▸ ▸ ▸ **Module 7**
Other UML Diagrams

IBM Software Group

IBM®

IBM Software Group

Essentials of Visual Modeling with UML 2.0
Module 7: Other UML Diagrams

Rational. software
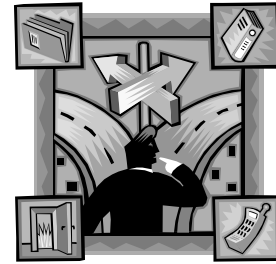
Topics

Objectives

Objectives

- ◆ Demonstrate how to read and interpret a:
  - State machine diagram
  - Component diagram
  - Deployment diagram

2

IBM

Where Are We?

Where Are We?

☆◆ State machine diagrams
   ◆ Component diagrams
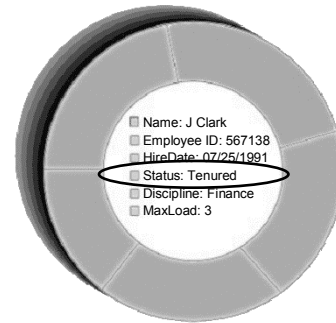   ◆ Deployment diagrams

3

IBM

Review: An Object Has State

The **state** of an object is one of the possible conditions that an object may exist in, and it normally changes over time.

The state of an object is usually implemented by a set of properties called attributes, along with the values of the properties and the links the object may have with other objects.

State is not defined by a "state" attribute or set of attributes. Instead, state is defined by the total of an object's attributes and links. For example, if Professor Clark's status changed from Tenured to Retired, the state of the Professor Clark object would change.

Example: Professor

Example: Professor

- ◆ There are a sequence of events before an instructor becomes a University professor.
  - ▪ Assistant professor (achieves tenure by producing a number of quality publications)
  - ▪ Tenure/Associate professor
  - ▪ Professor (based on seniority)
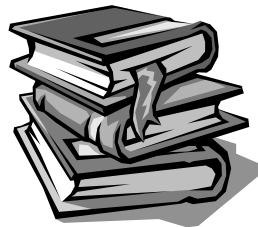
5

IBM

The example of someone becoming a college professor serves us well in explaining how a state machine works. When you are hired for a teaching position at the university, you begin as an assistant professor. Based on your successful publication of numerous quality papers, you receive tenure and the title of assistant professor. There is not an automatic time line for becoming an associate professor, it is based on producing and publishing quality papers. However after you have tenure, seniority makes you available for the next available professorship.

What Are State Machine Diagrams?

What Are State Machine Diagrams?

- ◆ A state machine diagram models dynamic behavior.
- ◆ It specifies the sequence of states in which an object can exist:
  - ▪ The events and conditions that cause the object to reach those states
  - ▪ The actions that take place when those states are reached

Assistant Professor → Tenured

6

IBM

A state machine diagram is typically used to model the discrete stages of an object's lifetime. They show the sequences of state that an object goes through, the events that cause a transition from one state to another, and the actions that result from the state change. State machine diagrams are closely related to activity diagrams.

Each state represents a named condition during the life of an object in which it satisfies some condition or waits for some event. A state machine diagram typically contains one start and multiple end states. Transitions connect the various states on the diagram. Like activity diagrams, decisions, and synchronizations may also appear on state machine diagrams.

State machines are used to model the dynamic behavior of a model element, and more specifically, the event-driven aspects of the system's behavior. State machines are specifically used to define state-dependent behavior, or behavior that varies depending on which state the model element is in. Model elements whose behavior does not vary with the state, do not require state machines to describe their behavior. These elements are typically passive classes whose primary responsibility is to manage data.

Special States

Special States

- ◆ The initial state is the state entered when an object is created.
  - ▪ An initial state is mandatory.
  - ▪ Only one initial state is permitted.
  - ▪ The initial state is represented as a solid circle.
- ◆ A final state indicates the end of life for an object.
  - ▪ A final state is optional.
  - ▪ A final state is indicated by a bull's eye.
  - ▪ More than one final state may exist.

7

IBM

The **initial state** indicates the default starting place for the state machine or sub-state. An initial state is represented as a filled black circle.

The **final state** indicates the completion of the execution of the state machine or the enclosing state. A final state is represented as a filled black circle surrounded by an unfilled circle.

Initial and final states are actually pseudo-states. Neither may have the usual parts of a normal state, except for a name.

What Are Events?

- ◆ An event is the specification of a significant occurrence that has a location in time and space.
  - An event is an occurrence of a stimulus that can trigger a state transition.
  - Example:
    - Successful publication of numerous papers

| Assistant Professor | → Event → | Tenured |
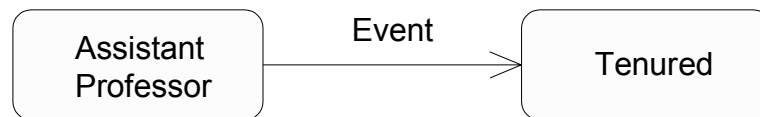
8

IBM

In the context of the state machine, an **event** can be defined as an occurrence of a stimulus that can trigger a state transition. Events may include signals, calls, the passing of time, or a change in state.

A signal or call may have parameters whose values are available to the transition, including expressions for the guard conditions and action.

It is also possible to have a triggerless transition, represented by a transition with no event trigger. These transitions, also called completion transitions, are triggered implicitly when their source state has completed its action. They are implicitly triggered on the completion of any internal 'do activity' in the state.

What Are Transitions?

- ◆ A transition is a change from an originating state to a successor state as a result of some stimulus.
  - ▪ The successor state could possibly be the originating state.
- ◆ A transition may take place in response to an event.
- ◆ Transitions can be labeled with event names.

Assistant Professor → maxPapers → Tenured

Transition   Event Name
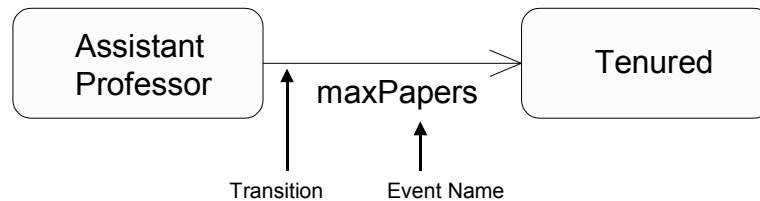
9

IBM

A **Transition** can be defined as:

A relationship between two states indicating that an object in the first state performs certain actions and enters a second state when a specified event occurs and specified conditions are satisfied. On such a change of state, the transition is said to "fire." Until the transition fires, the object is said to be in the "source" state. After it fires, it is said to be in the "target" state.

- You can show one or more state transitions from a state as long as each transition is unique. Transitions originating from a state can not have the same event unless there are conditions on the event.
- The icon for a state transition is a line with an arrowhead pointing toward the destination state.
- Label each state transition with the name of at least one event that causes the state transition. You do not have to use unique labels for state transitions because the same event can cause a transition to many different states but it is recommended that they have unique labels.
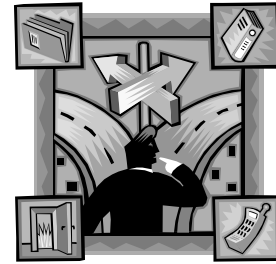
Example: State Machine
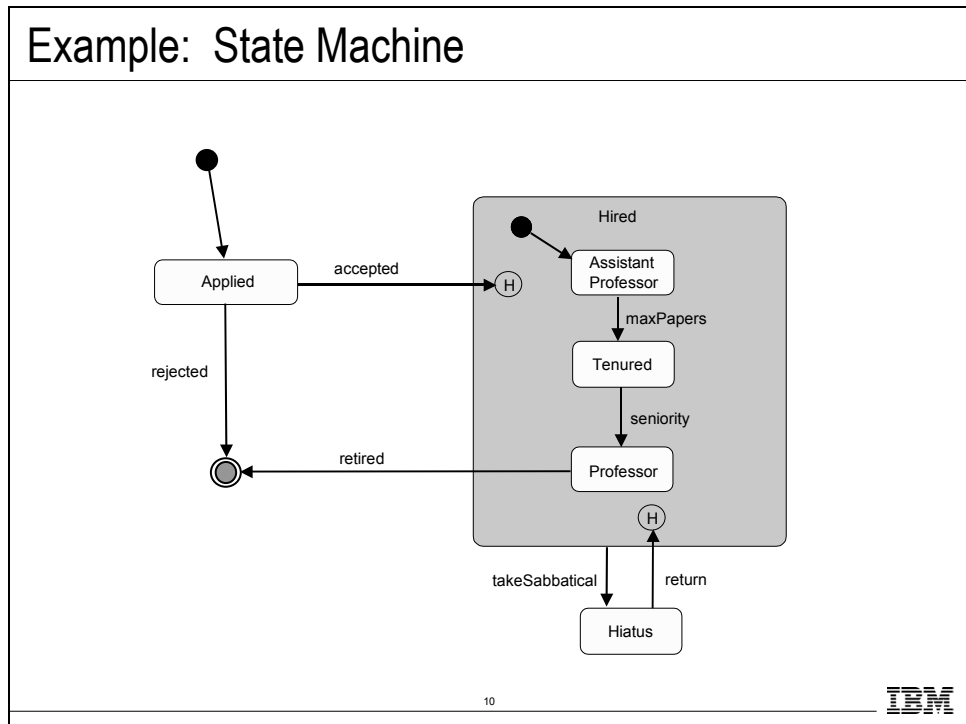
Where Are We?

Where Are We?

- ◆ State machine diagrams
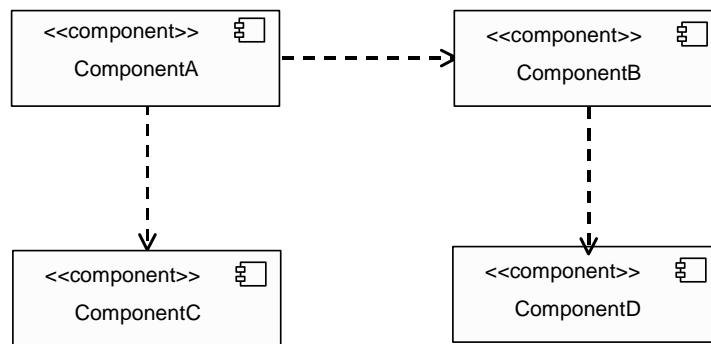- ☆ ◆ Component diagrams
- ◆ Deployment diagrams

11

IBM

What Is a Component Diagram?

- ◆ A diagram that shows the organizations and dependencies among components

<<component>> ComponentA		<<component>> ComponentB

<<component>> ComponentC		<<component>> ComponentD

12

IBM

Component diagrams commonly contain:

- Components
- Interfaces
- Realization, implementation and usage dependencies
- Classes
- Artifacts
- Ports

You use component diagrams to model a static view of a system. There is no significant distinction between a component diagram and a general class diagram. Both are examples of structure diagrams which depict elements in a specification that are irrespective of time.

Interfaces and ports are not covered in any detail in this course. Interfaces allow you to encapsulate, or hide, the implementation of a supplier. Ports are a structural notation used for interfaces.

What Is a Component?

What Is a Component?

- ◆ A modular part of a system that hides its implementation behind a set of external interfaces.
  - ▪ Part of a logical or physical system
- ◆ It conforms to and provides the physical realization of a set of interfaces.
- ◆ It specifies the physical dependency to interfaces it requires.

<<component>> ComponentName

Provided Interface Name

Required Interface Name

13

IBM
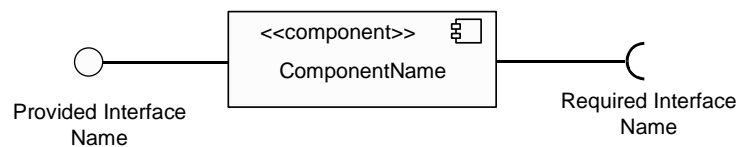
Graphically, a component is shown as a rectangle with the keyword «component». Optionally, in the right hand corner, a component icon can be displayed. This icon is a small rectangle with two smaller rectangles protruding from its left hand side. The name of the component (as a type) is placed inside the outer rectangle.

A **component** is a modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. A component defines its behavior in terms of provided and required interfaces.

A component represents a modular piece of a logical or physical system. Components satisfying the same interfaces may be freely substituted.

Where Are We?

Where Are We?

- ◆ State machine diagrams
- ◆ Component diagrams
- ☆ ◆ Deployment diagrams

14

IBM

What Is a Deployment Diagram?

- ◆ The deployment diagram shows:
  - ▪ Configuration of processing nodes at run-time
  - ▪ Communication links between these nodes
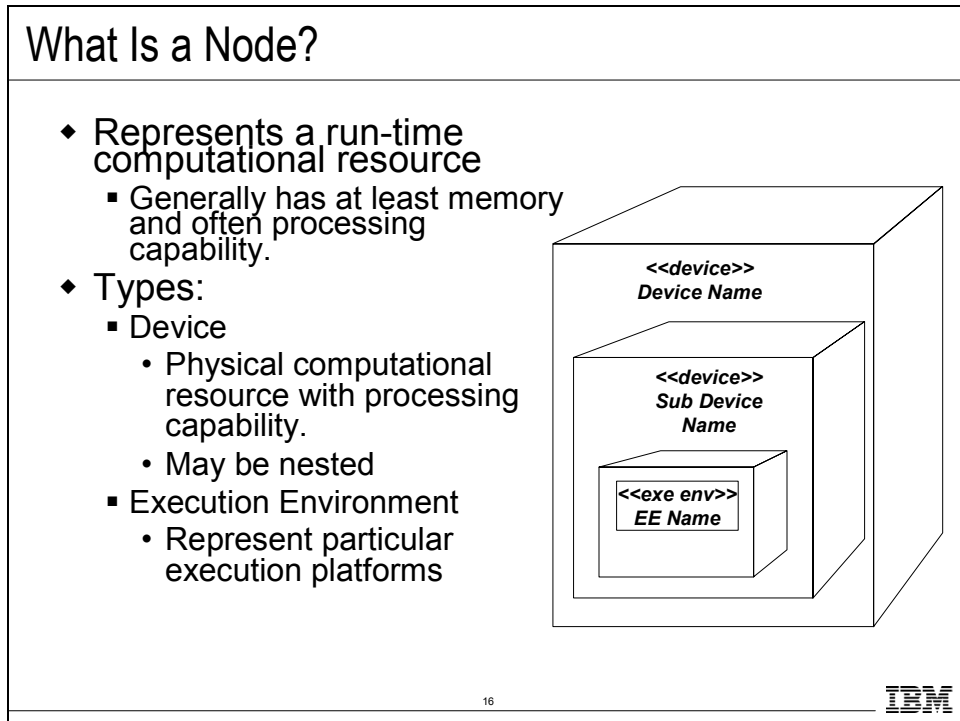 - ▪ Deployed artifacts that reside on them

15

IBM

The purpose of the deployment diagram is to capture the configuration of processing elements, and the connections between processing elements in the system.

The deployment diagram consists of one or more:

- **Nodes** (processing elements with at least one processor and memory)
- **Connectors** between nodes and/or devices

The deployment diagram also maps **processes** on to these processing elements, allowing the distribution of behavior across nodes to be represented.

What Is a Node?

What Is a Node?

- ◆ Represents a run-time computational resource
  - ▪ Generally has at least memory and often processing capability.
- ◆ Types:
  - ▪ Device
    - • Physical computational resource with processing capability.
    - • May be nested
  - ▪ Execution Environment
    - • Represent particular execution platforms

<<device>>
Device Name

<<device>>
Sub Device Name

<<exe env>>
EE Name

16

IBM

The essential elements of a deployment diagram are nodes and their connectors. A *node* is a run-time physical object that represents a computational resource, generally having at least memory and often processing capability as well.

Execution Environment and *device* are types of *node* but the UML2 distinction between them is rather vague.

- **Devices** may have artifacts deployed for execution software that controls the functionality of the device itself (physical computational resource with processing capability). Typical examples include <<print server>>, <<application server>>, <<client workstation>>, <<mobile device>>, <<embedded device>>, <<processor>>, etc. Devices may be complex and contain other devices.
- **Execution Environments** represent particular execution platforms, such as an operating system (<<Win2K>>, <<VxWorks>>, etc.), a workstation engine (<<workstation engine>>), a database management system <<DB2>>), etc.

What Is a Connector?

What Is a Connector?

- ◆ A connector represents a:
  - ▪ Communication mechanism
   - • Physical medium
    - • Software protocol
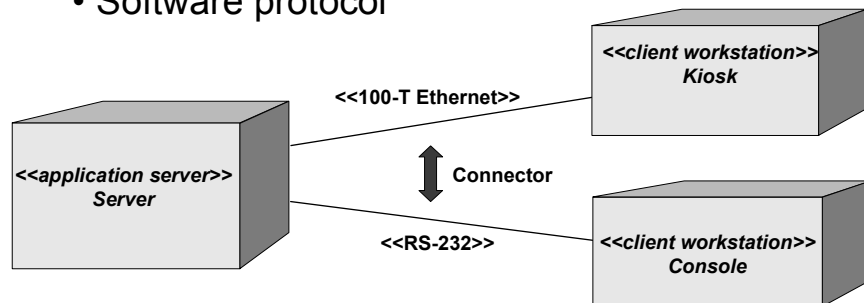
Connectors can be drawn between nodes. These connectors represent
communication mechanisms and can be described by physical mediums (for
example, Ethernet, fiber optic cable) or software protocol (for example, TCP/IP, RS-
232). A stereotype may be used to specify the type of connector.

Example: Deployment Diagram

Example: Deployment Diagram
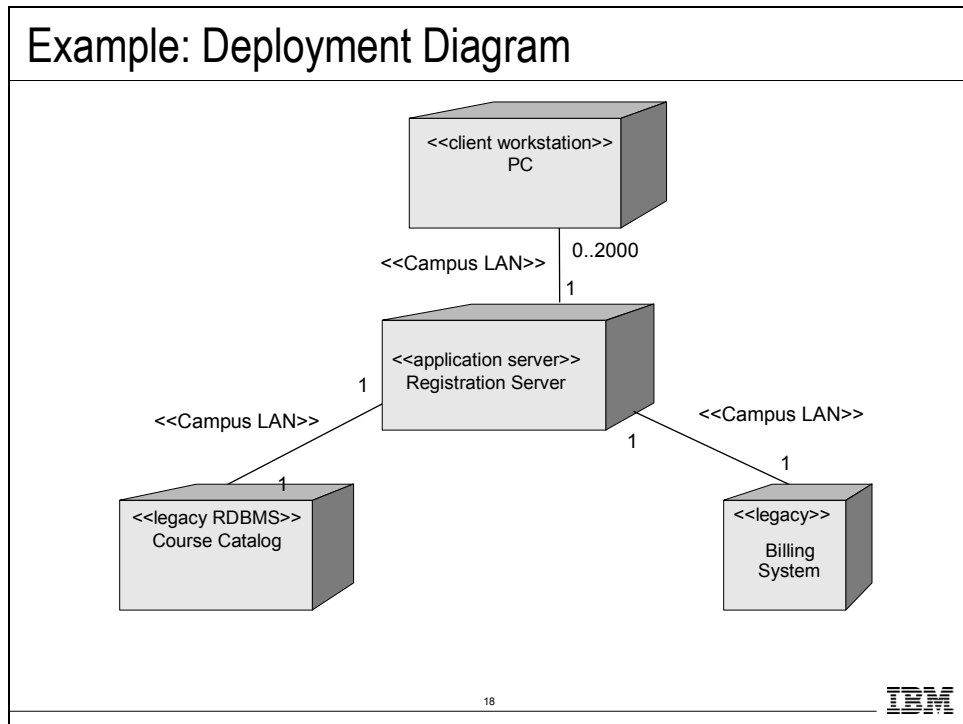
Deployment diagrams show the configuration of run-time processing nodes. A deployment diagram may be at the class or instance level.

Deployment diagrams allow you to capture the topology of the system nodes. This allows you to visually see potential bottlenecks.

A deployment diagram contains nodes connected by associations. The associations indicate a communication path between the nodes.

The above diagram illustrates the deployment view for the Course Registration System.

There are PC client workstations on the campus network.

The main business processing hardware is the Registration Server. It talks to the two machines that host the legacy systems. All nodes are on the campus network.

Example: Deployment Diagram with Processes
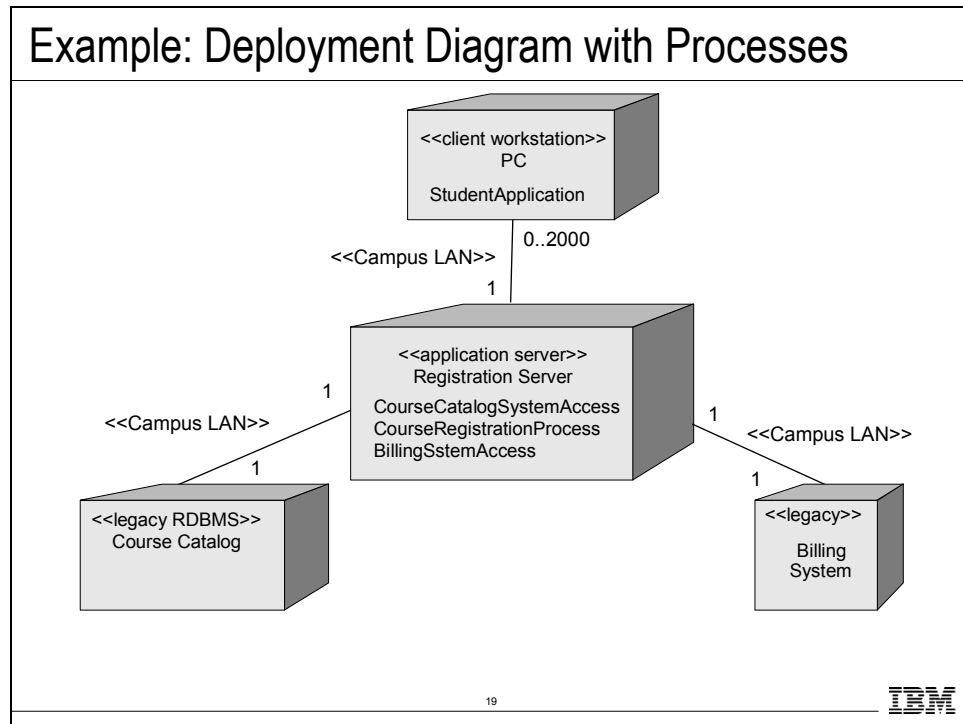
The above diagram illustrates a refined deployment view for the Course Registration System.

There are PC client workstations on the campus network. Each running a StudentApplication process.

The main business processing hardware is the Registration Server running the CourseCatlogSystemAccess, CourseRegistrationProcess, and BillingSystemAccess processes. The Registration Server talks to the two machines that host the legacy systems. All nodes are on the campus network.

Review

Review

- ◆ Define state. How do you determine the classes with significant state?
- ◆ What is a state machine diagram? Describe the different parts of the diagram.
- ◆ What is a component diagram?
- ◆ What is the purpose of a deployment diagram?

20

IBM

© Copyright IBM Corp. 2004