Component Diagram

- Represents Implementation perspective
- Reflect grouping of different design elements of system
- The remaining two UML diagrams that fall under the category of Implementation are the Component and Deployment diagrams

- **Component diagrams** show how the physical components of a system are organized.
- A **component** is a distributable unit of software.
- The component diagram allows you to combine deployment nodes with components to show which components run on each node (i.e. hardware).

- Physical aspects are the elements like executables, libraries, files, documents etc which resides in a node.
- So component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

Component

A component is a physical, replaceable part of a system that conforms to, and provides the realization of, a set of interfaces.

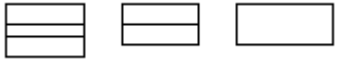
examples:

- dynamic link library (DLL)
- COM+ component
- Enterprise Java Bean (EJB)

- So the purpose of the component diagram can be summarized as:
- Visualize the components of a system.
- Construct executables by using forward and reverse engineering.
- Describe the organization and relationships of the components.

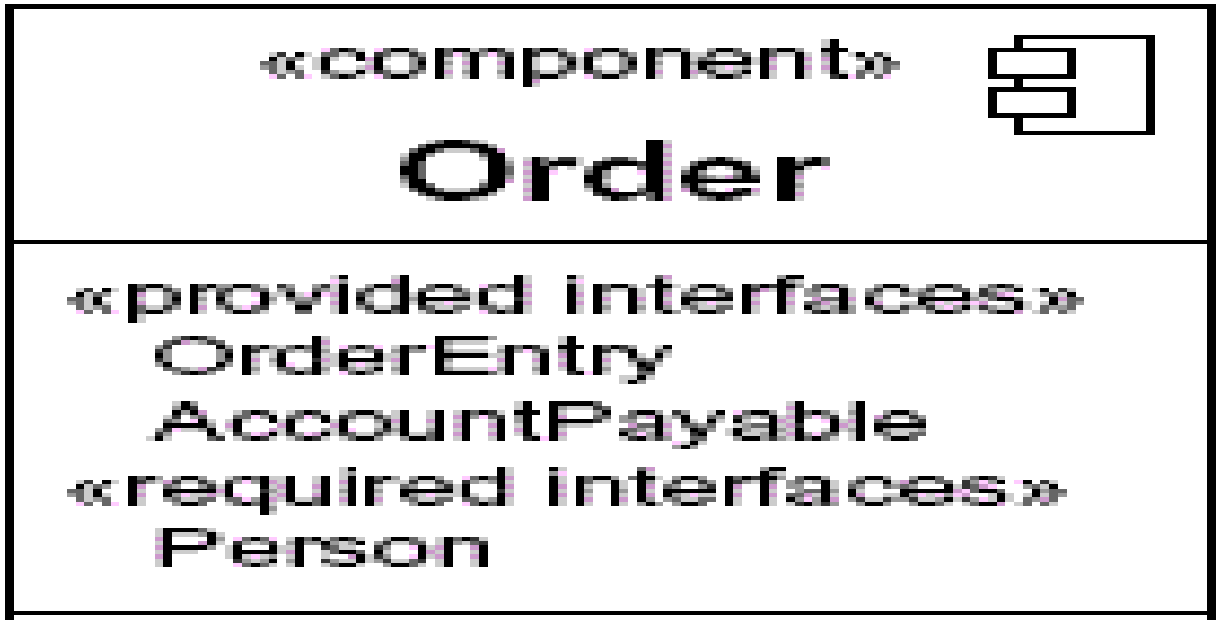
Component Elements

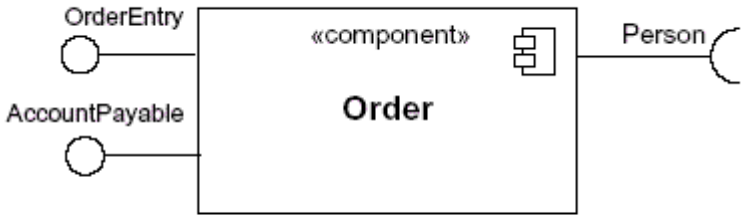
- Component
  - Interacting objects within system

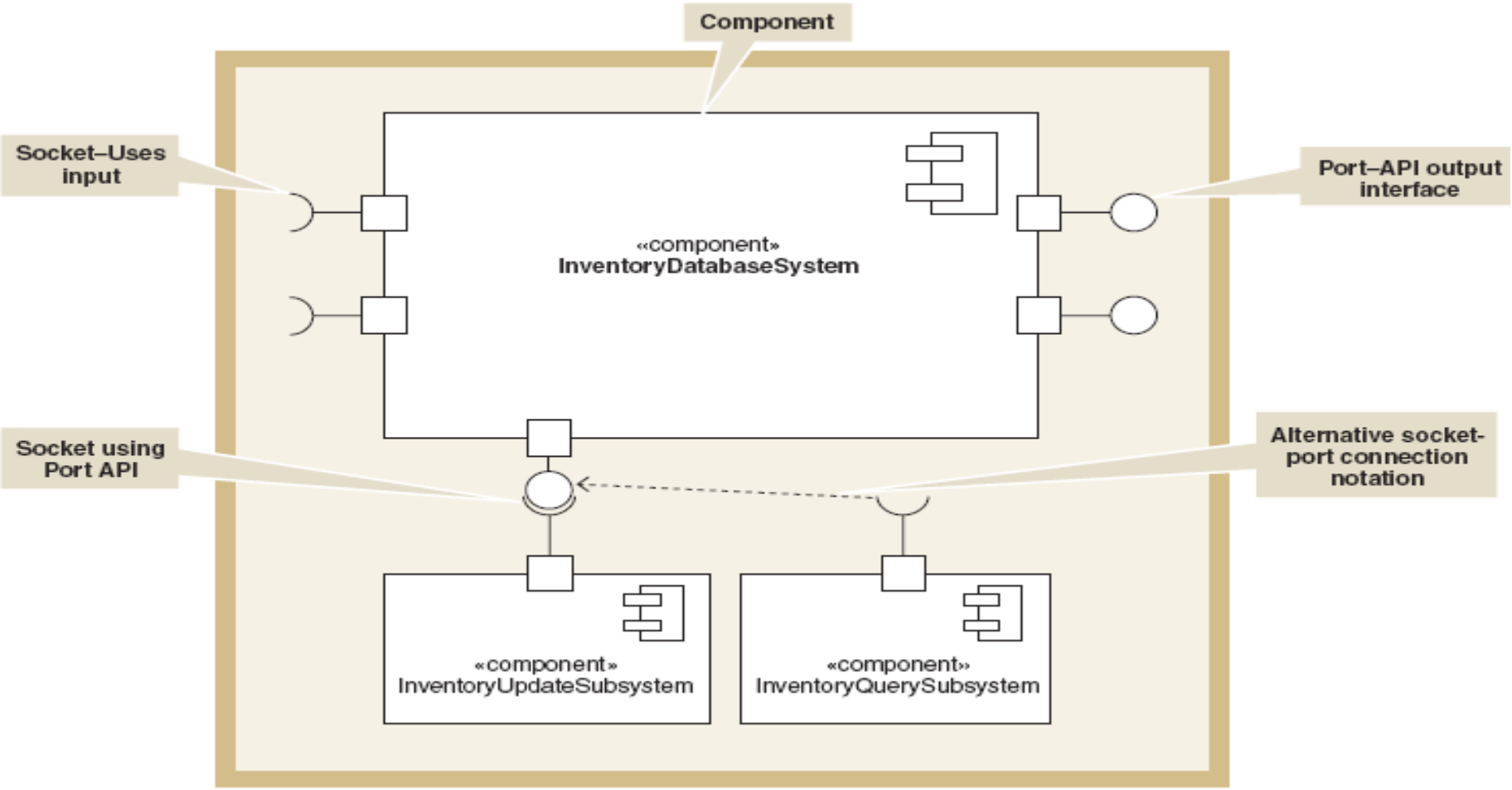


- Class/Interface/Object

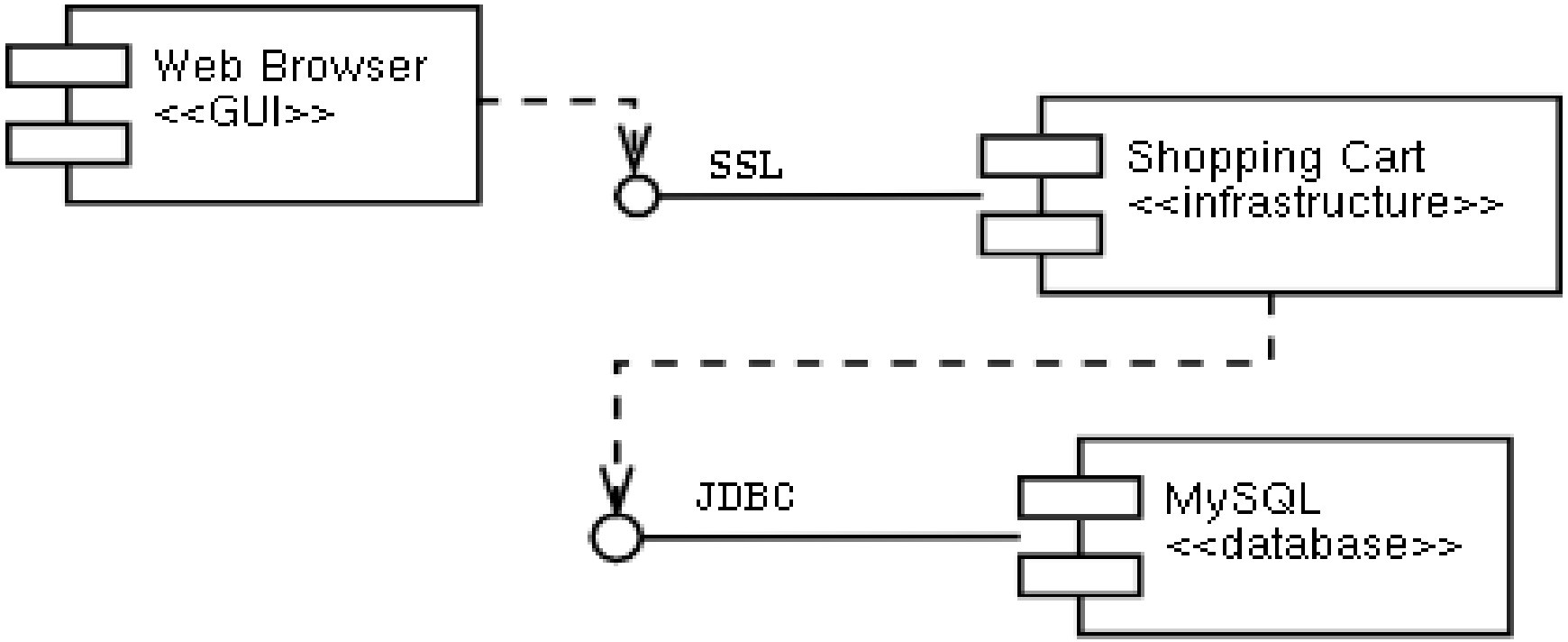
- Relation/Association


«component»
Order
«provided interfaces»
OrderEntry
AccountPayable
«required interfaces»
Person


OrderEntry
AccountPayable
«component»
Order
Person


Component
Socket–Uses input
Port–API output interface
«component»
InventoryDatabaseSystem
Socket using Port API
Alternative socket-port connection notation
«component»
InventoryUpdateSubsystem
«component»
InventoryQuerySubsystem

Component Diagram Example



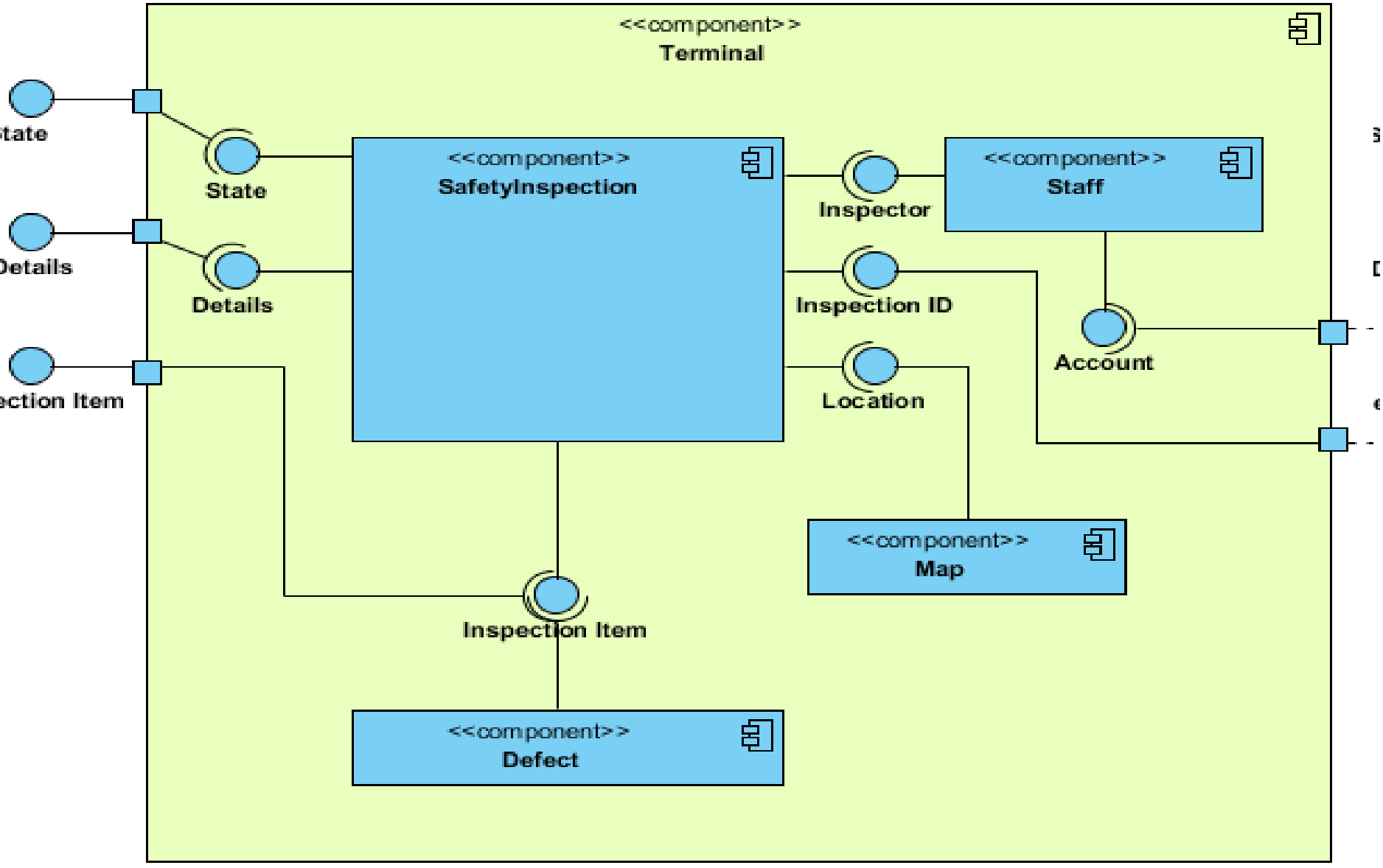
SSL = Secure Sockets Layer
JDBC = Java Database Connectivity

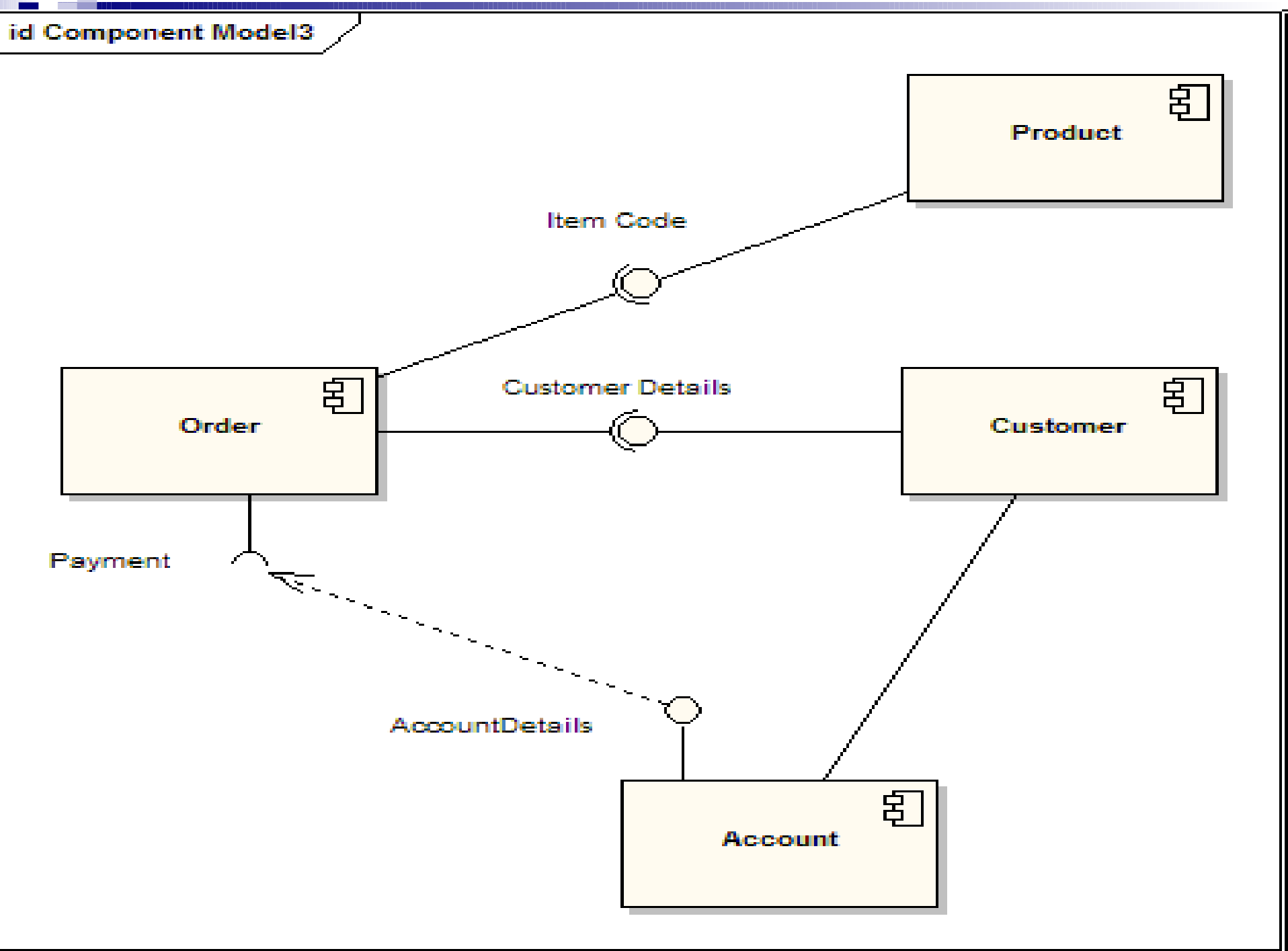
- So the purpose of this diagram is different, Component diagrams are used during the implementation phase of an application. But it is prepared well in advance to visualize the implementation details.
- Initially the system is designed using different UML diagrams and then when the artifacts are ready component diagrams are used to get an idea of the implementation.
- This diagram is very important because without it the application cannot be implemented efficiently. A well prepared component diagram is also important for other aspects like application performance, maintenance etc.

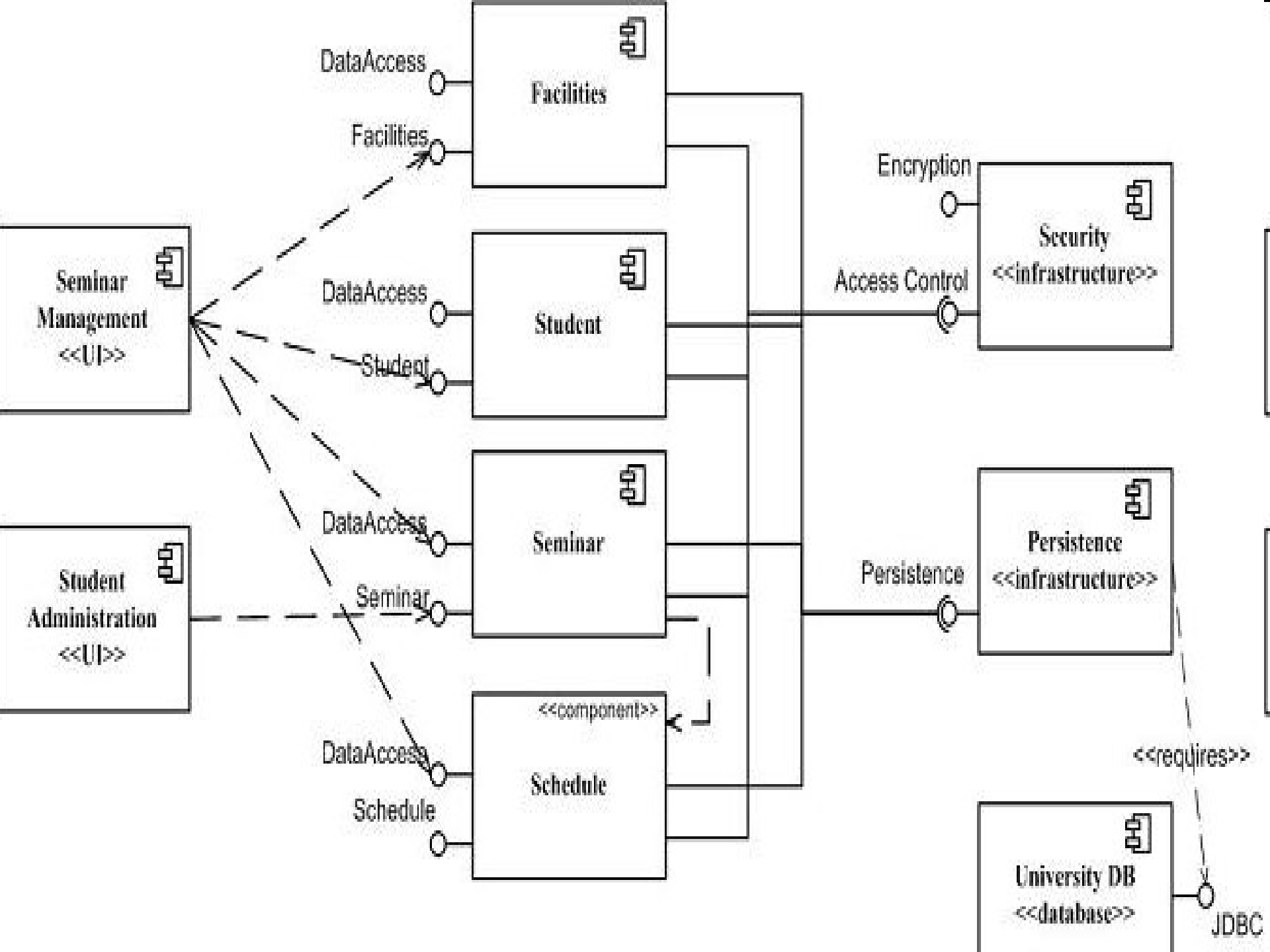
- So before drawing a component diagram the following artifacts are to be identified clearly:
- Files used in the system.
- Libraries and other artifacts relevant to the application.
- Relationships among the artifacts.

- Now after identifying the artifacts the following points needs to be followed:
- Use a meaningful name to identify the component for which the diagram is to be drawn.
- Prepare a mental layout before producing using tools.
- Use notes for clarifying important points.

- Now the usage of component diagrams can be described as:
- Model the components of a system.
- Model database schema.
- Model executables of an application.
- Model system's source code.


<<component>>
Terminal
State
Details
ection Item
State
Details
<<component>>
SafetyInspection
Inspector
<<component>>
Staff
Inspection ID
Account
Location
<<component>>
Map
Inspection Item
<<component>>
Defect

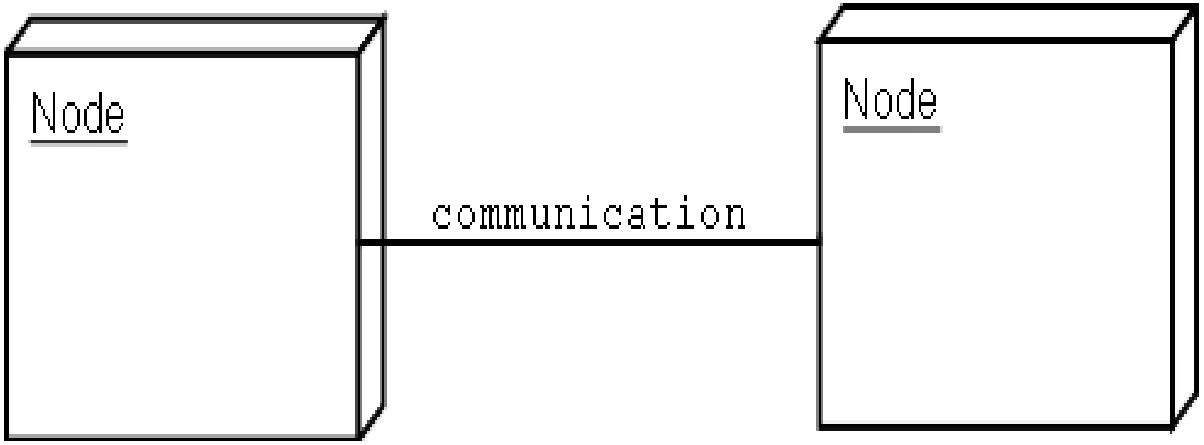

id Component Model3
Product
Item Code
Order
Customer Details
Customer
Payment
AccountDetails
Account


DataAccess
Facilities
Facilities
Seminar Management
<<UI>>
DataAccess
Student
Student
Student Administration
<<UI>>
DataAccess
Seminar
Seminar
<<component>>
DataAccess
Schedule
Schedule
Encryption
Security
<<infrastructure>>
Access Control
Persistence
Persistence
<<infrastructure>>
<<requires>>
University DB
<<database>>
JDBC

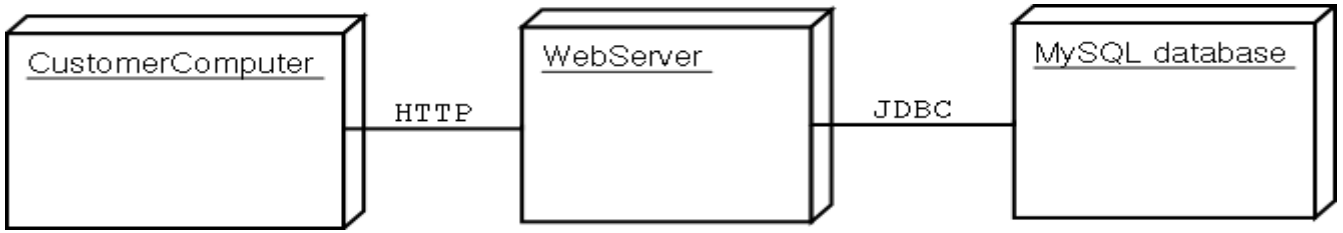
Deployment Diagram

- Represents physical relationships among software and hardware components as realized in running system
- Nodes represent computational elements (i.e. processor, server, etc.)

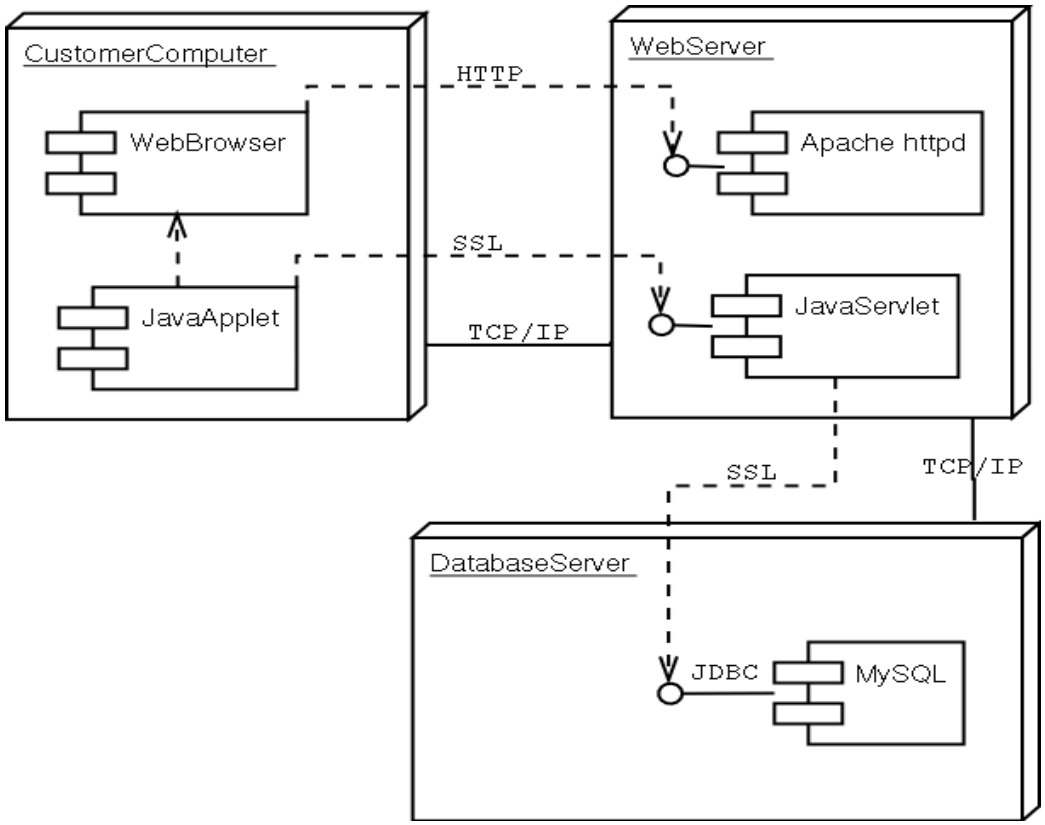
Deployment Diagram Skeleton



Example



Example


CustomerComputer
WebBrowser
JavaApplet
HTTP
SSL
TCP/IP
WebServer
Apache httpd
JavaServlet
SSL
TCP/IP
DatabaseServer
JDBC
MySQL

Courseware Example

- Construct the design elements for a system that can be used to manage courses/classes
- The organization offers a courses in areas such as learning management techniques and understanding different software languages and technologies
- Each course consists of a set of topics
- Tutors assigned courses to teach according to their specialty and availability
- Publishes and maintains calendar of courses and assigned tutors
- Course Administrators who manage content, assign courses to tutors, and define schedule

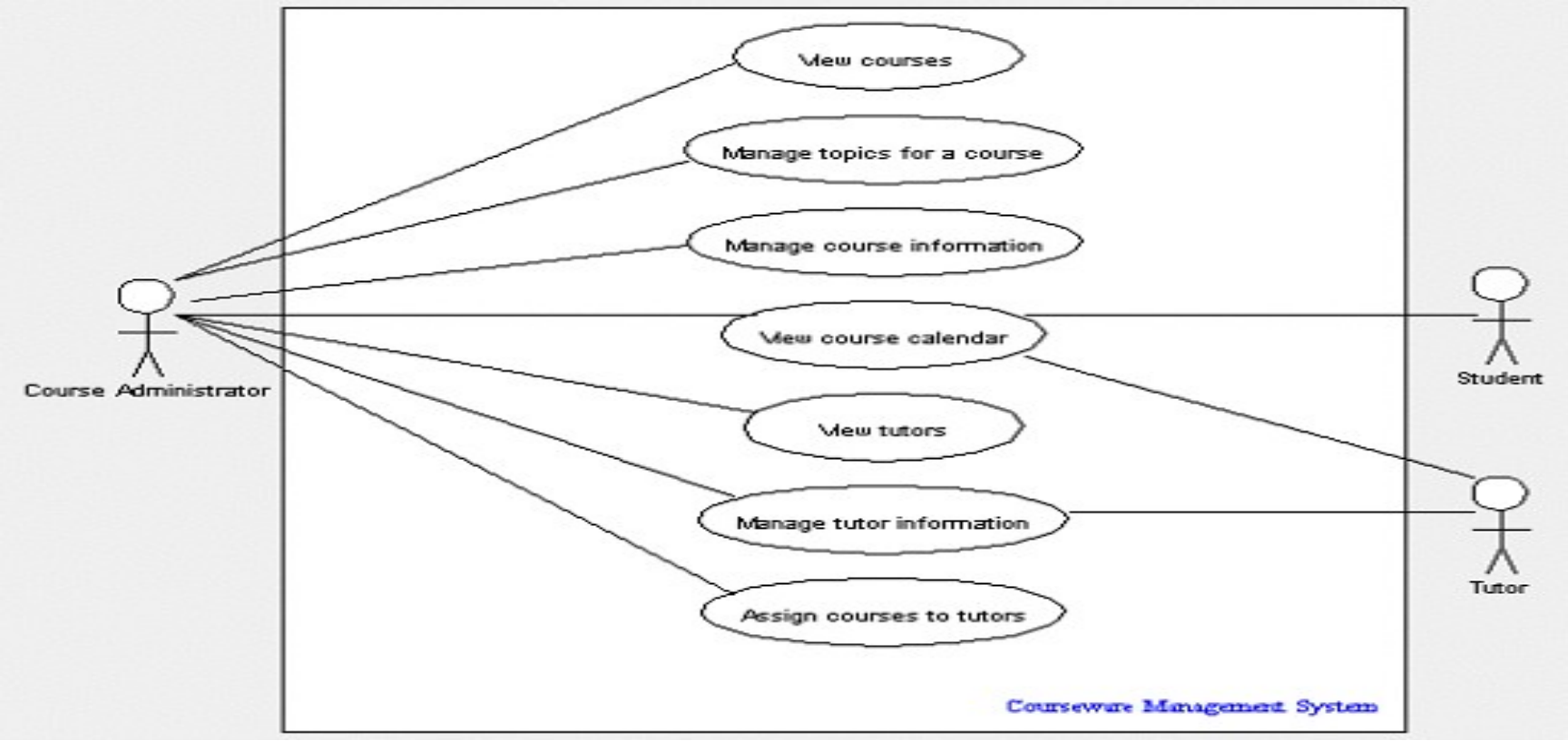
Identify Actors

- Tutors
- Course Administrators
- Students
- Course Administrator is main actor

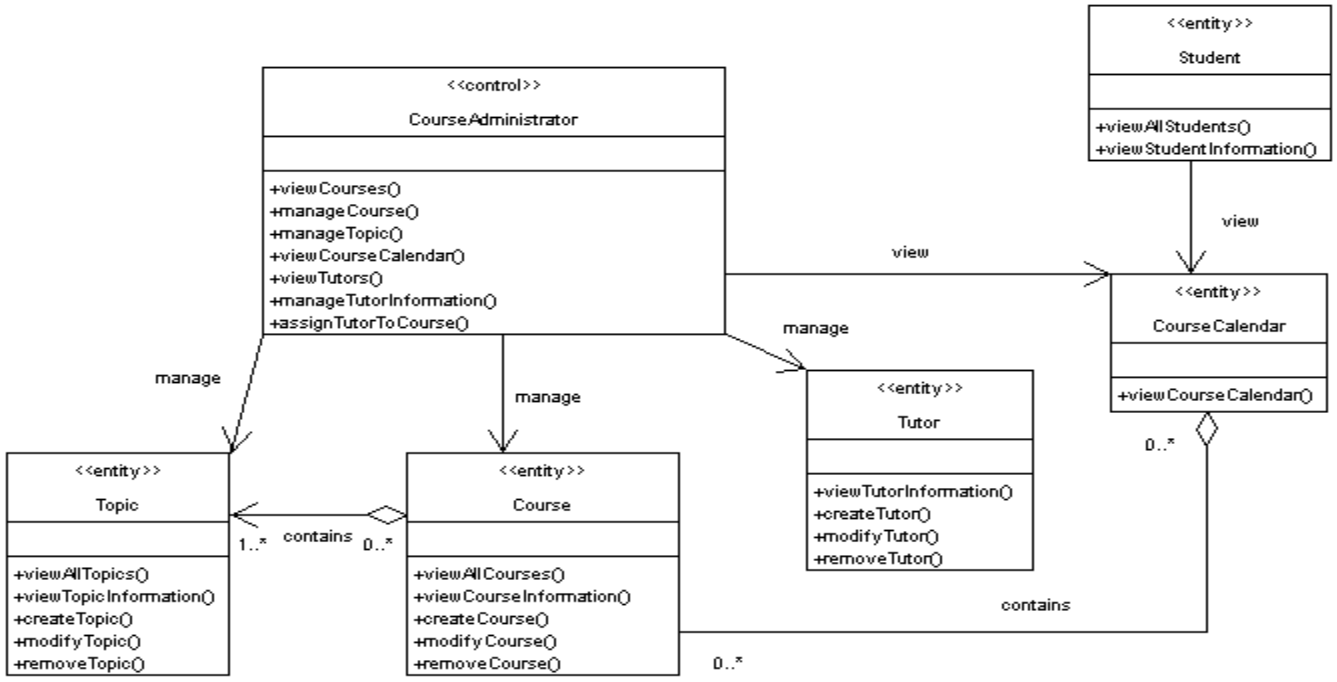
Use Case

- Manage courses
  - View courses
  - Manage topics for a course
  - Manage course information
- Manage course assignments
  - View course calendar
  - View tutors
  - Manage tutor information
  - Assign courses to tutors

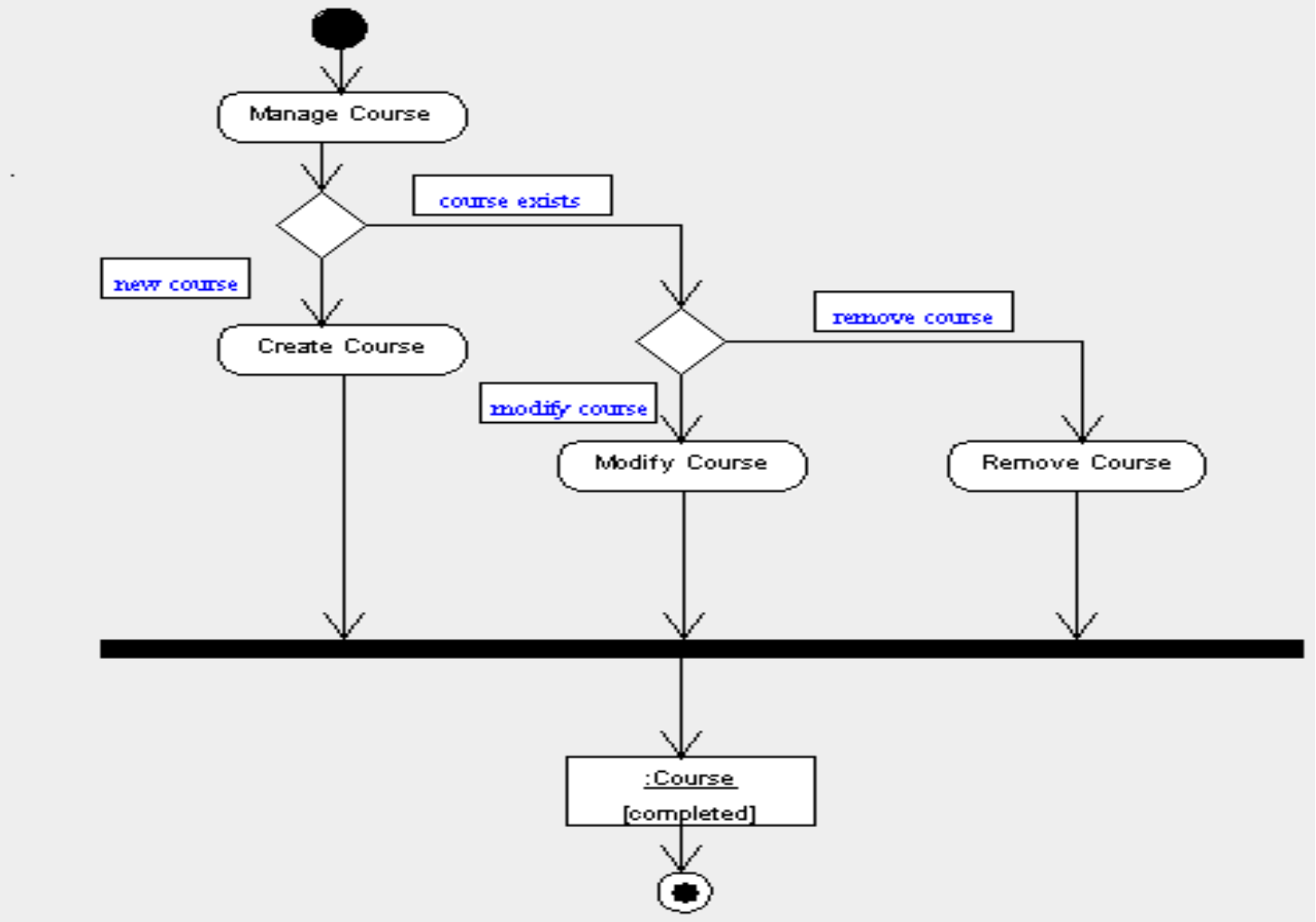
Use Case


View courses
Manage topics for a course
Manage course information
View course calendar
View tutors
Manage tutor information
Assign courses to tutors
Course Administrator
Student
Tutor
Courseware Management System

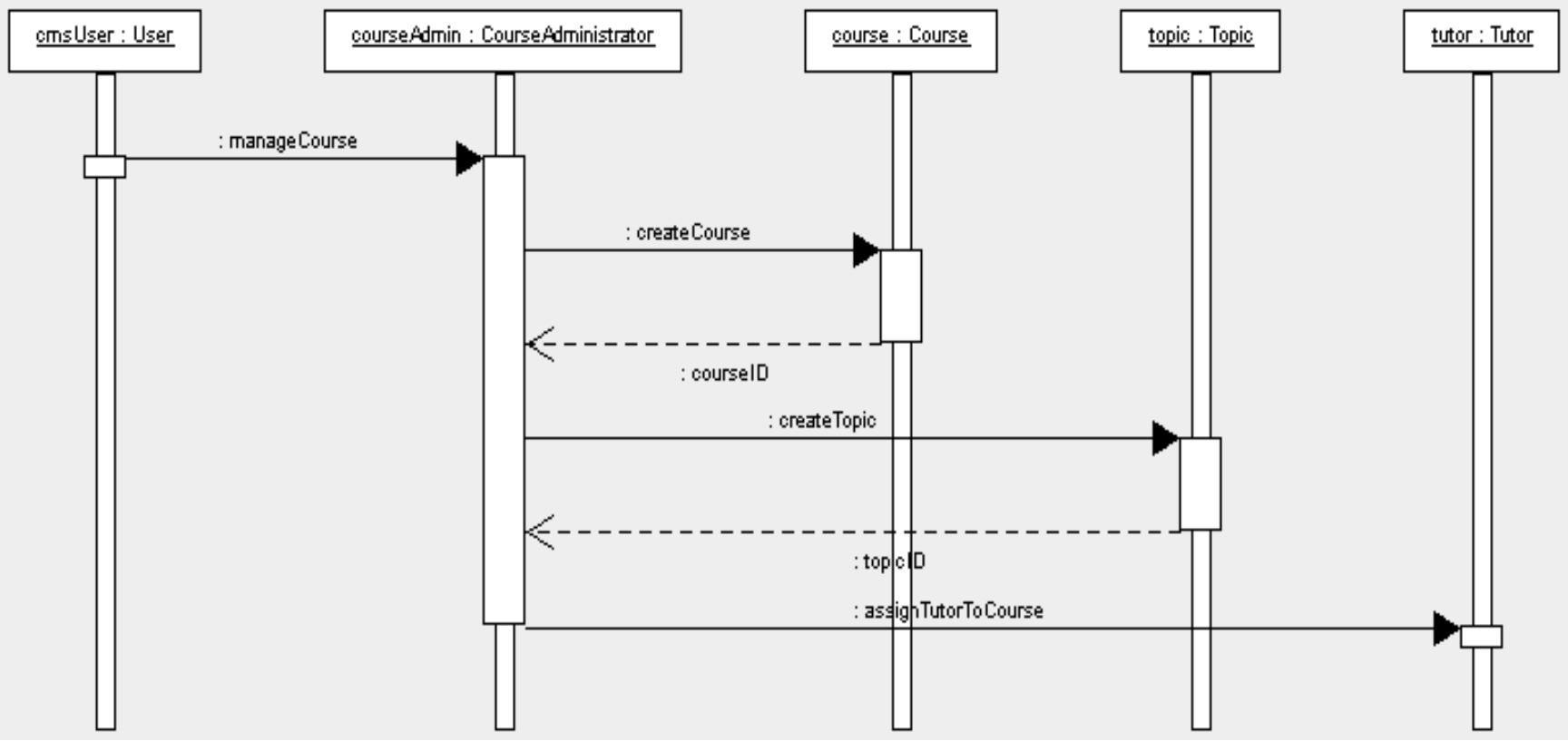
Class Diagram



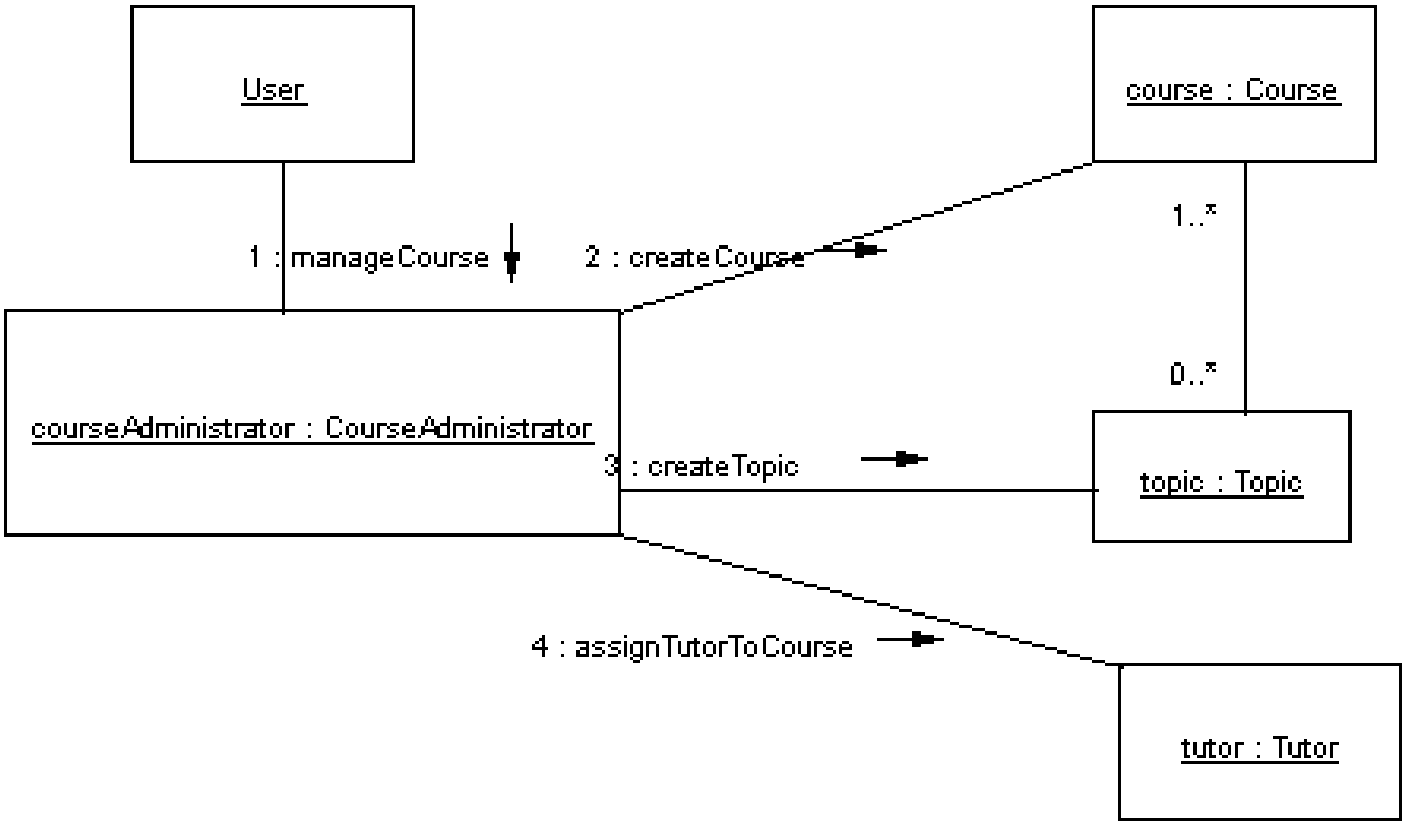
Activity Diagram



Sequence Diagram



Collaboration Diagram



Component Diagram