Yrkes Akademin

Vi hjälper dig att lyckas!

Component Based Development

Programming for Automotive - Basics

Component Based Development

- Component based development
  - A reuse-based approach to software development for defining, implementing and composing loosely coupled independent components into software systems
  - It shifts the focus from programming to composing systems
    - Focus on integration instead of implementation
    - There is no concept of developing softwares from scratch
- What is a component?
  - A modular, portable, replaceable, and reusable part of a system that encapsulates its implementation and exposes a set of interfaces

Component Based Development

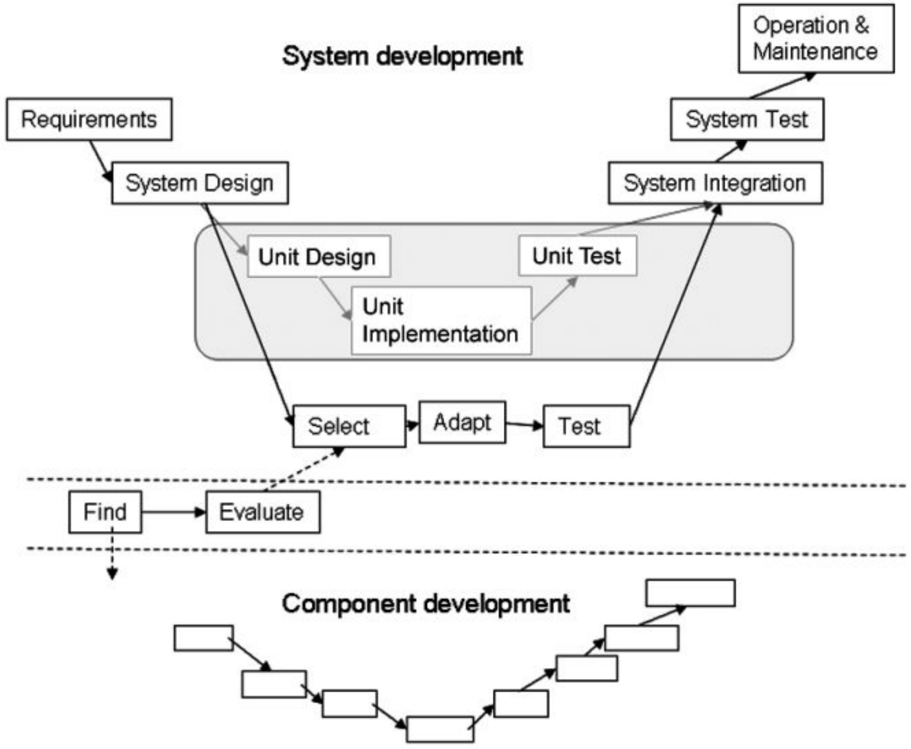
- Characteristics of Components
  - **Reusability** − They are designed to be reused in different applications
  - **Replaceable** − They may be freely substituted with other similar components.
  - **Extensible** − They can be extended from existing components to provide new behavior.
  - **Encapsulated** − They expose only the interfaces; not the details and implementation.
  - **Independent** − They are designed to have minimal dependencies on other components.
    - Stand-alone units and can be deployed and released independently
  - **Standard and well-defined interfaces** - They communicate to each other via interfaces.
  - **Customizable** - Parameterized interfaces make them customizable and adaptable
  - *Reusability implies **generality** and **flexibility**, and these requirements may significantly change the component characteristics*

Component Based Development

- Benefits of Components
  - **Faster development** - Separating development into components lets you build modular parts with smaller and focused APIs without impacting other parts of the system.
  - **Shortens the development time** - Code reuse saves you time.
  - **Reduced cost** − Buying a component costs lesser than developing a component.
  - **Easier TDD**: When building modular components it becomes much easier to implement unit-tests to validate the focused functionality of each component.
  - **Separation of concerns -** Each component addresses a specific concern, this simplifies the development, debugging and update cycle
  - **Reliability** - The overall system reliability increases since the reliability of each individual component is enhanced
  - **Easier for maintenance and evolution** - Easier to change/update/upgrade a component without affecting the rest of the system.

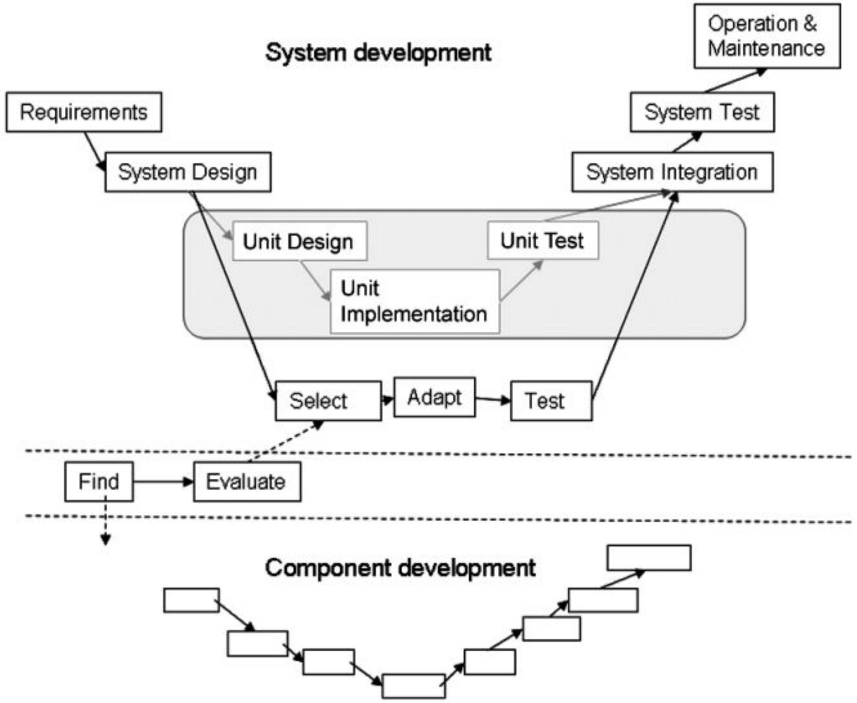
Component Based Development

- Development of Components
  - Components must be **documented**, **designed**, **implemented**, **verified**, **validated** and **deployed**
  - A **component model** defines standards for:
    - Properties of individual components must satisfy
    - Methods and mechanisms for composing
  - An adapted V-model development is used
    - Requirements Phase
    - Analysis & Design Phase
    - Implementation & Unit Testing Phase
    - Integration & Integration Testing Phase
    - Release Phase (delivery)
    - Maintenance Phase (repair, update, upgrade)



Component Based Development

- Component Based System Development
  - The main idea is building systems from existing components
  - Focus is on finding reusable component and relations between them
  - Is different from component development
  - An adapted V-model development is used:
    - Requirements Phase
    - Analysis & Design Phase
    - **Find, Verify and Store Components**
    - **Select, Adapt and Validate components**
    - Integration & Integration Testing Phase
    - System Testing Phase
    - Maintenance Phase (repair, update, upgrade)



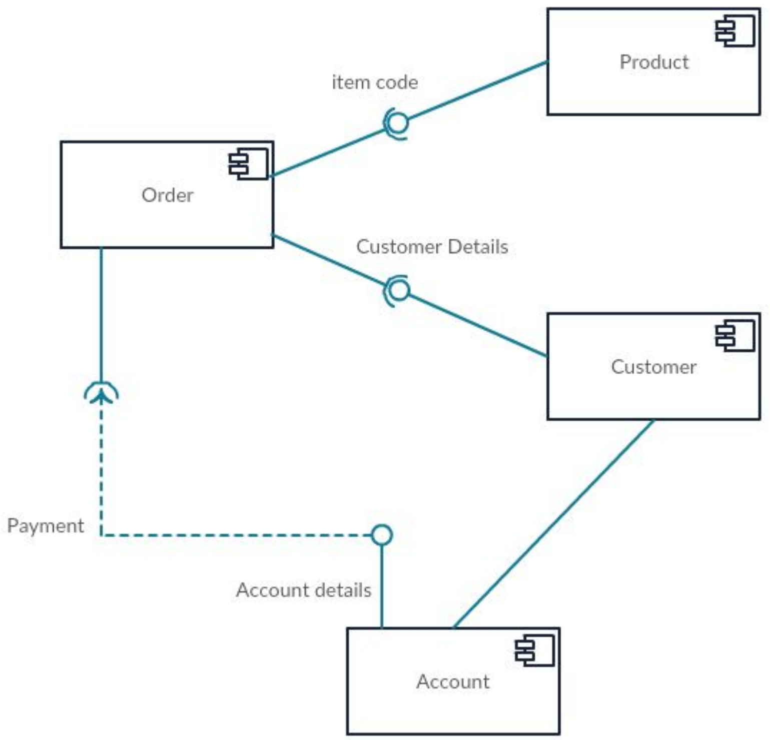
Component Based Development

- Unified Modeling Language (UML)
  - A standardized modeling language consisting of an integrated set of diagrams
  - Developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of softwares

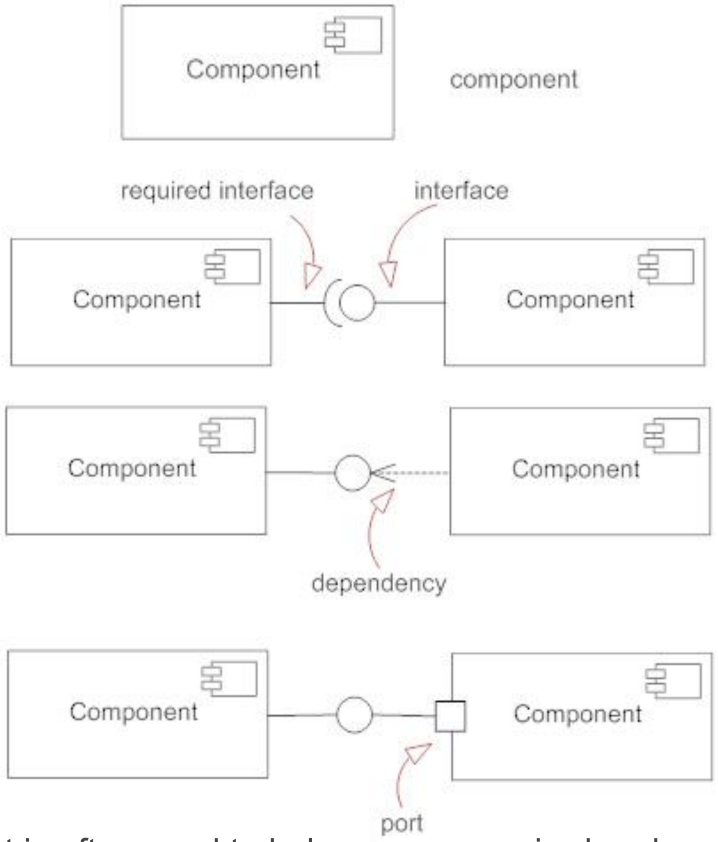
- Types of UML Diagrams
  - There are some types of UML diagrams and each of them serves a different purpose
  - The two most used types are
    - **Behavioral UML diagram:** Used to describe and visualize the behavior, actors and building components of a system
      - E.g. Activity Diagram, Timing Diagram, State Machine Diagram and etc.
    - **Structural UML diagram:** Used to describe and visualize the structure of a system or process
      - E.g. **Component Diagram**, Object Diagram, Class Diagram and etc.

Component Based Development

- UML Component Diagram Example



Component diagram of an online shopping system



A port is often used to help expose required and provided interfaces of a component

Component Based Development

- Some useful links
  - Component-based Development
  - Modular Design
  - Component-based software engineering
  - The 5 Essential Elements of Modular Software Design
  - Component Based Software Engineering
  - The Easy Guide to Component Diagrams
  - What is Component Diagram?
  - UML Component Diagrams
  - Component Diagram Tutorial
  - All You Need to Know About UML Diagrams
  - UML component diagram
  - UML 2 Component Diagrams