UML Component Diagrams

Introduction

- Usually, when analyzing large software systems we can break them into subsystems.

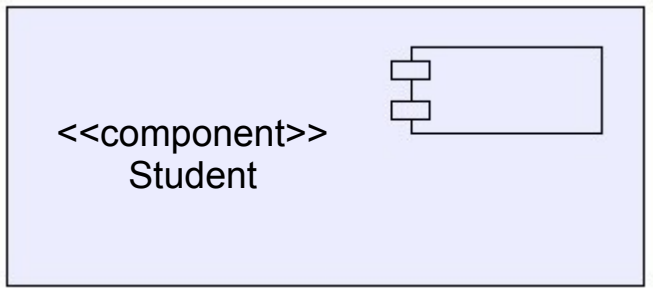
  The UML component diagram was developed to assist us.

- Each component is a replaceable executable piece of a larger system.

  The functionality provided by each component is specified by a set of interfaces the component realizes.

Introduction

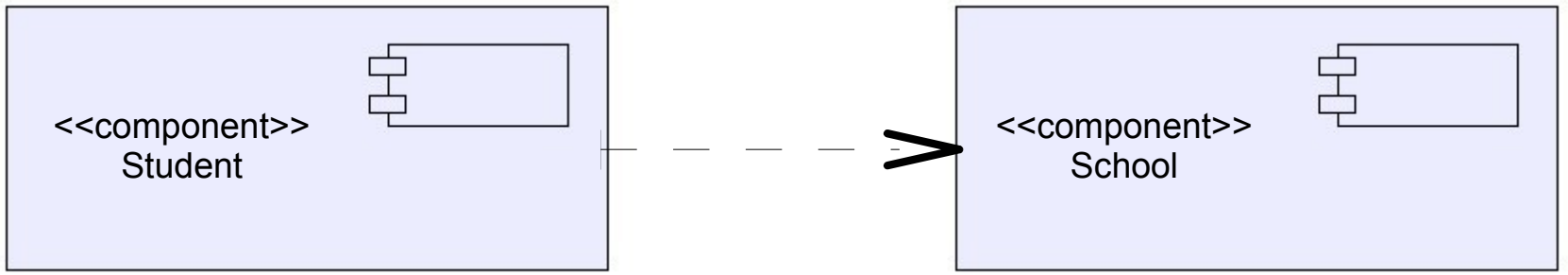
- Each component functions using interfaces available by others.
- The notation used to represent a component is:



- Within the component notation we will write its name.

Component Dependencies

- A dependency between one component and another one is depicted using a dashed line with an open arrow.



- The arrow direction shows the dependency direction.

Component Views

- The component has two possible views. The “Black Box View” and the “White Box View”.

Black Box View

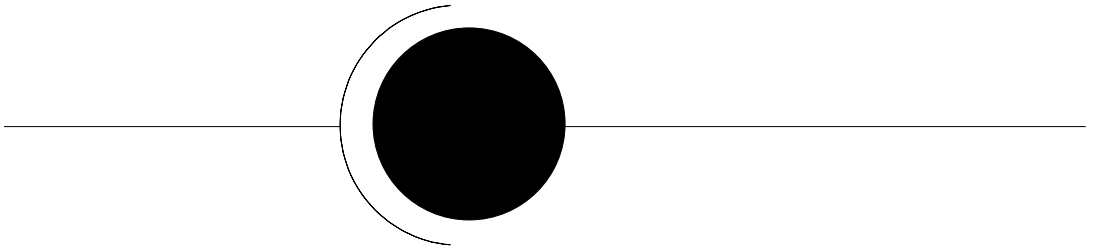
Shows the component from an outside perspective without getting into too many details.

White Box View

Shows the component including how it realizes the interfaces it provides. This is a more detailed view and is usually illustrated with a class diagram.

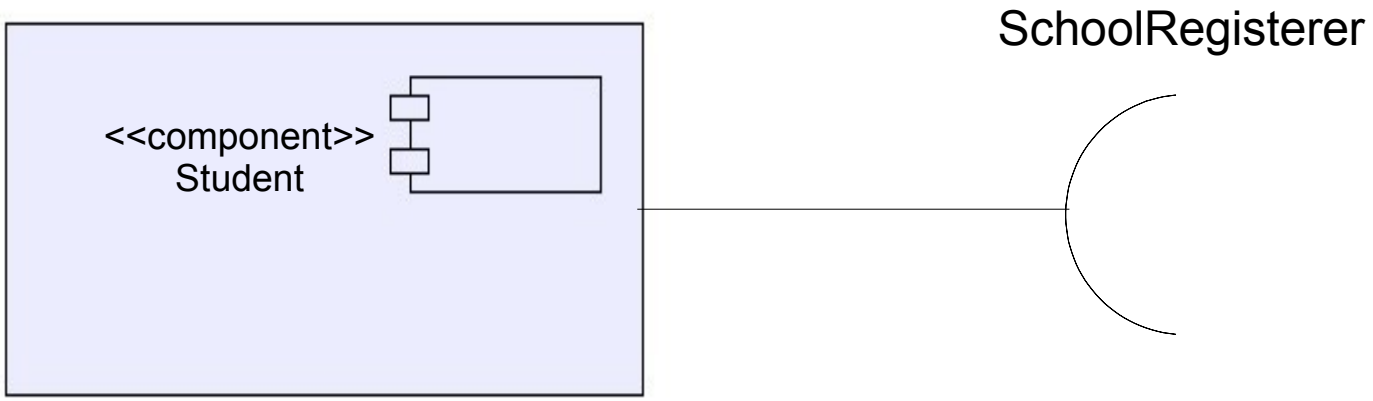
Black Box View

- The black box view doesn't specify anything about its internal implementation.
- The black box view does include details about the interfaces it provides and details about the interfaces it requires.
- The required and provided interfaces are represented using the assembly connectors, which are illustrated using the ball & socket icon:



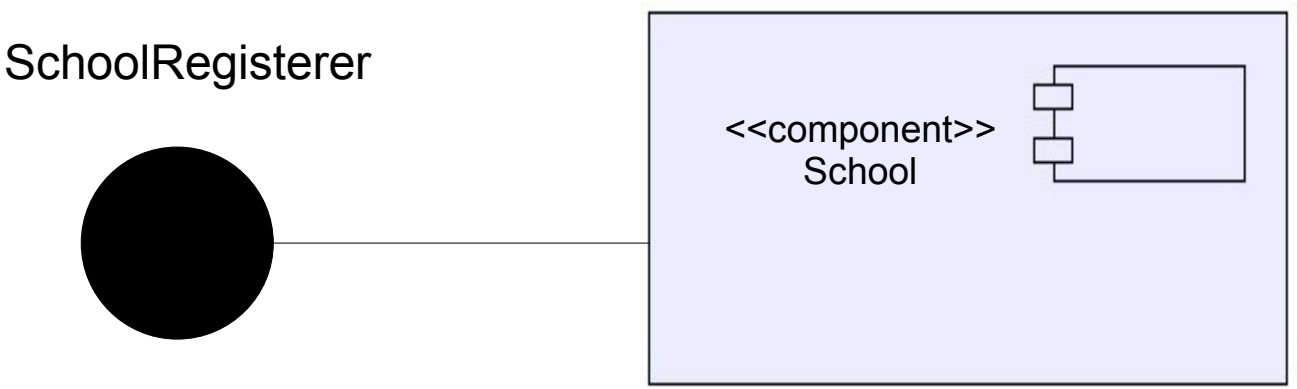
Black Box View

- The required interface is represented using the socket icon.
- The name of the required interface will be written near the connector symbol.



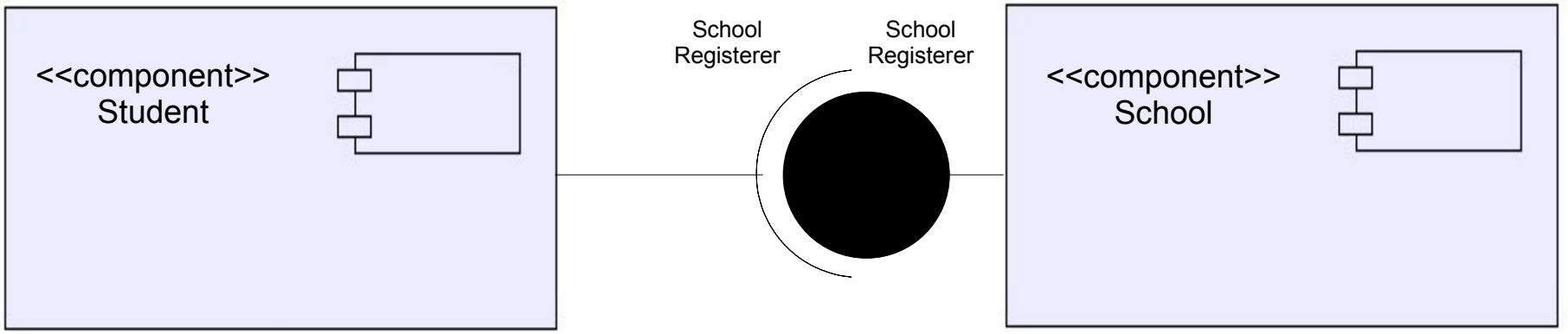
Black Box View

- The provided interface is illustrated using the ball icon.
- The name of the provided interface will be written near the connector symbol.



Black Box View

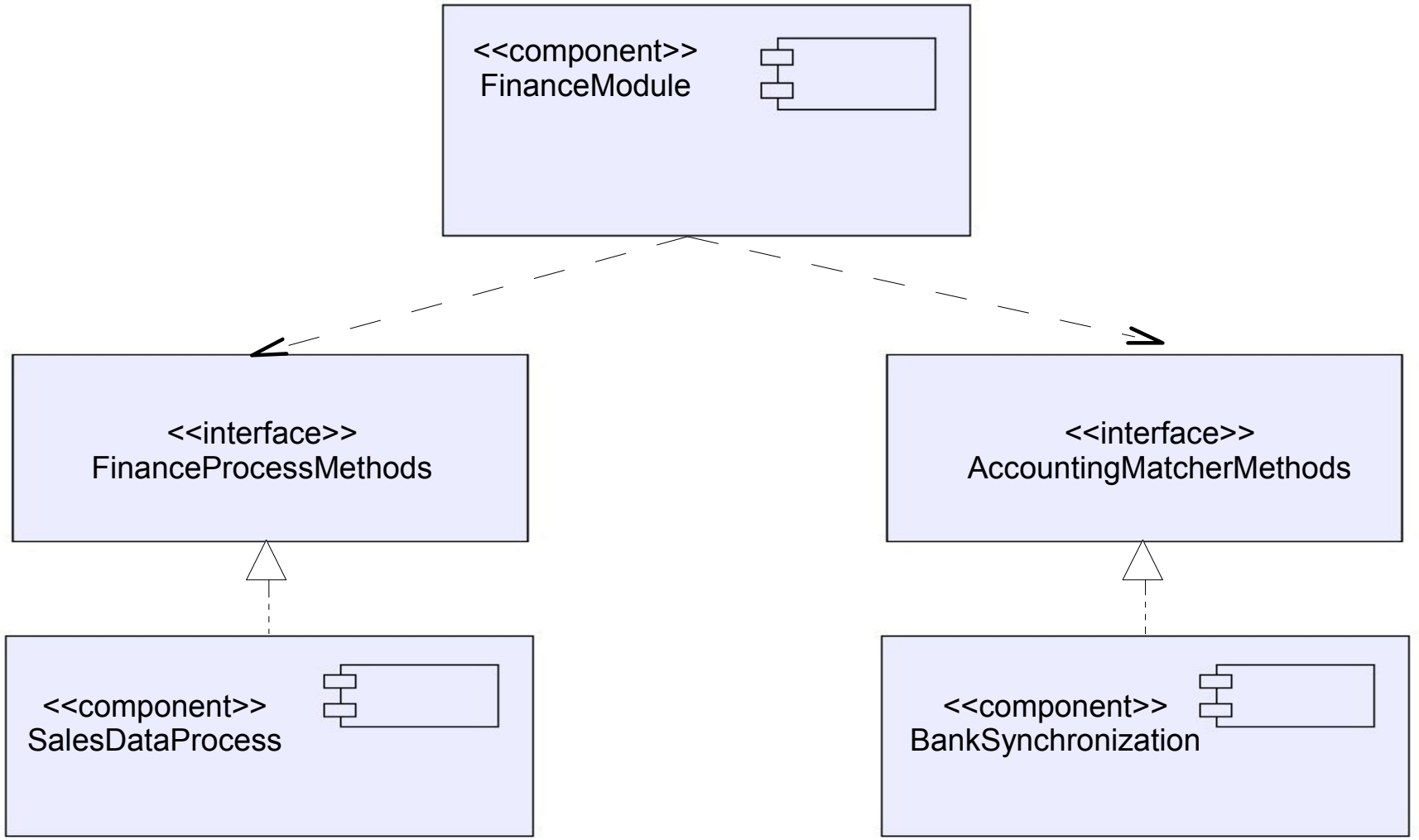
- Writing two components that match their required & provided interfaces is done by simply connecting them via their matching provided and required interfaces.



Black Box View

- The interface's notation is an empty rectangle in which we write the interface name and “<<interface>>” above it.
- A dependency arrow will be drawn from the component to the required interface.

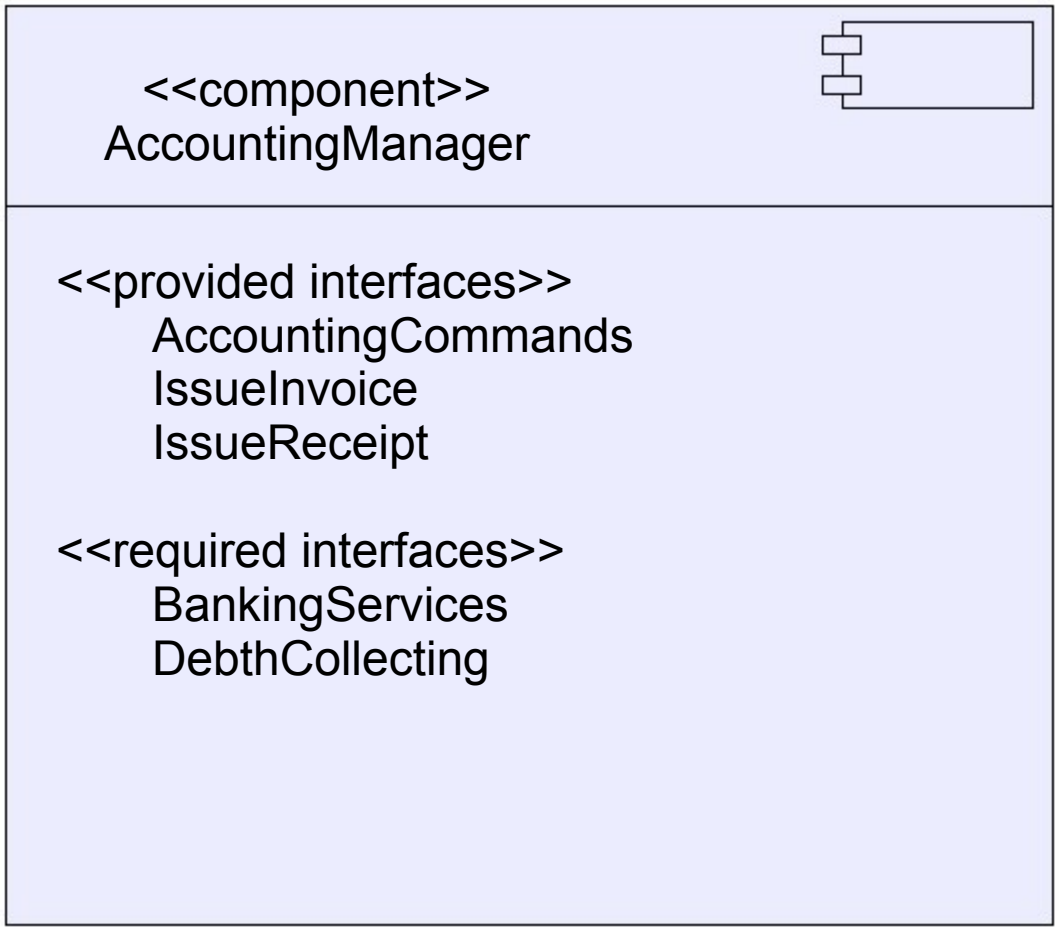
Black Box View



Black Box View

- More detailed info can be provided by drawing the component with compartments.
- The compartment on top includes the name of the class with the `<<component>>` stereotype above. The one on the bottom includes two lists. One is of the required interfaces and the other is of the provided ones. The `<<required>>` and `<<provided>>` stereotypes are used accordingly.

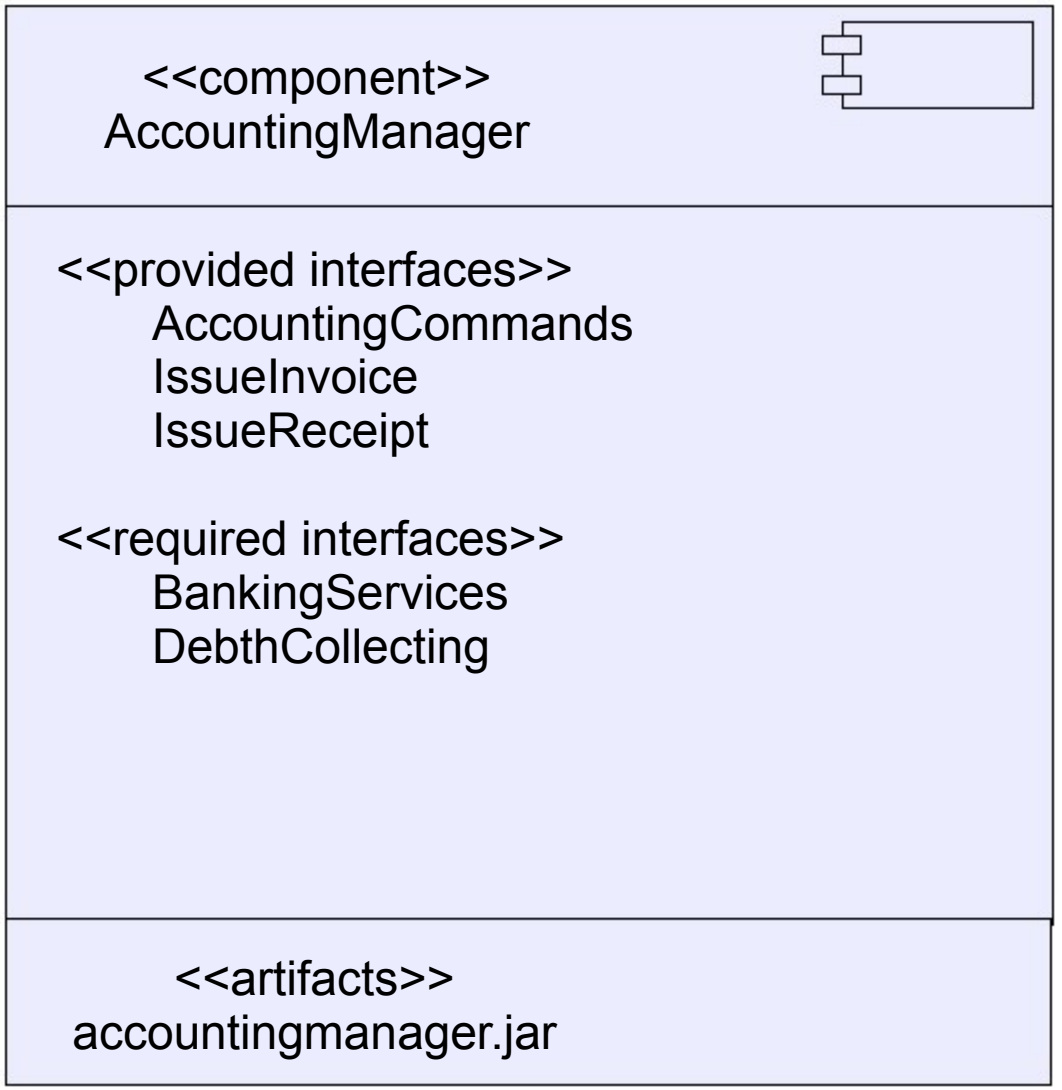
Black Box View



Black Box View

- A more detailed compartment diagram can include a third compartment stereotyped with `<<artifacts>>`, that lists the components' artifact\s.

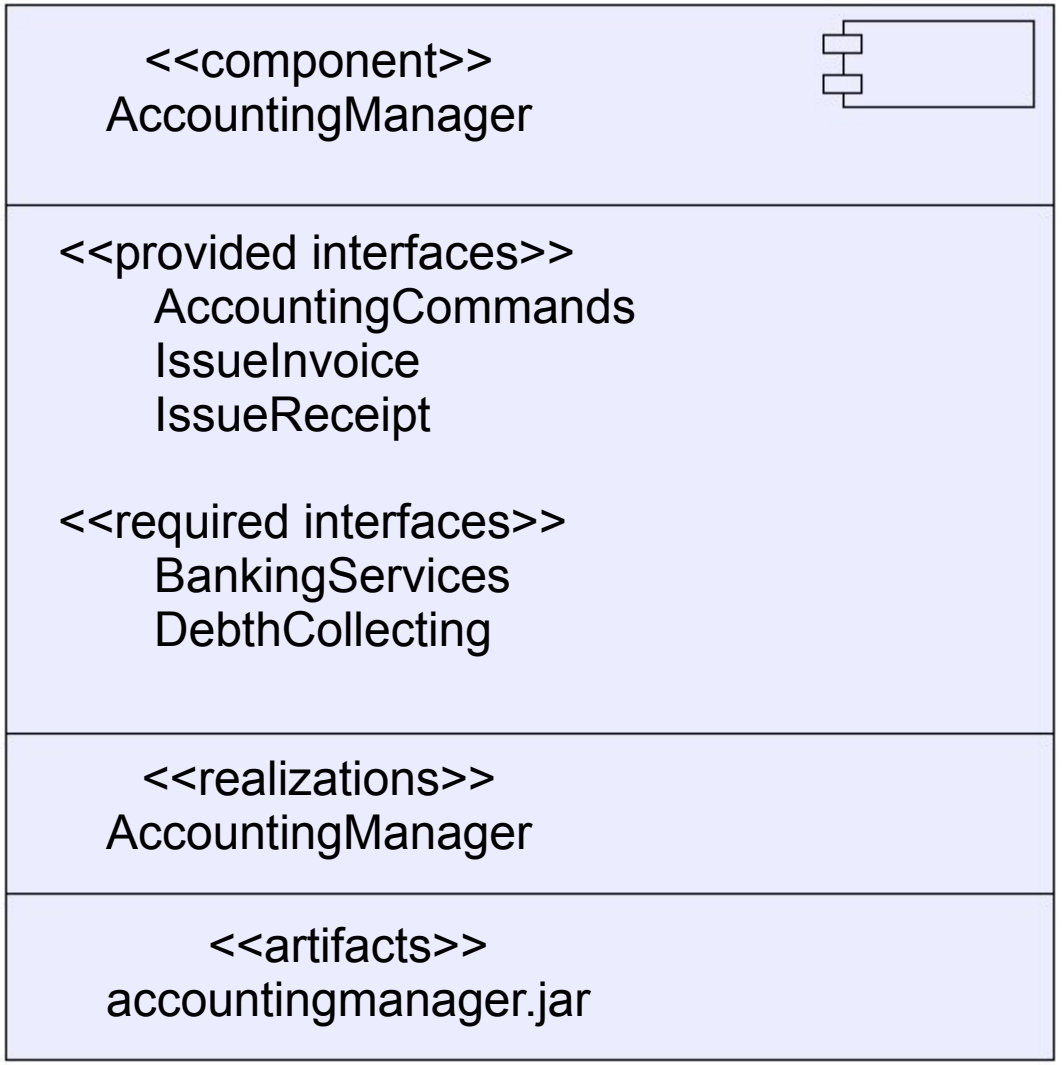
Black Box View



White Box View

- The white box view presents the way a component realizes the interfaces it provides.
- The realization compartment can be added to the compartment diagram and lists the exact classifiers that realize the provided interfaces. This new compartment should be labeled with the `<<realizations>>` stereotype.

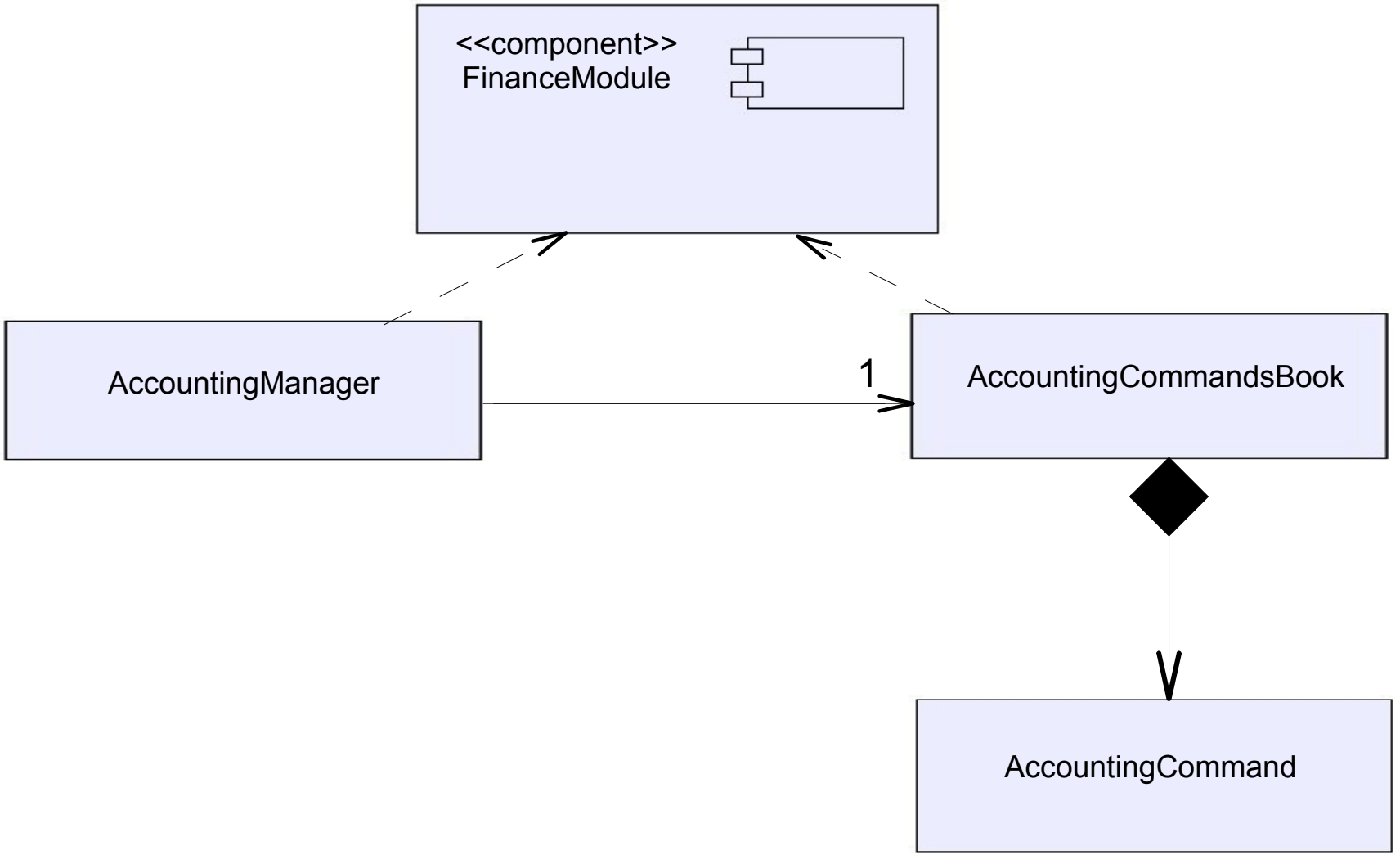
White Box View



White Box View

- To provide more information about the component structure we can create a diagram that shows the component, the classes it includes and the relationships between them.
- Between the classes and the component we will draw dependency relationships.

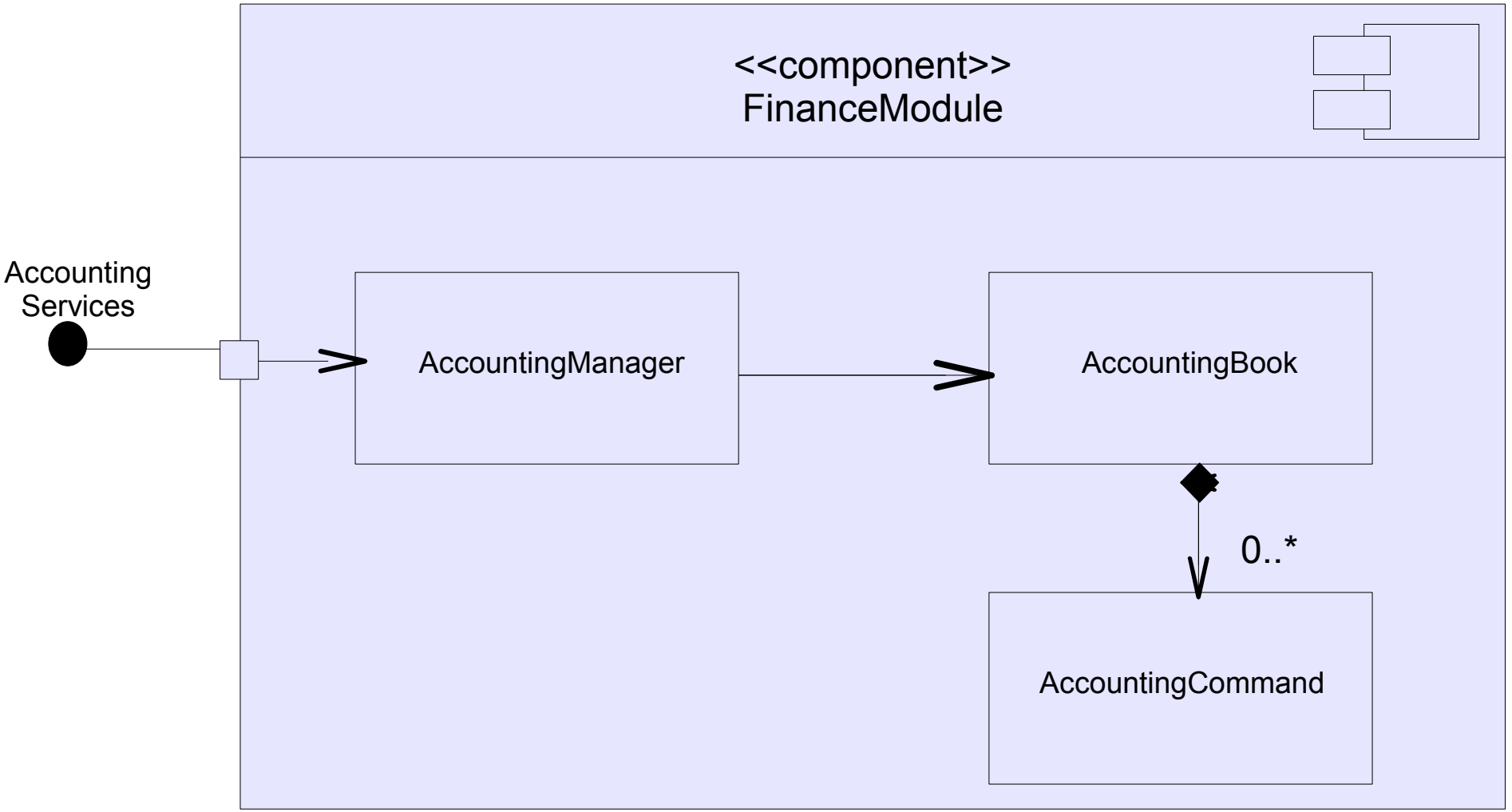
White Box View



White Box View

- An alternative can be drawing the classifiers within the component frame.
- A port, depicted as a small rectangle on one of the component's sides represents a required / provided (or both) functionality.
- Each one of the assembly connectors (both of type ball.. and of type socket...) are drawn as coming out of a port.
- If we draw the classifiers within the component frame then we can connect each port with its responsible class.

White Box View



Component Stereotypes

- The following is a summary list of the stereotypes you can apply to components:

Realization

When realize spec provided by another component. Another component that provides the specification.

Process

When capable of fulfilling functional requests. Transaction based. Holds data about the state.

Service

When fulfilling functional requests by others. Stateless based.

Component Stereotypes

Entity

When representing a business entity. Component for data storage.

Subsystem

A large component, part of a big system. Usually larger than a simple component.

Specification

Component with providing and requiring interfaces and without any implementation.

UML Component Diagrams

Introduction

- Usually, when analyzing large software systems we can break them into subsystems.

  The UML component diagram was developed to assist us.

- Each component is a replaceable executable piece of a larger system.

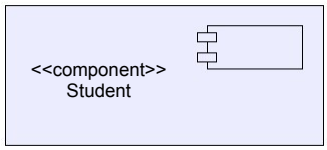
  The functionality provided by each component is specified by a set of interfaces the component realizes.

Visual Basic components were the first to reflect the components' based software development.

Java Beans and Enterprise Java Beans reflect another more advance approach for components based software development.

Introduction

- Each component functions using interfaces available by others.
- The notation used to represent a component is:



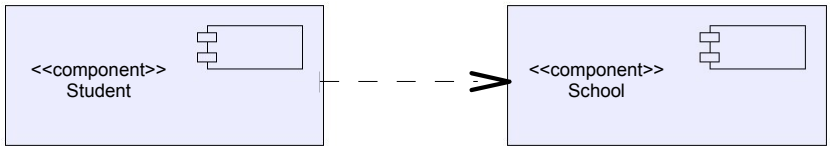
- Within the component notation we will write its name.

The small component icon (rectangle with two smaller rectangles on the right side) is optional.

In the past (UML1.4) the notation for component was:

Component Dependencies

- A dependency between one component and another one is depicted using a dashed line with an open arrow.



- The arrow direction shows the dependency direction.

The arrow direction shows the dependency direction. In this sample Student dependends on the School.

Component Views

- The component has two possible views. The “Black Box View” and the “White Box View”.

  Black Box View

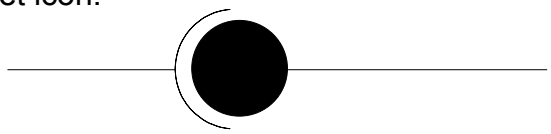
  Shows the component from an outside perspective without getting into too many details.

  White Box View

  Shows the component including how it realizes the interfaces it provides. This is a more detailed view and is usually illustrated with a class diagram.

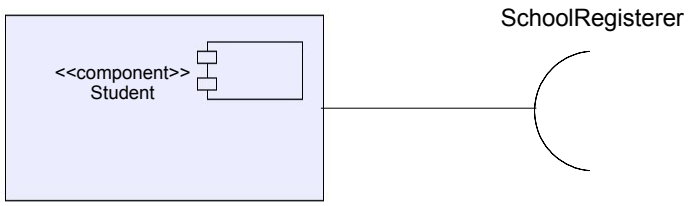
Black Box View

- The black box view doesn't specify anything about its internal implementation.
- The black box view does include details about the interfaces it provides and details about the interfaces it requires.
- The required and provided interfaces are represented using the assembly connectors, which are illustrated using the ball & socket icon:



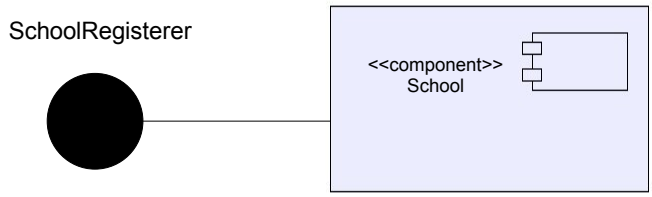
Black Box View

- The required interface is represented using the socket icon.
- The name of the required interface will be written near the connector symbol.



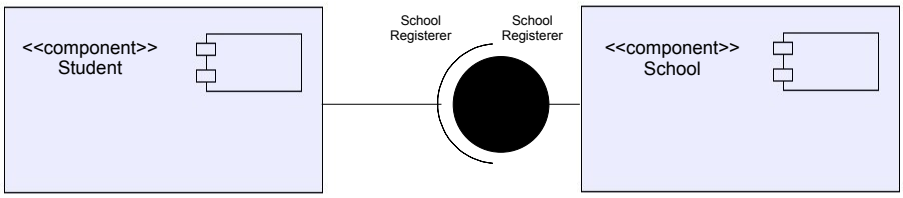
Black Box View

- The provided interface is illustrated using the ball icon.
- The name of the provided interface will be written near the connector symbol.



Black Box View

- Writing two components that match their required & provided interfaces is done by simply connecting them via their matching provided and required interfaces.



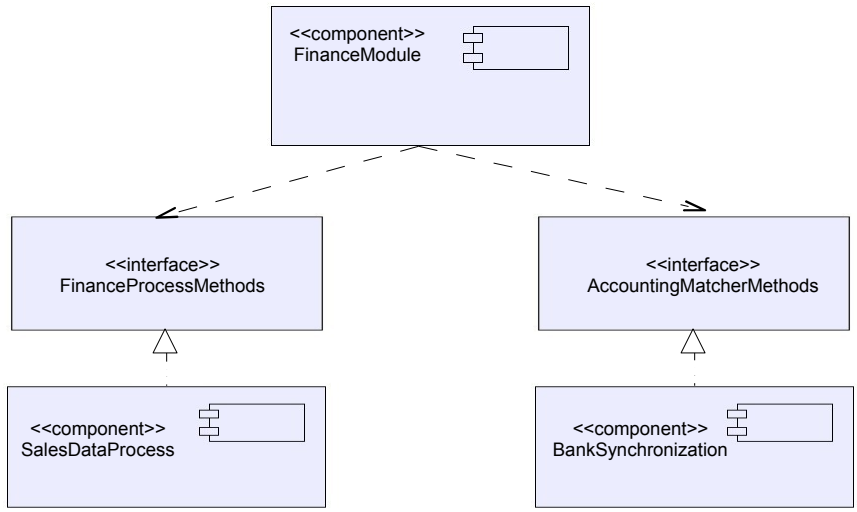
07/25/10 © 2008 Haim Michael. All Rights Reserved. 9

Drawing the component diagram including the assemblies connectors provides more info.

Black Box View

- The interface's notation is an empty rectangle in which we write the interface name and “<<interface>>” above it.
- A dependency arrow will be drawn from the component to the required interface.

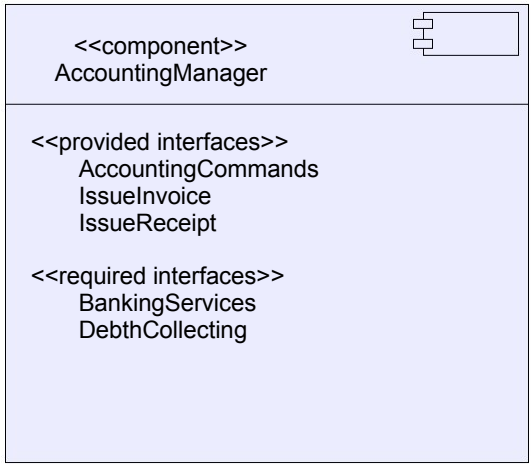
Black Box View



Black Box View

- More detailed info can be provided by drawing the component with compartments.
- The compartment on top includes the name of the class with the `<<component>>` stereotype above. The one on the bottom includes two lists. One is of the required interfaces and the other is of the provided ones. The `<<required>>` and `<<provided>>` stereotypes are used accordingly.

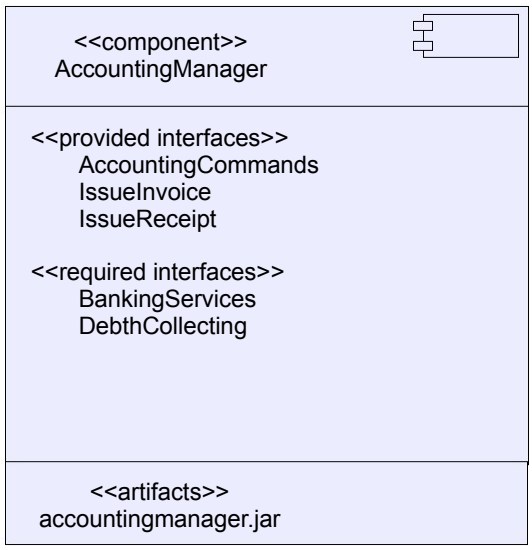
Black Box View



Black Box View

- A more detailed compartment diagram can include a third compartment stereotyped with `<<artifacts>>`, that lists the components' artifact\s.

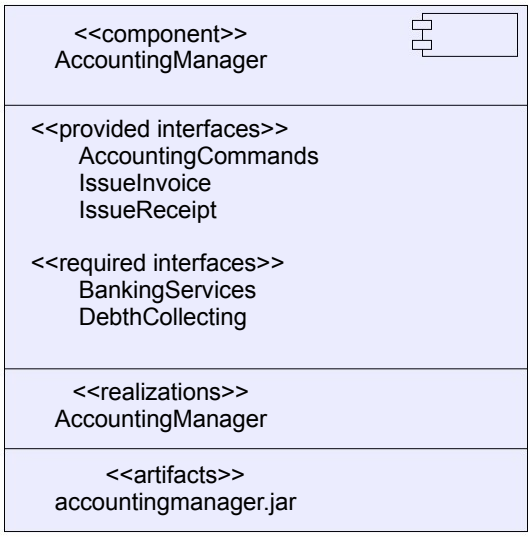
Black Box View



White Box View

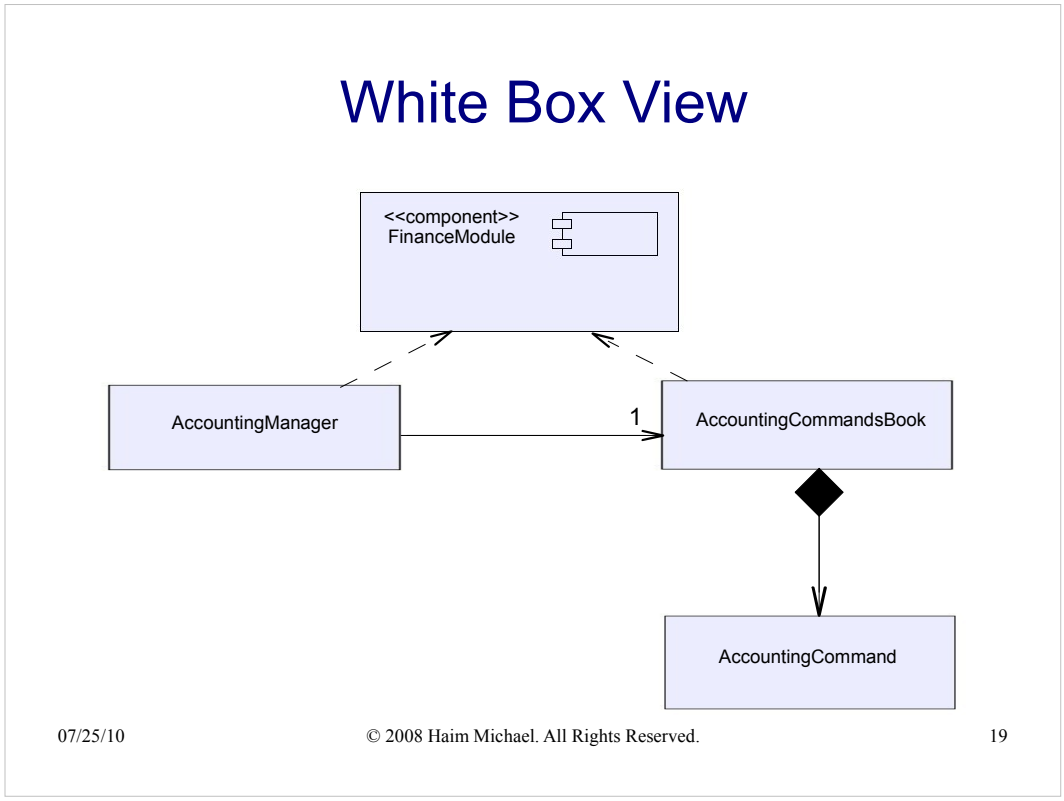
- The white box view presents the way a component realizes the interfaces it provides.
- The realization compartment can be added to the compartment diagram and lists the exact classifiers that realize the provided interfaces. This new compartment should be labeled with the `<<realizations>>` stereotype.

White Box View



White Box View

- To provide more information about the component structure we can create a diagram that shows the component, the classes it includes and the relationships between them.
- Between the classes and the component we will draw dependency relationships.



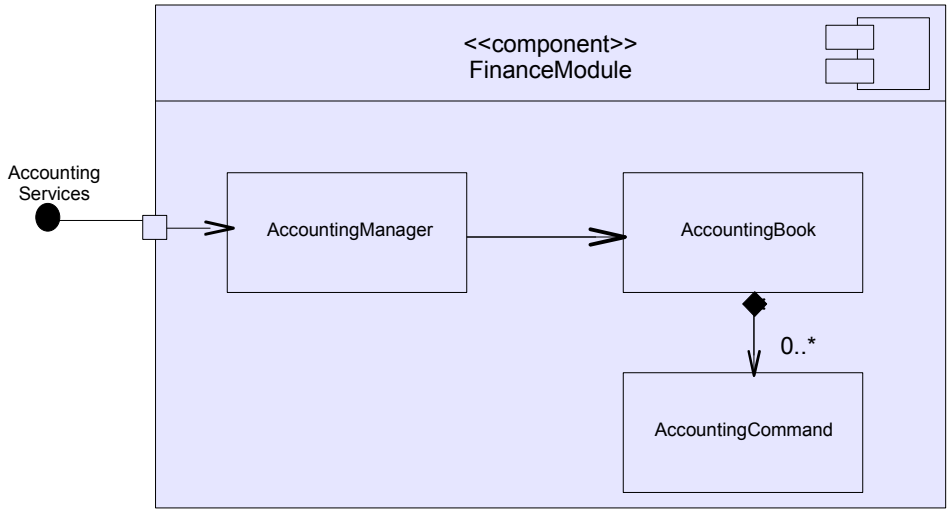
The dashed open arrow line represents dependency.

More details can be provided if the internals of each class will be detailed listing all members.

White Box View

- An alternative can be drawing the classifiers within the component frame.
- A port, depicted as a small rectangle on one of the component's sides represents a required / provided (or both) functionality.
- Each one of the assembly connectors (both of type ball.. and of type socket...) are drawn as coming out of a port.
- If we draw the classifiers within the component frame then we can connect each port with its responsible class.

White Box View



Component Stereotypes

- The following is a summary list of the stereotypes you can apply to components:

  Realization

  When realize spec provided by another component. Another component that provides the specification.

  Process

  When capable of fulfilling functional requests. Transaction based. Holds data about the state.

  Service

  When fulfilling functional requests by others. Stateless based.

Component Stereotypes

Entity

When representing a business entity. Component for data storage.

Subsystem

A large component, part of a big system. Usually larger than a simple component.

Specification

Component with providing and requiring interfaces and without any implementation.

A component stereotyped with <<entity>> is usually a persisted component and usually doesn't have any functionality. Entity Bean (EJB) is a good sample for entity component.

A good example for process component is the session bean (EJB).

