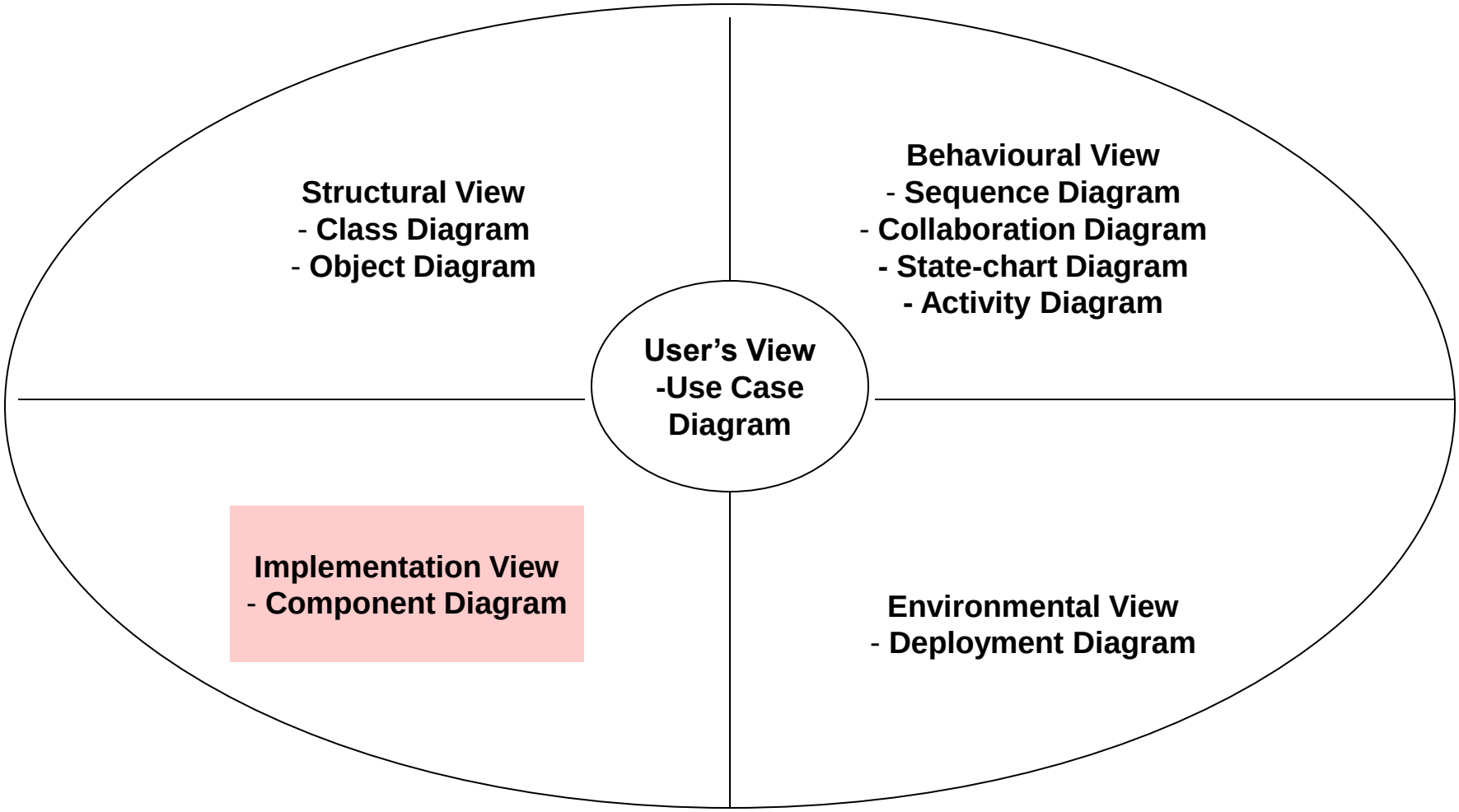
Chapter 5

Implementation View

UML Diagrams



Diagrams and views in UML

Component Diagram cont…

- So far diagrams deals with conceptual entities.
  - Class diagram represent a concept: an abstraction of item that fit into this category.
  - State diagram represent a concept: changes in the state of object.
- Component diagrams deals with separate types of entities: a software component.

Component Diagram cont…

- It is an **Implementation diagrams.**
  - Describe the different elements required for implementing a system
- It is a **Structure diagrams.**
  - A type of diagram that depicts the elements of a specification that are irrespective of time.

Software Component

- It is a modular part of a system.
  - A component resides in a computer and not in the mind of the analyst.
  - A component provides interfaces to other components.
- A component is an autonomous unit within a system.

Component

- The components can be used to define software systems of arbitrary size and complexity
- UML component diagrams enable to model the high-level software components, and the interfaces to those components
- Important for component-based development (CBD)
- Component and subsystems can be flexibly REUSED and REPLACED

Component

- A dependency exists between two elements if changes to the definition of one element may cause changes to the other.
- 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.

Types

- Deployment components
- Work product components
- Execution components

Component in UML 2.0

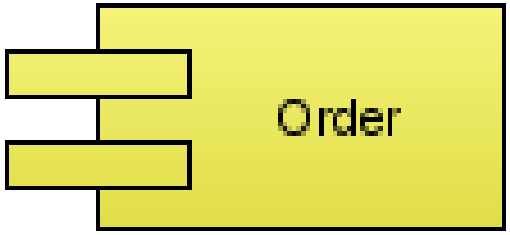
- Modular unit with well-defined interfaces that is replaceable within its environment
- **Autonomous** unit within a system
  - Has one or more provided and required interfaces
  - Its internals are hidden and inaccessible
  - A component is encapsulated
  - Its dependencies are designed such that it can be treated as independently as possible

Case Study

- Development of an application collecting students' opinions about courses
- A student can
  - Read
  - Insert
  - Update
  - Make data permanent about the courses in its schedule
- A professor can only see statistic elaboration of the data
- The student application must be installed in pc client (sw1, sw2)
- The manager application must be installed in pc client (in the manager's office)
- There is one or more servers with DataBase and components for courses management

Component notation

- A component is a subtype of Class which provides for a Component having attributes and operations
- Being able to participate in Associations and Generalizations
- A Component may form the abstraction for a set of realizing Classifiers that realize its behavior
- A Component may optionally have an internal structure and own a set of Ports that formalize its interaction points



Component elements

- A component can have
  - Interfaces

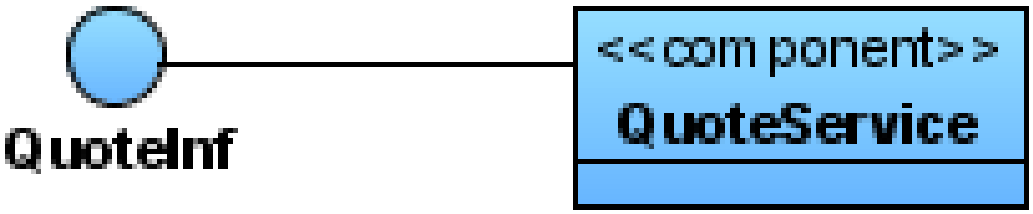
    An interface represents a declaration of a set of operations and obligations
  - Usage dependencies

    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

Interface

- An interface is a kind of classifier that represents a declaration of a set of coherent public features and obligations
- An interface specifies a contract; any instance of a classifier that realizes the interface must fulfill that contract.
- The obligations that may be associated with an interface are in the form of various kinds of constraints (such as pre- and post-conditions) or protocol specifications, which may impose ordering restrictions on interactions through the interface



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

Types of Interfaces

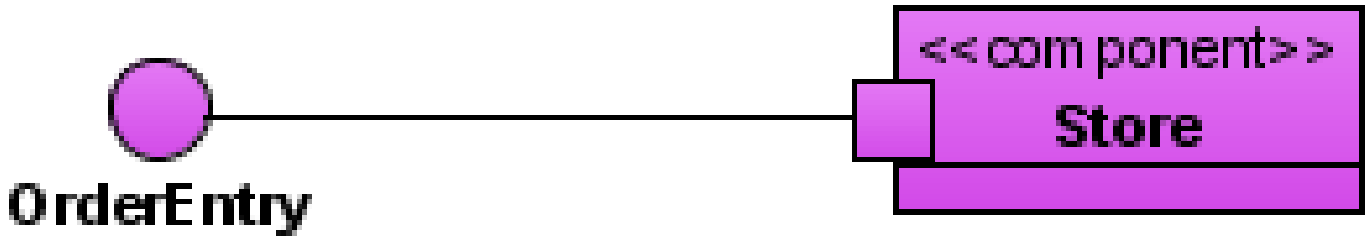
- Provided
- Required

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

Component has provided Port (typed by Interface)

- Ports represent interaction points between a component and its environment
- The interfaces associated with a port specify the nature of the interactions that may occur over a port.
- The provided interfaces of a port characterize requests to the component that its environment may make through this port.

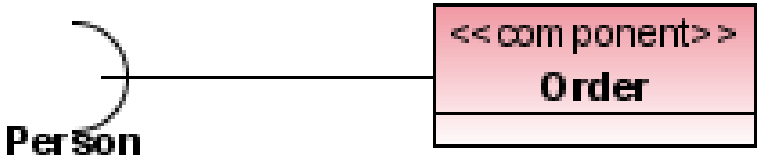


Required Interface

- 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

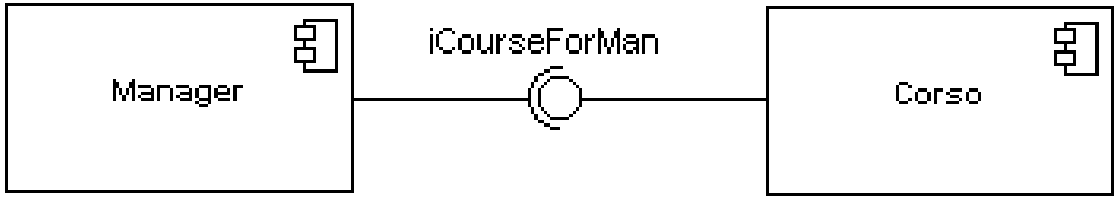
Component uses Interface

- The required interfaces of a port characterize the requests which may be made from the component to its environment through this port.



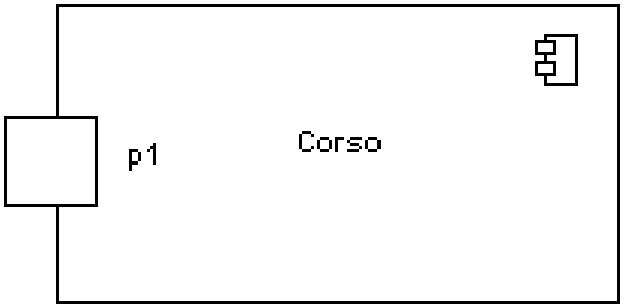
Interface

- 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



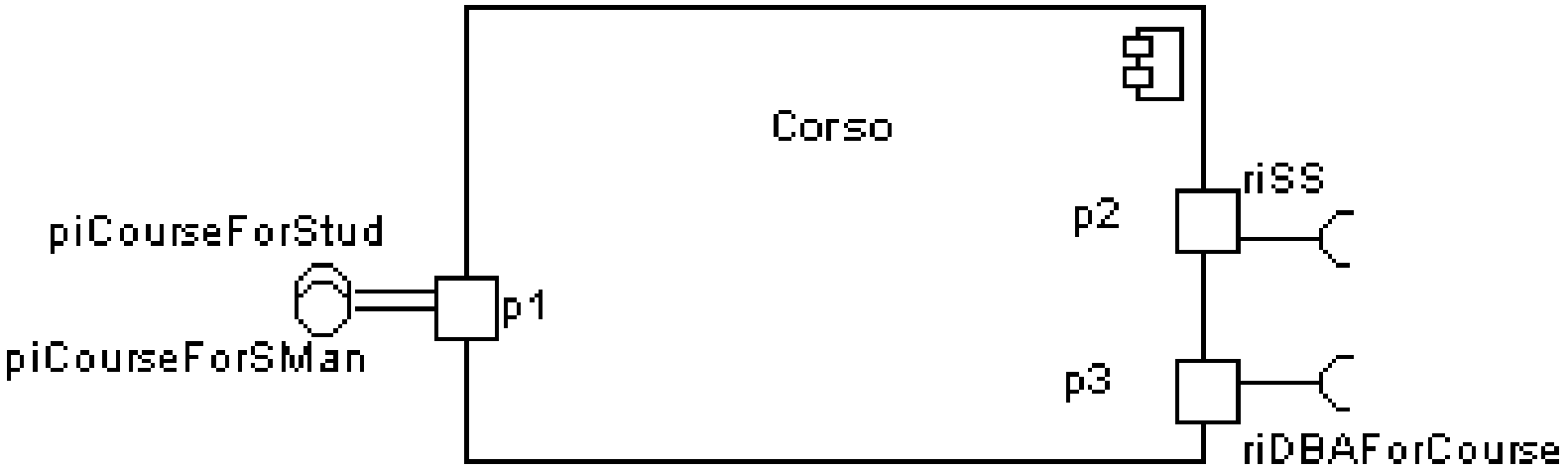
Port

- Specifies a distinct interaction point
  - Between that component and its environment
  - Between that component and its internal parts



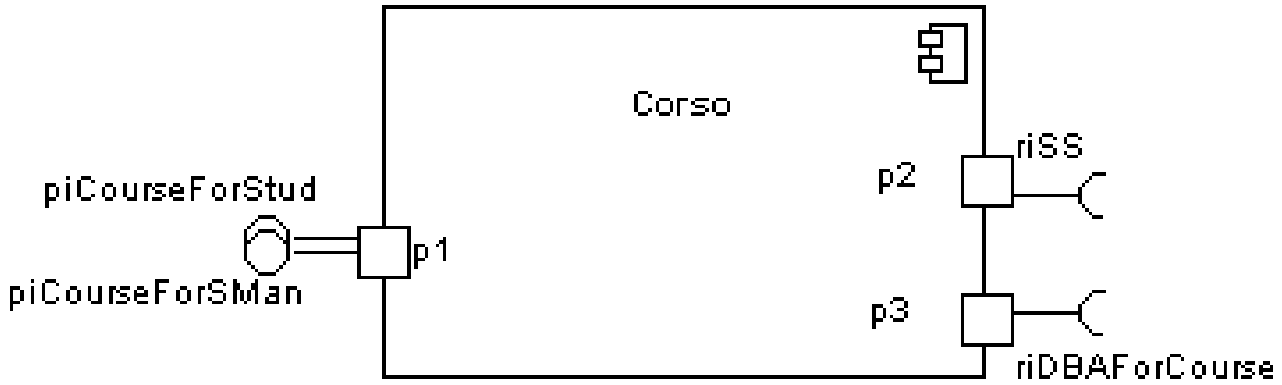
Port

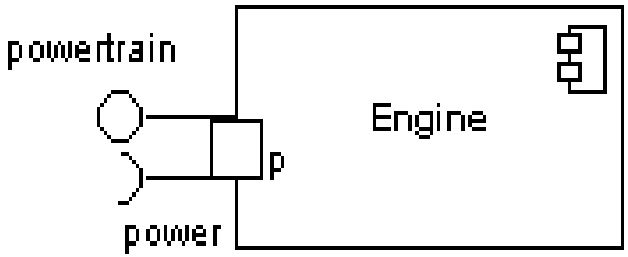
- Is shown as a small square symbol
- Ports can be named, and the name is placed near the square symbol
- Is associated with the interfaces that specify the nature of the interactions that may occur over a port



Port

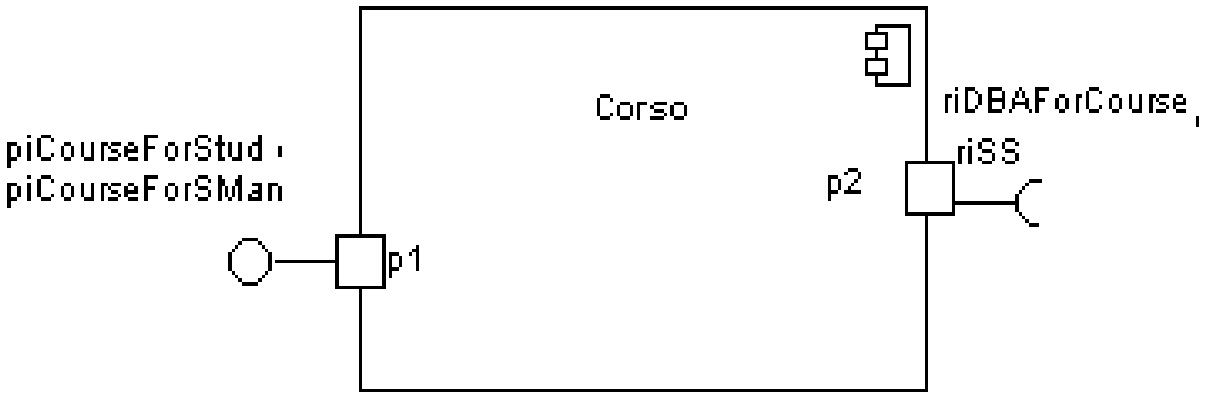
- Ports can support unidirectional communication or bi-directional communication





Port

- If there are multiple interfaces associated with a port, these interfaces may be listed with the interface icon, separated by a commas

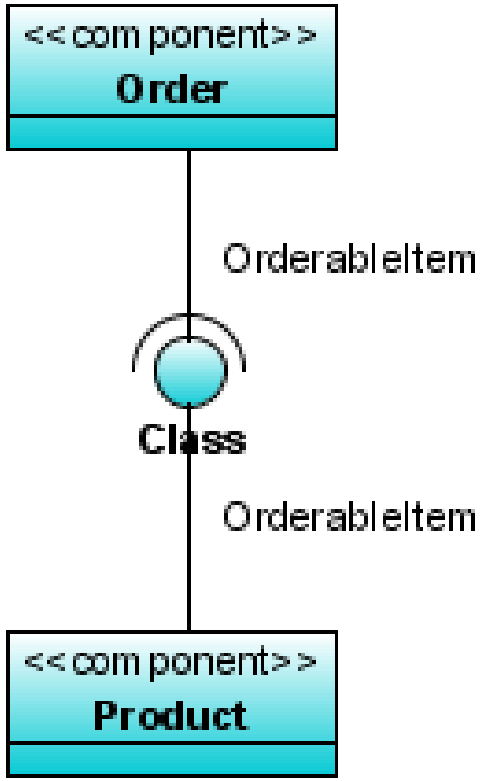


Port

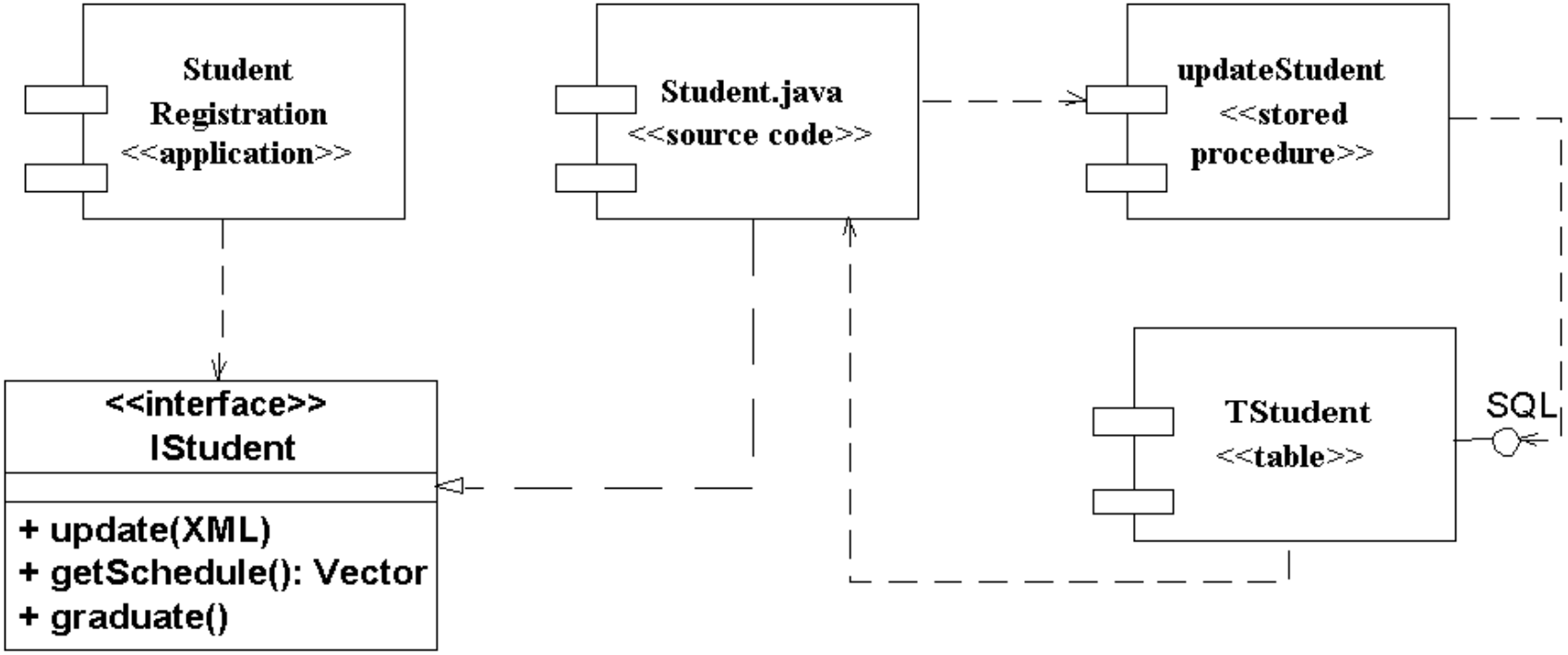
- All interactions of a component with its environment are achieved through a port
- The internals are fully isolated from the environment
- This allows such a component to be used in any context that satisfies the constraints specified by its ports
- Ports are not defined in UML 1.x

Assembly connector

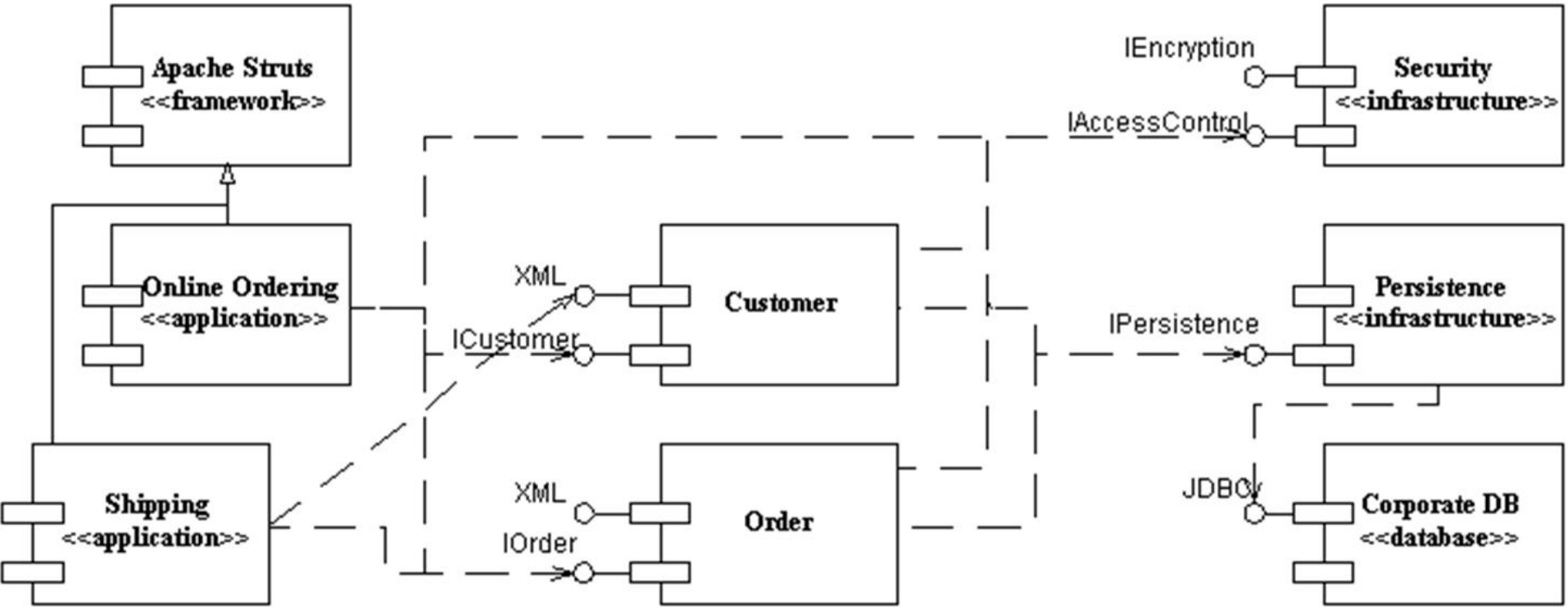
- An assembly connector is a connector between two components that defines that one component provides the services that another component requires.
- An assembly connector is a connector that is defined from a required interface or port to a provided interface or port



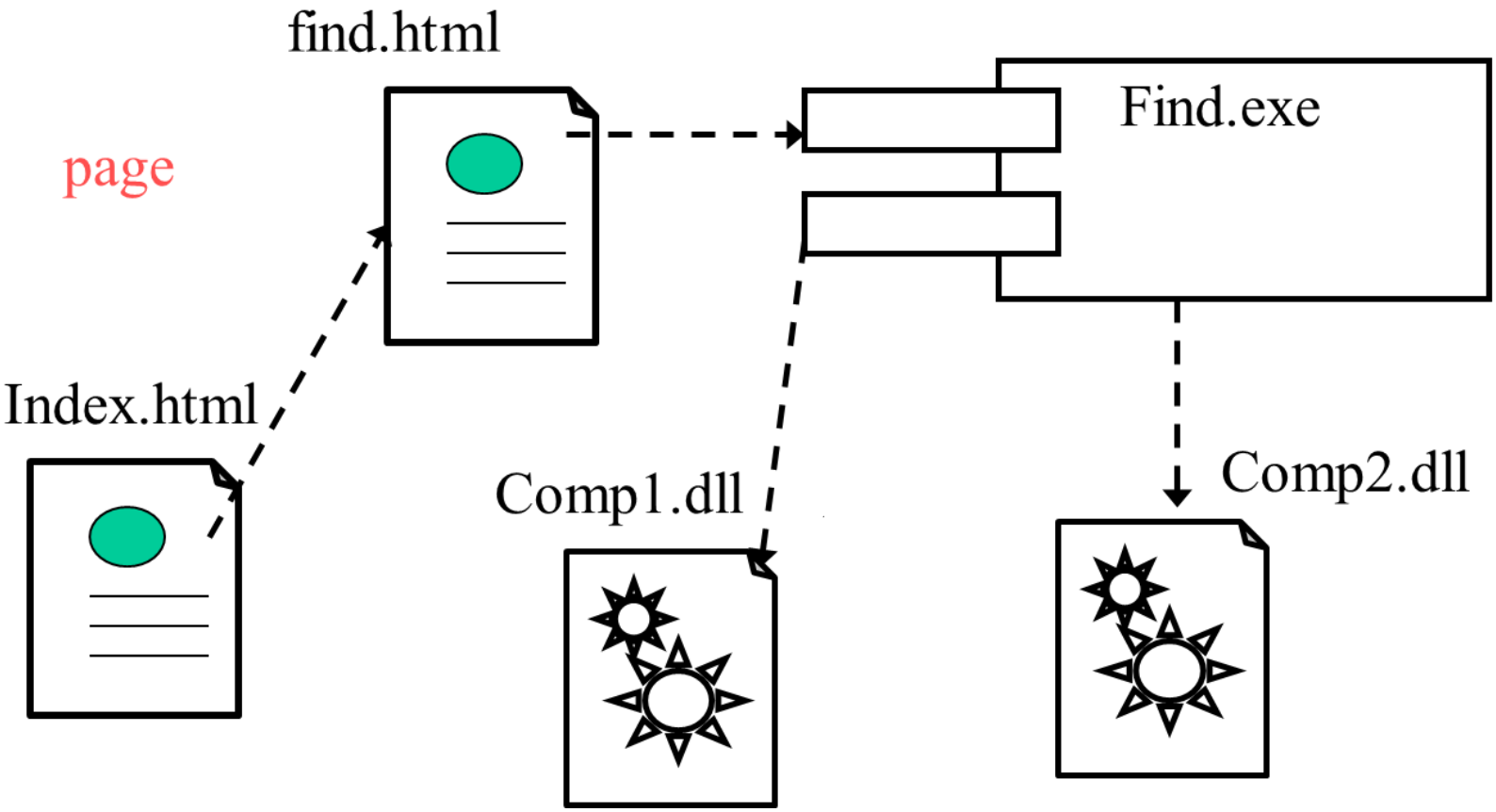
Sample interfaces



Example Component Diagram



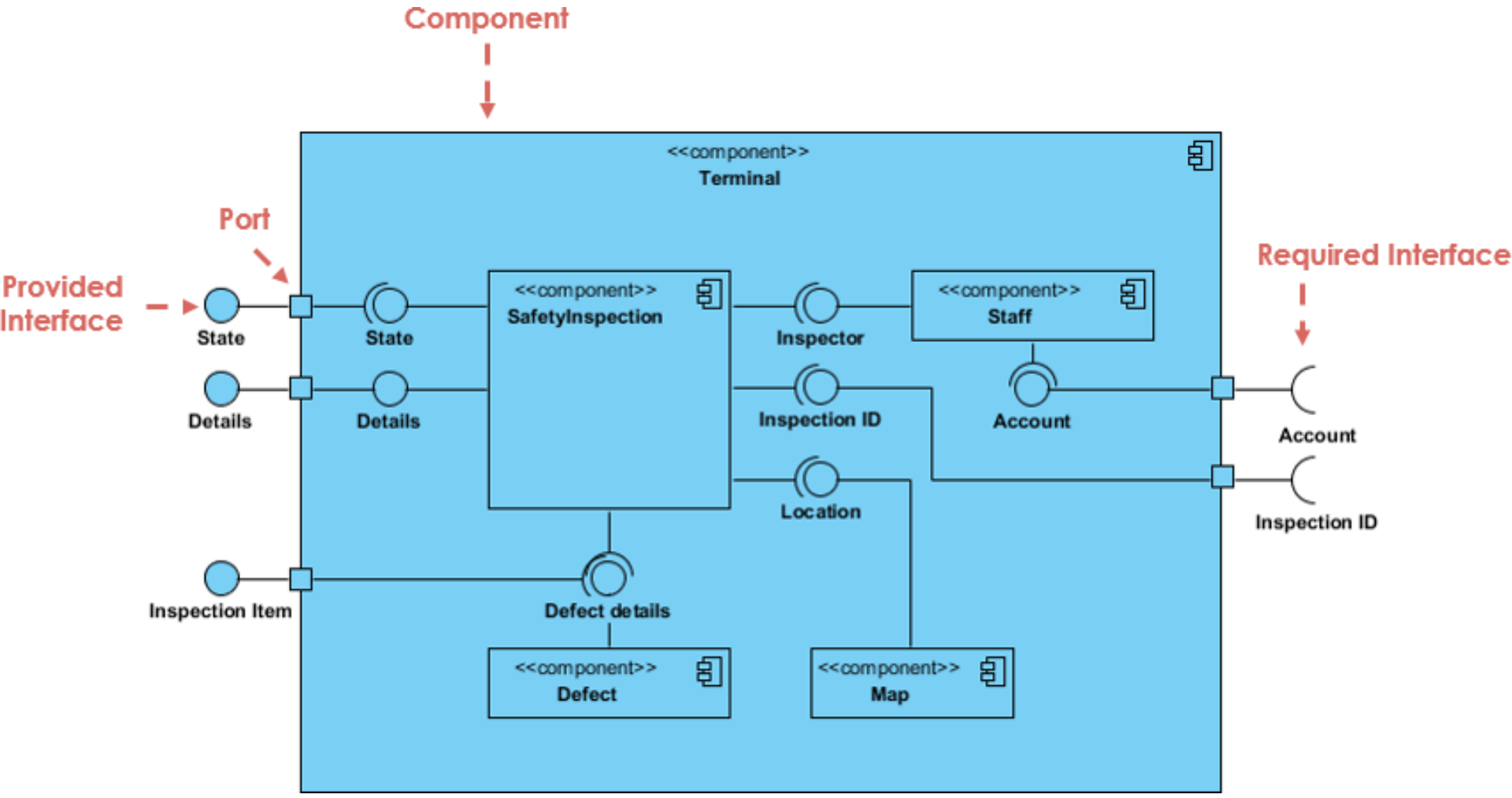
Component Diagram



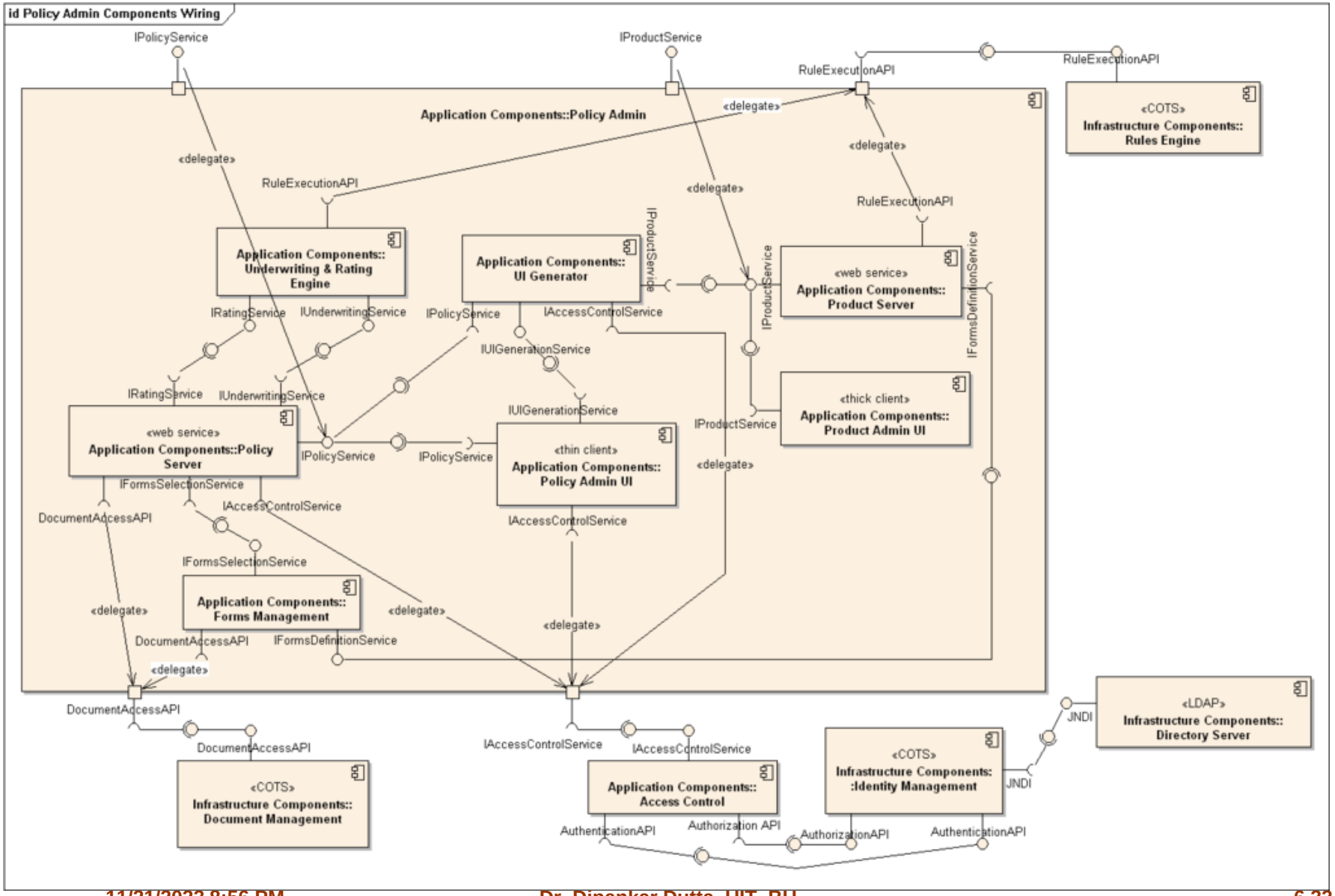
Common Stereotypes

Stereotype	Indicates
<<application>>	A "front-end" of your system, such as the collection of HTML pages and ASP/JSPs that work with them for a browser-based system or the collection of screens and controller classes for a GUI-based system.
<<database>>	A hierarchical, relational, object-relational, network, or object-oriented database.
<<document>>	A document. A UML standard stereotype.
<<executable>>	A software component that can be executed on a node. A UML standard stereotype.
<<file>>	A data file. A UML standard stereotype.
<<infrastructure>>	A technical component within your system such as a persistence service or an audit logger.
<<library>>	An object or function library. A UML standard stereotype.
<<source code>>	A source code file, such as a .java file or a .cpp file.
<<table>>	A data table within a database. A UML standard stereotype
<<web service>>	One or more web services.
<<XML DTD>>	An XML DTD.

The example shows the internal components of a larger component



Component Diagram



Thank you

Questions?