CSCI 4050/6050
Software Engineering

# Diagrams for System Architecture

## Component & Deployment Diagrams

Dr. Eman Saleh eman.saleh@uga.edu

## Lecture Outline

- Introduction about components
- Components  and component diagrams
- Elements of the component
- Component view: black-box view and white-box view
- Deployment diagrams

## INTRODUCTION

An Uml diagram classification:

- **Static diagrams**
  - Use case diagram, Class diagram
- **Dynamic diagrams**
  - State diagram, Activity diagram, Sequence diagram, and communication diagram
- **Implementation diagrams**
  - Component diagram, Deployment diagram, and Network diagrams

**Implementation diagrams** describe the different elements required for implementing a system

## INTRODUCTION

**Another classification:**

- **Behavior diagrams**
  - A type of diagram that depicts behavior of a system

    This includes activity, state machine, and use case diagrams, interaction diagrams
- **Interaction diagrams**
  - A subset of behavior diagrams which emphasize object interactions. This includes communication, activity, and sequence diagrams
- **Structure diagrams**
  - A type of diagram that depicts the elements of a specification that are irrespective of time. This includes class, composite structure, component, and deployment diagrams.

UML components diagrams are **structure diagrams**

UML components diagrams are **implementation diagrams**

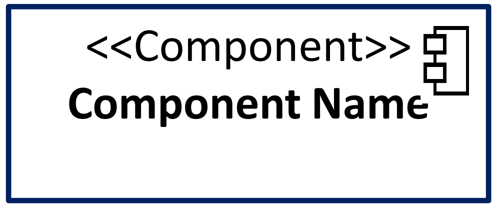
## A Component

- A component is an encapsulated, reusable, and replaceable part of your software.
- Reducing coupling between software components
- Reusing existing components

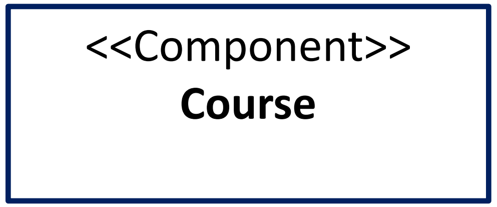
- A Component is a physical piece of a system, such as a compiled object file, piece of source code, shared library or Enterprise Java Bean (EJB).

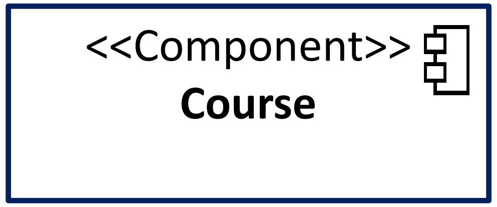
## Component Notation

- A component is shown as a rectangle with
  - A keyword <<component>>



- Optionally, in the right hand corner a component icon can be displayed
  - A component icon is a rectangle with two smaller rectangles jutting out from the left-hand side
  - This symbol is a visual stereotype
- The component name

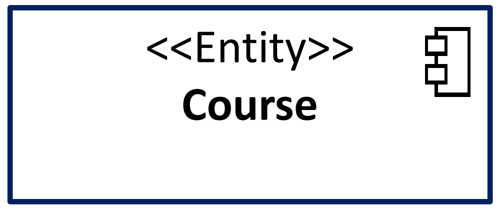




- Components can be labelled with a stereotype
- there are a number of standard stereotypes ex: <<entity>>, <<subsystem>>

## Standard stereotypes

- <<executable>> : A program that may run on a node
- <<application>> : Consists of several executables
- <<file>> : File containing source code or data
- <<library>> :Static or dynamic library
- <<document>> : A document
- <<page>> :HTML page
- Technology specific
  - <<ActiveX>>, <<JavaBean>>, <<Applet>>, <<DLL>>, <<CORBA Component>>



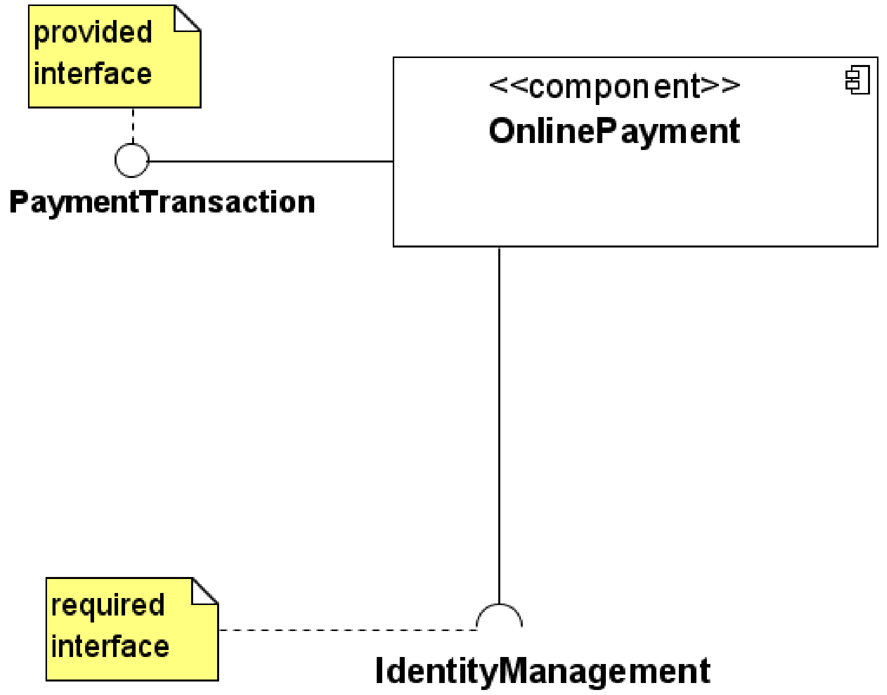
## Interface

- A component defines its behaviour in terms of provided and required interfaces
- An interface
    - Is the definition of a collection of one or more operations
    - Provides only the operations but not the implementation
    - Implementation is normally provided by a class/ component
    - In complex systems, the physical implementation is provided by a group of classes rather than a single class

## Component Interfaces

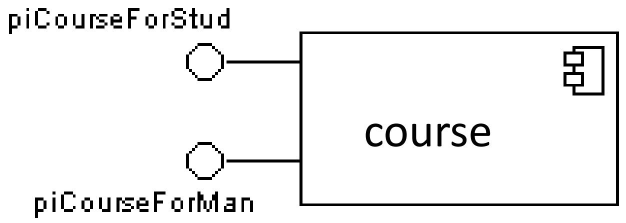
A provided interface of a component is an interface that the component Realizes

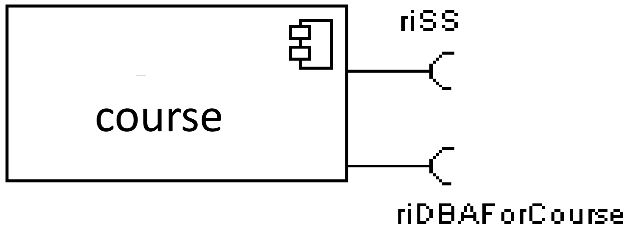
A required interface of a component is an interface that the component needs to function



## Componenet INTERFACES

- **A provided interface**
  - Characterize services that the component offers to its environment
  - Is modeled using a ball, labelled with the name, attached by a solid line to the component





- **A required interface**
  - Characterize services that the component expects from its environment
  - Is modeled using a socket, labelled with the name, attached by a solid line to the component
  - In UML 1.x were modeled using a dashed arrow

# Component Diagrams

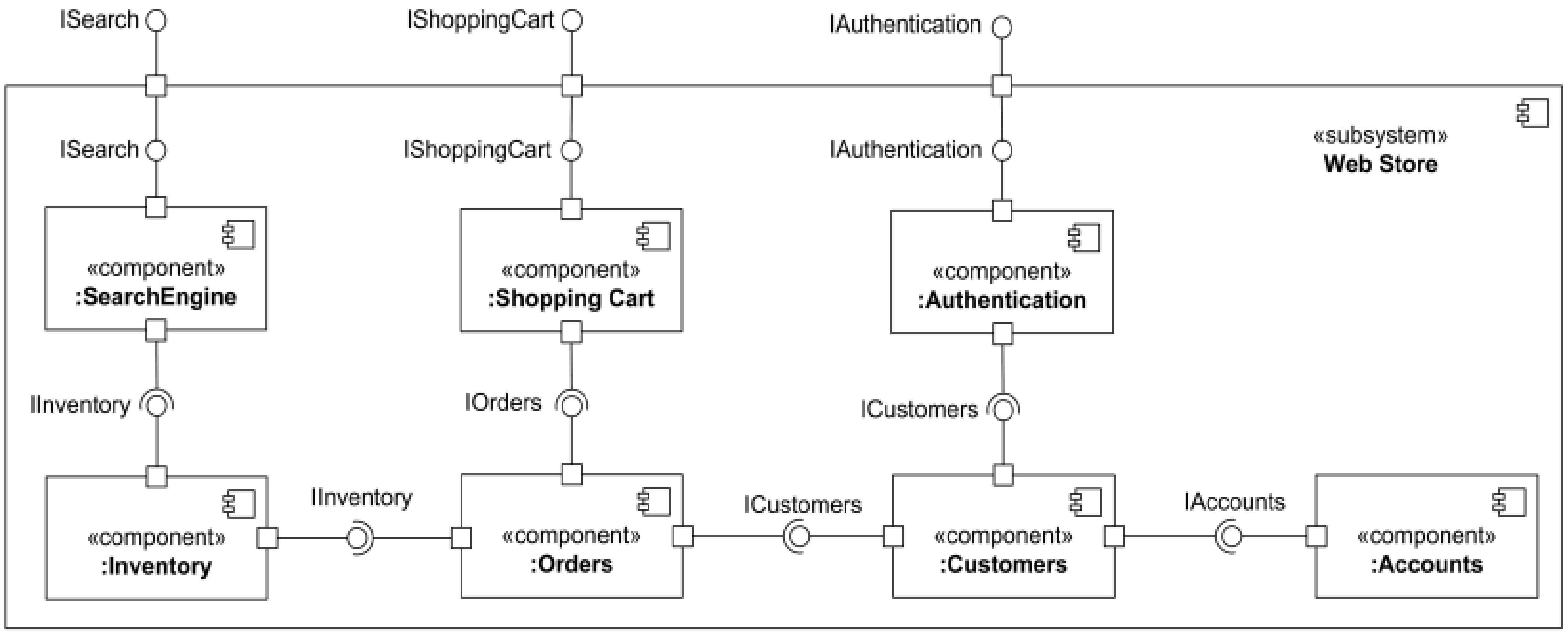
- UML component diagrams describe software components and their dependencies to each others.

Component Diagrams are often referred to as **“wiring diagrams”**
The wiring of components can be represented on diagrams by means of components and dependencies between them.

## Component Diagrams

- Model physical software components and the interfaces between them
- Show the structure of the code itself
- Can be used to hide the speciation detail (i.e., information hiding) and focus on the relationship between components
- Model the structure of software releases; show how components integrate with current system design
- Model source code and relationships between the files
- Specify the files that are compiled into an executable

# Component Diagram



## Component Elements

- A component can have
  - Interfaces

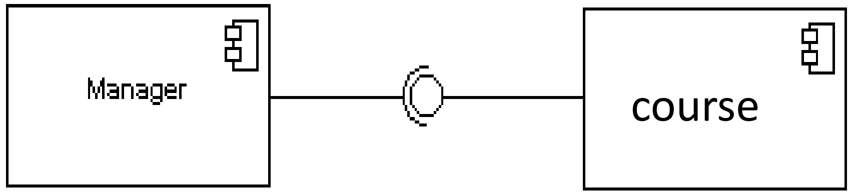
    An interface represents a declaration of a set of operations and obligations
  - Usage dependencies

    A usage dependency is relationship which one element requires another element for its full implementation
  - Ports

    Port represents an interaction point between  a component and its environment
  - Connectors
    - Connect two components
    - Connect the external contract of a component to the internal structure

## ASSEMBLY

- Two kinds of connectors:
  - Delegation
  - Assembly
- ASSEMBLY CONNECTOR
  - An assembly connector is notated by a “ball-and-socket” connection
  - A connector between 2 components defines that one component provides the services that another component requires
  - Must only be defined from a required interface to a provided interface

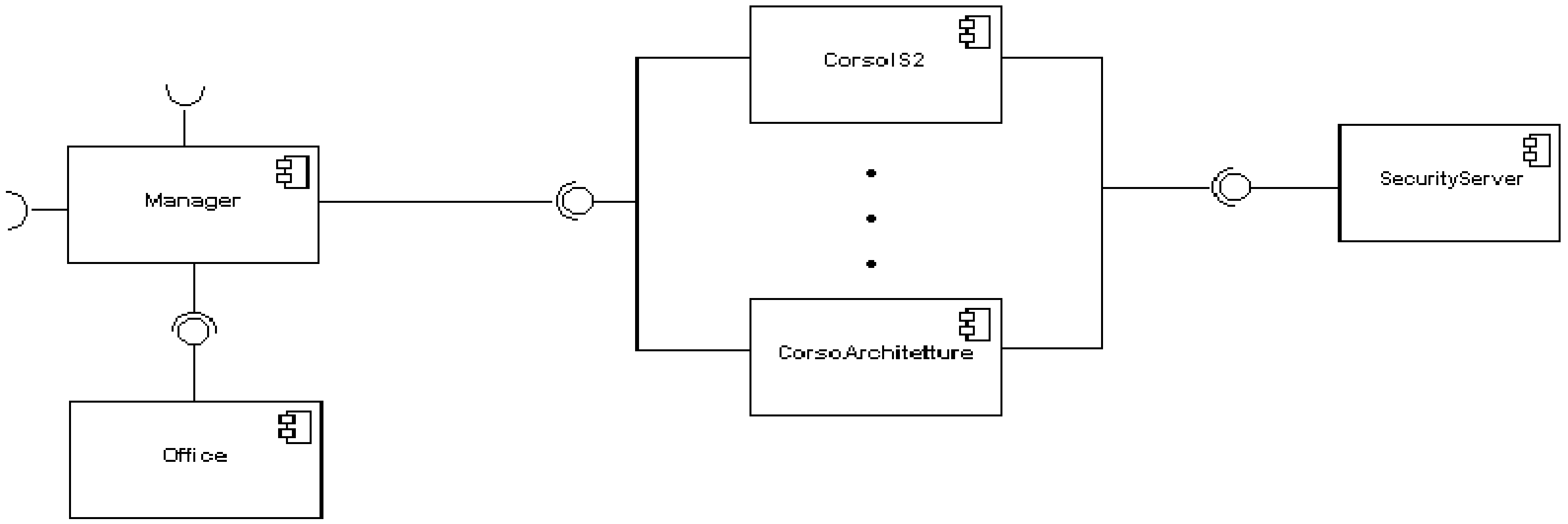


## SEMANTICS

- The semantics for an assembly connector :
  - Are that signals travel along an instance of a  connector originating in a required port and delivered to a provided port
  - **The interfaces provided and required must be compatible**
  - The interface compatibility between provided and required ports that are connected enables an existing component in a system to be replaced

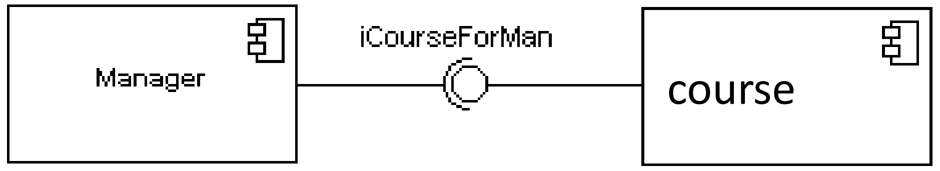
## SEMANTICS

- Multiple connections directed from a single required interface to provided interfaces indicates that the instance that will handle the signal will be determined at execution time

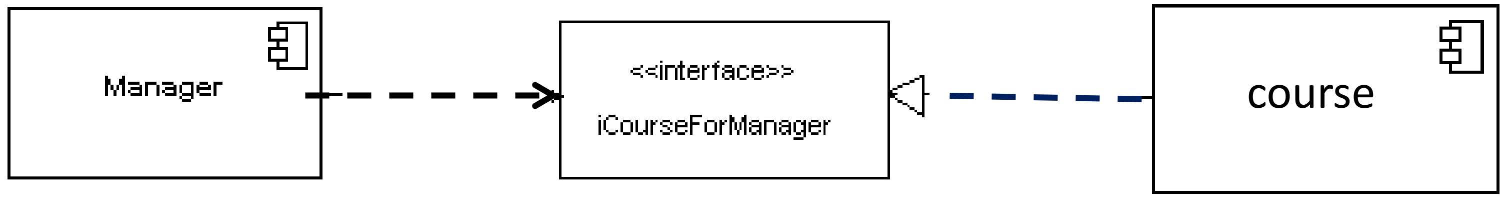


## INTERFACES

- Where two components/classes provide and require the same interface, these two notations may be combined



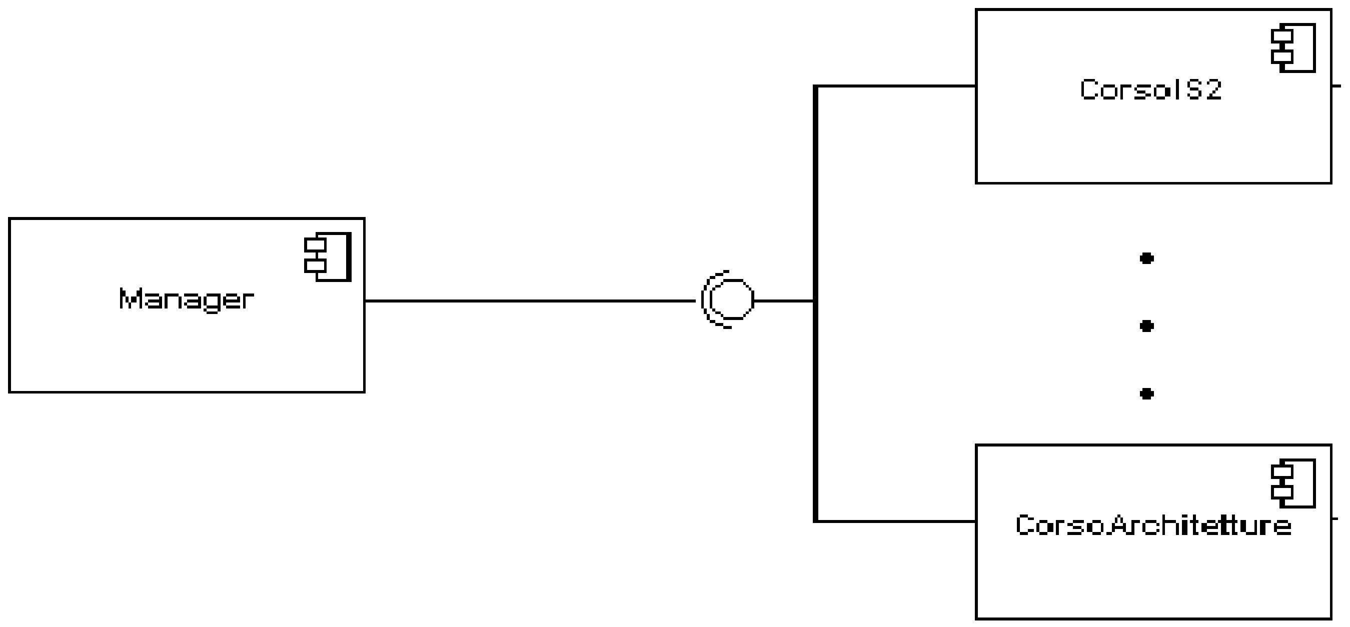
- The ball-and-socket notation hint at that interface in question serves to mediate interactions between the two components
- If an interface is shown using the rectangle symbol, we can use an alternative notation, **using dependency arrows**



## INTERFACE

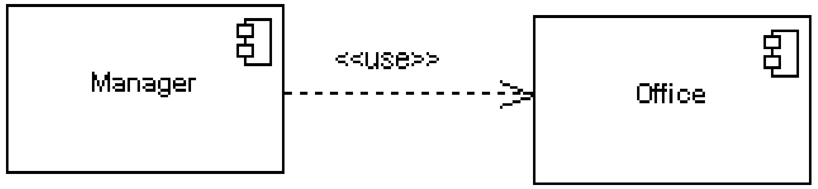
- In a system context where there are multiple components that require or provide  a particular interface, a notation abstraction can be used that combines by joining

  the interfaces



## DEPENDENCIES

- Components can be connected by usage dependencies

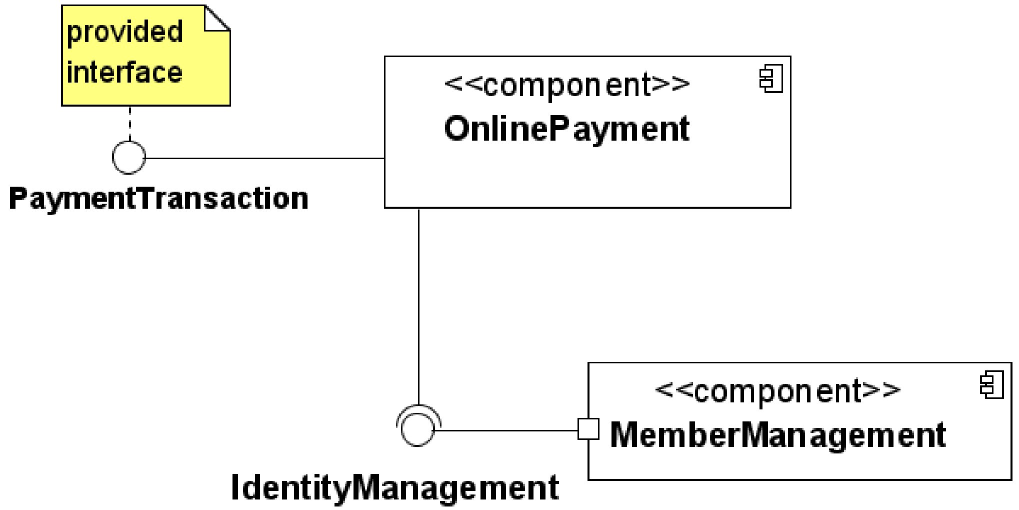


- Usage Dependency
  - A usage dependency is relationship which one element requires another element for its full implementation
  - Is a dependency in which the client requires the presence of the supplier
  - Is shown as dashed arrow with a <<use>> keyword
  - The arrowhead point from the dependent component to the one of which it is dependent

## PORT

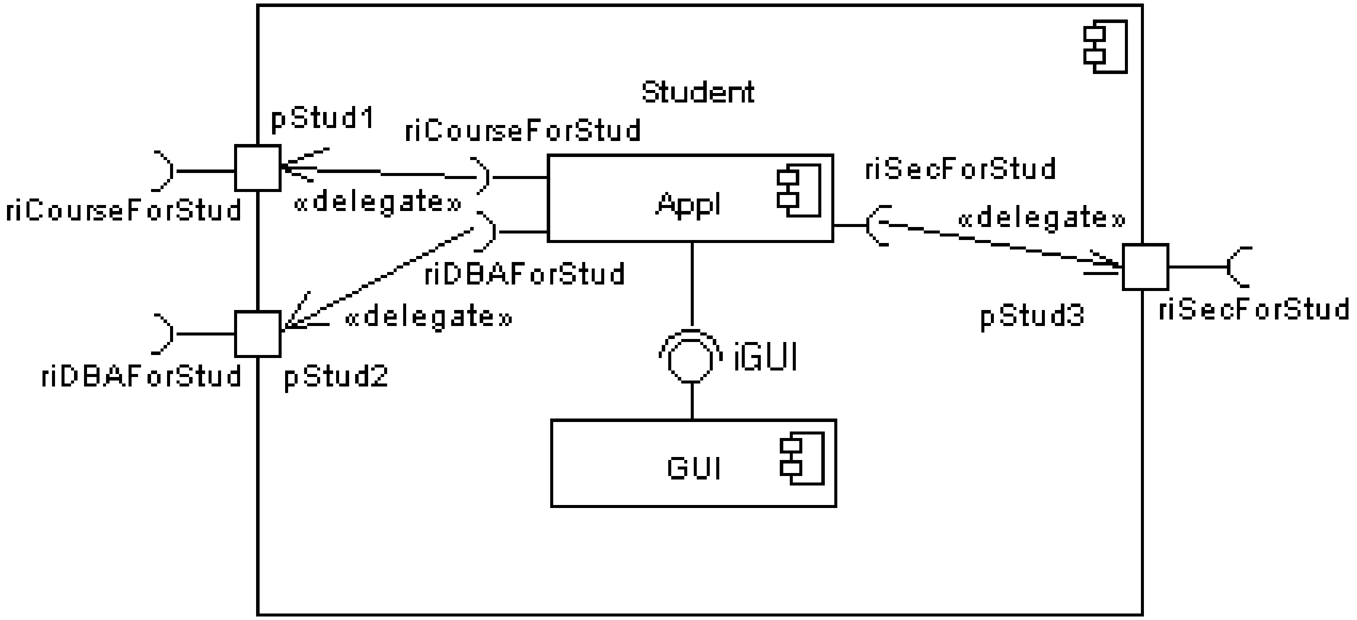
A port (definition) indicates that the component itself does not provide the required interfaces (e.g., required or provided). Instead, the component delegates the interface(s) to an internal class.All interactions of a component with its environment are achieved through a port

- The internals are fully isolated from the environment
- This allows such a component to be used in any context that satisfies the constraints specified by its ports
- Ports are not defined in UML 1.x

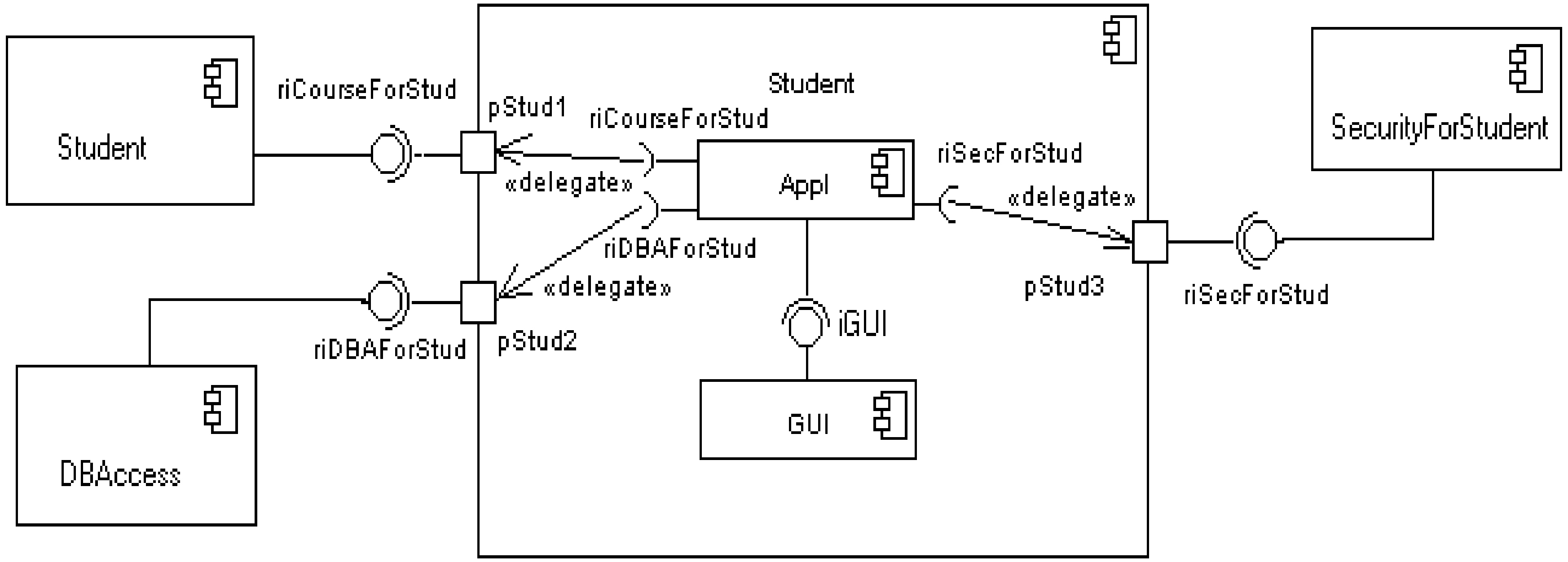


# Delegation

- DELEGATION CONNECTOR
  - Links the external contract of a component to the internal realization
  - Represents the forwarding of signals
  - Must only be defined between used interfaces or ports of the same kind



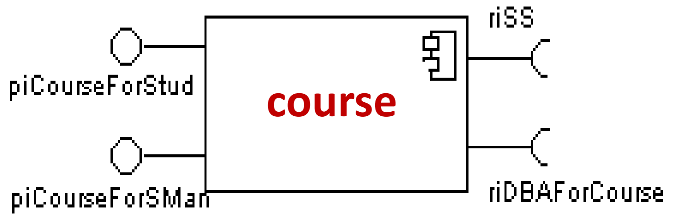
# Delegation and Assembly

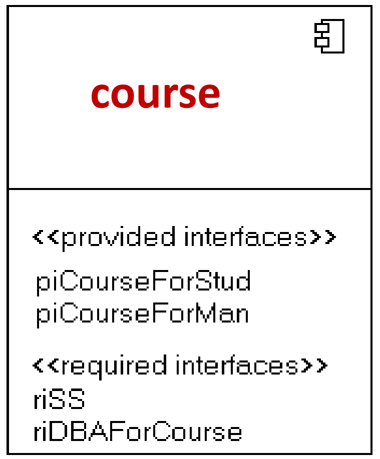


# EXTERNAL VIEW

- A component have an external view and an internal view

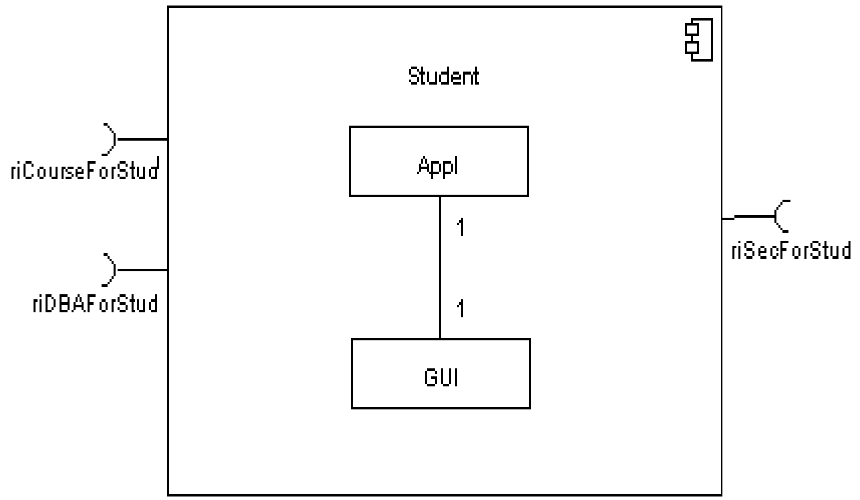
- An external view (or black box view) shows publicly visible properties and operations
- An external view of a component is by means of interface symbols sticking out of the component box
- The interfaces can be listed in the compartment of a component box





# INTERNAL VIEW

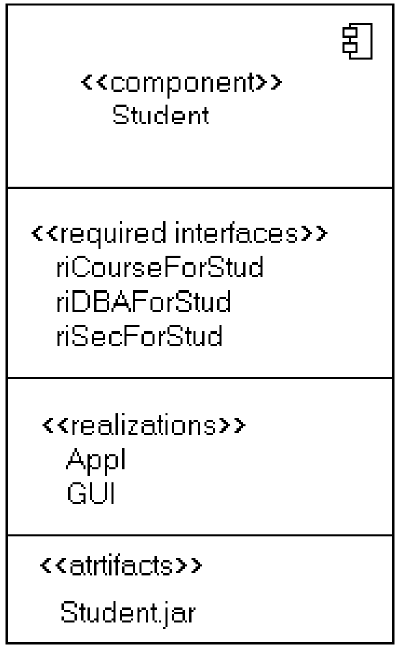
- An internal, or white box view of a component is where the realizing classes/components are nested within the component shape



- Realization is a relationship between two set of model elements
  - One represents a specification
  - The other represent an implementation of the latter

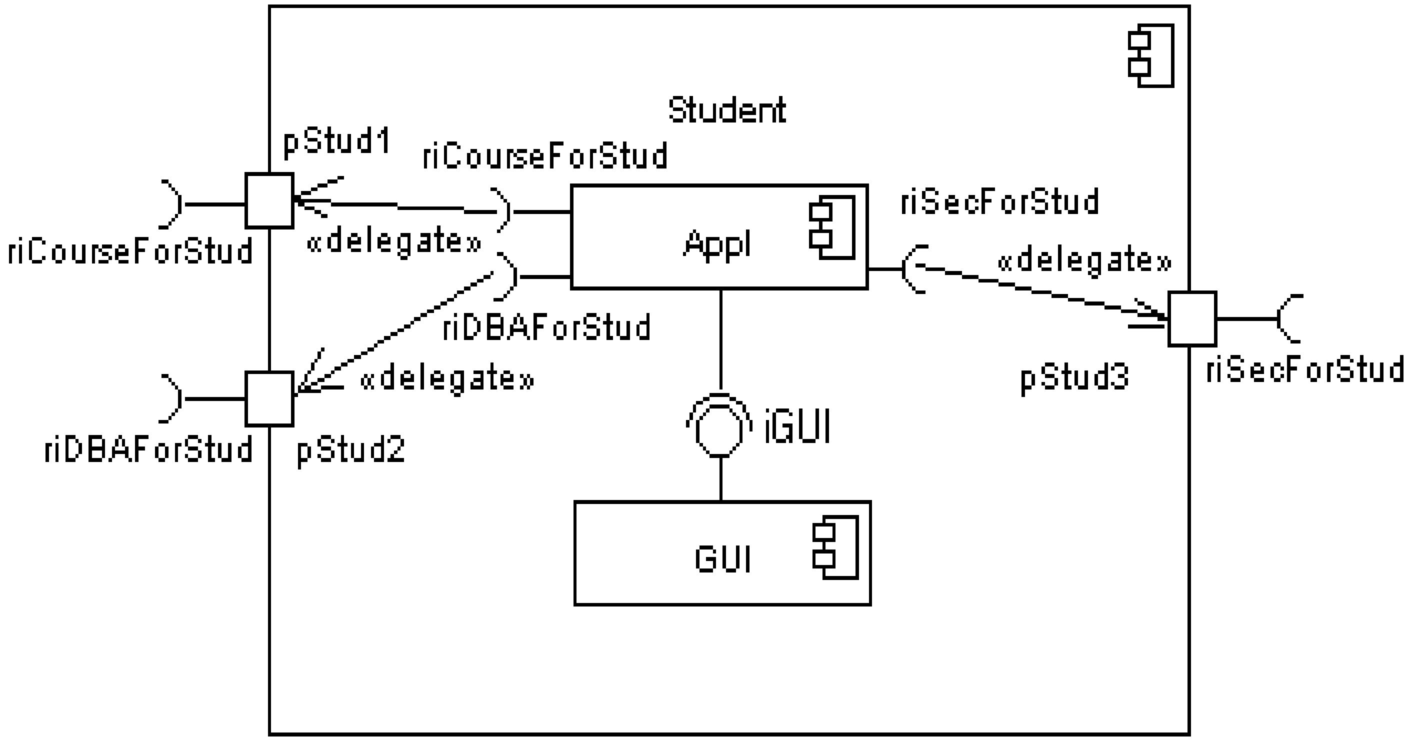
## INTERNAL VIEW

- The internal class that realize the behavior of a component may be displayed in an additional compartment
  - Compartments can also be used to display parts, connectors or implementation artifacts
  - An artifact is the specification of a phisycal piece of information



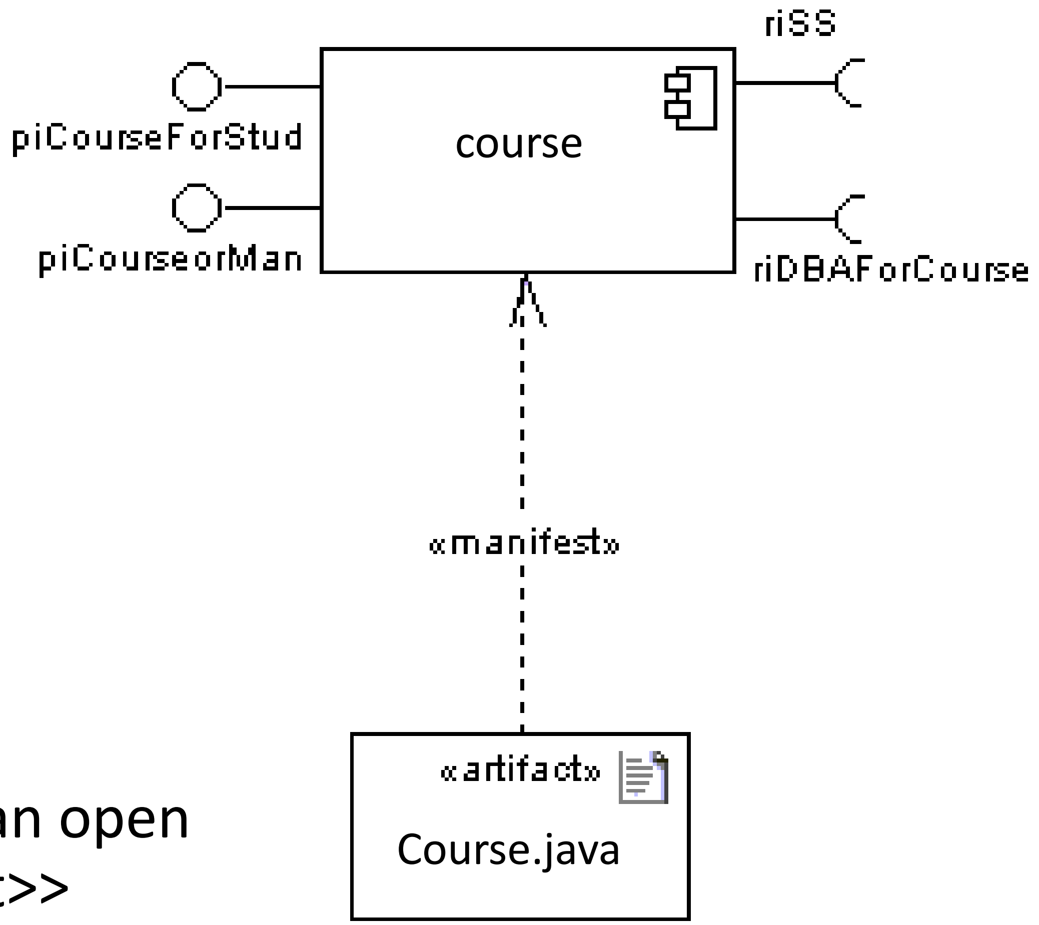
## INTERNAL VIEW

- Components can be built recursively

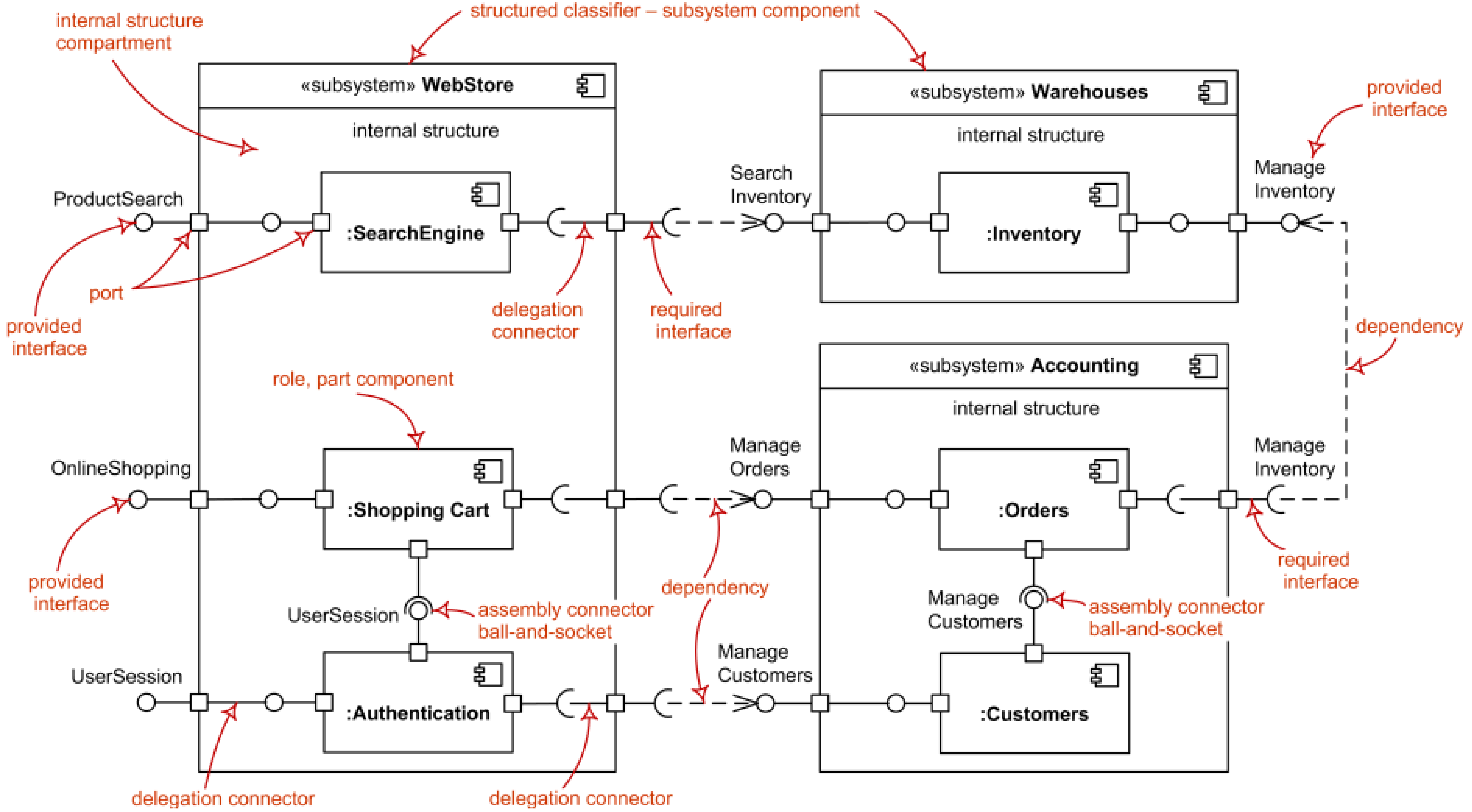


## The <<Manifest>> Relation

- An artifact manifest one or more model elements
- A <<manifestation>> is the concrete physical of one or more model elements by an artifact
- This model element often is a component



- A manifestation is notated as a dashed line with an open arrow-head labeled with the keyword <<manifest>>



Picture from umldiagrams.org

CSCI 4050/6050
Software Engineering

# Diagrams for System Architecture

## Deployment Diagrams

Dr. Eman Saleh eman.saleh@uga.edu

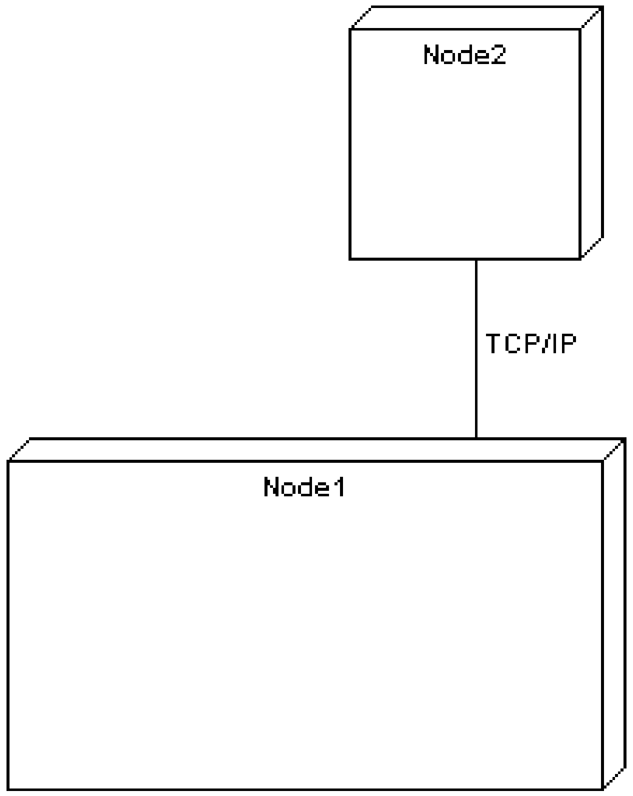
## DEPLOYMENT DIAGRAMS

- Deployment diagrams
  - Show the physical relationship between hardware and software in a system
  - Hardware elements:
    - Computers (clients, servers)
    - Embedded processors
    - Devices (sensors, peripherals)
  - Are used to show the nodes where software components reside in the run-time system

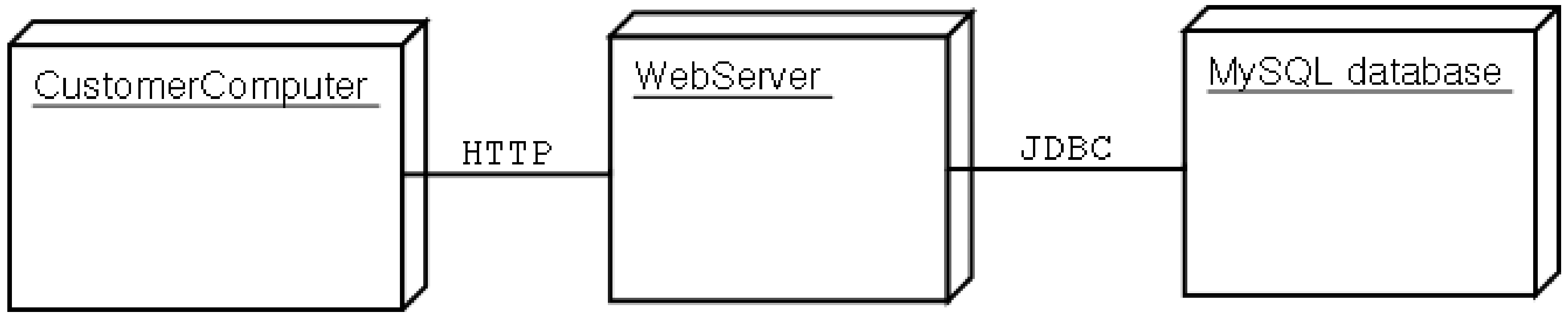
- There is a strong link between components diagrams and deployment diagrams

## DEPLOYMENT DIAGRAMS

- Deployment diagram
  - Contains nodes and connections
  - A node usually represent a piece of hardware in the system
- A connection depicts the communication path used by the hardware to communicate
- Usually indicates the method such as TCP/IP



## Deployment Diagram

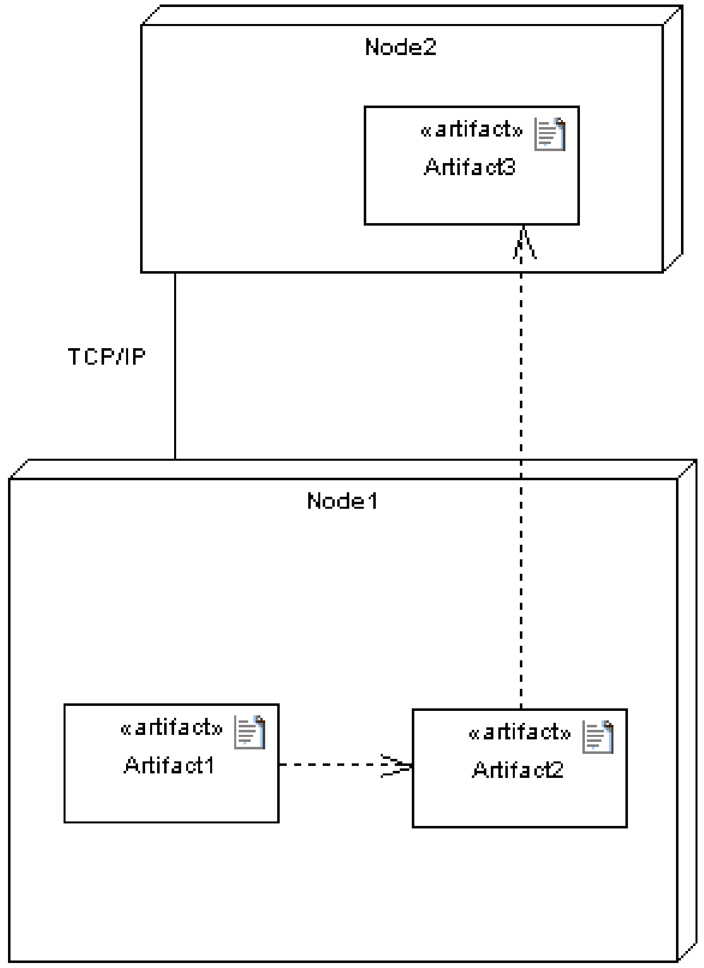


- Captures the distinct number of computers involved
- Shows the communication modes employed
- Component diagrams can be embedded into deployment diagrams effectively…

# DEPLOYMENT DIAGRAMS

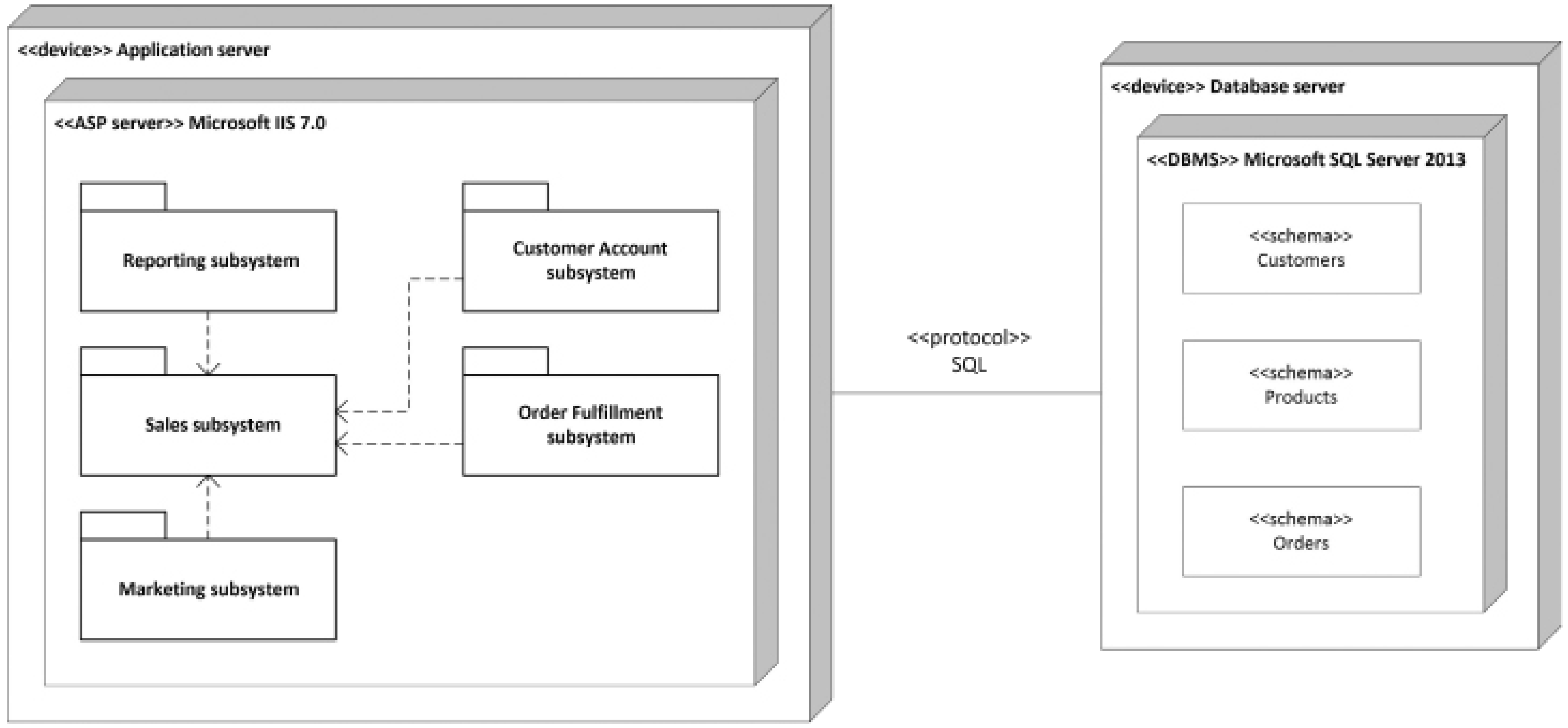
- Deployment diagrams contain artifacts

- **An artifact**
  - Is the specification of a phisycal piece of information
  - Ex: source files, binary executable files, table in a database system,….
  - An artifact defined by the user represents a concrete element in the physical world



## Diagrams for System Architecture

- Deployment Diagrams
  - How the components of a network are interconnected


<<device>> Application server
<<ASP server>> Microsoft IIS 7.0
Reporting subsystem
Customer Account subsystem
Sales subsystem
Order Fulfillment subsystem
Marketing subsystem
<<protocol>> SQL
<<device>> Database server
<<DBMS>> Microsoft SQL Server 2013
<<schema>> Customers
<<schema>> Products
<<schema>> Orders

# Exercise : Draw the Deployment Diagram

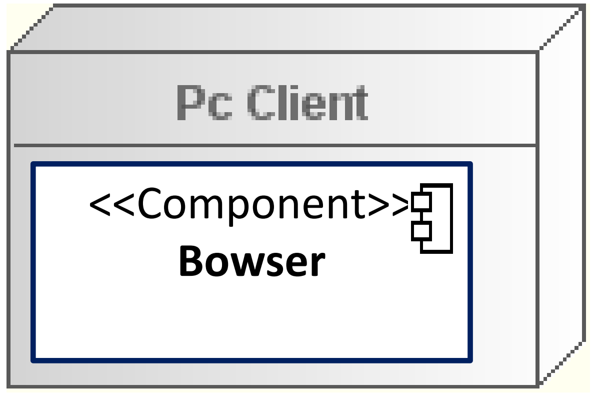
- Users access the Reporting Tool by using a browser running on their local machine and connecting via HTTP over their company's intranet to the Reporting Tool.

- This tool physically runs on the Application Server named w3reporting.myco.com.

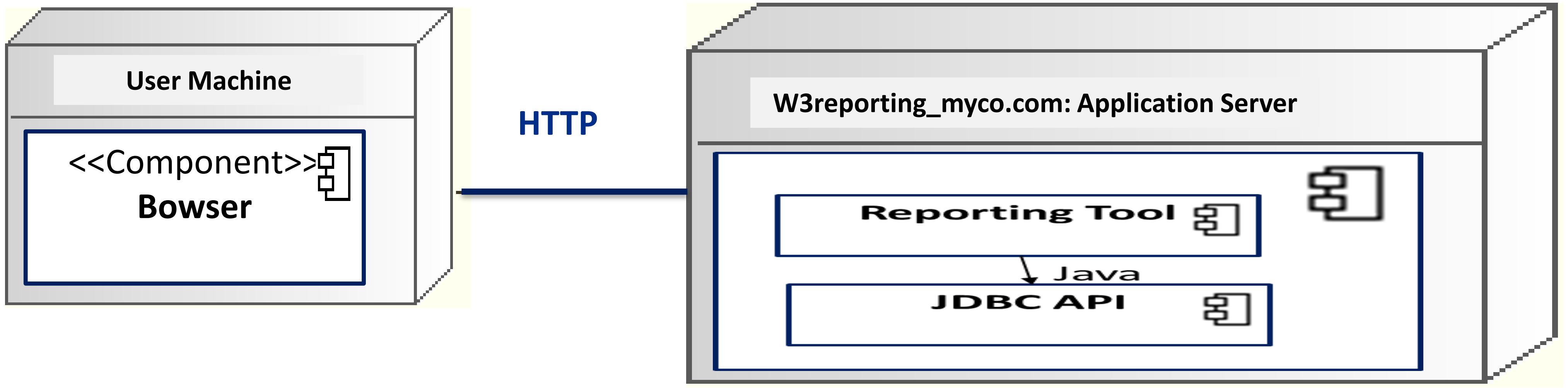
- The Reporting Tool connects to its reporting database using the Java language to IBM DB2's JDBC interface, which then communicates to the actual DB2 database running on the server named db1.myco.com using native DB2 communication.

- In addition to talking to the reporting database, the Report Tool component communicates via SOAP over HTTPS to the Billboard Service

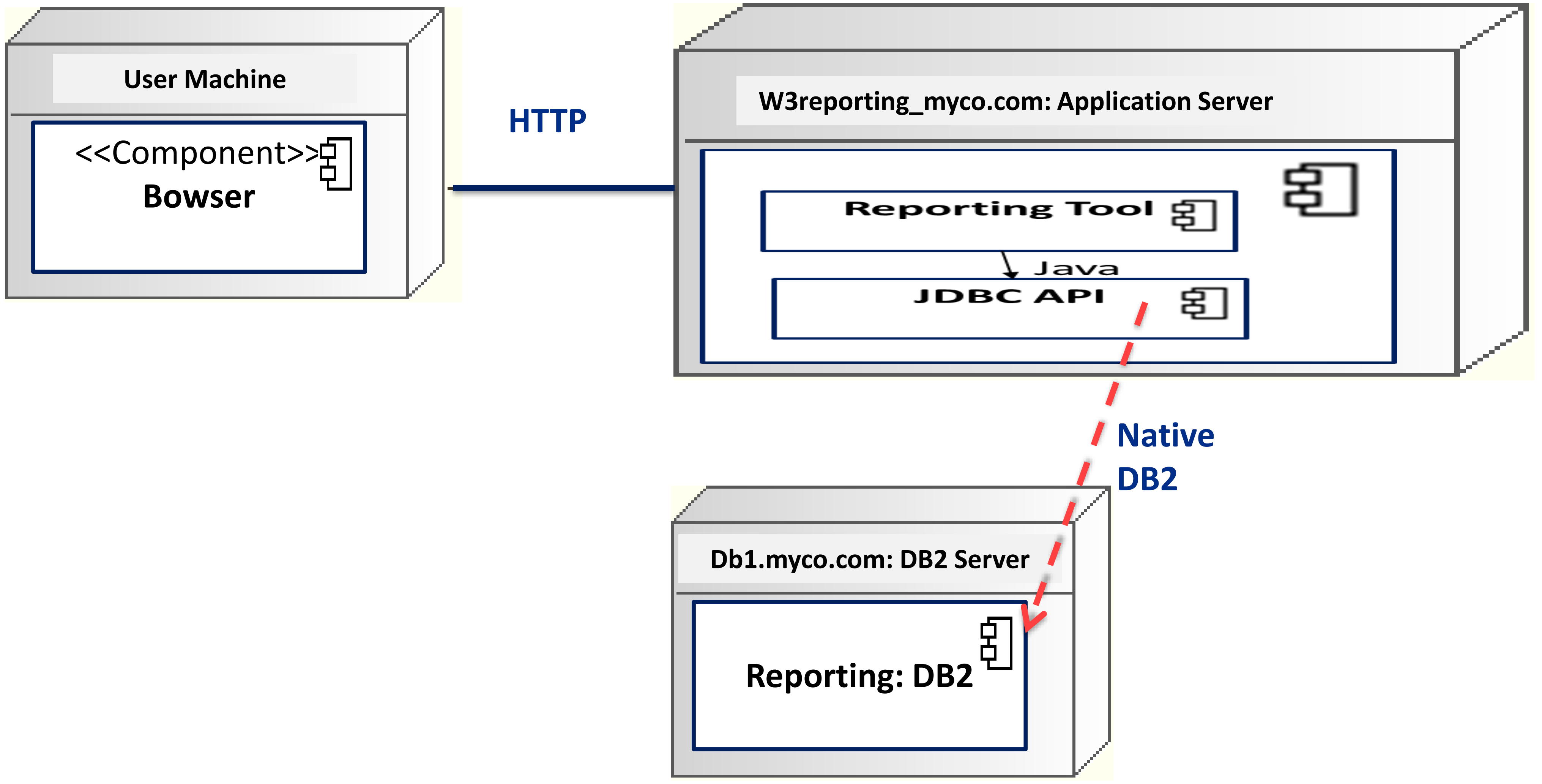
- Users access the Reporting Tool by using a browser running on their local machine and connecting via HTTP over their company's intranet to the Reporting Tool.
- This tool physically runs on the Application Server named w3reporting.myco.com.
- The Reporting Tool connects to its reporting database using the Java language to IBM DB2's JDBC interface, which then communicates to the actual DB2 database running on the server named db1.myco.com using native DB2 communication.
- In addition to talking to the reporting database, the Report Tool component communicates via SOAP over HTTPS to the Billboard Service



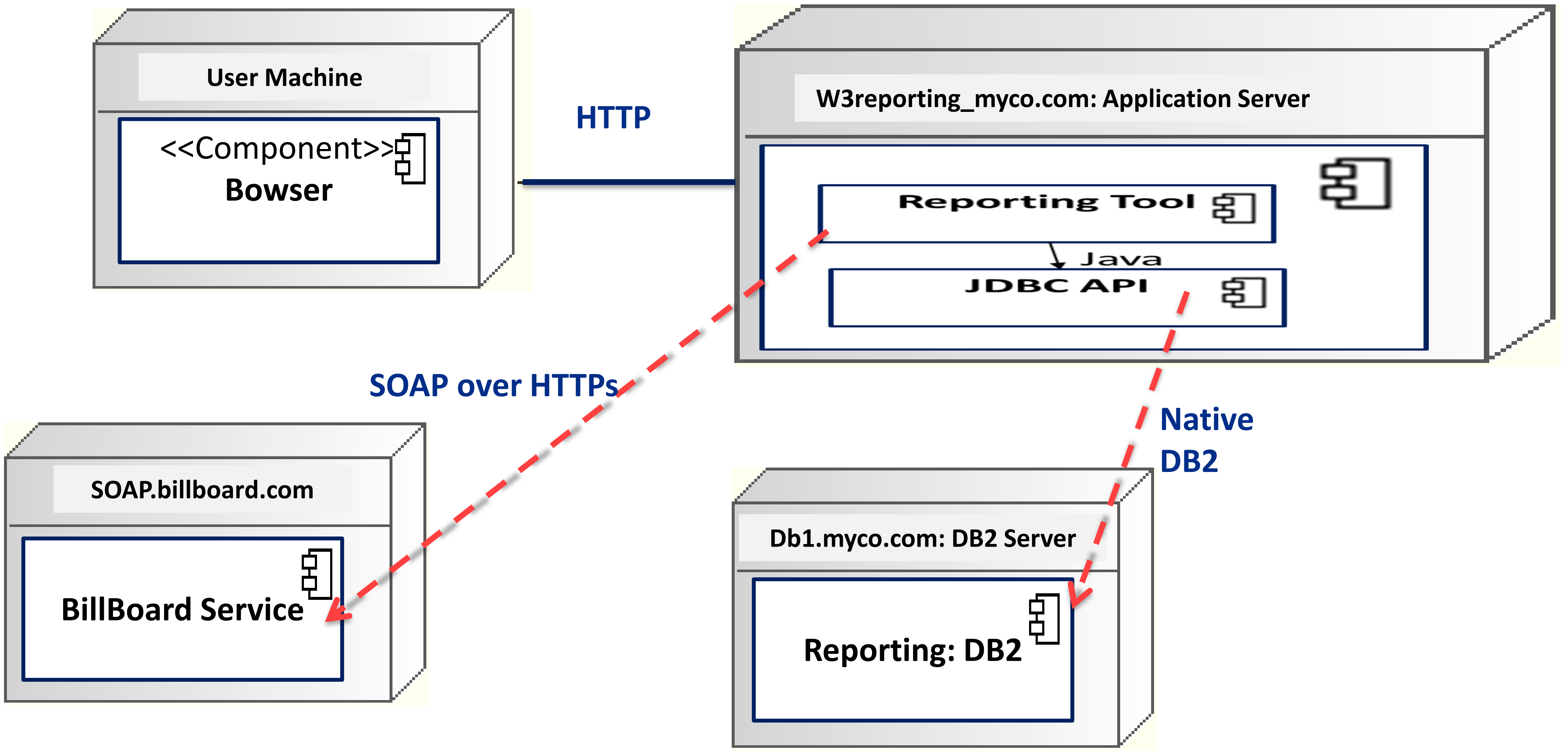
- Users access the Reporting Tool by using a browser running on their local machine and connecting via HTTP over their company's intranet to the Reporting Tool.
- This tool physically runs on the Application Server named w3reporting.myco.com.
- The Reporting Tool connects to its reporting database using the Java language to IBM DB2's JDBC interface, which then communicates to the actual DB2 database running on the server named db1.myco.com using native DB2 communication.
- In addition to talking to the reporting database, the Report Tool component communicates via SOAP over HTTPS to the Billboard Service



- Users access the Reporting Tool by using a browser running on their local machine and connecting via HTTP over their company's intranet to the Reporting Tool. This tool physically runs on the Application Server named w3reporting.myco.com.
- The Reporting Tool connects to its reporting database using the Java language to IBM DB2's JDBC interface, which then communicates to the actual DB2 database running on the server named db1.myco.com using native DB2 communication.

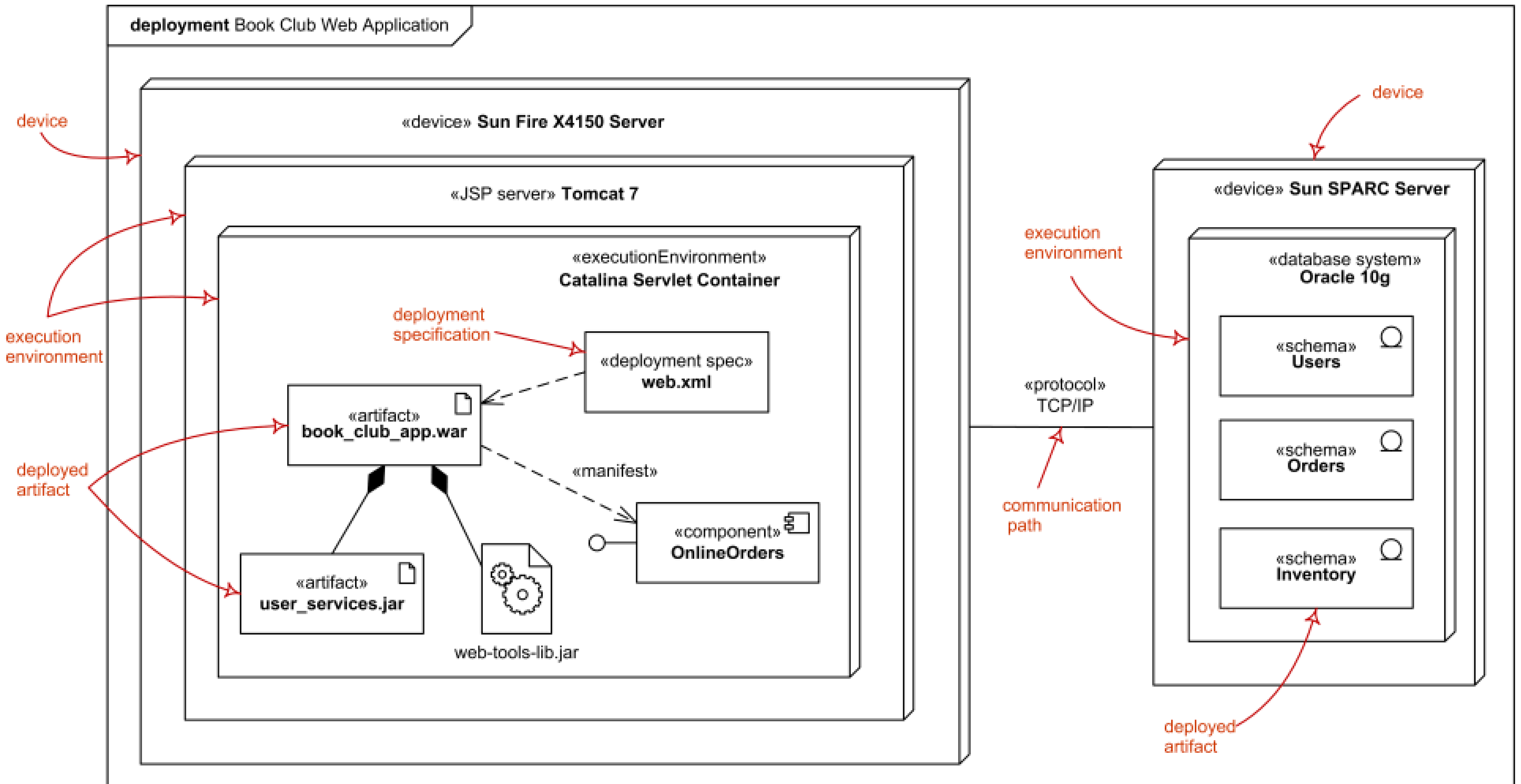


- Users access the Reporting Tool by using a browser running on their local machine and connecting via HTTP over their company's intranet to the Reporting Tool. This tool physically runs on the Application Server named w3reporting.myco.com.
- The Reporting Tool connects to its reporting database using the Java language to IBM DB2's JDBC interface, which then communicates to the actual DB2 database running on the server named db1.myco.com using native DB2 communication.
- In addition to talking to the reporting database, the Report Tool component communicates via SOAP over HTTPS to the Billboard Service



## Specification Level Deployment Diagram

Specification level (also called type level) deployment diagram shows some overview of **deployment** of **artifacts** to **deployment targets**, without referencing specific instances of artifacts or nodes.



## Communication Protocols Stereotypes

- <<Asynchronous>>:

– An asynchronous connection, perhaps via a message bus or message queue.

- <<HTTP>>:

– HyperText Transport Protocol, an Internet protocol.

- <<JDBC>>:

– Java Database Connectivity, a Java API for database access.

- <<ODBC>>:

– Open Database Connectivity, a Microsoft API for database access.

- <<RMI>>:

– Remote Method Invocation, a Java communication protocol.

- <<RPC>>:

– Communication via remote procedure calls.

- <<Synchronous>>:

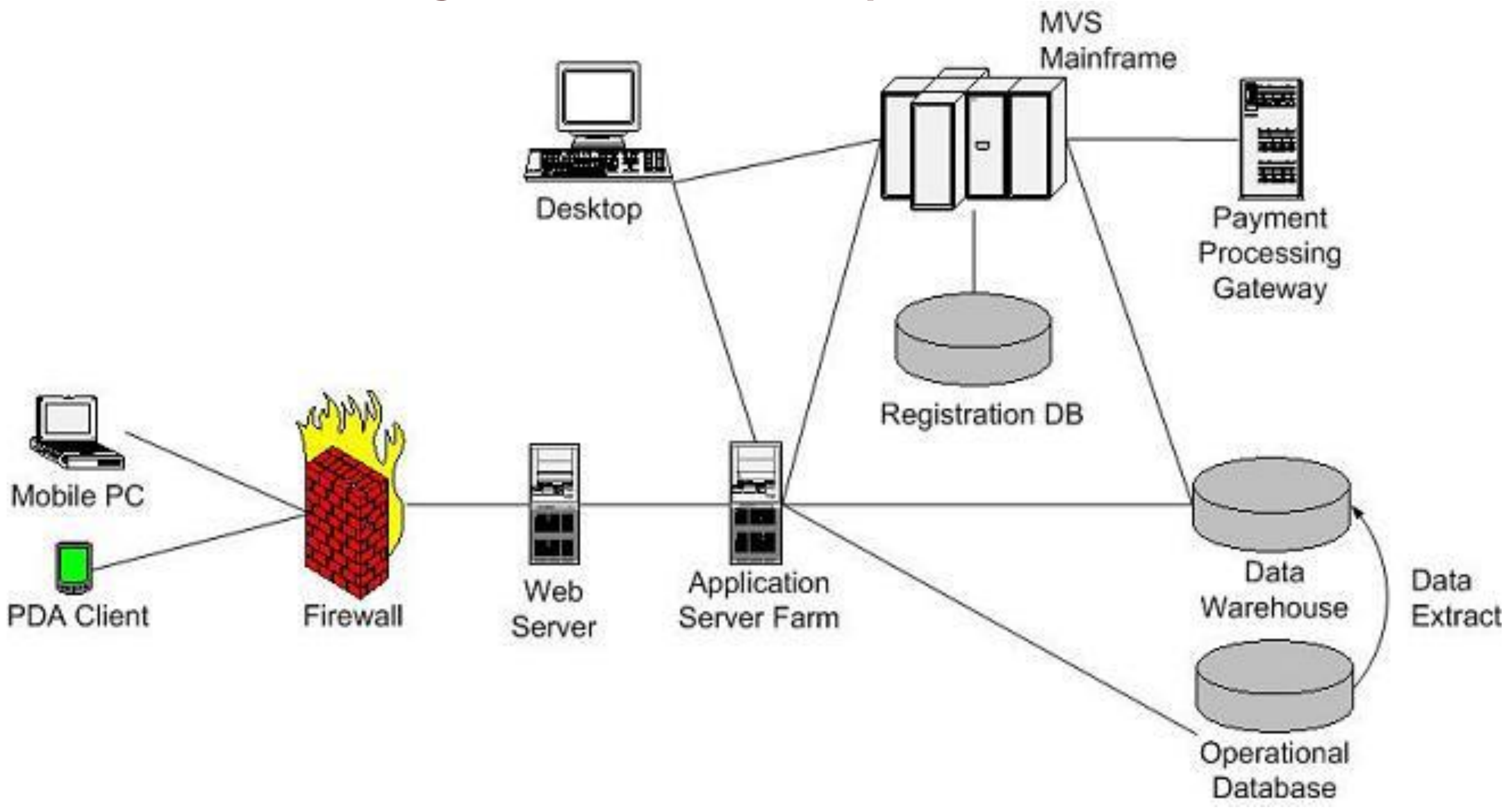
– A synchronous connect where the senders waits for a response from the receiver.

- <<web services>>:

– Communication is via Web Services protocols such as Simple Object Access Protocol (SOAP) and Universal Description, Discovery, and Integration (UDDI) used for building service registry centers

## Network Diagrams

- Network diagrams are often drawn using software-based drawing tools (figure below was drawn using Microsoft Visio)



## REFERENCIES

- The Diagrams of UML 2.0

  by Scott W. Ambler, 2003-2004

  www.agilemodeling.com/essays/umlDiagrams.htm
- UML overview

  By Mandar Chitnis, Pravin Tiwari, & Lakshmi Ananthamurthy

  http://www.developer.com/design/article.php/1553851
- UML-Diagrams.Org, last visited 7/18/2020

https://www.uml-diagrams.org/deployment-diagrams-overview.html