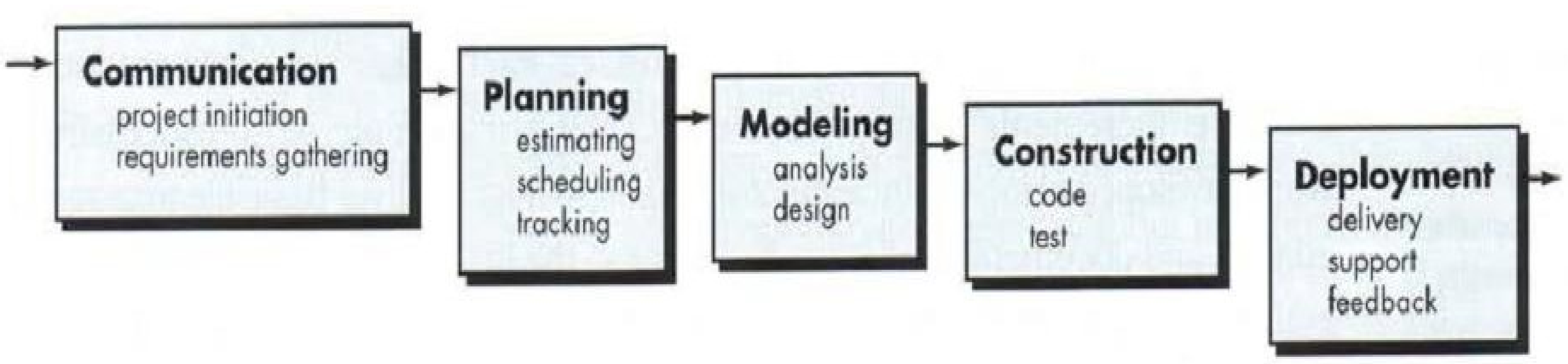
SDLC : Software Development Life Cycle

Software Development Life Cycle, SDLC for short, is a well-defined, structured sequence of stages in software engineering to develop the intended software product.



Waterfall Model

- The earliest SDLC approach that was used for software development
- Also known as Classic Life Cycle or Linear-Sequential Approach
- All the phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases.
- Each phase must be completed before the next phase can begin and there is no overlapping in the phases.



Waterfall Model

Applications / Used when:

- Requirements are very well documented, clear and fixed
- Product definition is stable
- Technology is understood and is not dynamic
- Ample resources with required expertise are available to support the product
- The project is short

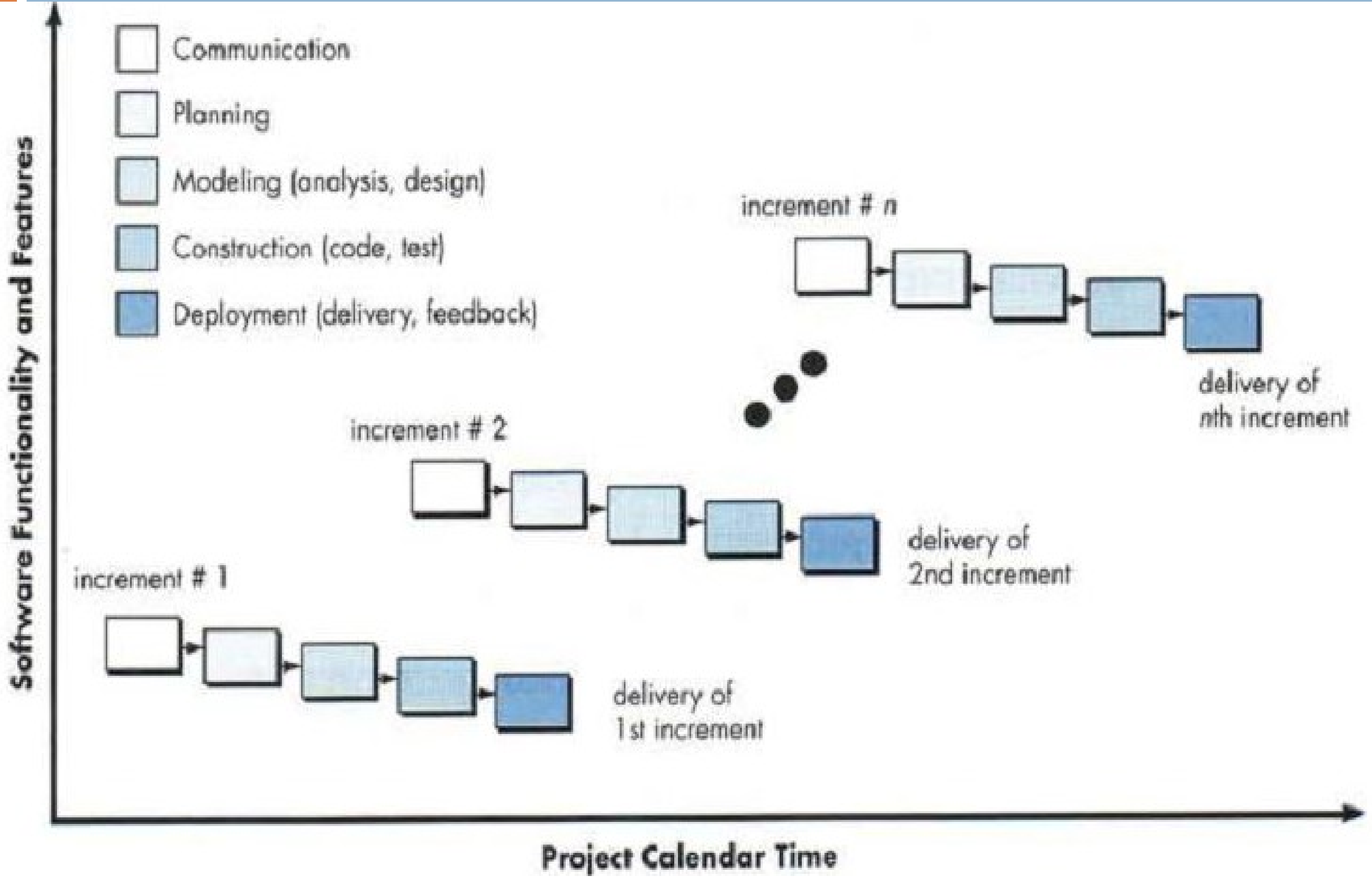
Waterfall Model

- **Advantages:**
  - Simple and easy to understand and use.
  - Easy to manage due to the rigidity of the model
  - Works well when quality is more important than cost or schedule
  - Easy to arrange tasks.
  - Clearly defined stages.: Provides structure to inexperienced staff
- **Disadvantages**
  - Cannot accommodate changing requirements.
  - Poor model for long and ongoing projects.
  - It is difficult to measure progress within stages
  - High amounts of risk and uncertainty.
  - All requirements must be known upfront

Incremental Model

- Also known as iterative model
- Iterative process starts with a simple implementation of a subset of the software requirements and iteratively enhances the evolving versions until the full system is implemented.
- During each iteration, the development module goes through the requirements, design, implementation and testing phases. Each subsequent release of the module adds function to the previous release.
- The process continues till the complete system is ready as per the requirement.
- The basic idea behind this method is to develop a system through repeated cycles (iterative) and in smaller portions at a time (incremental).
- During software development, more than one iteration of the software development cycle may be in progress at the same time.
- The first increment is called the “core product” i.e. basic requirements are addressed but other functionalities are unknown or undelivered.

Incremental Model


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

Incremental Model

- **Applications / Used when:**
- Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
- A need to get basic functionality to the market early
- A new technology is being used and is being learnt by the development team while working on the project.
- Resources with needed skill set are not available and are planned to be used on contract basis for specific iterations.

Incremental Model

- **Advantages:**
  - Some working functionality can be developed quickly and early in the life cycle. Hence customers get important functionality early
  - Parallel development can be planned.
  - Less costly to change the scope/requirements.
  - Testing and debugging during smaller iteration is easy
  - With every increment operational product is delivered.
  - It supports changing requirements.
  - Initial Operating time is less.
  - Better suited for large and mission-critical projects.
  - During life cycle software is produced early which facilitates customer evaluation and feedback.

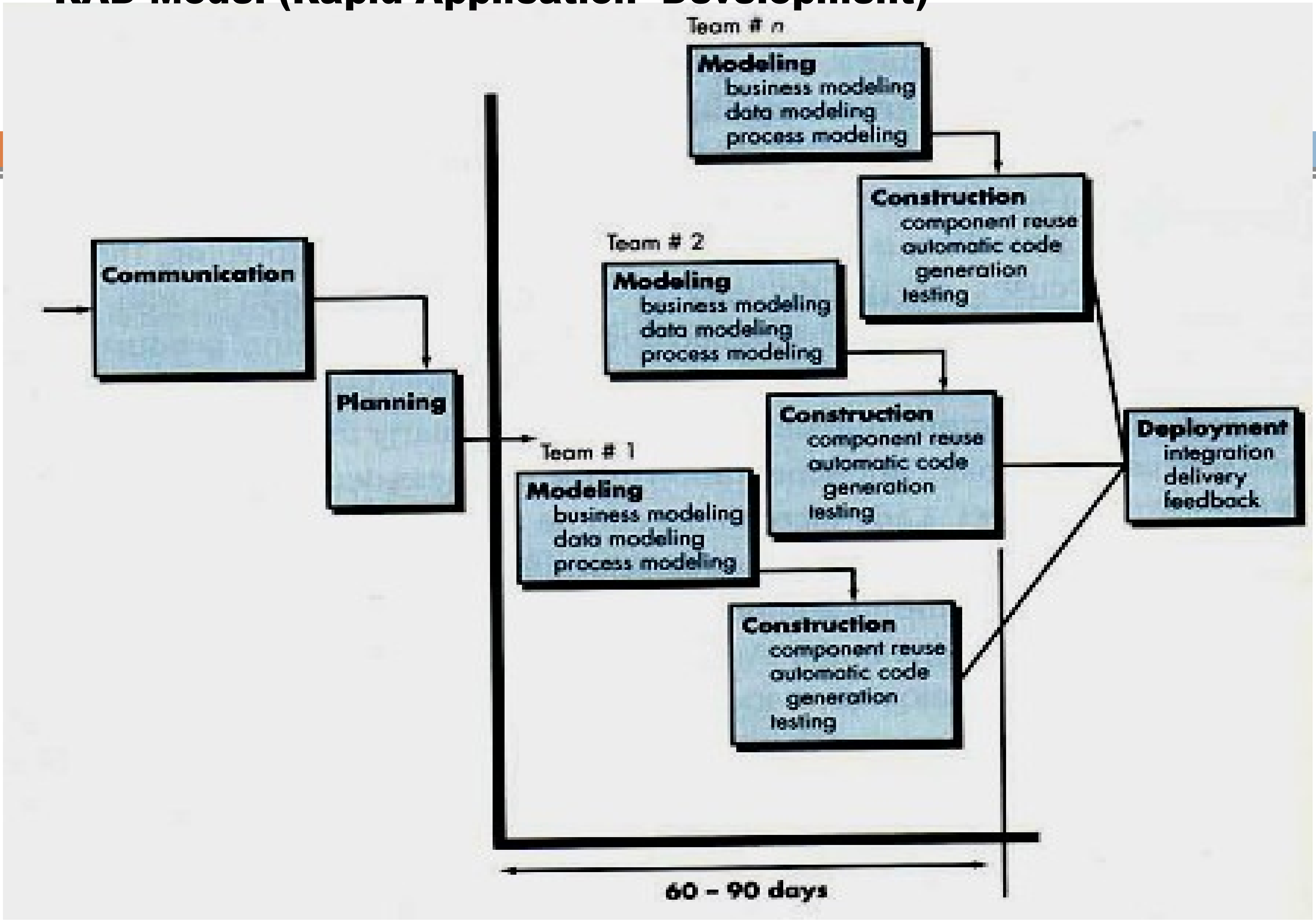
Incremental Model

- **Disadvantages:**
  - Requires good planning and design
  - Requires early definition of a complete and fully functional system to allow for the definition of increments
  - Although cost of change is lesser but it is not very suitable for changing requirements.
  - More management attention is required.
  - Defining increments may require definition of the complete system.
  - Not suitable for smaller projects.

RAD Model (Rapid Application Development)

- A type of incremental software process model
- The RAD model is a “high-speed” adaptation of the linear sequential model in which rapid development is achieved by using component-based construction.
- If requirements are well understood and project scope is constrained, the RAD process enables a development team to create a “fully functional system” within very short time periods (e.g., 60 to 90 days)
- Each major function can be addressed by a separate RAD team and then integrated to form a whole

RAD Model (Rapid Application Development)



RAD Model (Rapid Application Development)

- **Applications:**
  - Reasonably well-known requirements
  - System can be modularized
  - RAD should be used only when a system can be modularized in a way that each major function can be completed in less than 3 months
  - It should be used if there’s high availability of designers for modeling
  - It should be used only if the budget permits use of automated code generating tools

RAD Model (Rapid Application Development)

- **Advantages:**
  - Changing requirements can be accommodated
  - Progress can be measured
  - Increases reusability of components
  - Uses modeling concepts to capture information about business, data, and processes
- **Disadvantages:**
  - For large but scalable projects, RAD requires sufficient human resources to create the right number of RAD teams
  - RAD requires developers and customers who are committed to the rapid-fire activities necessary to get a system complete in a much abbreviated time frame. If commitment is lacking from either constituency, RAD projects will fail
  - Not all types of applications are appropriate for RAD. If a system cannot be properly modularized, building the components necessary for RAD will be problematic

Software prototyping

- **Software prototyping is building a prototype (dummy version) of the software during the requirements phase.**
- The prototype does not always hold the exact logic used in the actual software application but is evaluated by end users and their feedback is used to refine the prototype. When the user is satisfied, the prototype code is brought up to the standards needed for a final product.
- **Significance in software engineering:** Since a working model of the system is displayed, the users get a better understanding of the system being developed. For example, a software prototype of Messenger can be developed by the Software Engineer. The feedback of user on this prototype messenger the added

Software prototyping

- **Prototyping Approaches:**

  The prototyping paradigm can be either **close-ended** or **open-ended**.

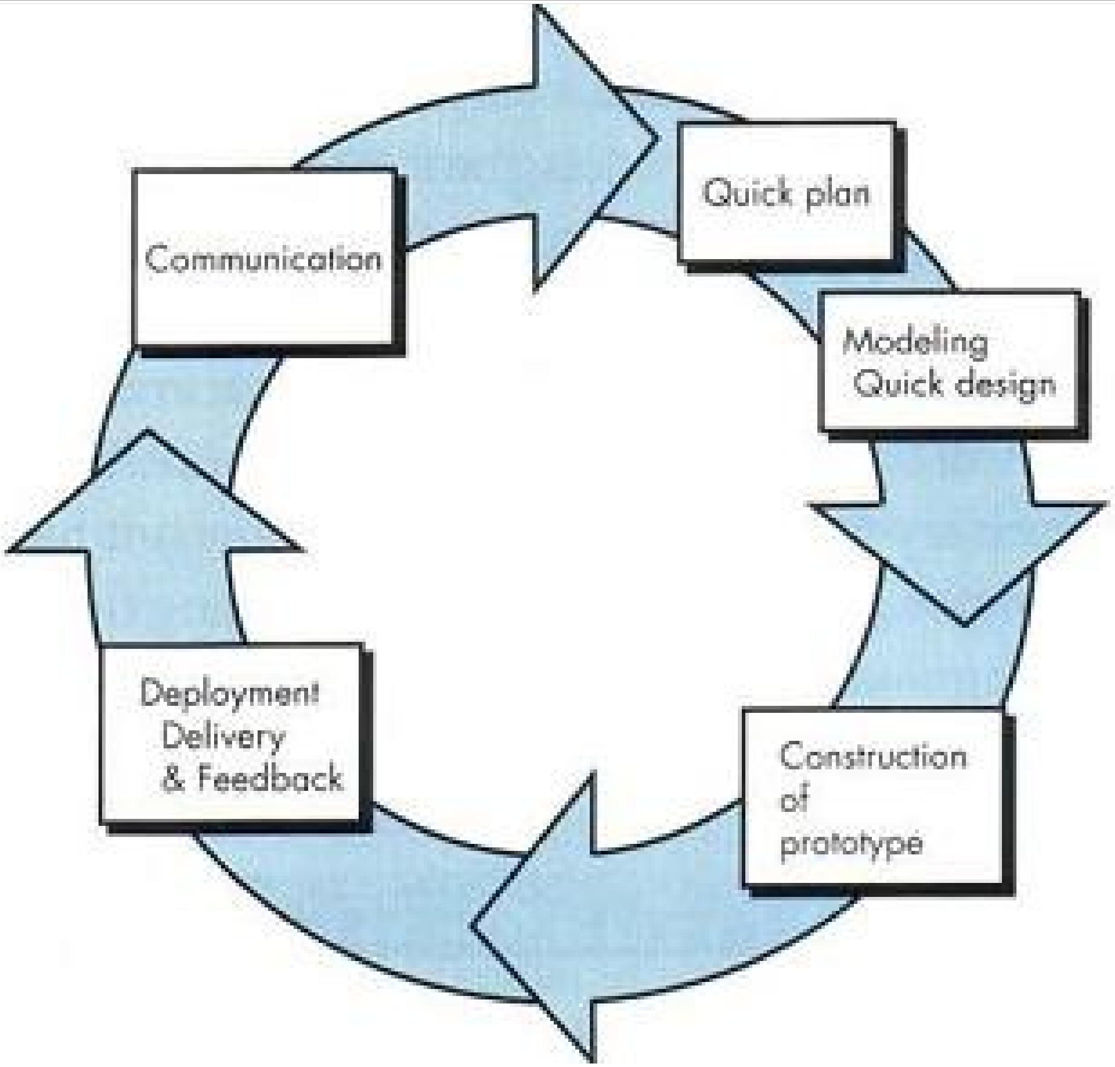
  The close-ended approach is often called throwaway prototyping. Using this approach, a prototype serves solely as a rough demonstration of requirements. It is then discarded, and the software is engineered using a different paradigm.

  An open-ended approach, called evolutionary prototyping, uses the prototype as the first part of an analysis activity that will be continued into design and construction. The prototype of the software is the first evolution of the finished system.

Prototype model

- Developers build a prototype during the requirements phase
- The prototype does not always hold the exact logic used in the actual software application and is an extra effort to be considered under effort estimation.
- Prototype is evaluated by end users
- Users give corrective feedback
- Developers further refine the prototype
- When the user is satisfied, the prototype code is brought up to the standards needed for a final product.
- **Applications / Used when:**
  - A customer defines a set of general objectives for software but does not identify detailed input, processing, or output requirements.
  - Requirements are unstable or have to be clarified
  - Low-risk projects

Prototype model



Prototype model

- **Advantages:**
  - Developers learn from customers
  - A more accurate end product
  - Steady, visible signs of progress produced
  - Reduces time and cost as the defects can be detected much earlier.
  - Since a working model of the system is displayed, the users get a better understanding of the system being developed.

- **Disadvantages:**
  - Risk of insufficient requirement analysis owing to too much dependency on prototype
  - Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
  - The effort invested in building prototypes may be too much if not monitored properly
  - The developer often makes implementation compromises in order to get a prototype working quickly.

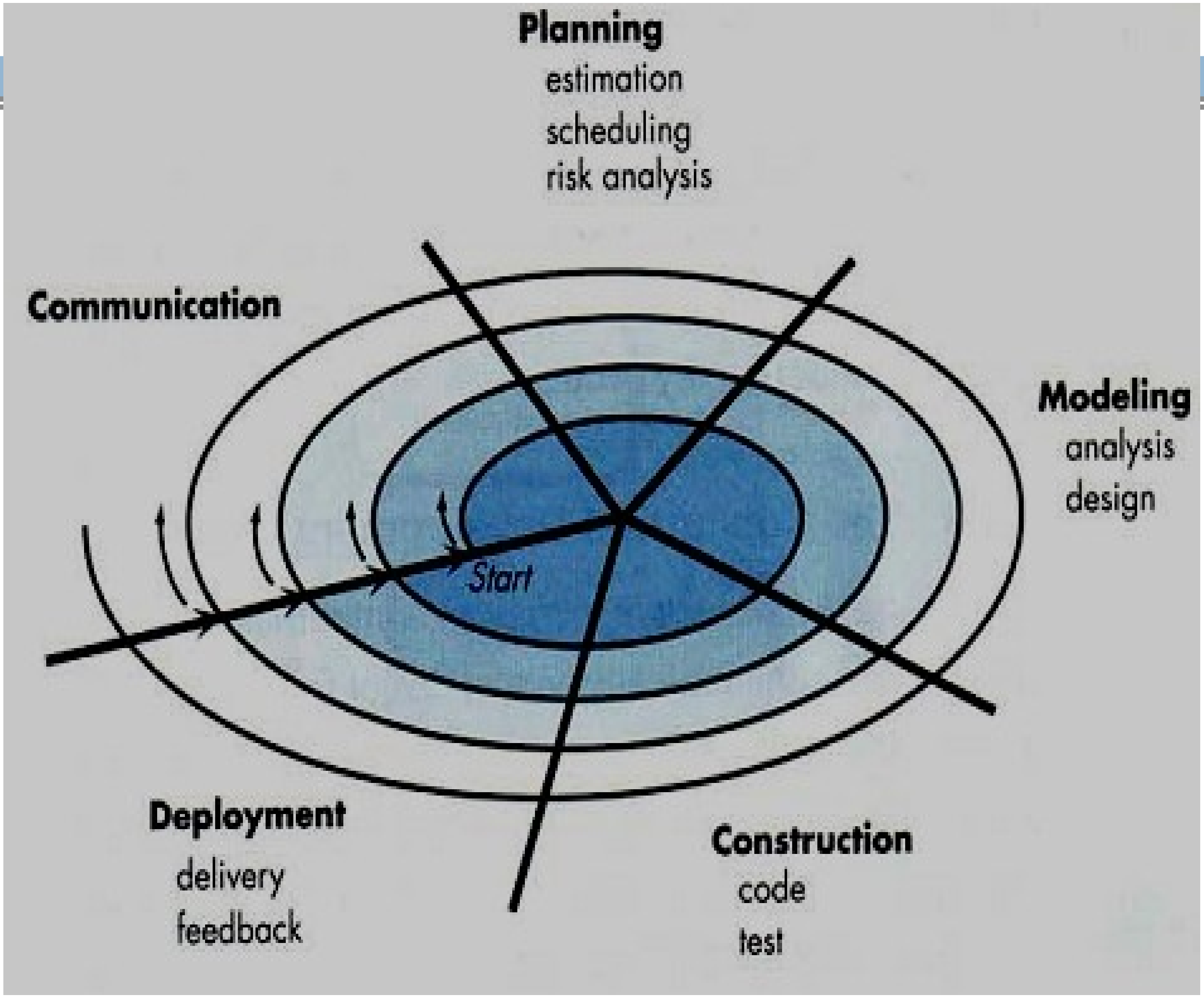
Spiral Model

- A combination of iterative development process model and waterfall model with very high emphasis on risk analysis.
- It provides the potential for rapid development of incremental versions of the software through each iteration around the spiral.
- During early iterations, the incremental release might be a paper model or prototype. During later iterations, increasingly more complete versions of the engineered system are produced.
- A spiral model is divided into a set of framework activities (C, P, M, C, D)
- Each of the framework activities represents one segment of the spiral path.

Spiral Model

- Cost and schedule are adjusted based on feedback derived from customer evaluation.
- In addition, the project manager adjusts the planned number of iterations required to complete the software.
- A “concept development project” starts at the core of the spiral and will continue until concept development is complete.
- If the concept is to be developed into an actual product, a “new development project” is initiated.
- Later, a circuit around the spiral might be used to represent a “product enhancement project”
- In this way, the spiral remains operative until the software is retired.

Spiral Model



Spiral Model

- **Applications:**
  - When there is a budget constraint and risk evaluation is important
  - For medium to high-risk projects
  - Customer is not sure of their requirements which is usually the case
  - Requirements are complex and need evaluation to get clarity
  - New product line which should be released in phases to get enough customer feedback
  - Significant changes are expected in the product during the development cycle

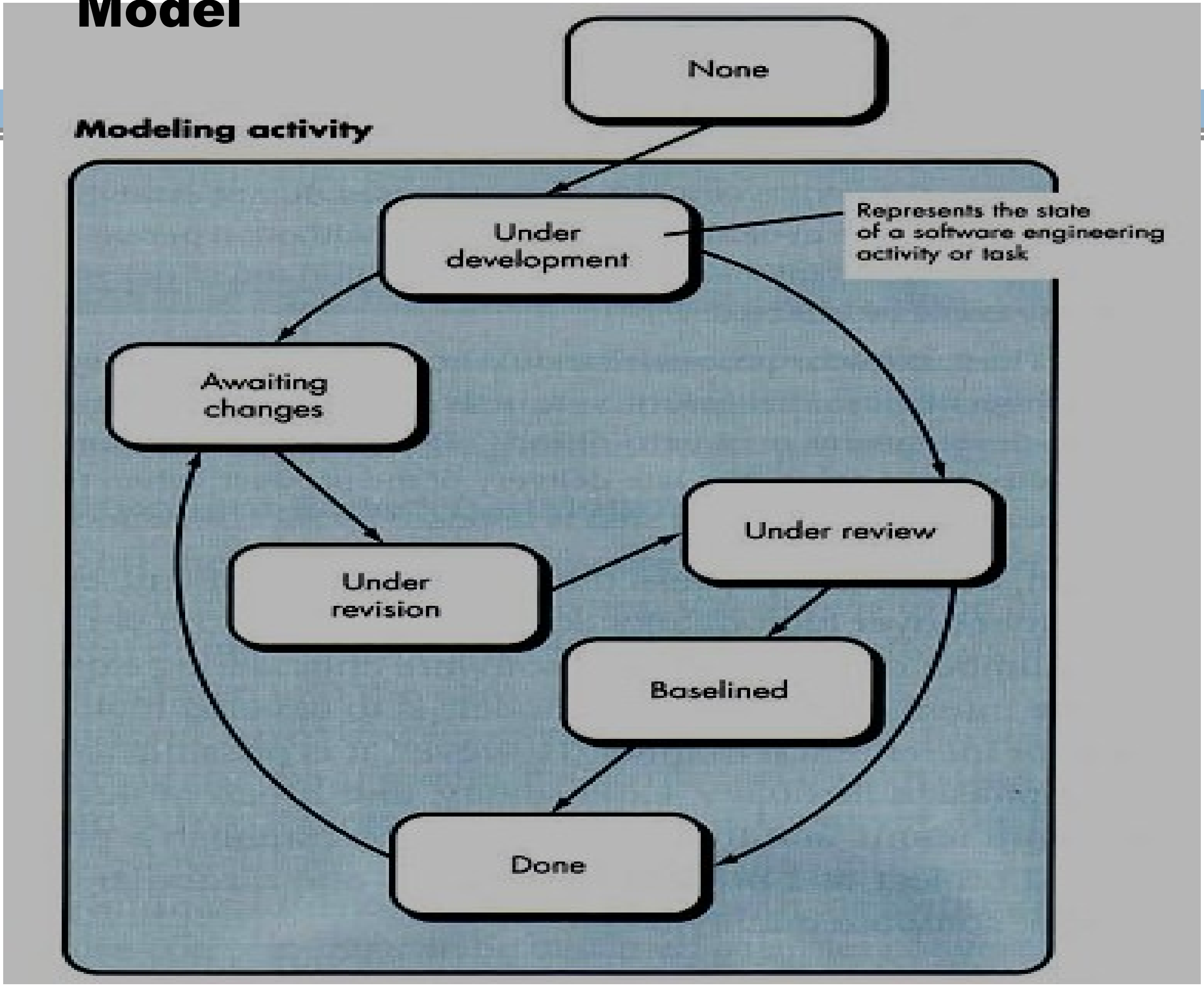
Spiral Model

- **Advantages:**
  - Changing requirements can be accommodated
  - Allows for extensive use of prototypes
  - Requirements can be captured more accurately.
  - Development can be divided into smaller parts and more risky parts can be developed earlier which helps better risk management.
- **Disadvantages:**
  - Management is more complex.
  - Not suitable for small or low risk projects and could be expensive for small projects.
  - Process is complex
  - Spiral may go indefinitely.
  - Large number of intermediate stages requires excessive documentation.

Concurrent Development Model

- Also known as concurrent engineering
- The concurrent process model can be represented schematically as a series of major technical activities, tasks, and their associated states.
- All activities exist concurrently but reside in different states.
- For example, early in a project the customer communication activity has completed its first iteration and exists in the ***awaiting changes state.***
- The analysis activity (which existed in n***one state*** while initial customer communication was completed) now makes a transition into the under ***development state***.
- If the customer indicates that changes in requirements must be made, the analysis activity moves from the ***under development state*** into the ***awaiting changes state.***

Concurrent Development Model



Concurrent Development Model

- **Applications:**
  - Project managers who track project status in terms of the major phases [of the classic life cycle] have no idea of the status of their projects
- **Advantages:**
  - To all types of software development
  - Provides an accurate picture of the current state of a project.
  - Each activity or task can be carried out concurrently.