Software Process Models

Unit I

Software Process

- Process is a ***framework for the tasks*** that are required to build high-quality software.
- A software process is ***a set of activities and associated results***, which produces a software product.
- Software engineers carry out these activities.

Software Process

- The common activities ca be categorized into the following:
  - ***Software specification*** **-** the functionality of the software and constraints on its operation must be defined.
  - ***Software development*** – the software to meet the specification must be produced.
  - ***Software validation*** – the software must be validated to ensure that it does what the customer wants.
  - ***Software evolution*** – the software must be evolved to meet the changing requirements of customers.

Software Process

- Different processes organize these activities in different ways and are described at different levels of detail.
- Different organizations may use different processes to produce the same types of product.
- However, some processes are suitable than the others for some types of applications.
- If inappropriate process is used, this will probably reduce the quality or the usefulness of the software product to be developed.

Process Model

- Process Model
  - The process model is a description of a software process.
  - Model is an abstraction (summary) of the actual process.
  - Process model may include activities which are part of software process, software product and role of people involved in software engineering.

Types of Process Models

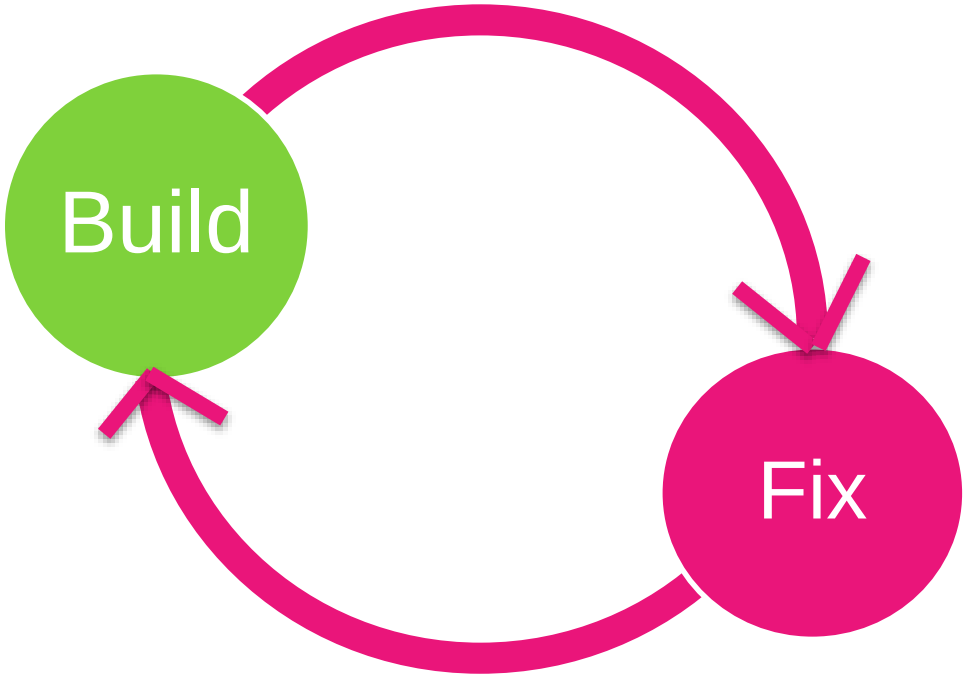
- General types of software process models:
  - ***A workflow model.*** This shows the sequence of activities in the process along with their inputs, outputs and dependencies. The activities in this model represent human-actions.
  - ***A dataflow model/ Activity model.*** This represents the process as a set of activities each of which carries out some data transformation. It shows how the input to the process is transformed to an output.
  - ***A role/action model.*** This represents the role of the people involved in the software process and the activities for which they are responsible.

General Process Models

- Build & Fix Model
- Waterfall Model (Linear Sequential Model)
- Evolutionary Models
  - Prototyping Model
  - Spiral Model
- Incremental Models
  - Iterative Model
  - RAD Model
- Unified Process Model

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

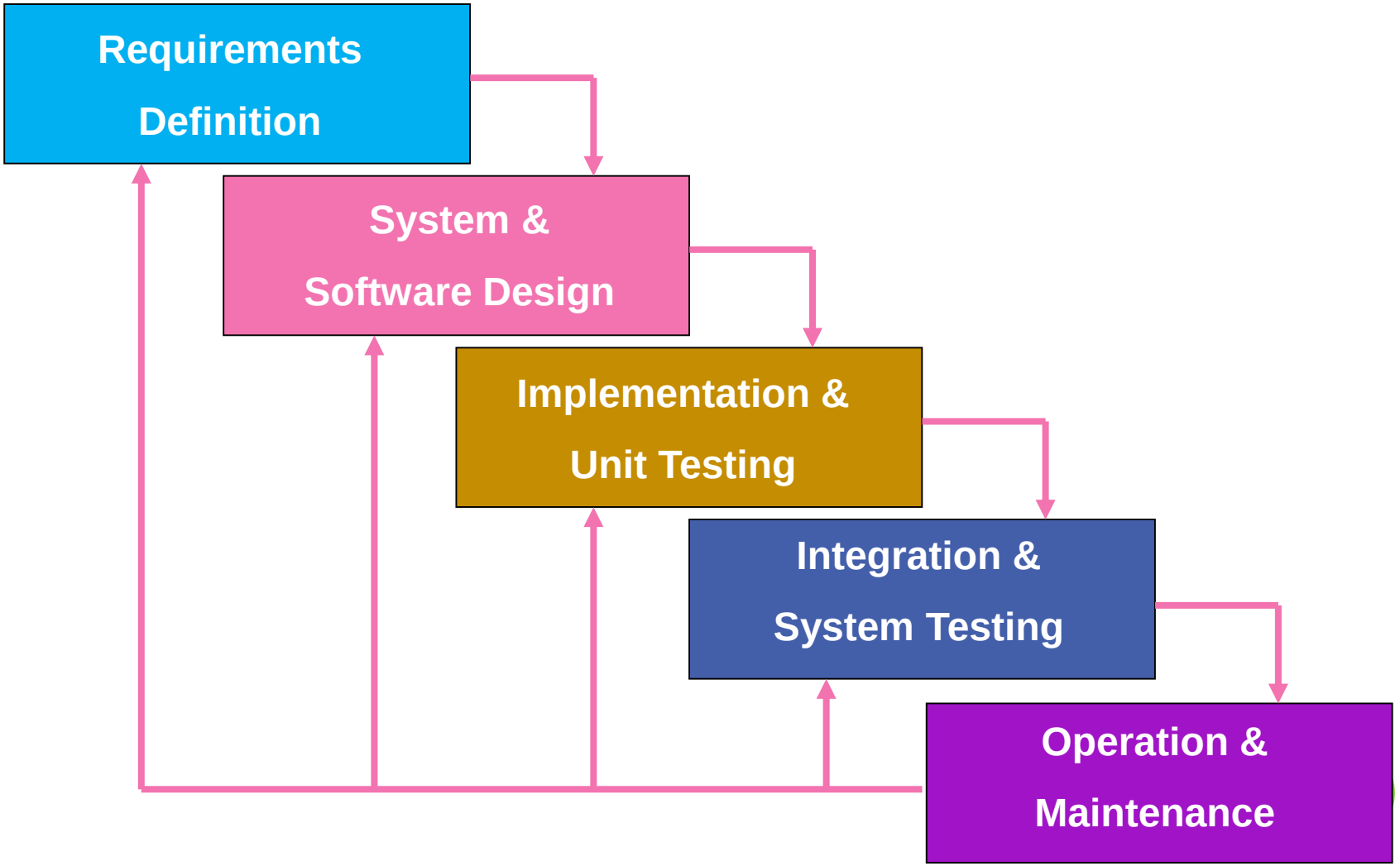
Build & Fix Model



Build & Fix Model

- Product is constructed without any specifications or any attempt to design.
- The developer simple builds the product that is reworked as many times as necessary to satisfy the client.
- This is an ad-hoc approach and not well defined.
- It is a simple two phase model:
  - The first phase is to write the code
  - The next phase is to fix it.
- Fixing may be error correction or addition of further functionality.

Waterfall Model



Waterfall Model

- Also known as **software-life cycle, classical life cycle** and **linear sequential model**.
- Derived from other engineering process.
- The oldest and widely used model.
- It suggests a systematic sequential approach from requirement analysis up to support phase.
- A product delivered after the linear sequence is complete.

Waterfall Model: Phases

- ***Requirements analysis and definition***.
  - The system services, constraints and goals are established by consultation with system users.
  - They are then defined in detail and serve as a system specification.
- ***System and software design***.
  - The system design process partitions the requirements to either hardware or software systems.
  - It establishes an overall system architecture.
  - Software design involves identifying and describing the fundamental software system abstractions and their relationships.

Waterfall Model: Phases

- ***Implementation and unit testing***.
  - During this stage, the software design is realized as a set of programs or program unit.
  - Unit testing involves verifying that each unit meets its specification.
- ***Integration and system testing***
  - The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met.
  - After testing, the software system is delivered to the customer.

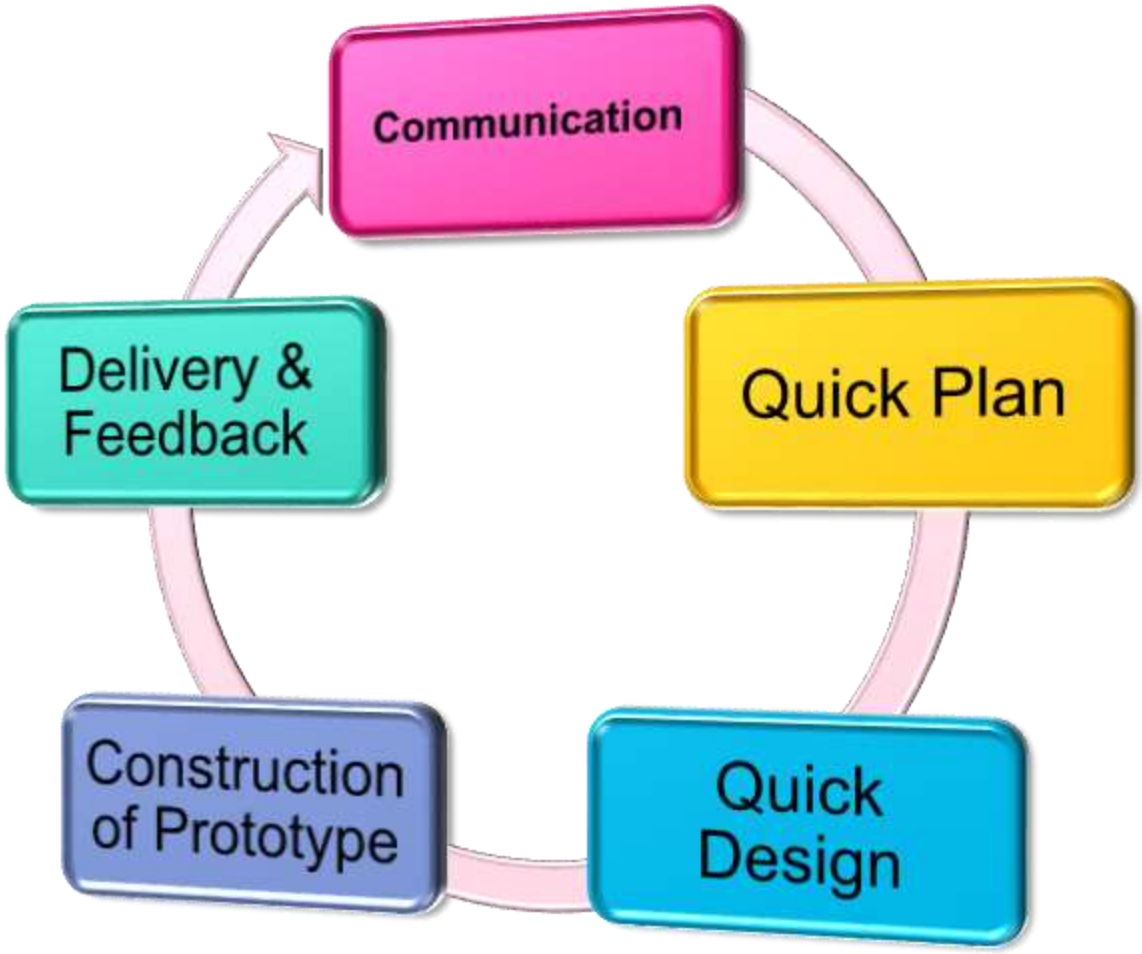
Waterfall Model: Phases

- ***Operation and maintenance***.
  - Normally this is the longest life-cycle phase.
  - The system is installed and put into practical use.
  - Maintenance involves correcting errors which were not discovered in earlier stages of the life cycle.
  - Thus, improving the implementation of system units and enhancing the system's services as new requirements are discovered.

Waterfall Model: Limitations

- A phase should not start until the previous phase is signed off.
- Iterations is costly and involve significant work.
- Software is put into use during final phase of study.
- Partitioning into the distinct stages of projects is inflexible.
- Requirements should be well understood.

Evolutionary Models: Prototyping Model



Evolutionary Models: Prototyping Model

- It is an effective paradigm for software engineering.
- Generally in this model, developers listen to the customer, design a prototype, and then test the prototype.
- The procedure is repeated until all the requirements of the system are identified, and a complex system can be designed.

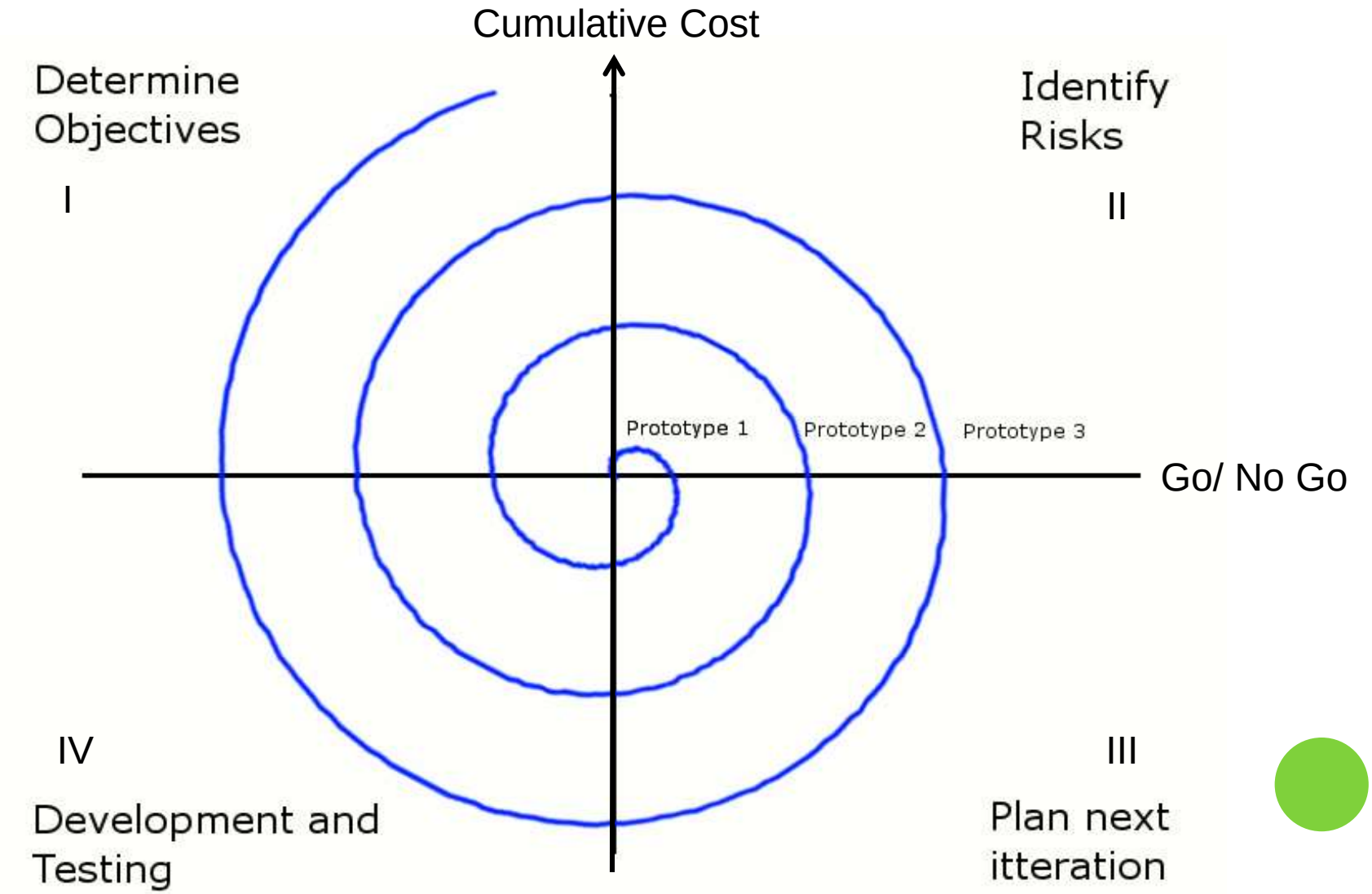
Evolutionary Models: Prototyping Model

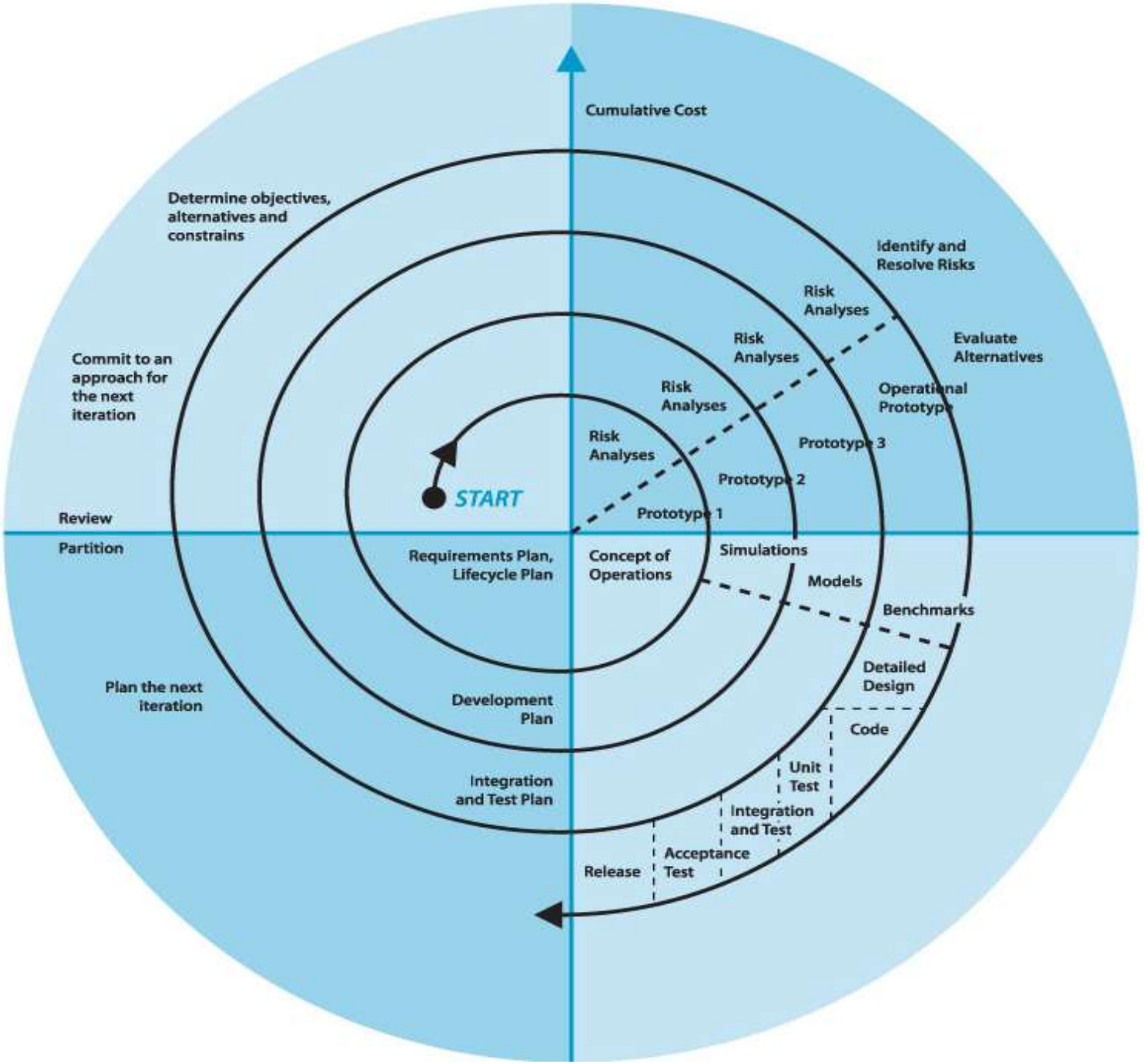
- Steps:
  - First of all developers meet the customers and consult them. General objectives are defined and known requirements are identified.
  - Areas where further definition is mandatory are outlined.
  - A quick design occurs. The design focuses on a representation of those aspects of the software that will be visible to customers/users
  - Then a prototype is constructed.
  - The prototype is evaluated to refine the software requirements.
  - Iteration can occur to satisfy the customer’s needs.

Evolutionary Models: Prototyping Model

- Suitable scenario:
  - General objectives of software are defined.
  - Detail requirements of input, processing and output are not identified.
  - Efficiency of algorithms is not sure.
  - Adaptability can take place.
- The model's objective:
  - Defined rules are used in the prototype to identify the detail requirements.
- **Can be used as a tool for Requirement Gathering**

Evolutionary Models: Spiral Model




Cumulative Cost
Determine objectives, alternatives and constrains
Identify and Resolve Risks
Risk Analyses
Evaluate Alternatives
Commit to an approach for the next iteration
Risk Analyses
Operational Prototype
Risk Analyses
Prototype 3
Risk Analyses
Prototype 2
START
Prototype 1
Review
Partition
Requirements Plan, Lifecycle Plan
Concept of Operations
Simulations
Models
Benchmarks
Detailed Design
Plan the next iteration
Development Plan
Code
Unit Test
Integration and Test Plan
Integration and Test
Acceptance Test
Release

Evolutionary Models: Spiral Model

- The radial dimension of the model represents cumulative costs
- Each path around the spiral is indicative of increased costs.
- The angular dimension represents the progress made in completing each cycle.
- Each loop of the spiral from X-axis clockwise through 360 degrees, represents one phase.
- One phase is split roughly into four sectors:
  - Planning: determination of objectives, alternatives, & constraints.
  - Risk Analysis: analyze alternatives & attempt to identify & resolve the risks involved.
  - Development: product development & testing.
  - Assessment: customer evaluation.

Evolutionary Models: Spiral Model

- During the first phase:
  - Planning is done
  - Risks are analyzed
  - Prototypes are built
  - Customers evaluate the prototype.
- During second phase:
  - A more refined prototype is built
  - Requirements are documented & validated
  - Customers are involved in assessing the new prototype.
- By the time third phase begins,
  - Risks are known
  - A somewhat more traditional development approach is taken up.

Evolutionary Models: Spiral Model

- It couples the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model.
- It provides the potential for rapid development of increasingly more complete versions of the software
- Each phase is completed with a review by the people concerned with the project.
- It is a *risk-driven process model.*
- It has two main distinguishing features
  - **Cyclic approach:** Incrementally growing a system's degree of definition and implementation while decreasing its degree of risk
  - **A set of *anchor point milestones*:** A combination of work products and conditions that are attained along the path of the spiral.

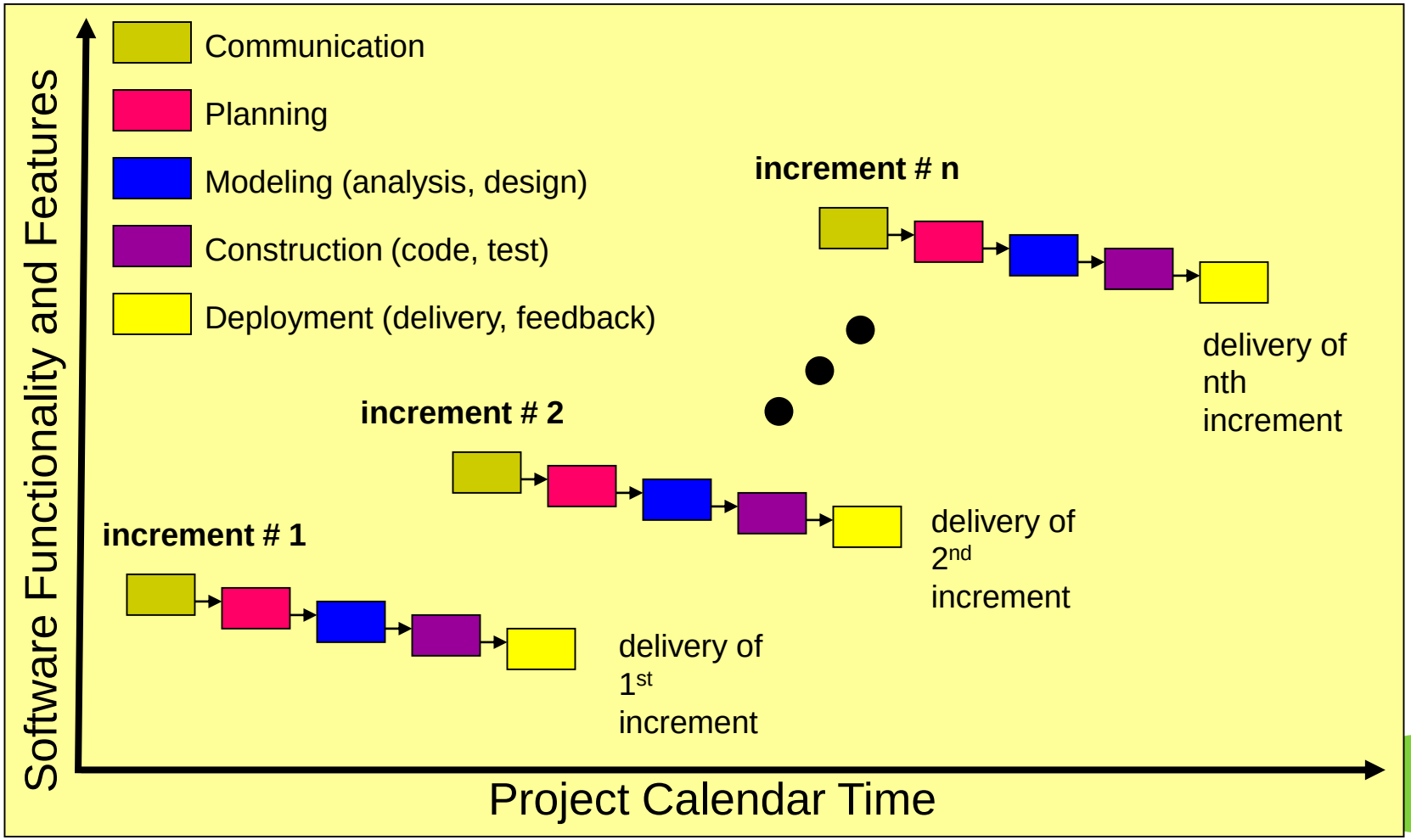
Evolutionary Models: Spiral Model

- Unlike other process models that end when software is delivered, the spiral model can be adapted to apply throughout the life of the computer s/w.
- The circuits around the spiral might represent
  - Concept development project
  - New Product development project
  - Product enhancement project
- The spiral model demands a direct consideration of technical risks at all stages of the project

Evolutionary Models: Spiral Model: Drawbacks

- It may be difficult to convince customers (particularly in contract situations) that the evolutionary approach is controllable.
- It demands considerable risk assessment expertise and relies on this expertise for success.
- If a major risk is not uncovered and managed, problems will undoubtedly occur.

Incremental Models: Iterative Enhancement Model



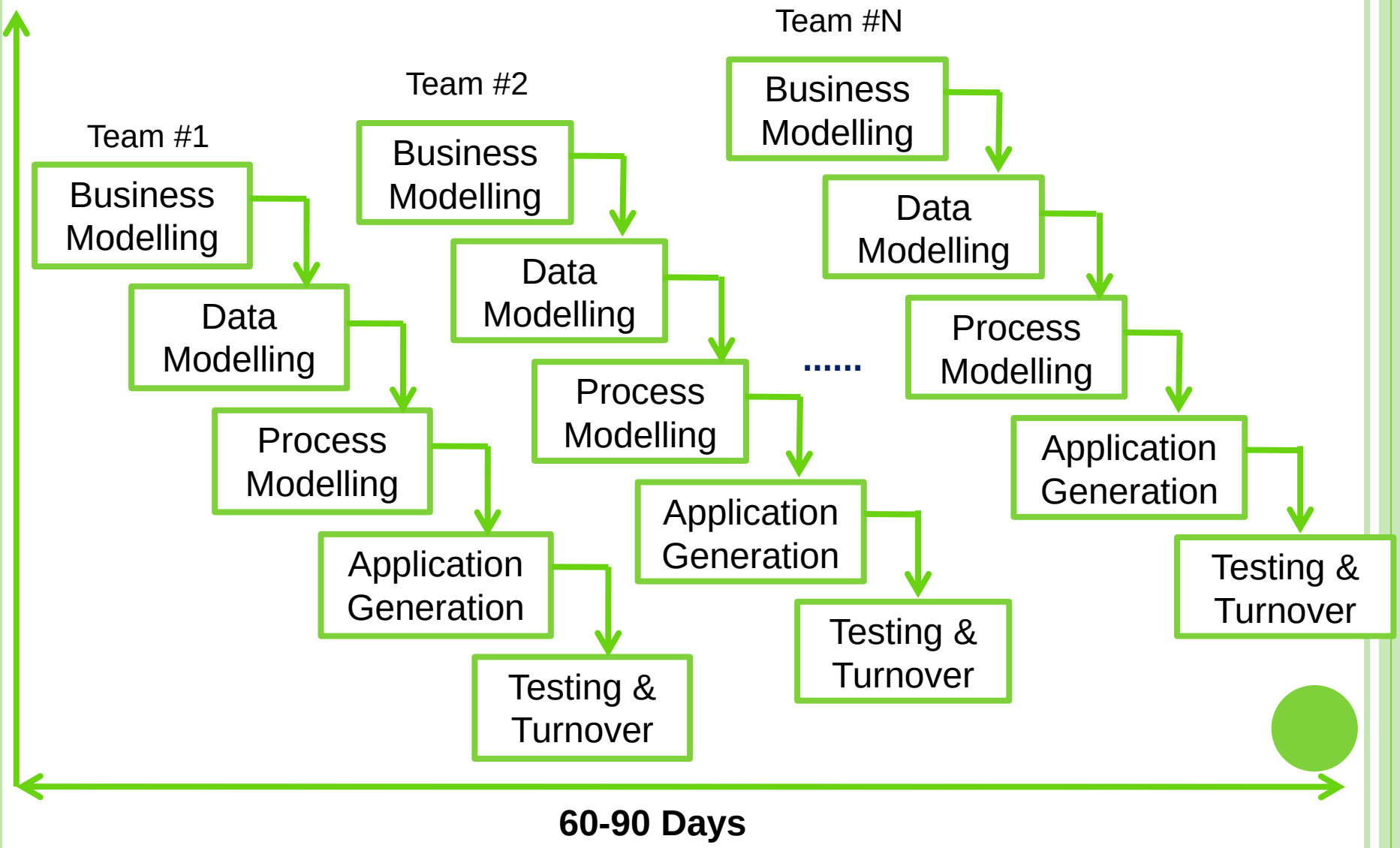
Incremental Models: Iterative Enhancement Model

- This model has the same phases as waterfall model but with less restrictions.
- Combines elements of the waterfall model applied in an iterative fashion
- Each linear sequence produces deliverable “increments” of the software
- The first increment is often a *core product*
- The core product is used by the customer (or undergoes detailed evaluation)
- Based on evaluation results, a plan is developed for the next increment

Incremental Models: Iterative Enhancement Model

- The incremental process model, like prototyping and other evolutionary approaches, is iterative in nature
- But unlike prototyping, the incremental model focuses on the delivery of an operational product with each increment
- Particularly useful when
  - Staffing is unavailable
- Increments can be planned to manage technical risks

Incremental Models: Rapid Application Development Model



Incremental Models: Rapid Application Development Model: Phases

- ***Business modeling:*** The information flow among business functions is modeled in a way that answers the following questions:
  - What information drives the business process?
  - What information is generated?
  - Who generates it?
  - Where does the information go?
  - Who process it?

Incremental Models: Rapid Application Development Model: Phases

- ***Data modeling.*** The information flow defined as part of the business modeling phase is refined into a set of data objects that are needed to support the business.
  - The characteristics (called attributes) of each object are identified and the relationship between these objects defined.
- ***Process modeling.*** The data objects defined in the data modeling phase are transformed to achieve the information flow necessary to implement a business function.
  - Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.

Incremental Models: Rapid Application Development Model: Phases

- ***Application generation.*** RAD assumes the use of fourth generation techniques.
  - Rather than creating software using conventional third generation programming languages.
  - The RAD process works to reuse existing program components (when possible) or create reusable components (when necessary).
  - In all the cases, automated tools are used to facilitate construction of the software.

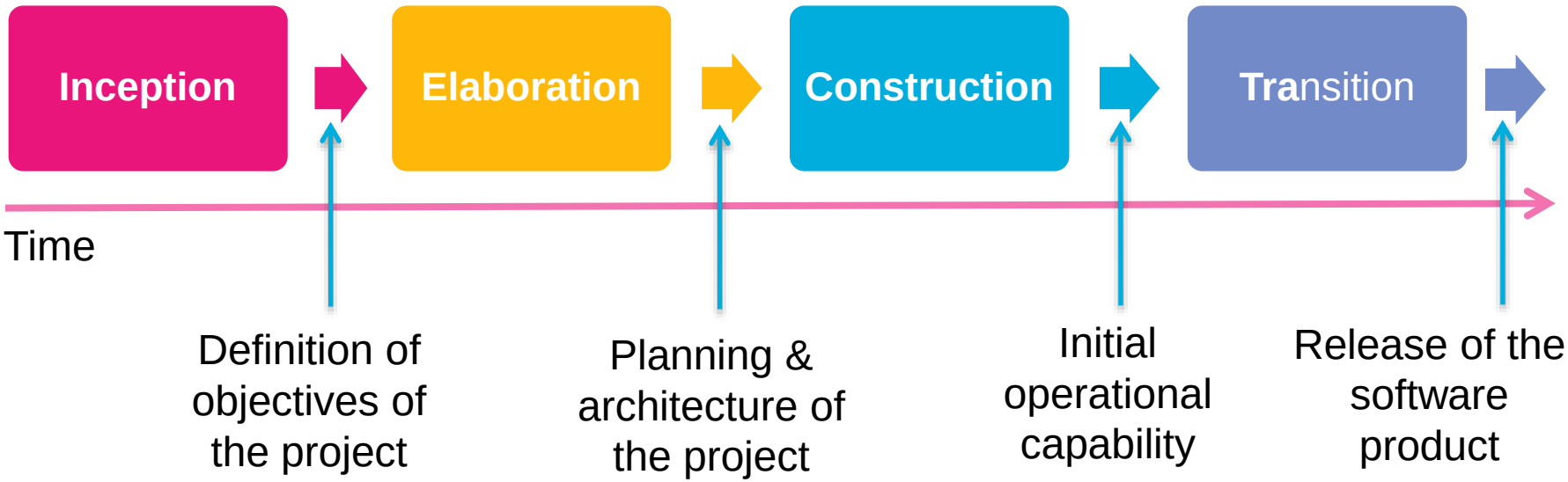
Incremental Models: Rapid Application Development Model: Phases

- ***Testing and turnover***. Since the RAD process emphasize reuse, many of the program components have already been tested.
  - This reduces overall testing time.
  - However, new components must be tested and all interfaces must be fully exercised.

Incremental Models: Rapid Application Development Model: Drawbacks

- RAD requires sufficient human-resources to create the right number of RAD teams in a large and scalable projects.
- RAD developers and costumers should be committed to the rapid-fire activities.
- Unsuitable cases:
  - A non-modularized system.
  - When technical risk is high.

Unified Process model



Unified Process Model: Phases

- ***Inception***
  - Encompasses both customer communication and planning activities
  - Fundamental business requirements are described through a set of preliminary use-cases
    - A **use-case** describes a sequence of actions that are performed by an actor (e.g., a person, a machine, another system) as the actor interacts with the software
  - A rough architecture for the system is also proposed

Unified Process Model: Phases

- ***Elaboration***
  - Encompasses customer communication and modeling activities
  - Refines and expands the preliminary use-cases
  - Expands the architectural representation to include five different views of the software
    - The use-case model
    - The analysis model
    - The design model
    - The implementation model
    - The deployment model
  - In some cases, elaboration creates an “executable architectural baseline” that represents a “first cut” executable system

Unified Process Model: Phases

- ***Construction***
  - Makes each use-case operational for end-users
  - As components are being implemented, **unit tests** are designed and executed for each
  - Integration activities (component assembly and **integration testing**) are conducted
  - Use-cases are used to derive a suite of **acceptance tests**

Unified Process Model: Phases

- ***Transition***
  - Software is given to end-users for **beta testing**
  - The software team creates the necessary support information –
    - User manuals
    - Trouble-shooting guides
    - Installation procedures
  - At the conclusion of the transition phase, the software increment becomes a usable software release