SFWR ENG 3A04: Software Design II

Dr. Ridha Khedri

Department of Computing and Software, McMaster University
Canada L8S 4L7, Hamilton, Ontario

Term 2

Acknowledgments: Material based on *Software Architecture Design* by Tao et al. (Chapter 3)

Outline of Part I

Outline of Part II

9 Introduction

10 UML for Software Architecture

- UML overview
  - Structural (Static) Diagrams
  - Behavioral (Dynamic) Diagrams

Part I

Review of Previous Lecture

Part II

Today's Lecture

Introduction to the concepts of the view models of software architecture

- Every software architecture must describe the collection of software components, connections and interactions between these components
- It has also to specify the configuration topology
- It MUST conform to the functional and non-functional requirements of the product

There are many effective ways to describe software architecture formally (ADL) or informally (UML)

- Box-and-line diagrams
  - Often used to describe the business concept and process at the analysis phase
  - Lines indicate the relationship among components (unlike UML, the semantics of lines may vary)
  - Lines may refer to dependency, control flow, data flow, and etc.
  - Lines may be associated with arrows to indicate the process direction and sequence

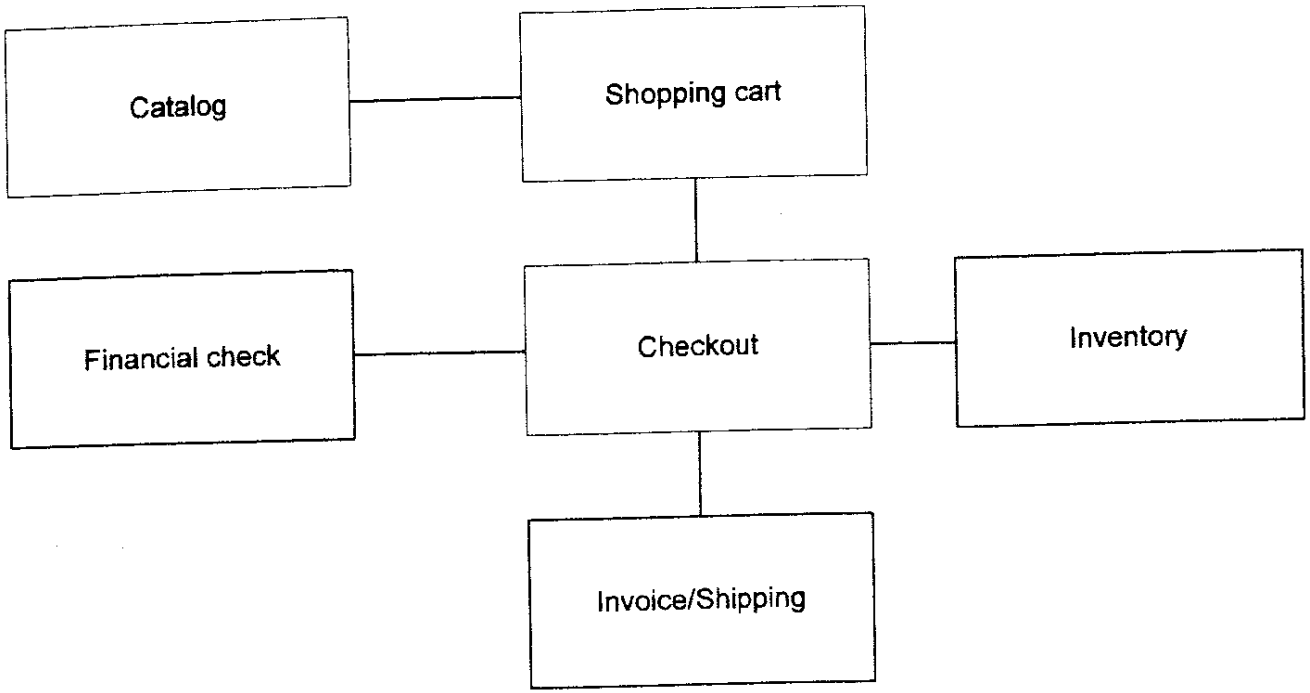

Figure: Block (box-and-line) diagram

- UML is one of the Object-Oriented solutions for software modeling and design
- The Architecture Description Languages (ADL) is another way to describe the software architecture formally and semantically
- The "4+ 1" view model is another way to show different views with different concerns for different aspects (F + NF Rqts)

- The "4+ 1" view model has 5 views:
  - Logical view: identifies software modules and their boundaries, interfaces, external environment, usage scenarios
  - Process view: addresses non-functional requirements such as module communication styles and performance issue at runtime environment
  - Development view: organizes the software units in a well defined ways according to the actual file or directory structure
  - Physical view: specifies the physical software, hardware, and networking node configuration, installation, and deployment for delivery purpose
  - User interface view: gives a look and feel view which may also impact other views

Models for Software Architecture

UML for Software Architecture (Overview)

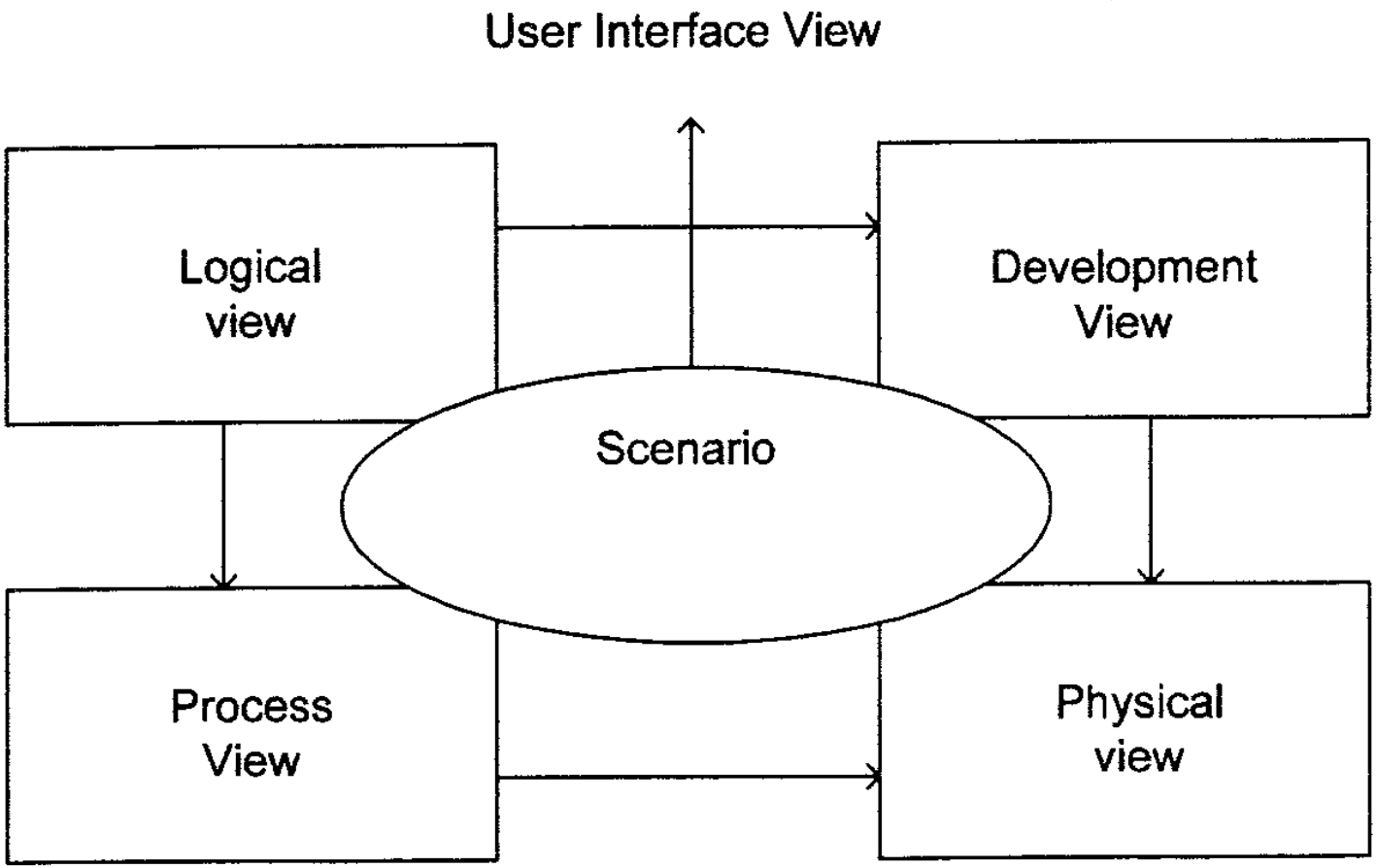

Figure: The "4+ 1" view model

UML for Software Architecture (Overview)

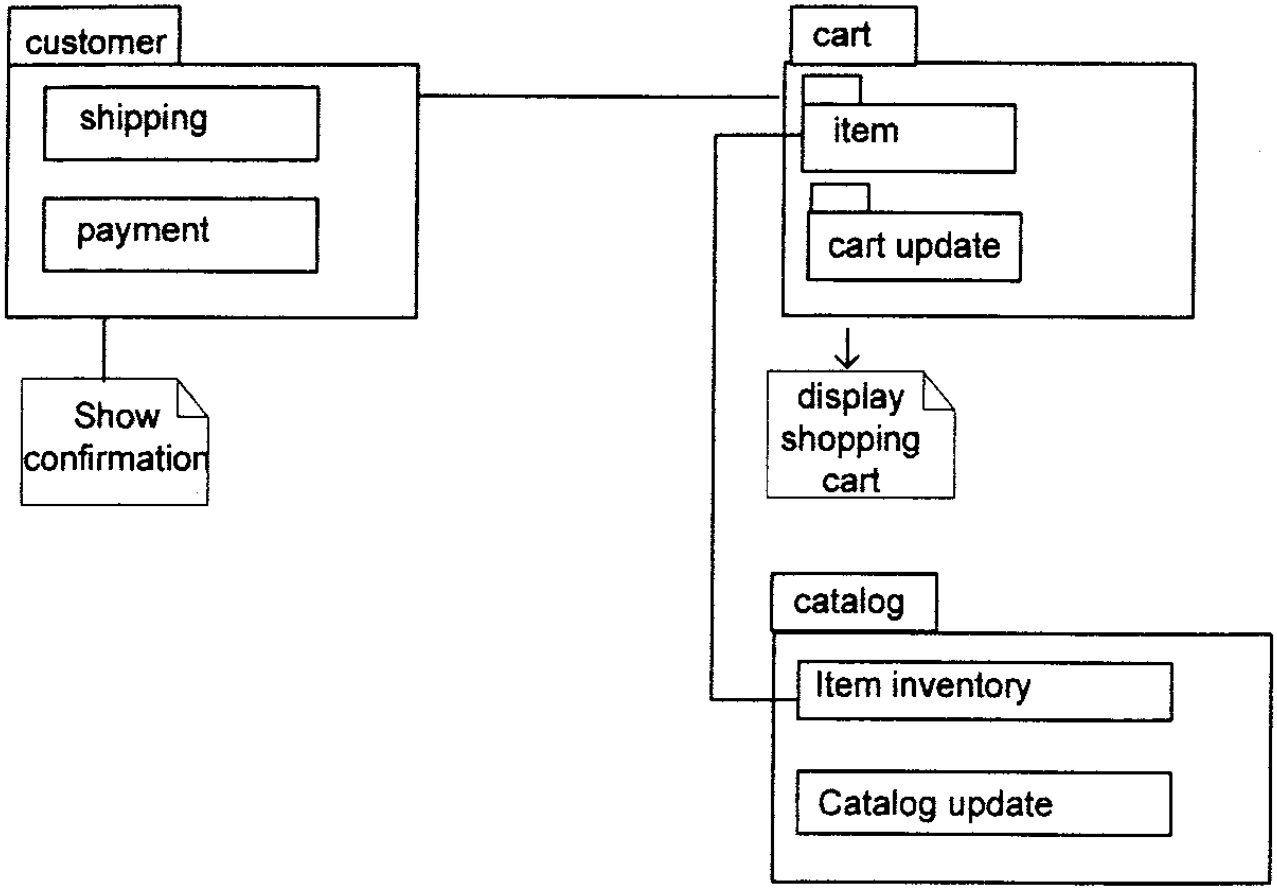

Figure: Package diagram in the development view

UML for Software Architecture (Overview)

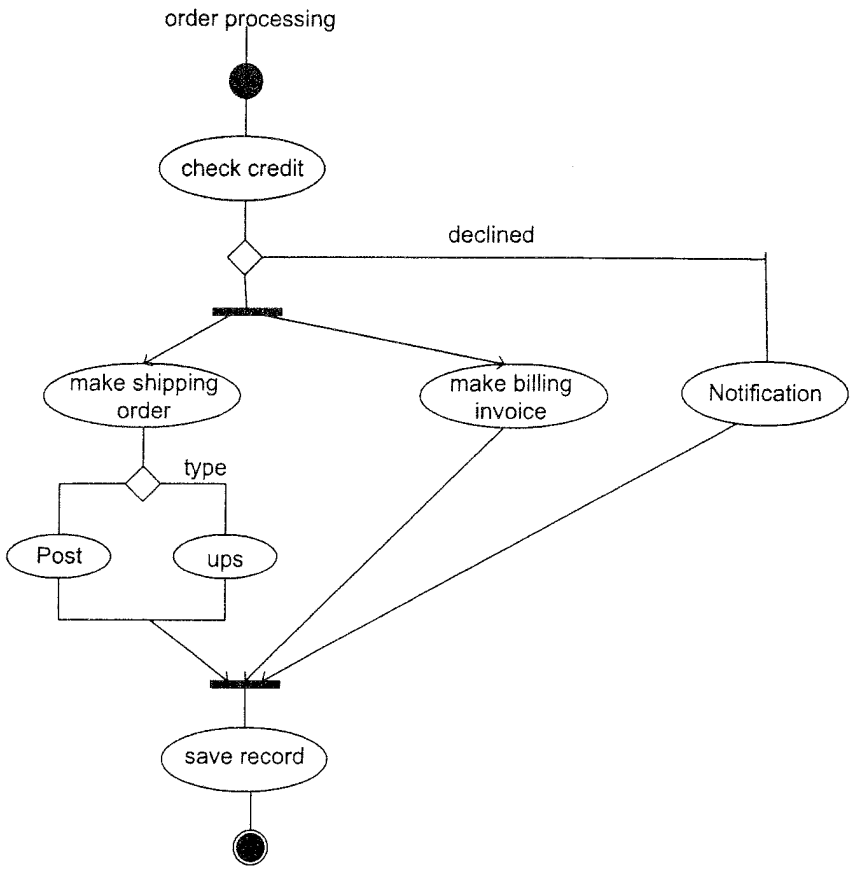

Figure: Activity diagram in the process view

UML for Software Architecture (Overview)

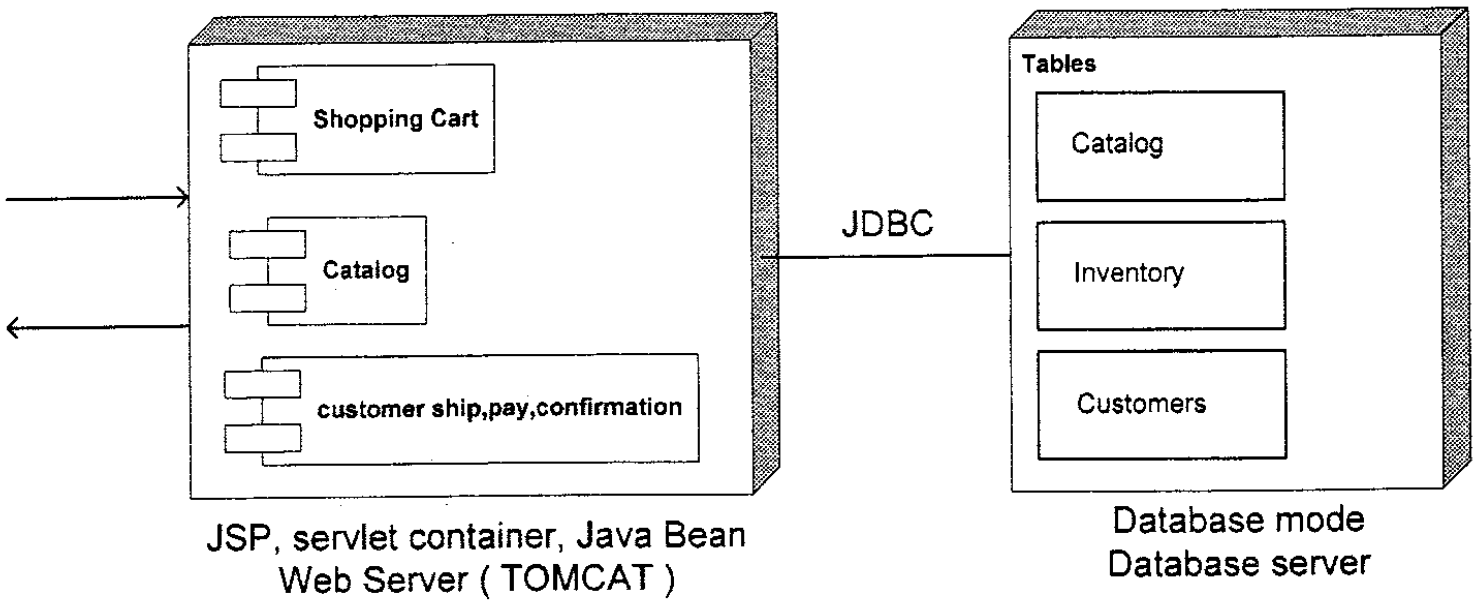

Figure: Deployment diagram in the physical view

UML for Software Architecture

- Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system
- It offers a standard way to write a system's blueprints (Business processes, functions, Prog. language, database schemas, etc.)
- It is a typical Object-Oriented analysis and design
- It provides many modeling diagrams which can be grouped into two major categories: Structural (static) and Behavioral (dynamic).

UML for Software Architecture

Structural software architecture describes the static structure of all software elements

- Class hierarchy
- Class library structure
- Relationships between classes
  - inheritance (is a)
  - aggregation (has a)
  - association (uses a)
  - messaging (method invocation)

UML for Software Architecture

- A static structural UML diagram depicts the control flow (time-independant) between software elements in the software system
  - Class diagram, component diagram, deployment diagram, etc.
  - All of them are independent from time
- Behavioral dynamic software architecture describes the behaviors of objects (i.e., instances of classes)
  - Object collaboration, interaction, activity, and concurrency
  - Exemples: sequence diagram, collaboration diagram, activity diagram

UML for Software Architecture

- They are many UML IDE (Integrated Development Environment) tools available (some are open source)
- The most popular UML tools are Rational Rose, Boland Together, and Microsoft Visio
- Some offer the capability of mapping from UML diagrams directly to coding framework in popular programming languages such as C++, C#, and Java

UML for Software Architecture (Overview)

Structural (Static) Diagrams

- Class Diagram :
  - Gives overview of classes for modeling and design
  - Shows how classes are statically related, but not how classes dynamically interact with each other
  - It is the foundation diagram of the system design
  - It is the most frequently used UML diagram
  - Class diagrams can be derived from use cases/Scenarios

UML for Software Architecture (Overview)

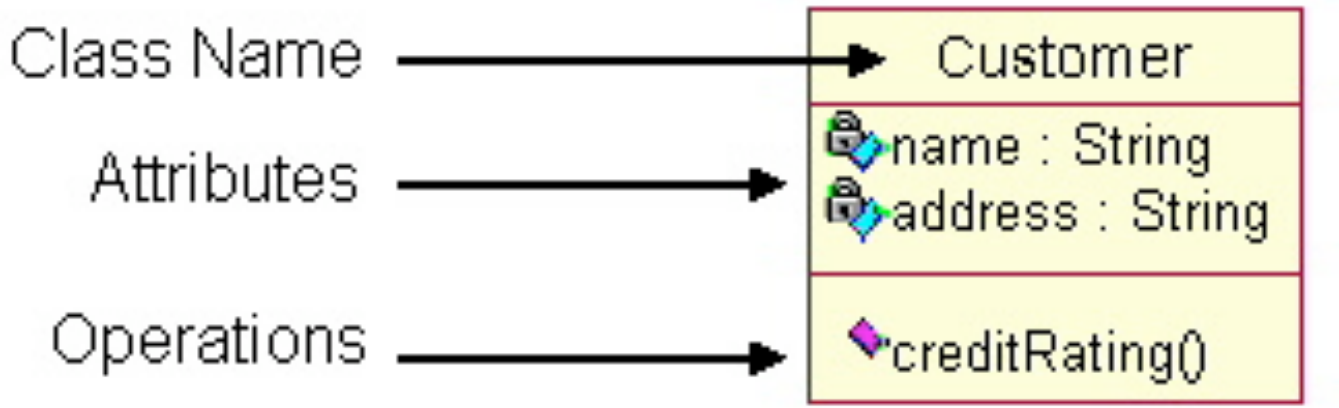

Figure: Elements of a class

UML for Software Architecture (Overview)

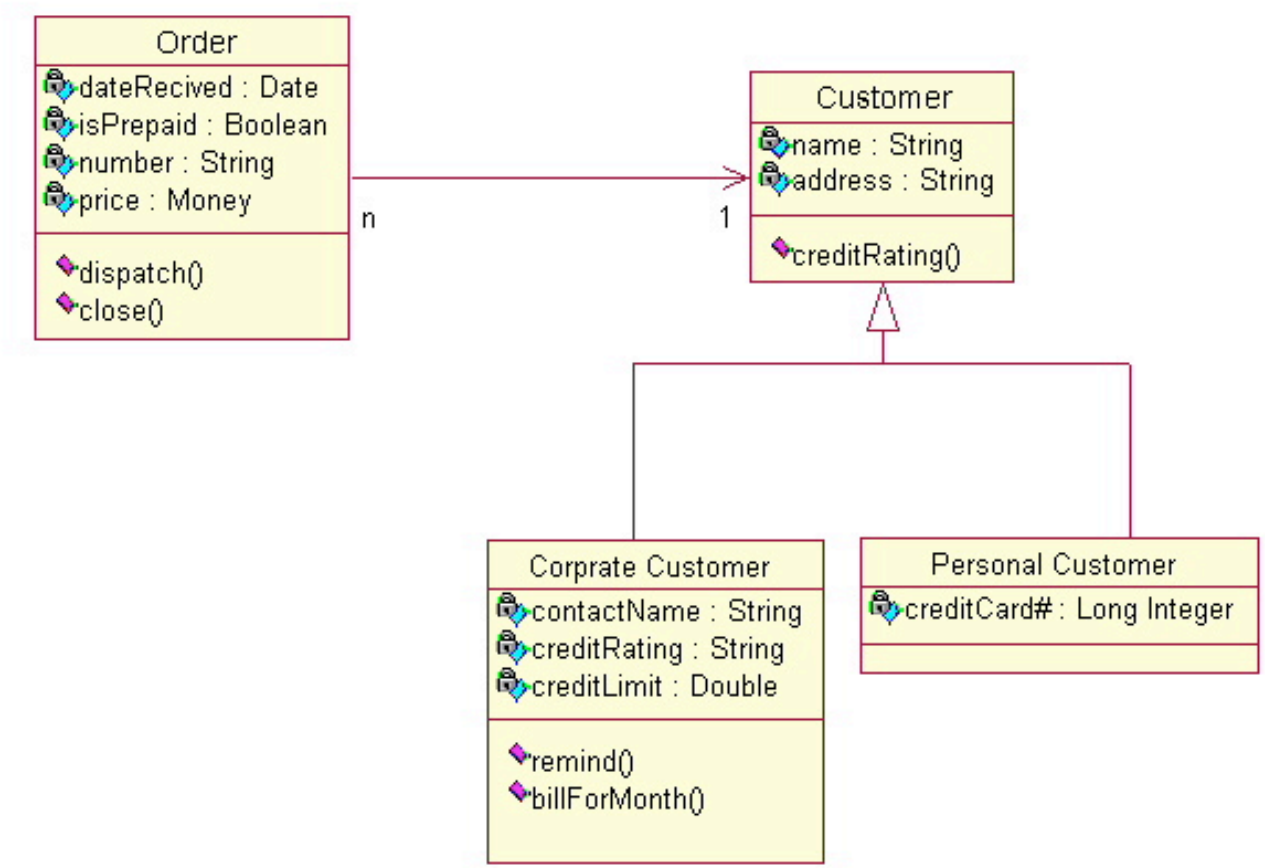

Figure: Class diagram (Example 1, different notation for composition)

UML for Software Architecture (Overview)

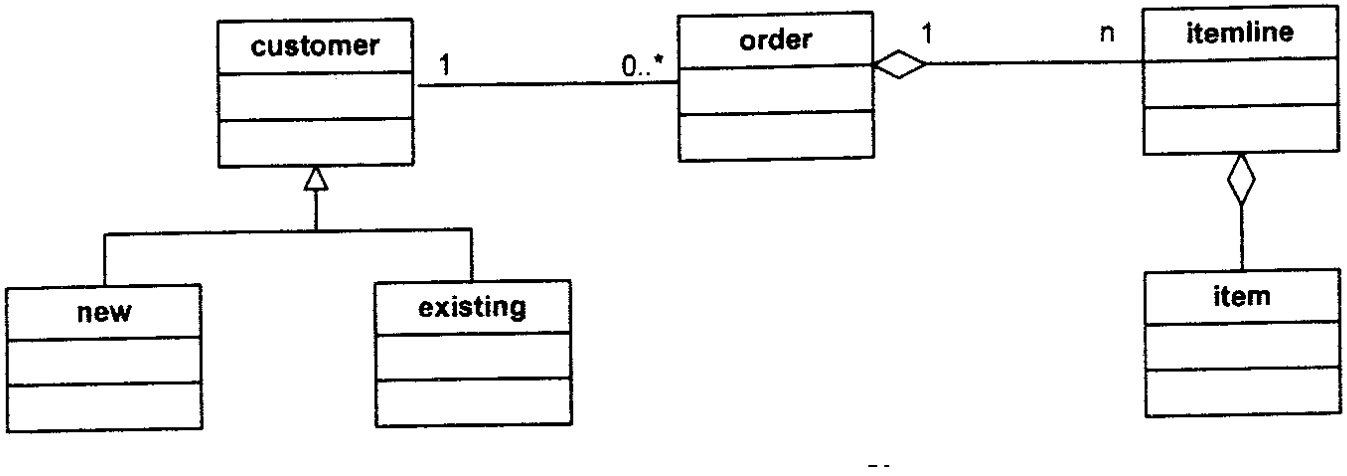

Figure: Class Diagram (Example 2)

UML for Software Architecture (Overview)

Structural Diagrams (Class Diagram):

- Relationships (connectors)
  - Composition/Aggregation (HAS A)
    - In composition, the components of a class HAVE the same lifespan as their owner
    - In aggregation, the components of a class DO NOT HAVE the same lifespan as their owner
    - In composition, components CANNOT be involved in another composition
    - In aggregation, components CAN be involved in another composition

UML for Software Architecture (Overview)

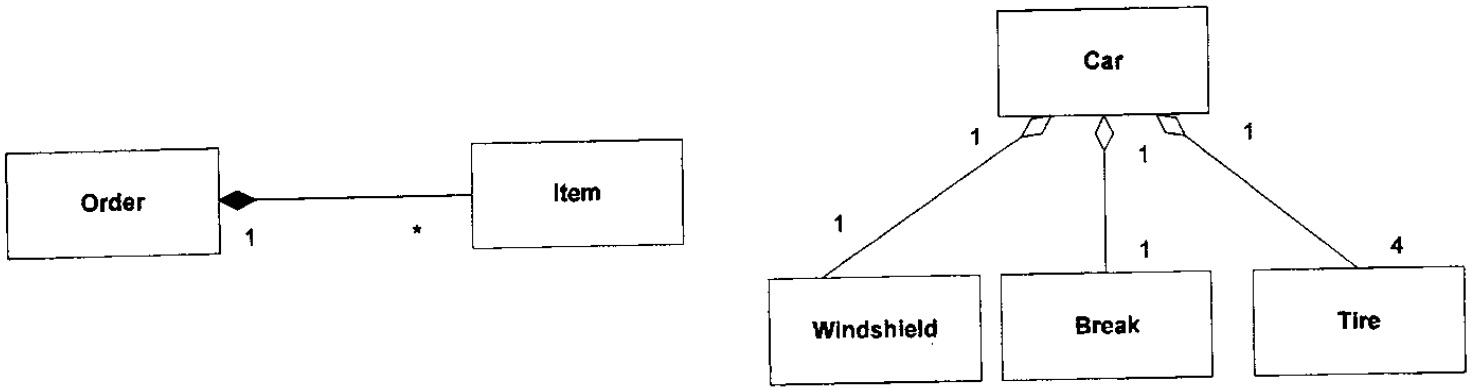

Figure: Composition (left) Aggregation (right)

UML for Software Architecture (Overview)

Structural Diagrams (Class Diagram):

- Relationships (Connectors) –Continued–
  - Association (USES A)
    - Association link has the following parts: name of the association, end type at each end of the association link, and multiplicity at each end
    - Composition can actually be regarded as one specific type of association
  - Dependency
    - A class X depends on another class Y, if changes to the elements Y will lead to the changes of X

UML for Software Architecture (Overview)

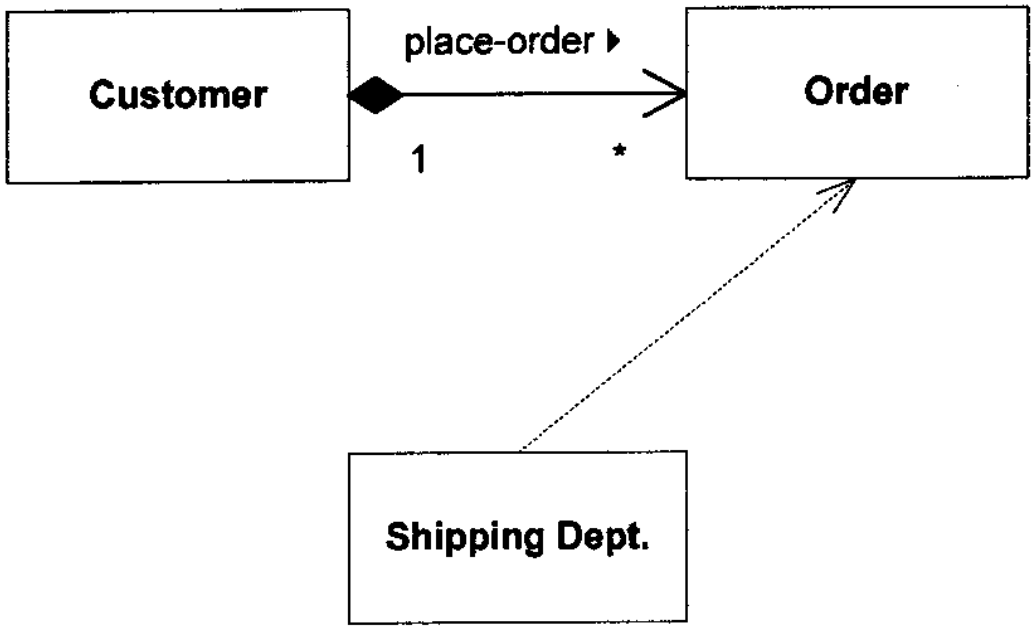

Figure: Association and Dependency (dotted arrow line)

UML for Software Architecture (Overview)

Structural Diagrams (Class Diagram):

- Relationships (Connectors) –Continued–
  - Inheritance (IS A)
    - Used when two or more classes have attributes and operations in common
    - When a class *A* inherits from a class *B*, A will inherit all attributes and operations of B unless otherwise specified
      (a private attribute will not be inherited by derived classes)
    - Be very careful about the use of inheritance (Weakens the encapsulation of an 00 design)

UML for Software Architecture (Overview)

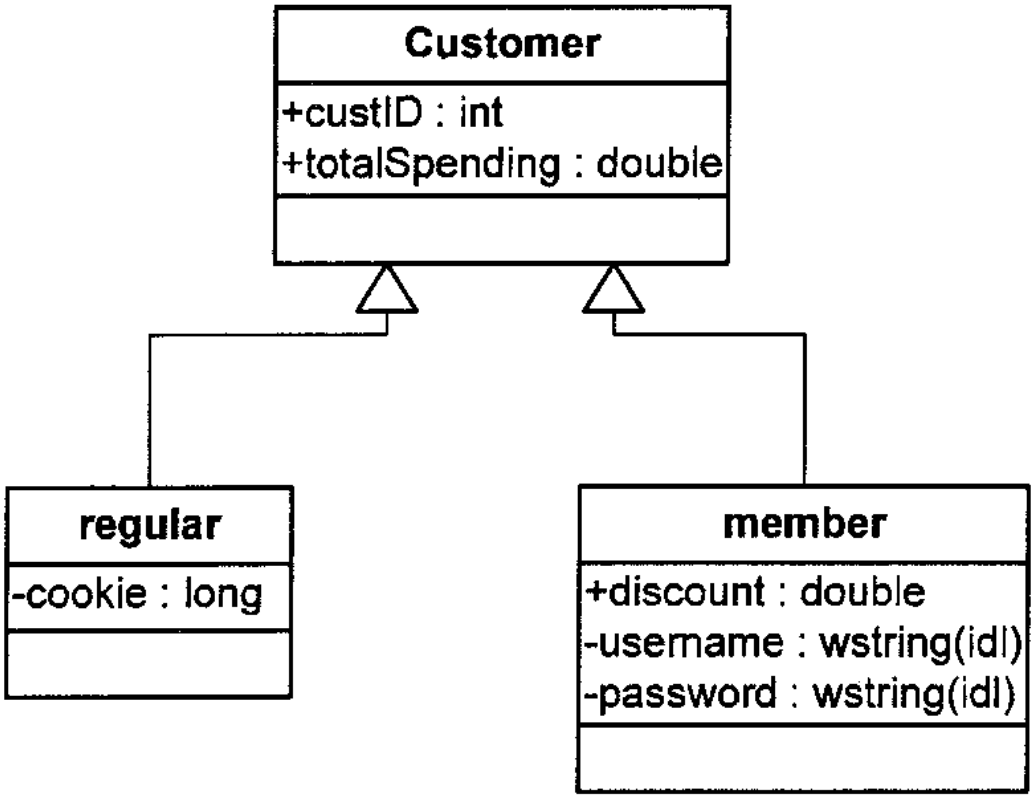

Figure: Inheritance relationship

UML for Software Architecture (Overview)

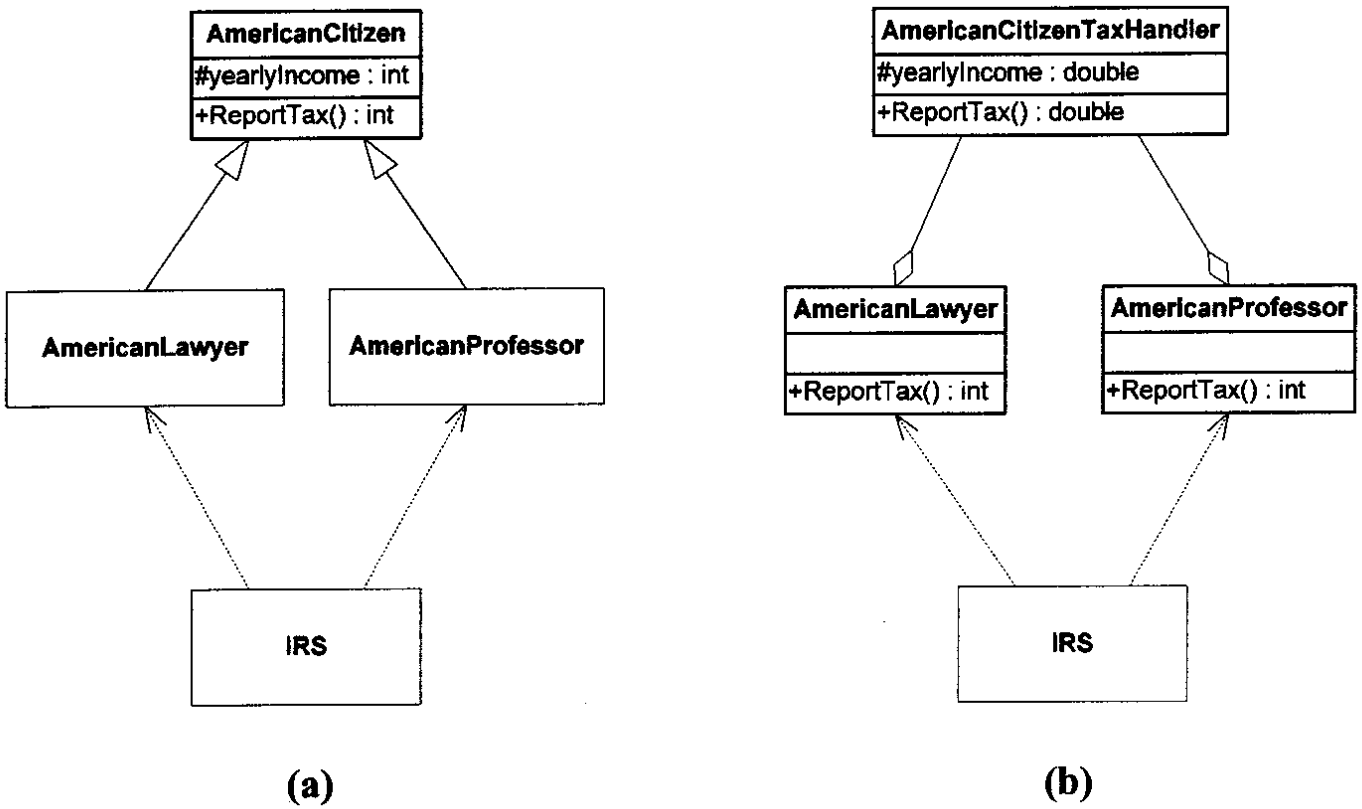

Figure: Composition vs. inheritance

UML for Software Architecture (Overview)

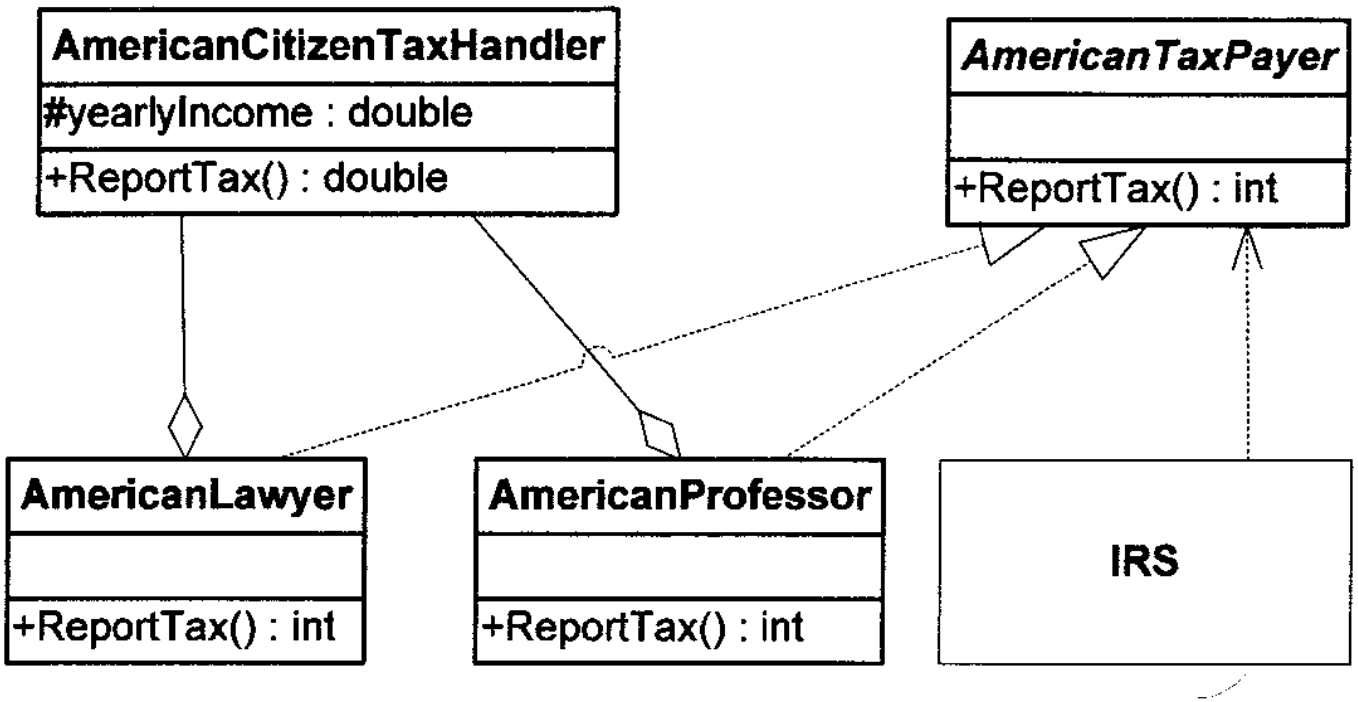

Figure: A refined design of the previous example

UML for Software Architecture (Overview)

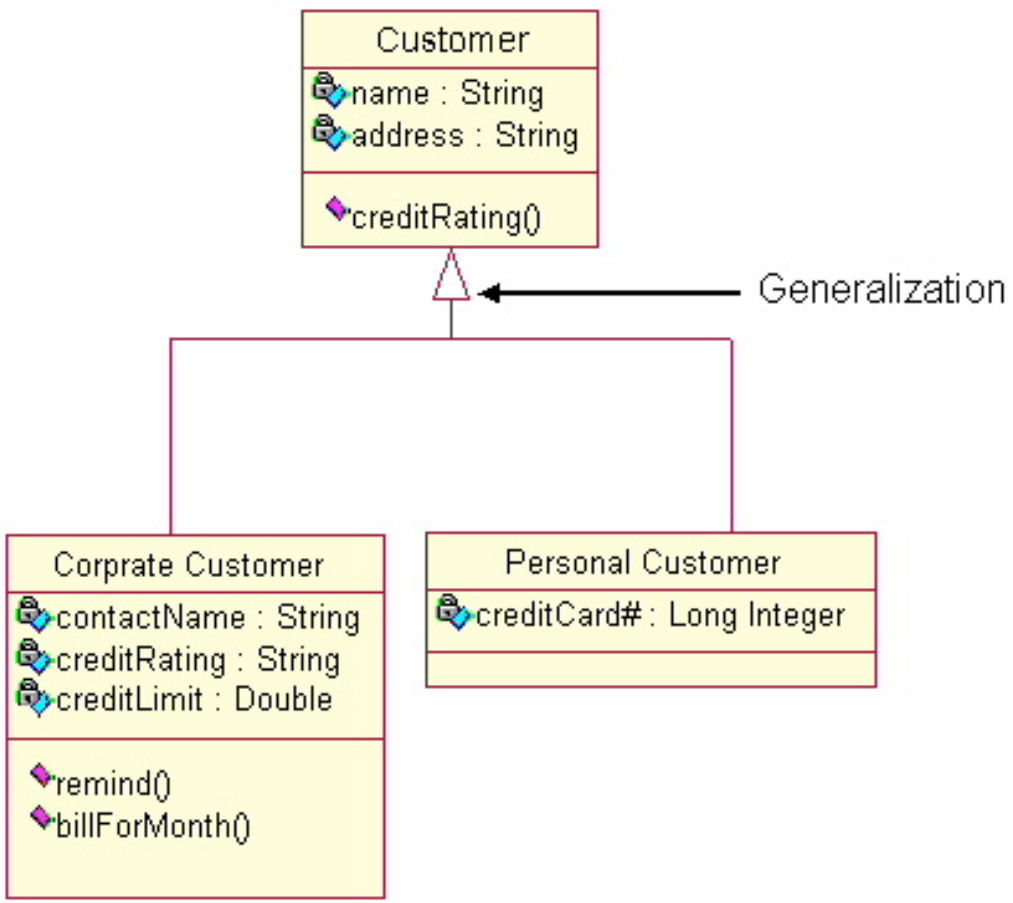

Figure: Class relationships: generalization

UML for Software Architecture (Overview)

Structural (Static) Diagrams

- Object Diagram:
  - Gives the objects and their relationship at a runtime
  - Presents 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

UML for Software Architecture (Overview)

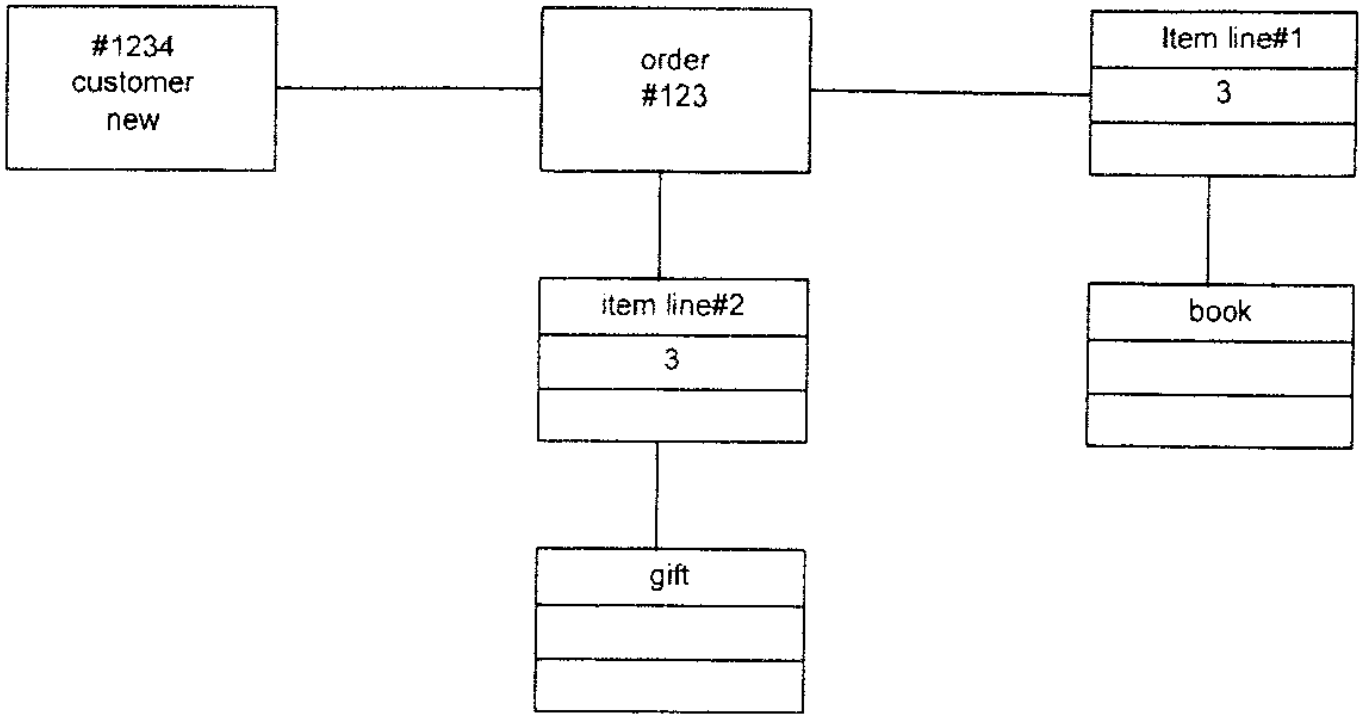

Figure: Object Diagram

UML for Software Architecture (Overview)

Structural Diagrams (Continued)

- Composite Structure Diagram:
  - Describes the inner structure of a component
    - all classes within the component
    - interface of the component
- Component Diagram:
  - Describs all components of a system
  - Gives their interrelationships, interactions, and their interface
  - It is an outline of composition structure of components or modules

UML for Software Architecture (Overview)

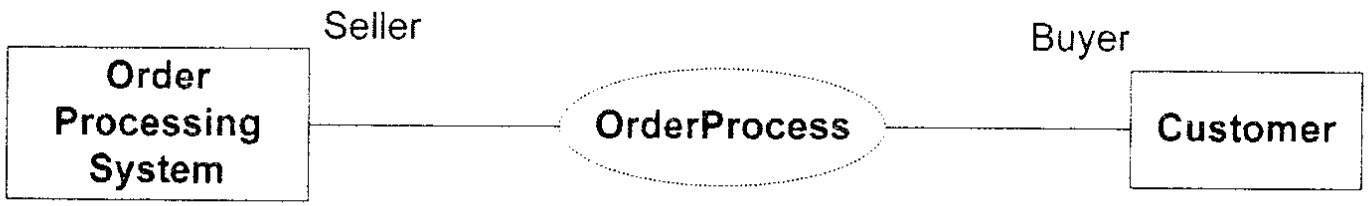

Figure: Composite Structure Diagram

UML for Software Architecture (Overview)

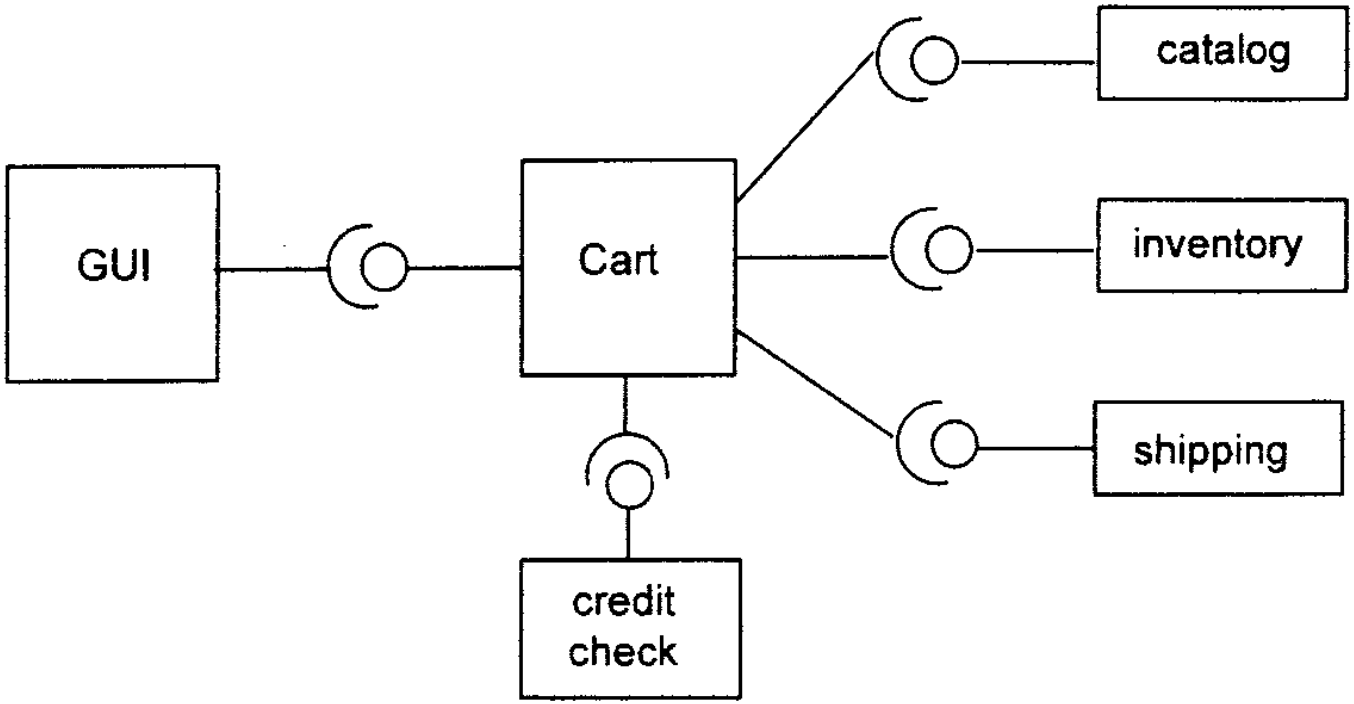

Figure: Component Diagram

UML for Software Architecture (Overview)

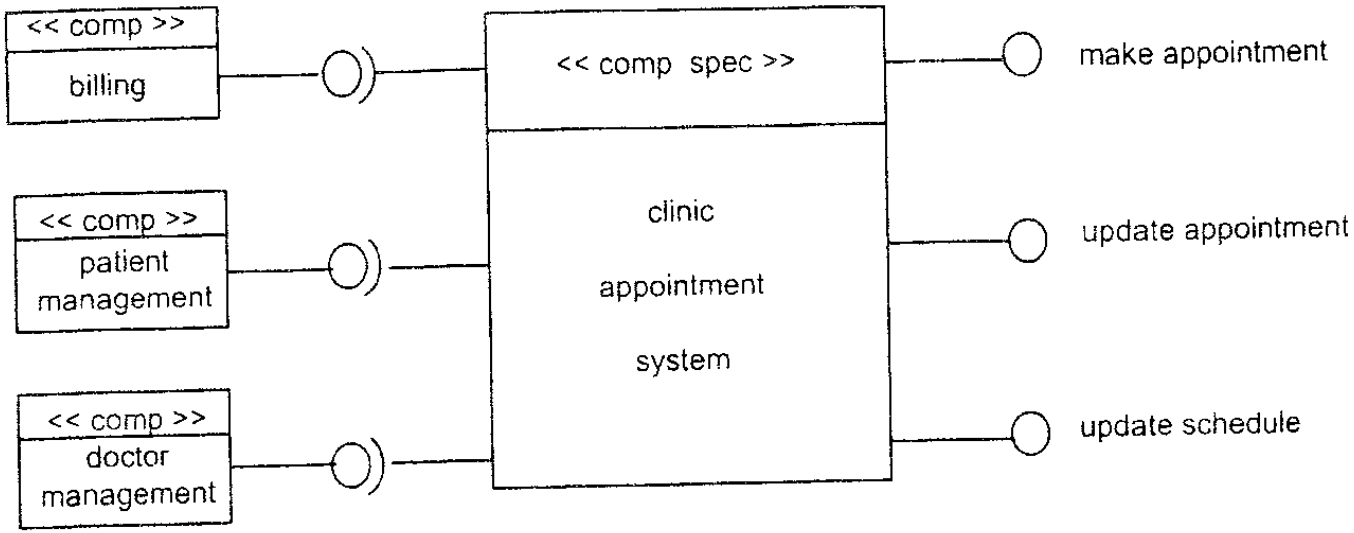

Figure: Component Diagram (Example 2)

UML for Software Architecture (Overview)

Structural Diagrams (Continued)

- Package Diagram:
  - Describes the package structure and their organization
  - Covers classes in the package and packages within another package
- Deployment Diagram:
  - Describes system hardware, software, and network connections for distributed computing
  - Covers server configuration and network connections between server nodes in real-world setting

UML for Software Architecture (Overview)

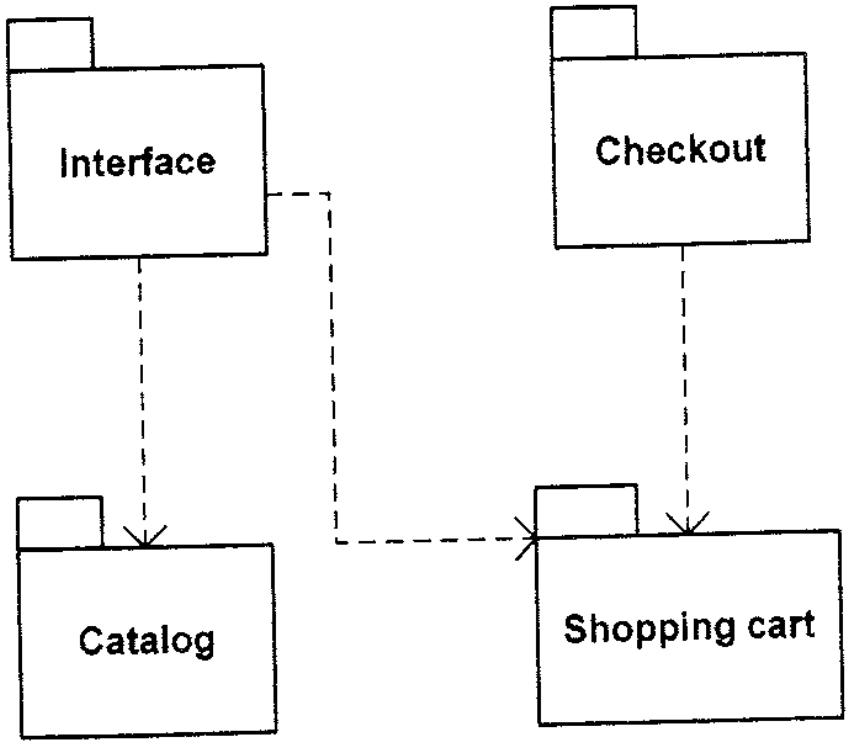

Figure: Package Diagram

UML for Software Architecture (Overview)

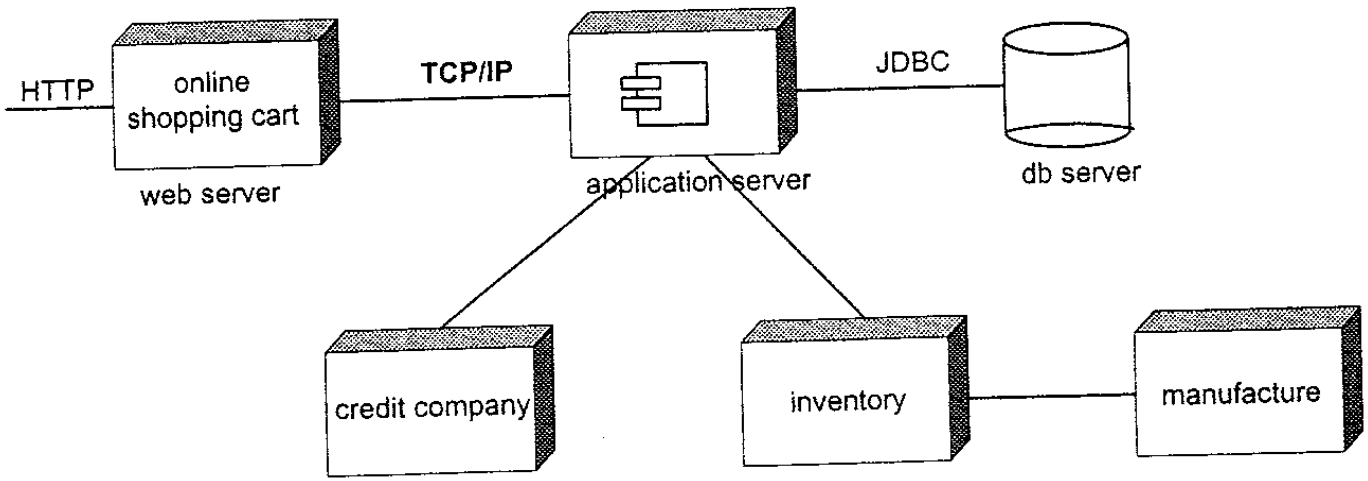

Figure: Deployment Diagram

UML for Software Architecture (Overview)

Behavioral (Dynamic) Diagrams

- Use Case :
  - Derived from use case study scenario
  - An overview of use cases, actors, and their communication relationships
  - Demonstrations for how the system reacts to Business Events from the environment
  - Used to capture system requirements
- Activity Diagram:
  - An outline of activity's data and control flow
  - A workflow-oriented diagram
  - Covers decision points, threads of a complex process
  - Describes how activities are orchestrated

UML for Software Architecture (Overview)

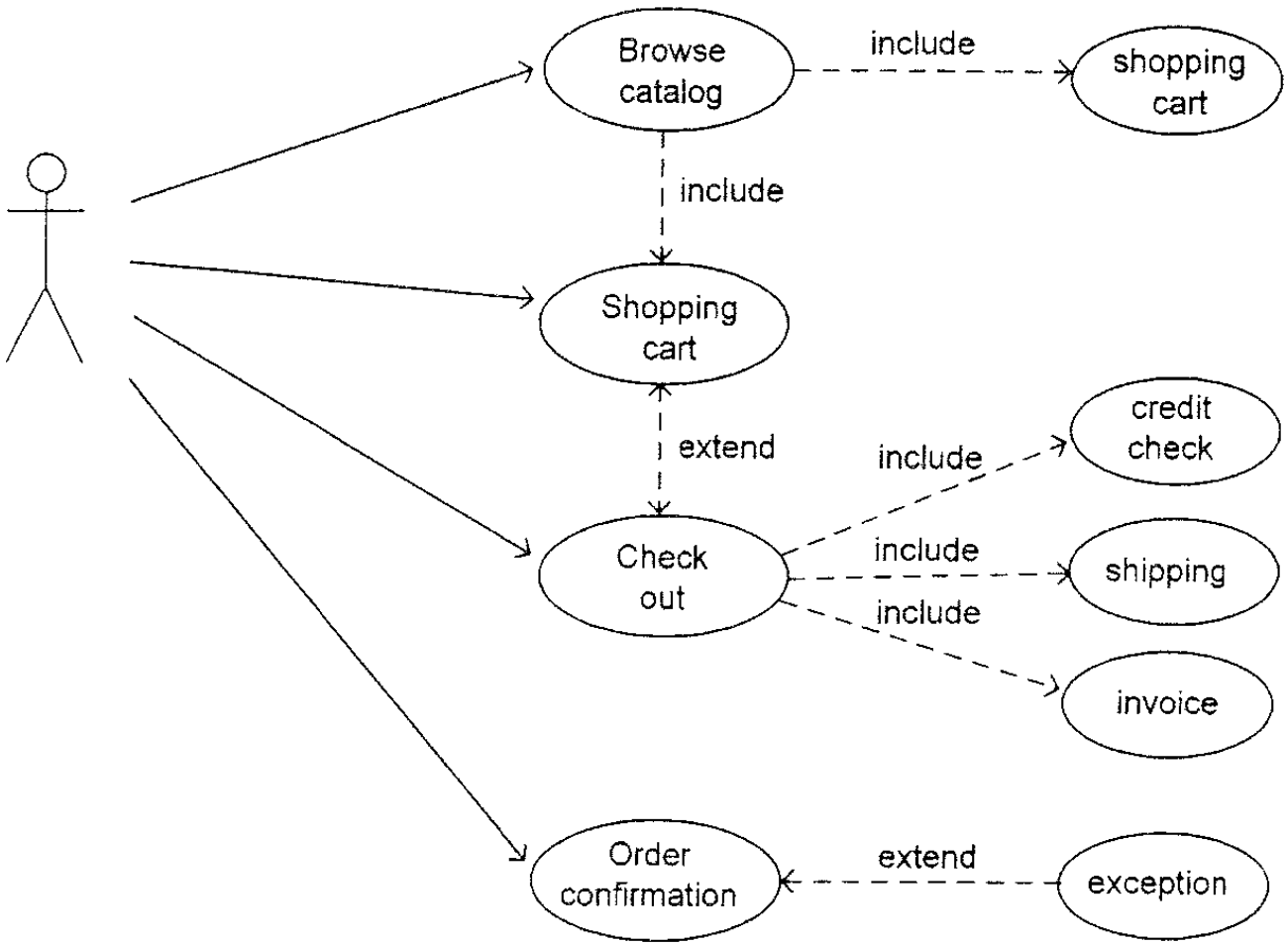

Figure: Use Case

UML for Software Architecture (Overview)

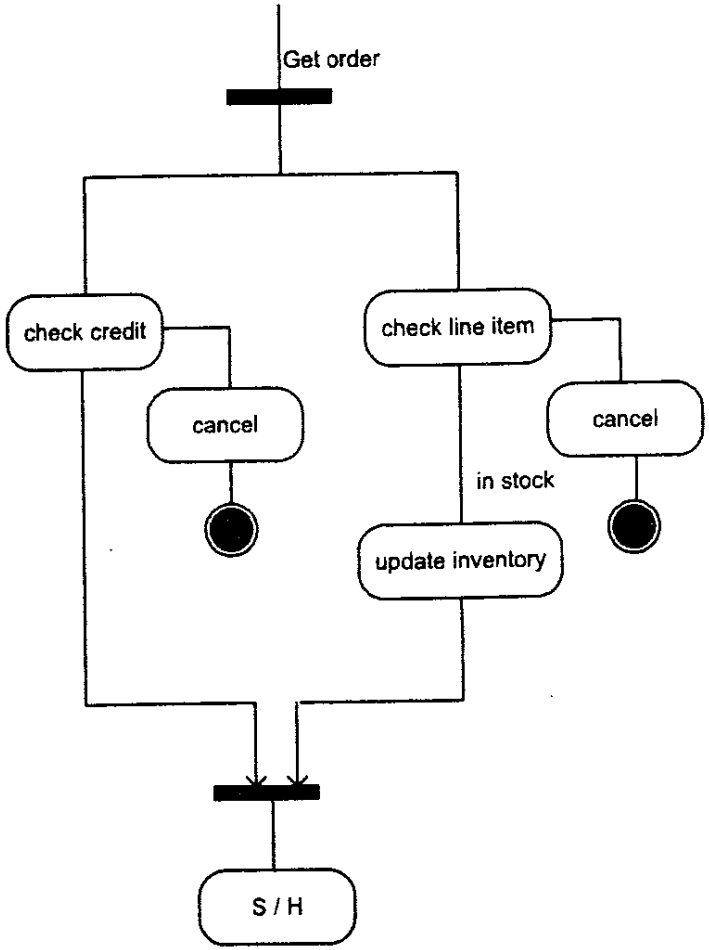

Figure: Activity Diagram

UML for Software Architecture (Overview)

Behavioral Diagrams (Continued)

- State Machine:
  - Uses FSM (Automaton) to give the life cycle of an object
  - The diagram consists of states and the transitions
  - Transitions are usually caused by external events
  - They can also represent internal moves of the object
  - Combines activity and sequence diagrams to provide control flow overview (system + business process)
- Interaction Overview:
  - Combines activity and sequence diagrams to provide control flow overview of the system and business

UML for Software Architecture (Overview)

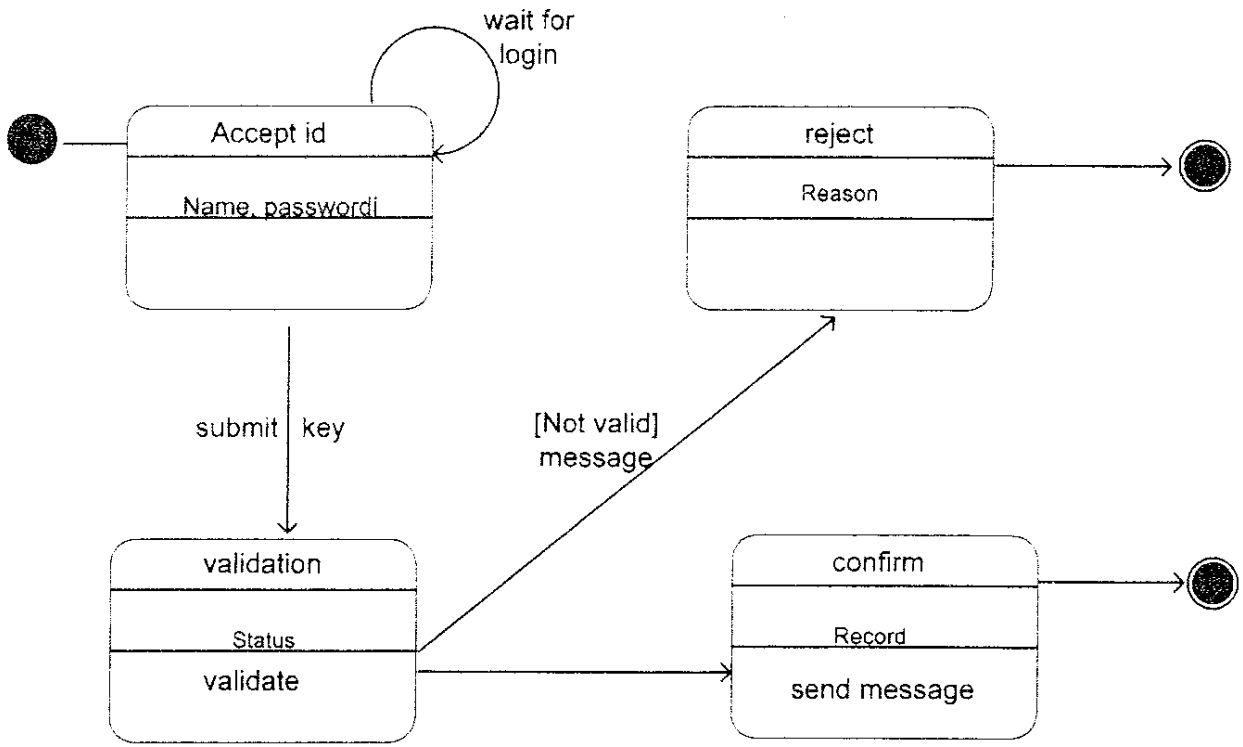

Figure: State Machine

UML for Software Architecture (Overview)

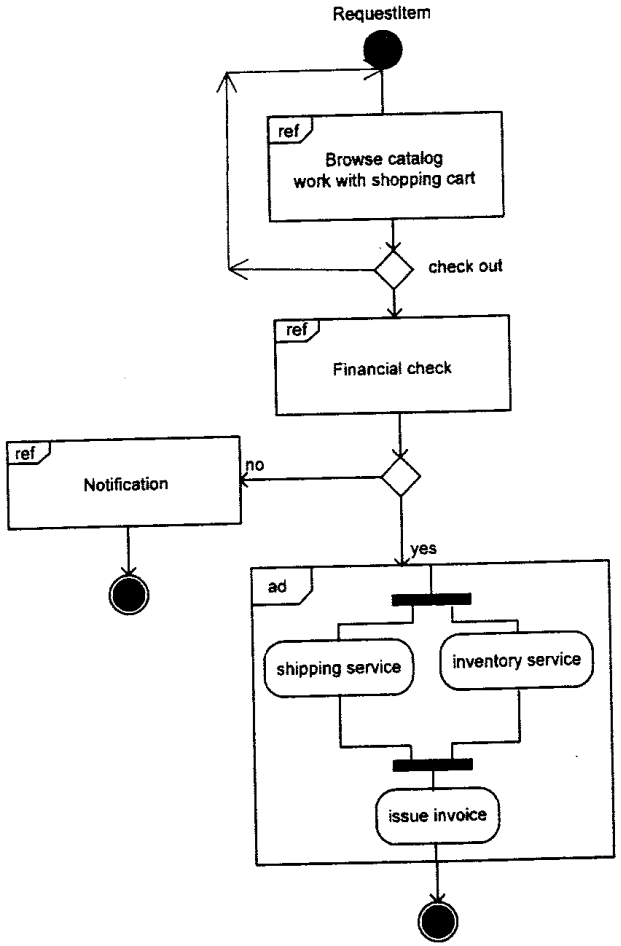

Figure: Interaction Overview

UML for Software Architecture (Overview)

Behavioral Diagrams (Continued)

- Sequence diagram:
  - One of the most important and most widely used UML diagrams
  - It shows the chronological sequence of messages between objects
  - Usually one sequence diagram corresponds to one use case
  - An object can send a synchronous message to another object by a synchronous message line with a full arrowhead
  - An object can also send asynchronous message to another object by a asynchronous message line with a half arrowhead

UML for Software Architecture (Overview)

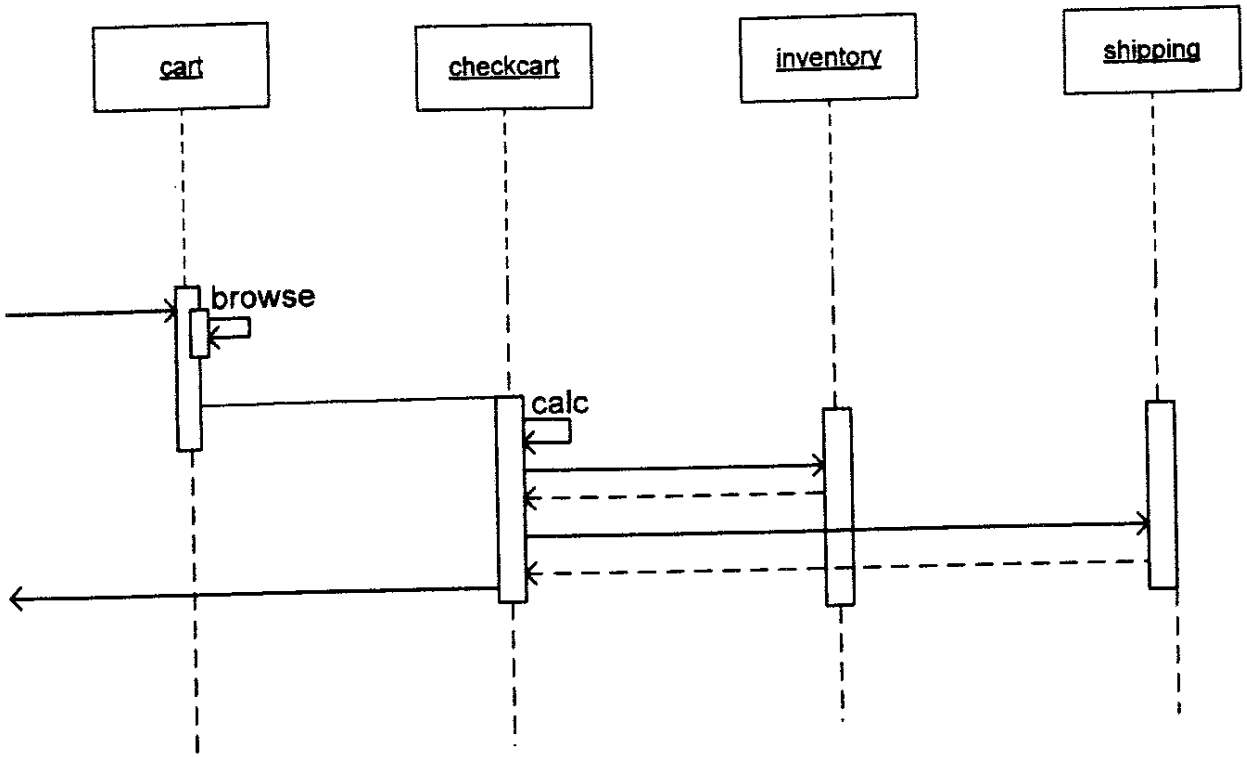

Figure: Sequence diagram

UML for Software Architecture (Overview)

Behavioral Diagrams (Continued)

- Communication (Collaboration in UML I.x) Diagram:
  - It describes message passing sequence, flow control, and object coordination
  - It depicts how an object in the system receives messages from other objects and sends messages to other objects
  - Every communication diagram is equivalent to a sequence diagram (can be converted to)

UML for Software Architecture (Overview)

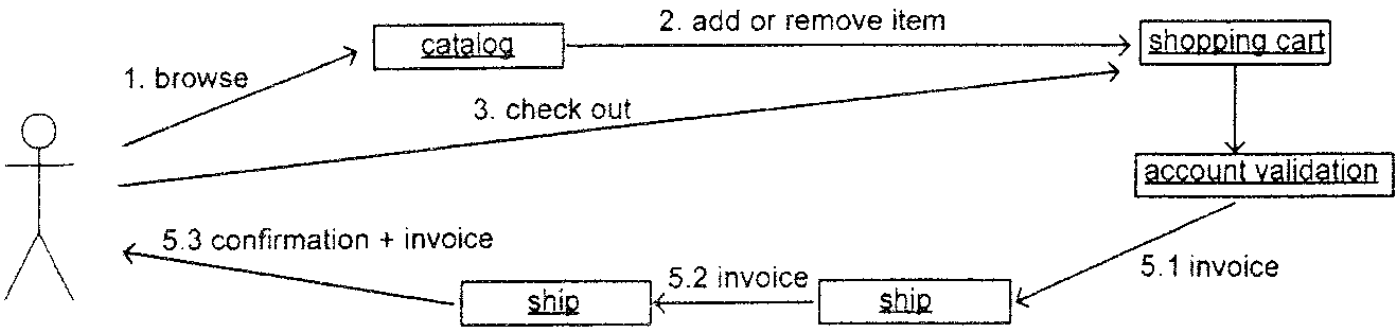

Figure: Communication (Collaboration) Diagram

UML for Software Architecture (Overview)

Behavioral Diagrams (Continued)

- Timing Diagram (UML 2.0):
  - It combines the state diagram and time sequence
  - It shows the dynamic view of state change caused by external events over time
  - It is often used in timing critical system

UML for Software Architecture (Overview)

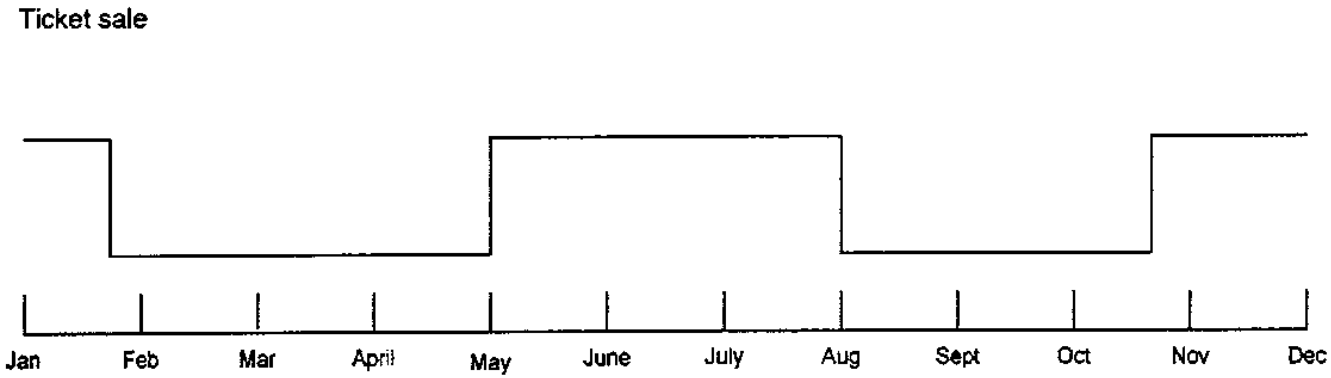

Figure: Timing diagram