


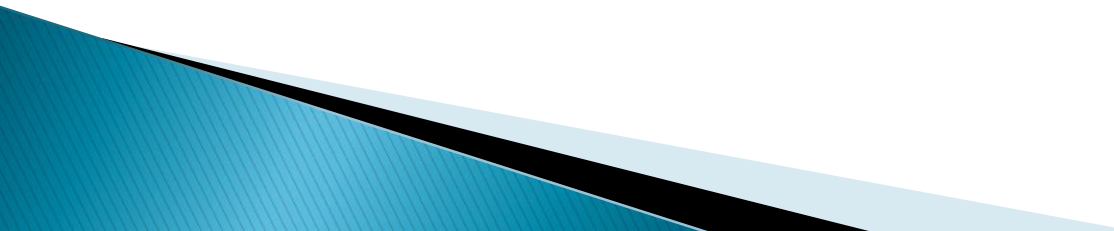
# Software Model Process Con't



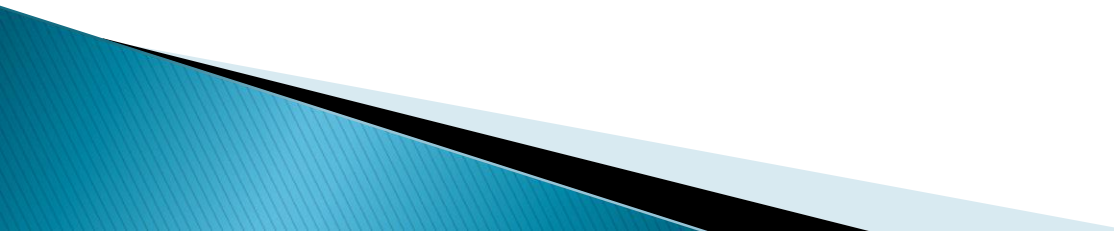
# Evolutionary Model

- ▶ Business and product requirement **often change** as development proceed
  - ▶ Software engineer need a process model that has been **explicitly designed** to accommodate a product that evolves over time
  - ▶ Evolutionary models are **iterative**. Enables software engineers to develop increasingly more complete version of the software
- 

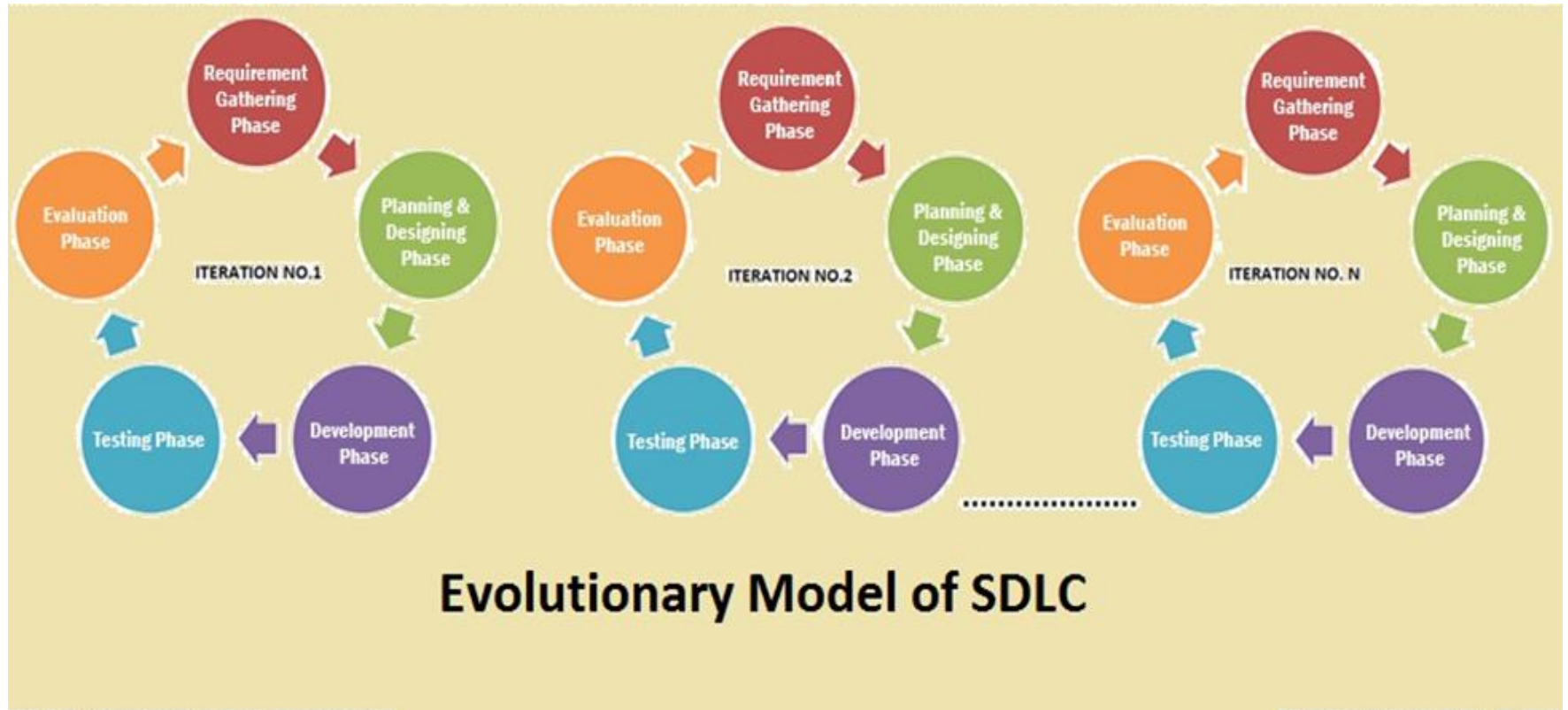
# Evolutionary Model

- ▶ It is better for software products that have their feature sets redefined during development because of user feedback and other factors.
  - ▶ The Evolutionary development model divides the development cycle into smaller, incremental waterfall models in which users are able to get access to the product at the end of each cycle.
- 

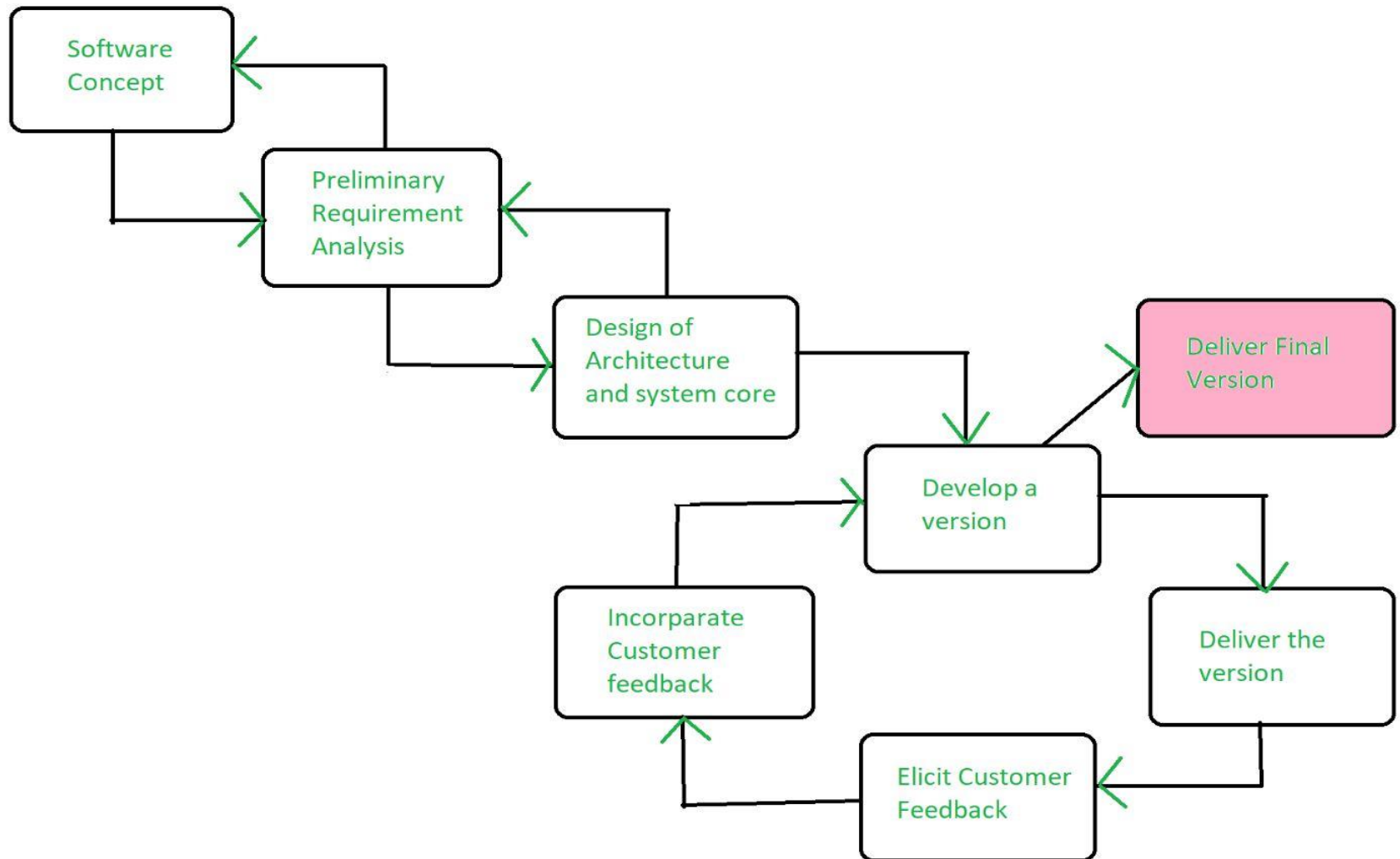
# Evolutionary Model

- ▶ Feedback is provided by the users on the product for the planning stage of the next cycle and the development team responds, often by changing the product, plan or process.
  - ▶ Therefore, the software product evolves with time.
- 

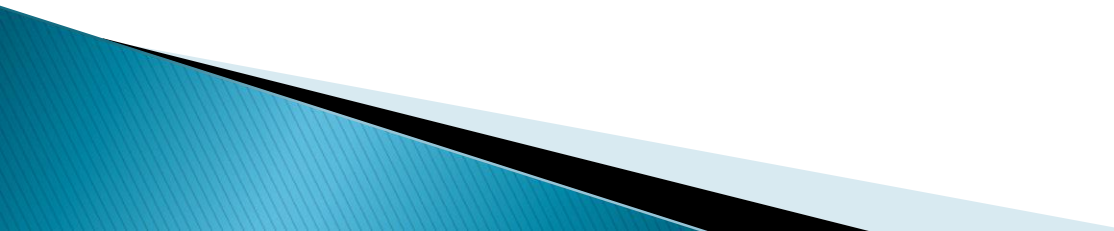
# Evolutionary Model



# Evolutionary Model



# Application of Evolutionary Model:

1. It is used in large projects where you can easily find modules for incremental implementation. Evolutionary model is commonly used when the customer wants to start using the core features instead of waiting for the full software.
  2. Evolutionary model is also used in object oriented software development because the system can be easily portioned into units in terms of objects.
- 

# Advantages

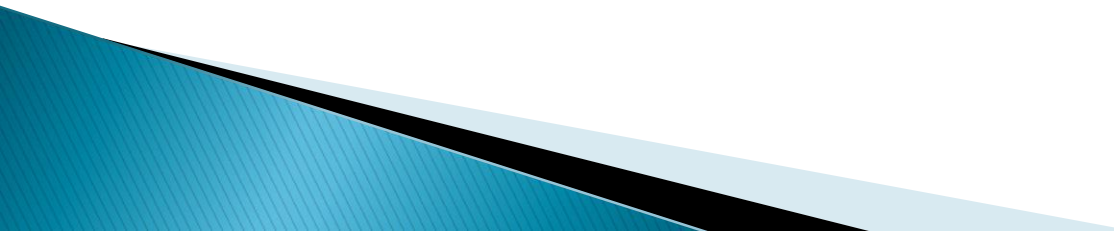
- **Large project:** Evolutionary model is normally useful for very large products.
- User gets a **chance to experiment with a partially developed software** much before the complete version of the system is released.
- Evolutionary model helps to accurately **elicit user requirements** during the delivery of different versions of the software.
- The core modules get tested thoroughly, thereby **reducing the chances of errors** in the core modules of the final products.
  - ▶ Evolutionary model **avoids the need to commit large resources** in one go for development of the system



# Disadvantages

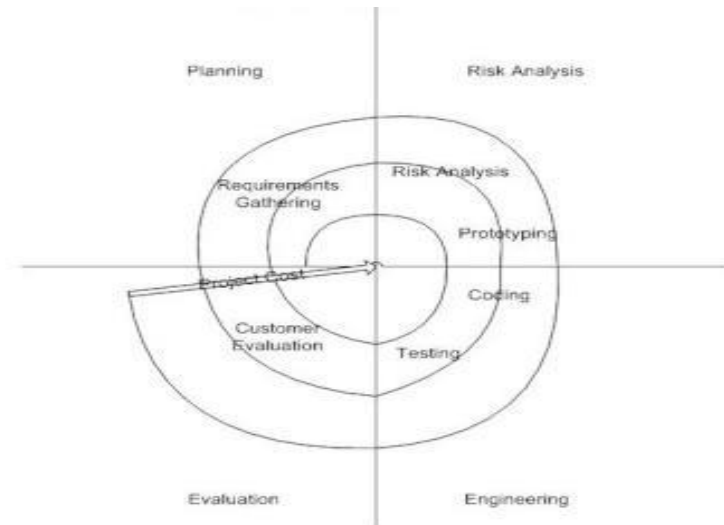
- Sometimes it is hard to divide the problem into several versions that would be acceptable to the customer which can be incrementally implemented and delivered.
- It is a slow process because it takes more time for development.
- Many changes can disturb the rhythm of the development team.
- It is a thrown away prototype when the users are confused with it.

# The Spiral

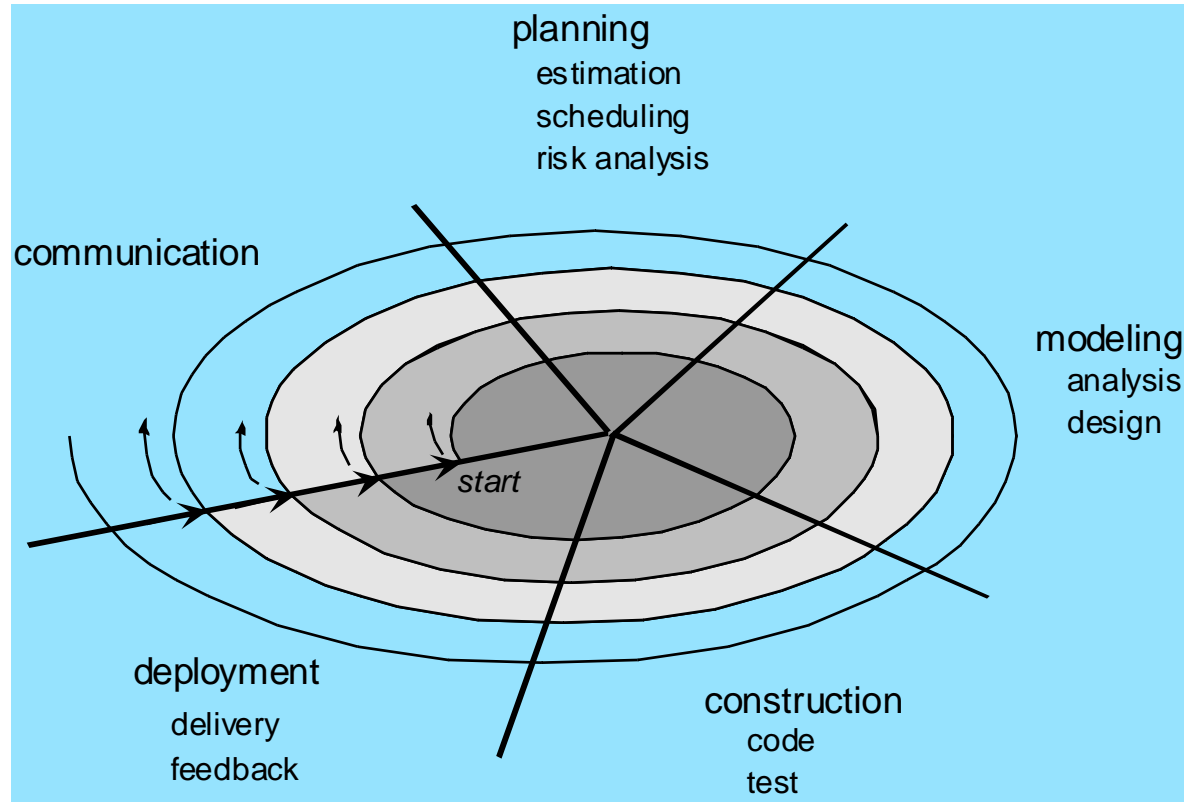
- ▶ Proposed by Boehm
  - ▶ Evolutionary software process model that couples the **iterative** nature of prototyping with the controlled and **systematic aspect** of the waterfall model
  - ▶ **Each loop** in the spiral **represents a phase** of the software process.
  - ▶ The important distinction between spiral model and other software models is the **explicit consideration of risk**
- 

# Spiral Process Model

- ▶ The Spiral Model is an evolutionary software process model that couples the iterative nature of prototyping with the controlled and systematic aspects of the Linear Sequential Model.
- ▶ Using the Spiral Model the software is developed in a series of incremental releases. Unlike the Iterative Model where in the first product is a core product, in the Spiral Model the early iterations could result in a paper model or a prototype.
- ▶ However, during later iterations more complex functionalities could be added.



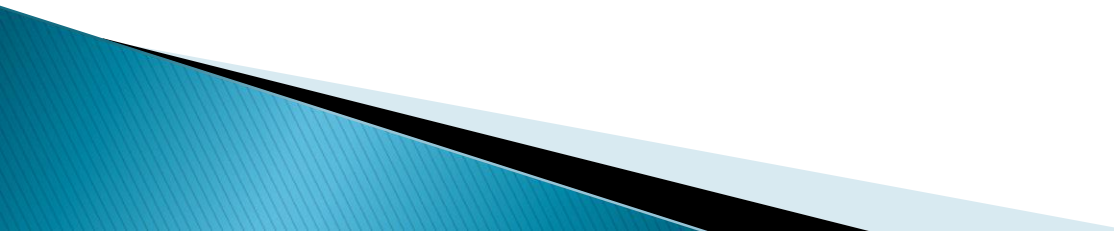
# Evolutionary Models: The Spiral (Cont')



# Task Regions of Spiral Process Model

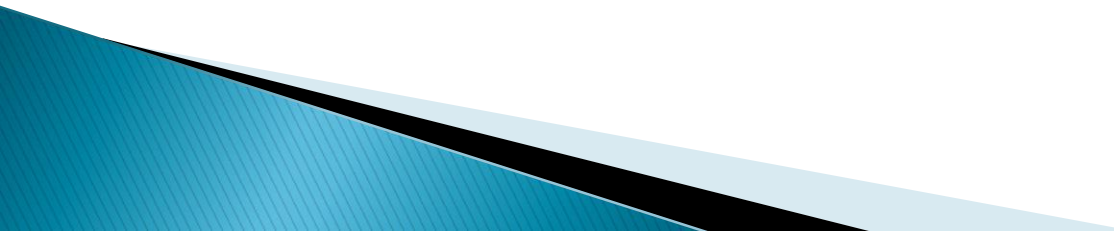
- ▶ A Spiral Model is divided into a number of framework activities, also called task regions. These task regions could vary from 3-6 in number and they are:
  - **Customer Communication** - tasks required to establish effective communication between the developer and customer.
  - **Planning** - tasks required to define resources, timelines and other project related information /items.
  - **Risk Analysis** - tasks required to assess the technical and management risks.
  - **Engineering** - tasks required to build one or more representation of the application.
  - **Construction & Release** - tasks required to construct, test and support (eg. Documentation and training)
  - **Customer evaluation** - tasks required to obtain periodic customer feedback so that there are no last minute surprises.

# When to use Spiral model


- ▶ When costs and risk evaluation is important
  - ▶ For medium to high-risk projects
  - ▶ Long-term project commitment unwise because of potential changes to economic priorities
  - ▶ Users are unsure of their needs
  - ▶ Requirements are complex
  - ▶ New product line
  - ▶ Significant changes are expected (research and exploration)
- 

# The Spiral (Cont')

## Advantages

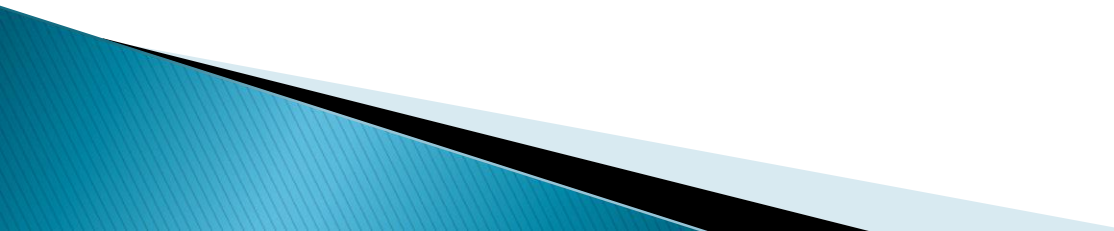
- ▶ High amount of risk analysis
  - ▶ Good for large and mission-critical projects.
  - ▶ Software is produced early in the software life cycle.
- 

# Disadvantages

- ▶ One should possess considerable risk-assessment expertise.
  - ▶ It has not been employed as much proven models (e.g. the Waterfall Model) and hence may prove difficult to 'sell' to the client.
  - ▶ Can be a costly model to use.
  - ▶ Risk analysis requires highly specific expertise.
  - ▶ Project's success is highly dependent on the risk analysis phase.
  - ▶ Doesn't work well for smaller projects.
- 



# Component – based development

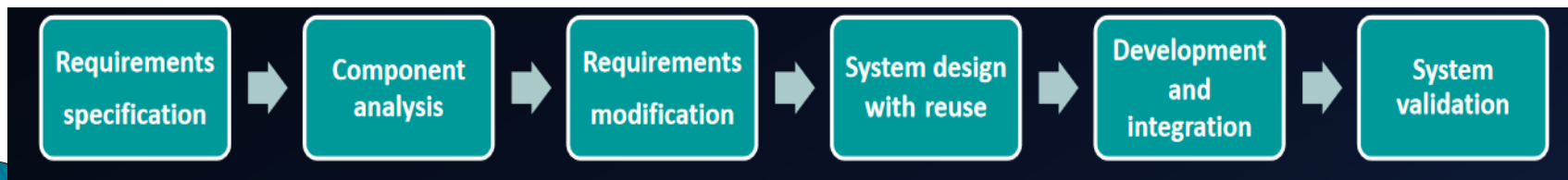
- ▶ **development**—the process to apply when **reuse** is a development objective
  - ▶ The model composes applications from **prepackage** software components
  - ▶ Components can be designed as either **conventional software module** or **OO classes** or Package of classes.
- 

# Component – based development

- A software process is represented as a set of work phases that is applied to design and build a Software product.
- To represent software processes, we will use different types of models which are called software process models. Each model represents a process from a specific perspective.
- The three most widely used software process models are :-
  - ☐ The Waterfall Model
  - ☐ Evolutionary Development
  - ☐ Component-based Software Engineering (CBSE)

# CBSD

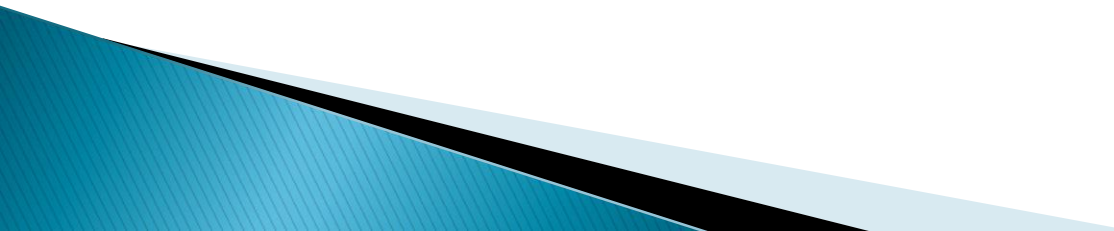
- ▶ Component-based software engineering (CBSE) is an approach to software development that is based on the existence of a significant number of reusable components.
- ▶ It focuses on integrating these components into a system rather than developing them from scratch.
  - A generic process model for CBSE is given below:-



# Stages of CBSE

- ▶ **Requirements specification** : A software requirements specification (SRS) is a document that captures complete description about how the system is expected to perform.
- ▶ **Component analysis**: A search is made for the components to implement the given requirements specification.
- ▶ **Requirements modification**: During this phase, the requirements are analyzed using information about the components that have been discovered. They are then modified to reflect the available components. If the modifications are impossible, the component analysis activity may be re-entered to search for alternative solutions.

# Con't

- ▶ **System design with reuse:** During this phase, the framework of the system is designed or an existing framework is reused. The designers take into account the components that are reused and they will organize the framework accordingly.
  - ▶ **Development and integration:** The components are integrated to create the new system. System integration, in this model, may be part of the development process rather than a separate activity.
  - ▶ **System Validation** is a set of actions used to check the compliance of any element with its purpose and functions. These actions are planned and carried out throughout the life cycle of the system.
- 

## ▶ **Advantages**

- Reduces development time
- Increases productivity
- Reduction in cost for development
- Reliability is increased
- Flexibility

## ▶ **Disadvantages**

- Development of components
  - Quality of components is questionable
  - Component maintenance costs
  - Sensitive to changes
- ▶ Sometime finding ideal/matching components can be quite challenging or impossible

Question??