

# Software Architecture and Design I

## SOEN 343

Instructor: Dr. Rodrigo Morales

<https://moar82.github.io/>

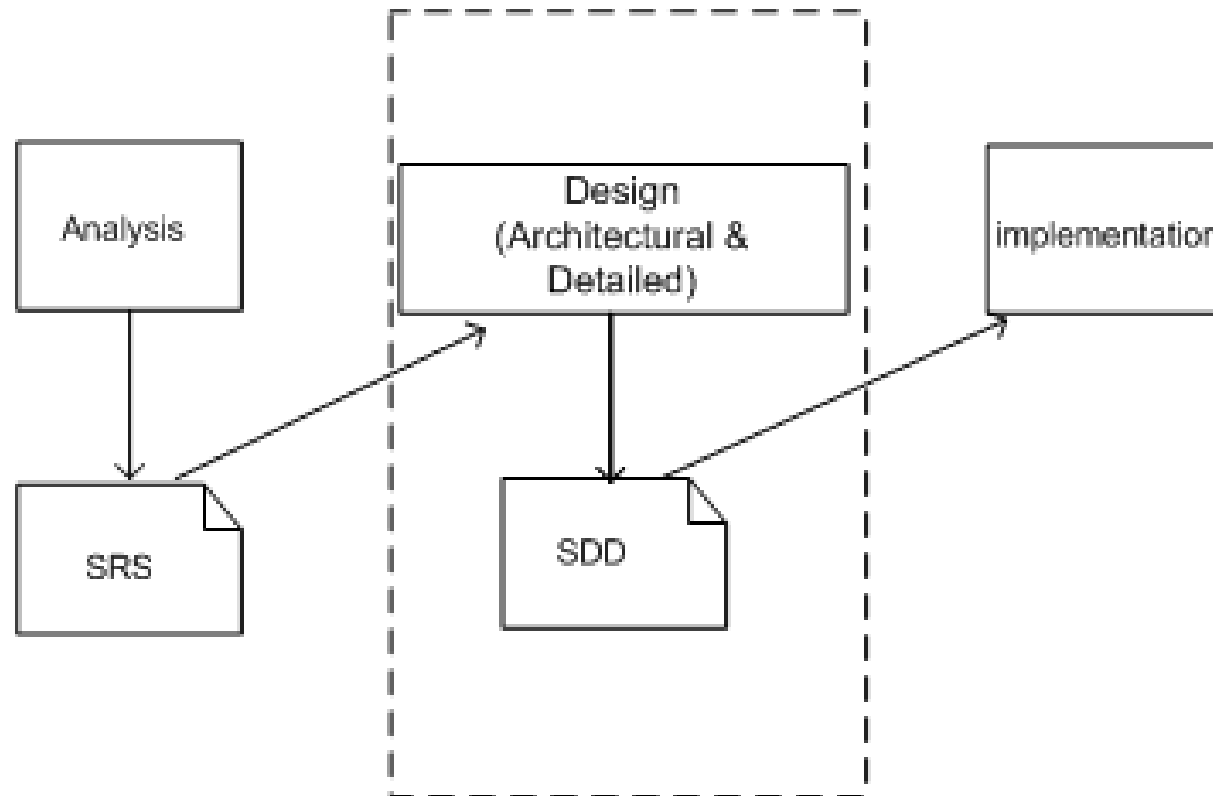
rodrigo.moralesalvarado@concordia.ca

## Lecture 2a: Models for Software Architecture


# Objectives

- Introduce concepts of the view models of software architecture
- Discuss the UML notations as modeling tools for software architecture specification
- Introduce UML Static and behavioral diagrams
- Introduce the concept of architectural models
- Introduce the 4+1 Model

# A simplified software development life cycle

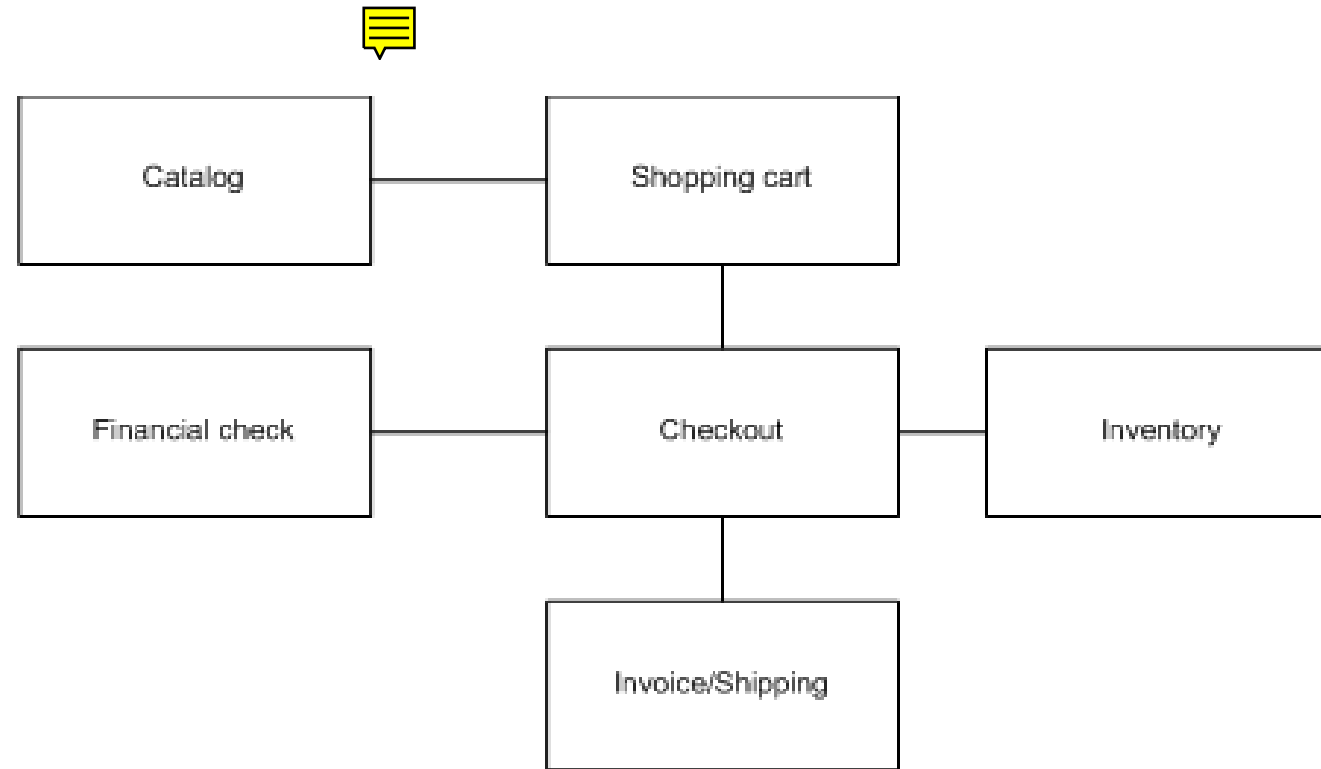


# Software architect tasks

- Specify at high-level of  abstraction a software system by employing decomposition, composition, architectural styles and quality attributes



# Example



# UML for Software Architecture

- UML is a graphical language for visualizing and documenting the artifacts of a software-intensive system
- UML offers a standard way to draw a system's blueprints
- It is based on OO analysis and design



# UML (static) structural diagrams

- Class
- Object
- Composite structure
- Component
- Package
- Deployment

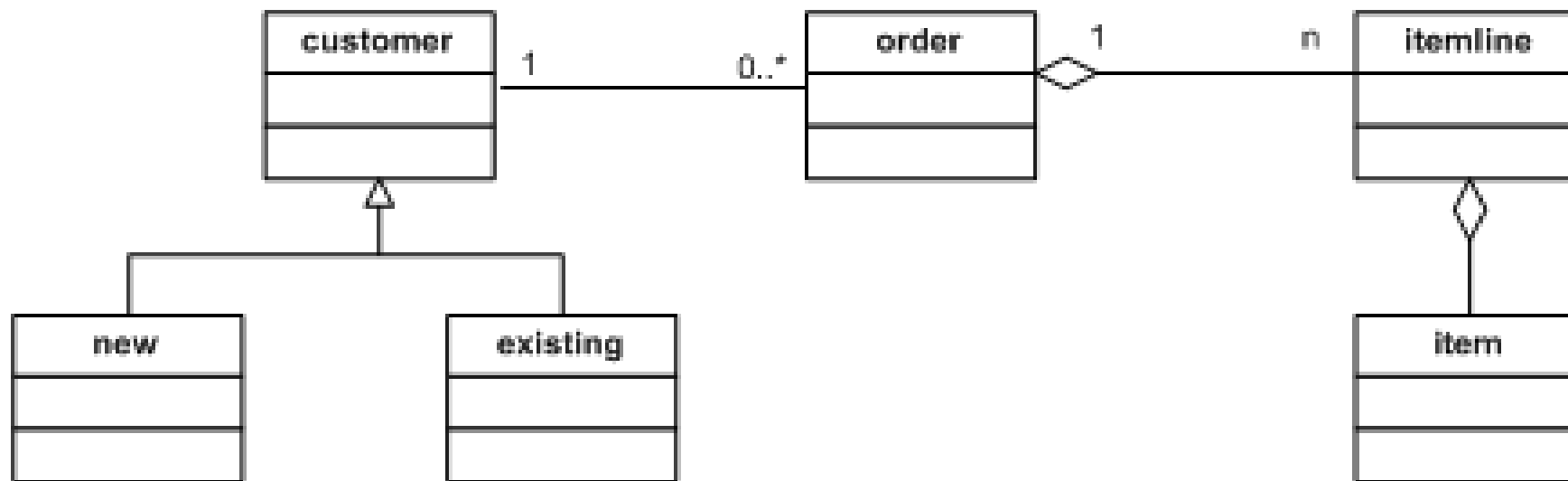
# UML Behavioral (dynamic) diagrams

- Use case
- Activity
- State Machine
- Sequence
- Interaction Overview
- Communication
- Time Sequence

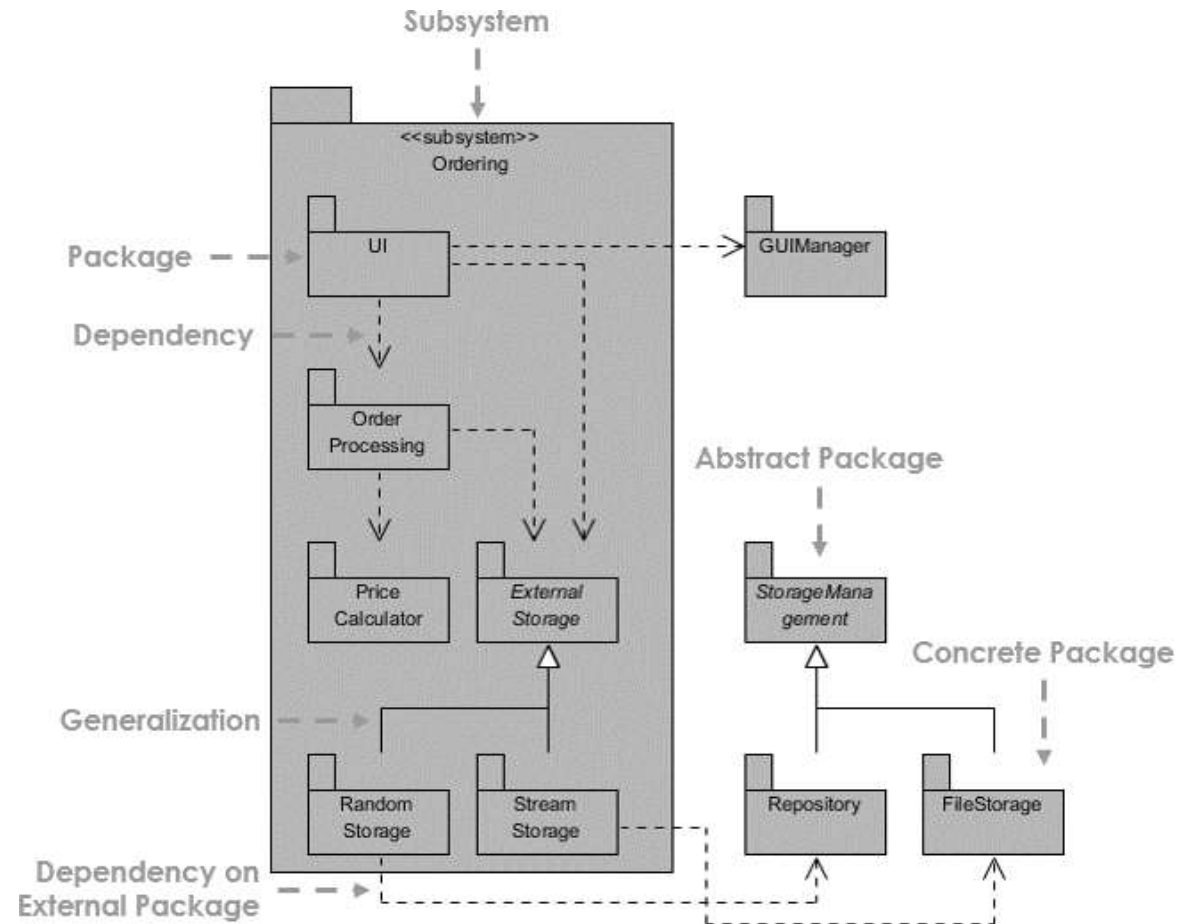


# UML (static) structural diagrams

# Class Diagram

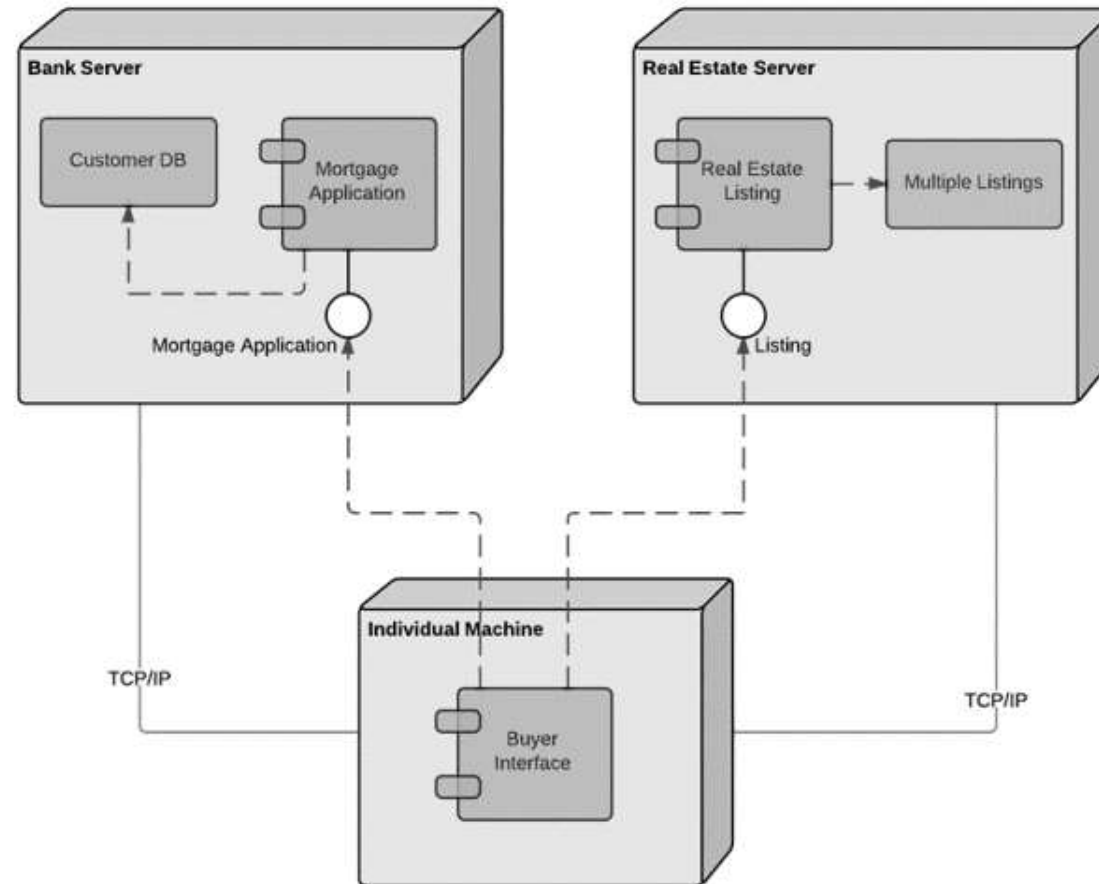


# Package Diagram



<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-package-diagram/>

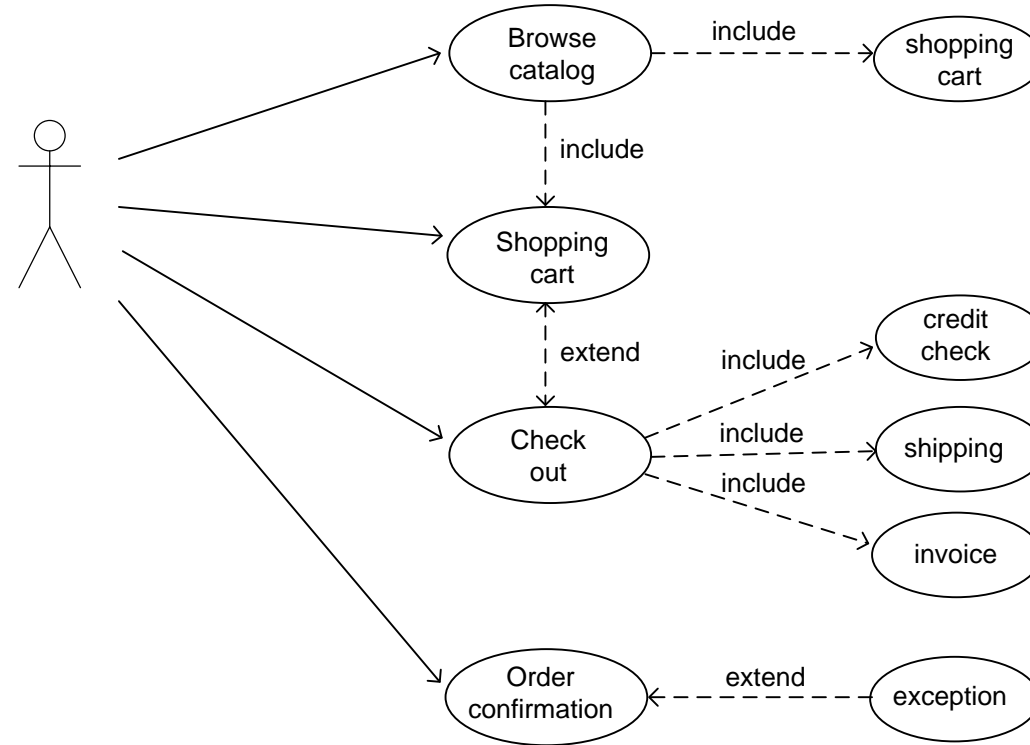
# Deployment Diagram



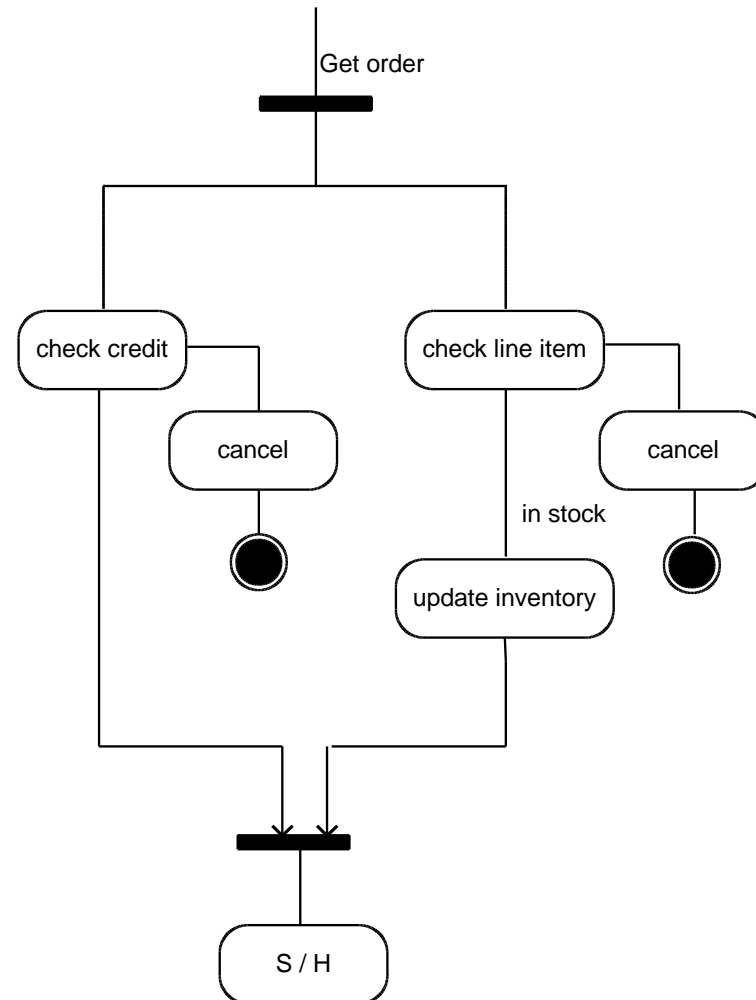
<https://www.lucidchart.com/pages/uml-deployment-diagram>

# Behavioral Diagrams

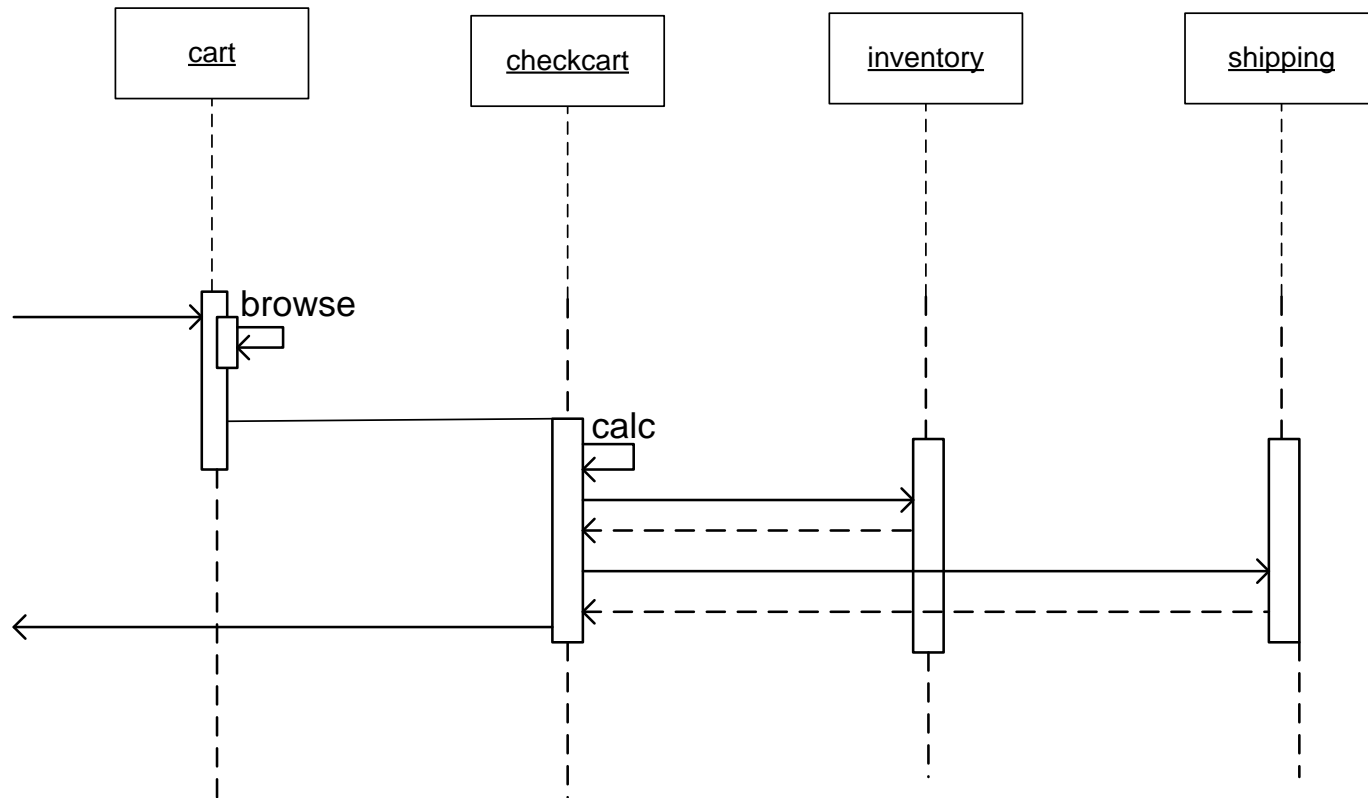
# Use Case Diagram



# Activity Diagram



# Sequence Diagram



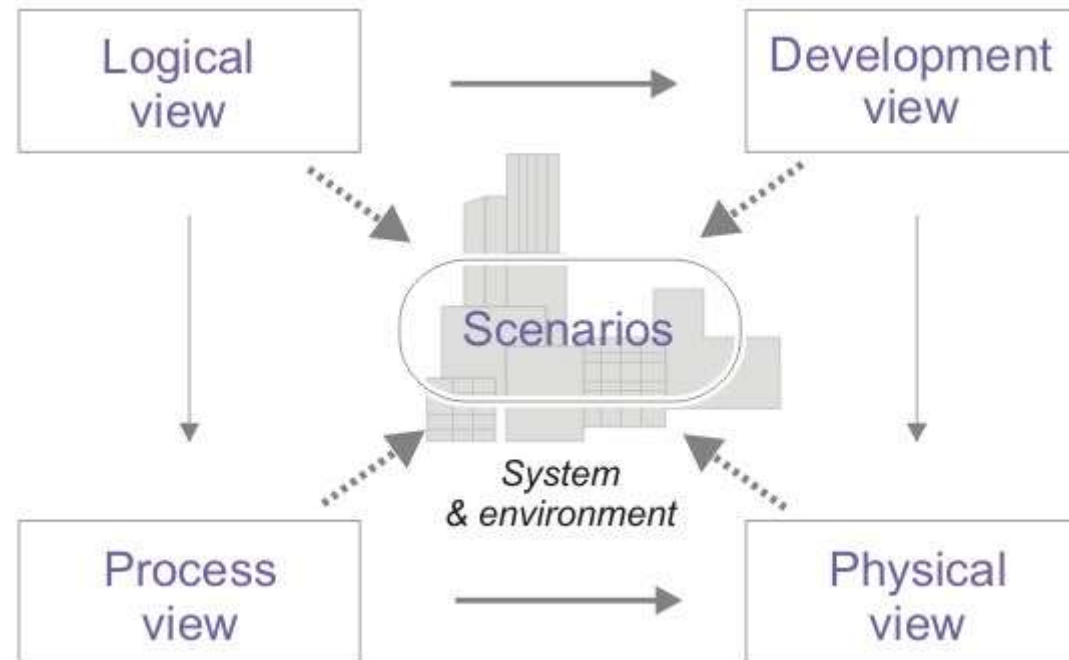


# Architecture View Models

# Architecture View Models

- A model is a simplified description of a system from a particular perspective
- There is no single view that can present all aspects of complex software to stakeholders
- Specific views models provide partial representations of the software architecture to a specific stakeholder:
  - system users, the analyst/designer, the developers, the system integrator, etc.

# 4+1 Model by Krutchen (1995)



# The Logical or Conceptual View



- The view basically is an abstraction of the system's functional requirements
- It specifies system decomposition into conceptual entities (objects), and connection between them (associations)
- The logical view is supported by UML diagrams such as class, sequence diagram, state, and activity diagram
- The stakeholders of the logical view are the end-user, analysts, and designers

# The Development or Module View



- The development view describes the software static organization of the system modules, which it's derived from the logical view
- Modules such as namespaces, class library, sub-system, or packages are building blocks that group classes for further development and implementation
- This view addresses the sub-system decomposition and organizational issue
- UML package diagrams are used to support this view
- The stakeholders of this view can be programmers and software project manager

# The Process View

- The process view focuses on the dynamic aspects of the system, i.e., its execution time behavior
- This view is an abstraction of processes or threads dealing with process synchronization and concurrency
- It contributes to many non-functional requirements and quality attributes such as scalability and performance requirements
- The UML activity diagram support this view
- The stakeholders of this view are the developers and integrators
- Many architectural styles such as pipe & filter, multi-tier, and others can be applied in the process view

# The Physical View



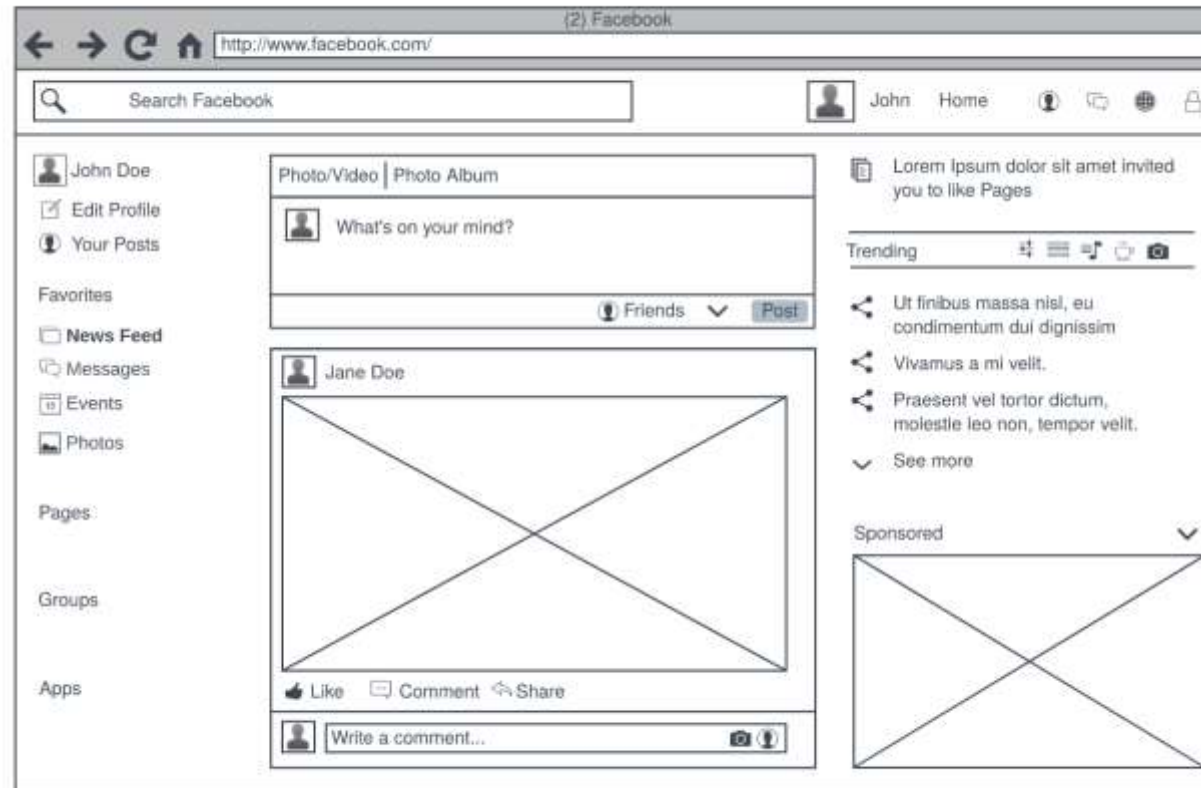
- The physical view describes installation, configuration, and deployment of the software application
- It concerns itself with how to deliver the deployable system
- The physical view shows the mapping of software onto hardware
- It is particularly of interest in distributed or parallel systems
- UML deployment diagram support this view
- The stakeholders of this view are system installers, system administrators, system engineers, and operators

# The scenario view

- The scenario view describes the functionality of the system
- This view provides a foundation for other 4 views and lets them work together seamlessly and coherently
- It helps designers discover architectural elements during design and helps to validate the architectural design afterward
- The scenario view helps to make the software architecture consistent and validated against functional and non-functional requirements
- This view may be provided as a series of screen snapshots or a dynamic, interactive prototype demo or wireframe diagram



# Wireframe diagram example:



# 4+1 View Summary

- The view is an architecture verification technique for studying and documenting software architectural design
- Each view provides a window for the different aspects of the system
- It covers all aspects of a software architecture for all stakeholders
- The views are interconnected; thus, based on the scenarios view, we can start with logical view, and then move to development or process view and, finally go to physical view
- The user interface view is also established during this process

# References

1. Chapter 3. Qian, Kai, et al. Software architecture and design illuminated. Jones & Bartlett Learning, 2010
2. P. B. Kruchten, "The 4+1 View Model of architecture," in *IEEE Software*, vol. 12, no. 6, pp. 42-50, Nov. 1995
3. <https://www.uml.org/>

# Appendix A. UML diagrams

## 1. Structural (Static) Diagrams

Diagram	Description
Class	An overview of classes for modeling and design. It shows how classes are statically related, but not how classes dynamically interact with each other.
Object	Objects and their relationship at a runtime. An overview of particular instances of a class diagram at a point of time for a specific case. It is based on the class diagram.
Composite Structure	Describes the inner structure of a component including all classes within the component, interface of the component, and etc.
Component	Describes all components in a system, their interrelationships, interactions, and the interface of the system. It is an outline of composition structure of components or modules.
Package	Describes the package structure and organization. It covers classes in the package and packages within another package.
Deployment	Describes system hardware, software, and network connections for distributed computing. It Covers server configuration and network connections between server nodes in real-world setting.

## 2. Behavioral (Dynamic) Diagrams

Diagram	Description
Use case	Derived from use case study scenarios. It is an overview of use cases, actors, and their communication relationships to demonstrate how the system reacts to requests from external users. It is used to capture system requirements.
Activity	Outline of activity's data and control flow between related objects. An activity is an action for a system operation or a business process, such as those outlined in the use case diagram. It also covers decision points and threads of complex operational processes. It describes how activities are orchestrated to achieve the required functionality.
State Machine	Describes the life cycle of an objects using a finite state machine. The diagram consists of states and the transitions between states. Transitions are usually caused by external stimuli or events. They can also represent internal moves of the object.
Sequence	Describes time sequence of messages passed between objects in timeline.
Interaction Overview	Combines activity and sequence diagrams to provide control flow overview of the system and business process.
Communication	Describes sequence of messaging passing among objects in the system. Equivalent to sequence diagram, except that it focuses on the object's role. Each communication link is associated with a sequence order number plus the passed messages.
Time Sequence	Describes the changes in state, condition, events over time by messages.