8. Behavioural State Machines and Other UML diagrams

IT 3106– Object Oriented Analysis and Design

Level II - Semester 3

Overview

In this section students will learn to

- draw state transition diagrams that provide additional analysis techniques for classes with significant dynamic behavior.
- timing, interaction Overview, Composite Structure, Component, Deployment and Profile Diagrams.

Intended Learning Outcomes

At the end of this lesson students will be able to

- understand the processes used to create behavioral state machines, and CRUDE matrices,
- understand the relationship between the behavioral models and the structural and functional models,
- understand the Timing, interaction Overview, Composite Structure, Component, Deployment and Profile Diagrams.

List of sub topics

8 Behavioural State Machines and Other UML diagrams (5 hours)

- 8.1 Behavioural State Machines [Ref 1: Pg. 221-229]
  - 8.1.1 States, Events, Transitions, Actions, and Activities
  - 8.1.2 Elements of a Behavioral State Machine
  - 8.1.3 Guidelines for Creating Behavioral State Machines
  - 8.1.4 Creating a Behavioral State Machine
- 8.2 CRUDE Analysis [Ref 1: Pg. 229-232]
- 8.3 Verifying and Validating the Behavioral Model [Ref 1: Pg. 233-235]
- 8.4 Other UML Diagrams [Ref 1: Pg. 34-36, Ref 4]
- Timing, interaction Overview, Composite Structure, Component, Deployment and Profile Diagrams

Ref 1: Alan Dennis, Barbara Haley, David Tegarden, Systems analysis design, An Object-Oriented Approach with UML: an Object-Oriented approach, 5th edition, John Wiley & Sons, 2015, ISBN 978-1-118-80467-4

8.1 Behavioural State Machines

- Some of the classes in the class diagrams represent a set of objects that are quite dynamic and they pass through a variety of states over the course of their existence.
  - Example: a patient can change over time from being new to current to former based on his or her status with the doctor's office.
- A behavioral state machine is a dynamic model that shows the different states through which a single object passes during its life in response to events, along with its responses and actions.

8.1 Behavioural State Machines

More Examples:

- After an appropriate amount of time, a washing machine changes its state from Washing to Rinsing.
- Hotel room changes its state to available, reserved and occupied.

- UML State diagram captures these kinds of changes.
- UML *State Transition Diagrams* shows:
  - Life history showing the different states of a given object.
  - The events or messages that cause a transition from one state to another.
  - The actions that results from a state change.
- *State Diagrams* are created only for classes with significant dynamic behaviour.

  eg. ***Hotel Room*** in a Hotel Reservation System

8.1 Behavioural State Machines

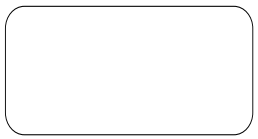
Modeling Dynamic Behaviour

- Interaction diagrams can be studied to determine the dynamic objects.
  - Objects receive and send many messages.
- If you have an attribute called *status*.
  - This can be a good indicator of various states.

States

Condition or Situation during the life of an object

- eg. HotelRoom object can be in one of the following states.
  - Occupied, Available, Reserved
- eg. Course object (in a course registration system) can be in one of the following states.
  - Initialization,Open, Close,Cancel



UML Notation for a State

State Transitions

- A State Transition represents a change from an originating state to a successor state.
- An action can accompany a state transition.
- A State Transition is represented by an arrow that points from the originating state to the successor state.

UML Notation for State Transition

Special States

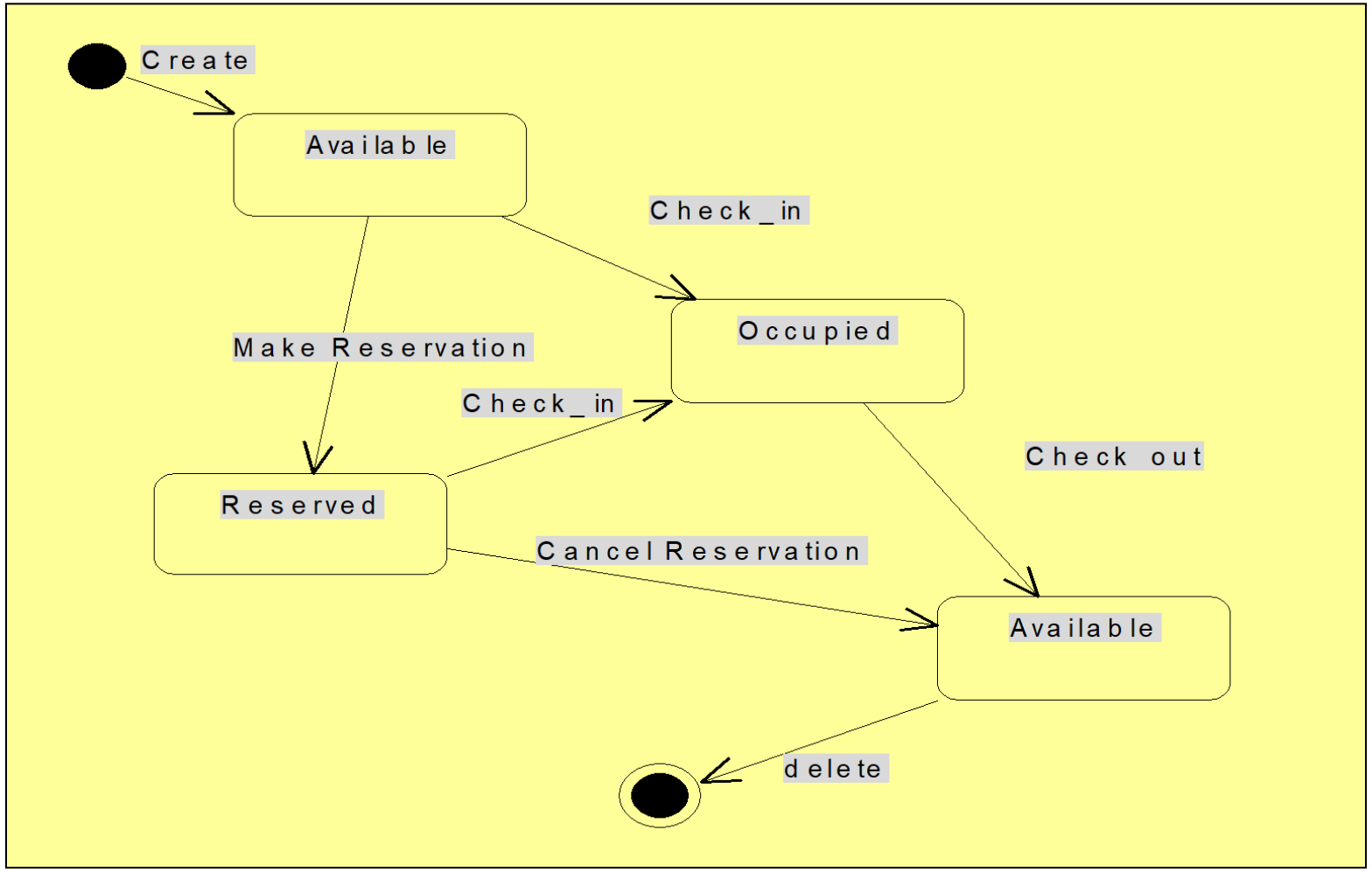
- There are two special states that are added to the state transition diagram.
- **Start** state – Each diagram must have one and only one start state.
- **Stop** state – An object can have multiple stop states.

Start State Stop State

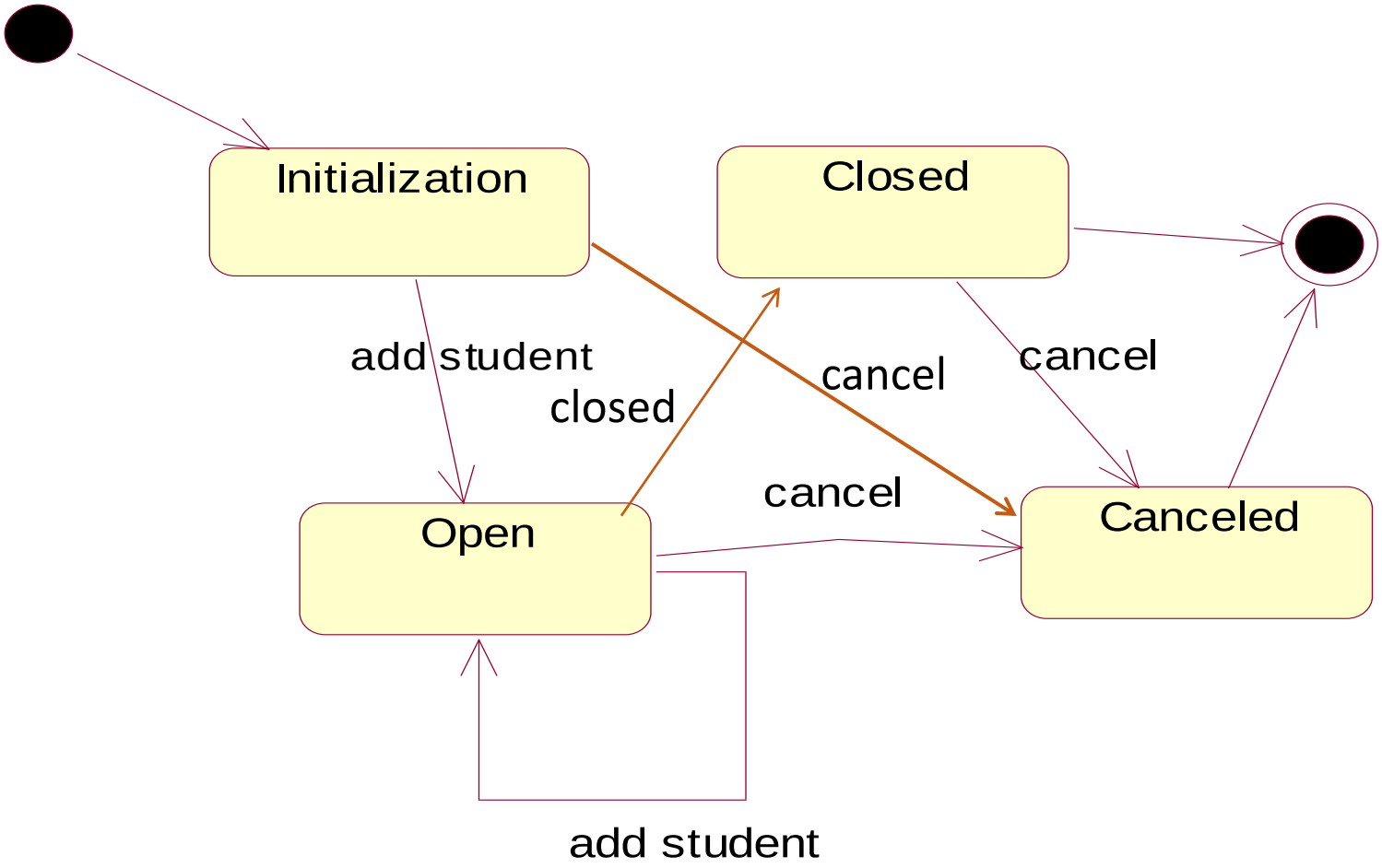
Guidelines for creating State Diagrams

- Examine your class diagram to identify which classes undergo a complex series of state changes (having significant dynamic behavior).
- The second step is to identify the various states that an object will have over its lifetime. This includes establishing the boundaries of the existence of an object by identifying the initial and final states of an object (start and stop state).
- The third step is to determine the sequence of the states that an object will pass through during its lifetime.
- The fourth step is to identify the transitions between the states of the objects and to add the events, actions, and guard conditions associated with the transitions.
- The fifth step is to validate the behavioral state machine by making sure that each state is reachable and that it is possible to leave all states except for final states.

State Diagram –Hotel Room Class



State Diagram– Course Class



State Transition Details

- A State Transition may have the following associated with:
  - an **action** and/or

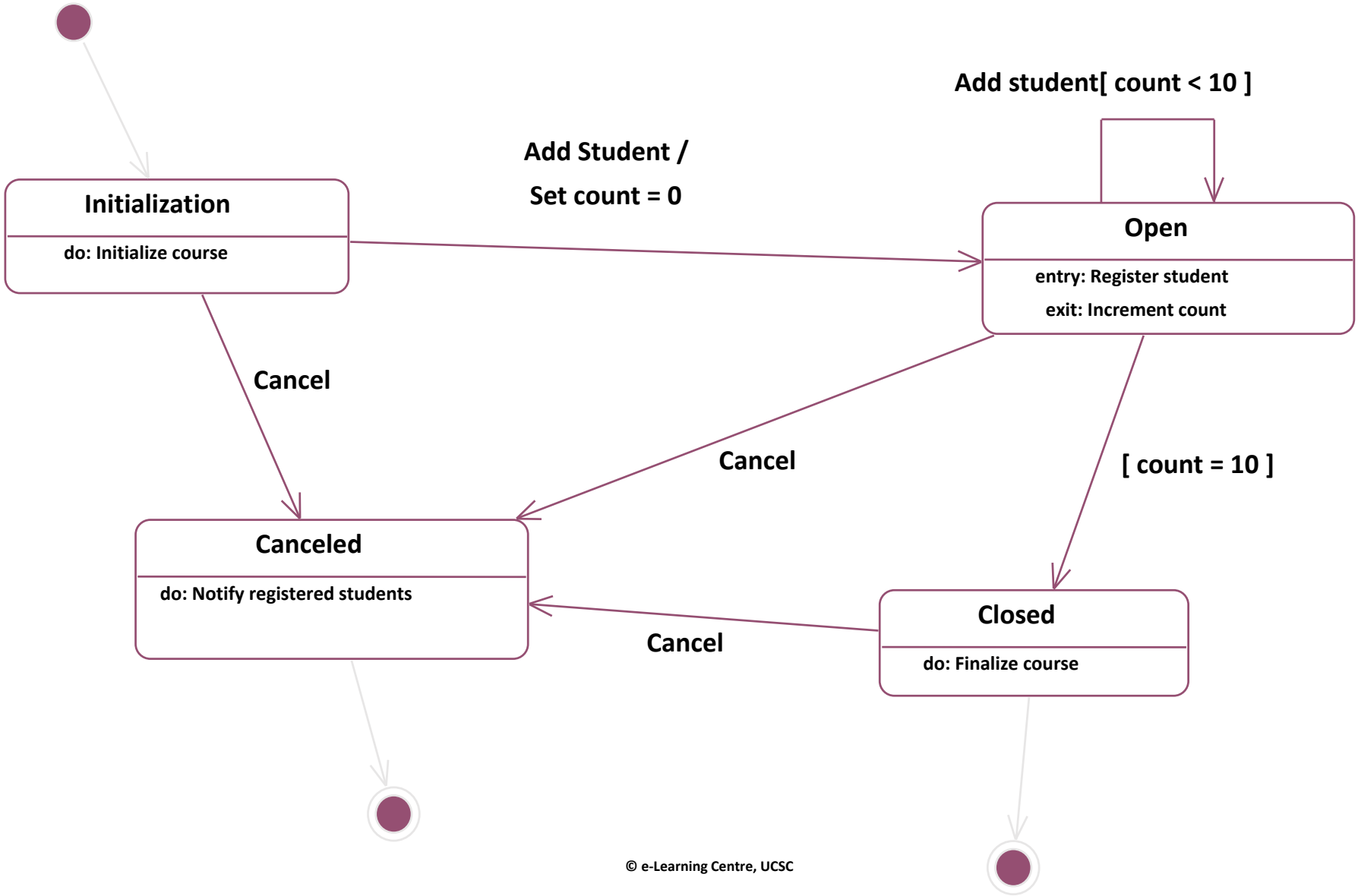
    (behaviour that occurs when the state transition occurs.)
  - a **guard condition**

    (allows state transition only if it is true.)
- A State Transition may also trigger an event

  A message that is sent to another object in the system.

State Diagram

Course Offering with State Details



State Details

- ***Activity***:
  - Behaviour that an object carries out while it is in a particular state.
  - An activity is shown inside the state itself, preceded by the word *do* and a colon.
- **Entry Action**:
  - Behaviour that occurs while the object is transitioning into the state.
  - Shown inside the state, preceded by the word *entry* and colon.

State Details cont…

- **Exit Action:**
  - Occurs as part of the transition out of a state.
  - Shown inside the state, preceded by the word *exit* and colon.
- The behaviour in an activity, entry action, or exit action can include sending an event to some other object.

State Details con…

- In this case, the activity, entry action, or exit action is preceded by a ^

Do:^Target.Event(Arguments)

Target - object receiving the event
Event - message being sent
Arguments – parameters of the message being sent

Example:
Do:^CourseRoster.Create

UML 2.0 State Diagrams

- UML 2.0 has added some new state relevant symbols called **connection points.**
- They represent points of entry into a state or exists out of a state.
- Let us look at the different states of a book in a library.

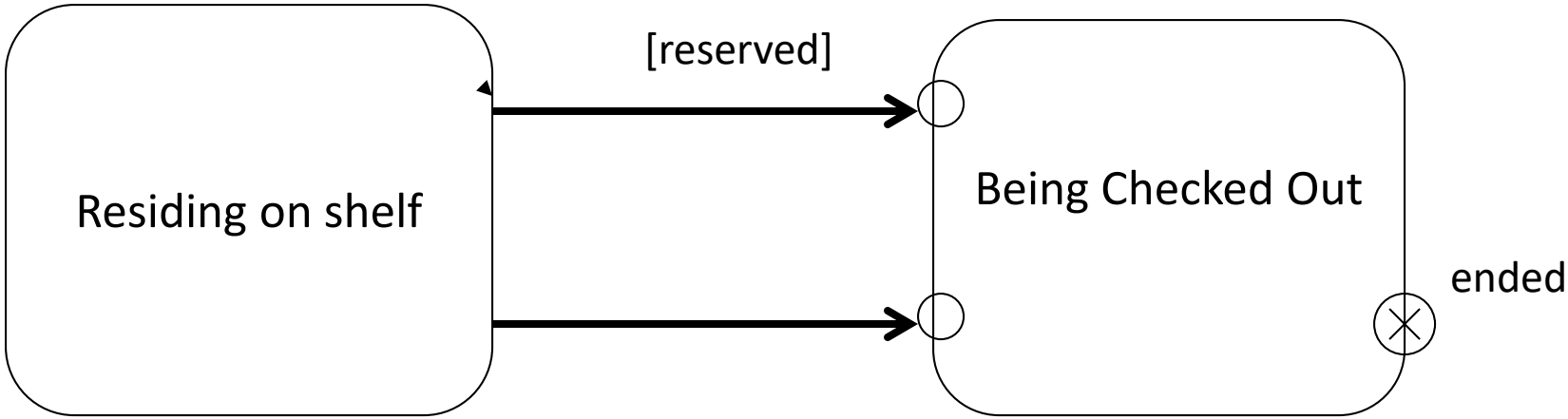
UML 2.0 State Diagrams

- At first the book is residing on a shelf.
- If a borrower has called in to reserve the book, a librarian retrieves the book and brings it into the state of *“Being checked out”.*
- If a borrower comes to the library, browses through the shelves, selects the book, and decides to borrow it.
- Again it enters into the state of *“Being checked out”,* but in a different way.

UML 2.0 State Diagrams

- You can think of each way of getting to the Being-checked-out state as going through a separate **entry point**
- Suppose the borrower is trying to borrow more than the allotted limit or has number of unpaid fines.
- If that is the case the book abruptly exits via an **exit point,** from *"Being-checked-out" state*

Entry points and exit point in a UML state diagram



Why are State diagrams important?

The state diagram:

- models the changes that just one object goes through.
- help analysts, designers, and developers understand the behavior of the objects in a system.

A class diagram and an object diagram:

- shows only static aspects of a system.
- do not show the dynamic details of the behaviors.

Why are State diagrams important?

- Developers, in particular, have to know
  - how objects are supposed to behave because they have to implement these behaviors in software.
  - Developers have to make that object do something.
- State diagrams ensure that they won't have to guess about what the object is supposed to do.

8.2 CRUDE Analysis

CRUDE analysis

- A useful technique to identify how the underlying objects in the problem domain work together to collaborate in support of the use cases.
- Uses a *CRUDE matrix*, in which each interaction among objects is labeled with a letter for the type of interaction:

  C for Create, R for Read or Reference, U for Update, D for Delete, and E for Execute.

- Also can be used to identify complex objects.
  - These objects are candidates for state modeling with a behavioral state machine.

8.2 CRUDE Analysis

	Receptionist	PatientList	Patient	UnpaidBills	Appointments	Appointment
Receptionist		RU	CRUD	R	RU	CRUD
PatientList			R			
Patient						
UnpaidBills						
Appointments						R
Appointment						

Eg. CRUDE Matrix for the "*Make Patient Appointment*" Use Case in a Medical care system

- CRUDE analysis also can be used to identify complex objects.
- The more (C)reate, (U)pdate, or (D)elete entries in the column associated with a class, the more likely the instances of the class have a complex life cycle.
- As such, these objects are candidates for state modeling with a behavioral state machine.

Verifying and Validating the Behavioral Model

The sequence and communication diagrams

- models the interaction among instances of classes that work together to support the business processes included in a system, whereas,

The behavioral state machine

- describes the state changes through which an object traverses during its lifetime, and

The CRUDE matrix

- represents a system-level overview of the interactions among the objects in the system.
- combines walkthroughs to more completely verify and validate the behavioral models.

8.3 Verifying and Validating the Behavioral Model

- First, every actor and object included on a sequence diagram must be included as an actor and an object on a communication diagram, and vice versa.
- Every message that is included on a sequence diagram must appear as a message on an association in the corresponding communication diagram, and vice versa.
- The sequence number included as part of a message label in a communications diagram implies the sequential order in which the message will be sent. Therefore, it must correspond to the top-down ordering of the messages being sent on the sequence diagram.

8.3 Verifying and Validating the Behavioral Model cont..

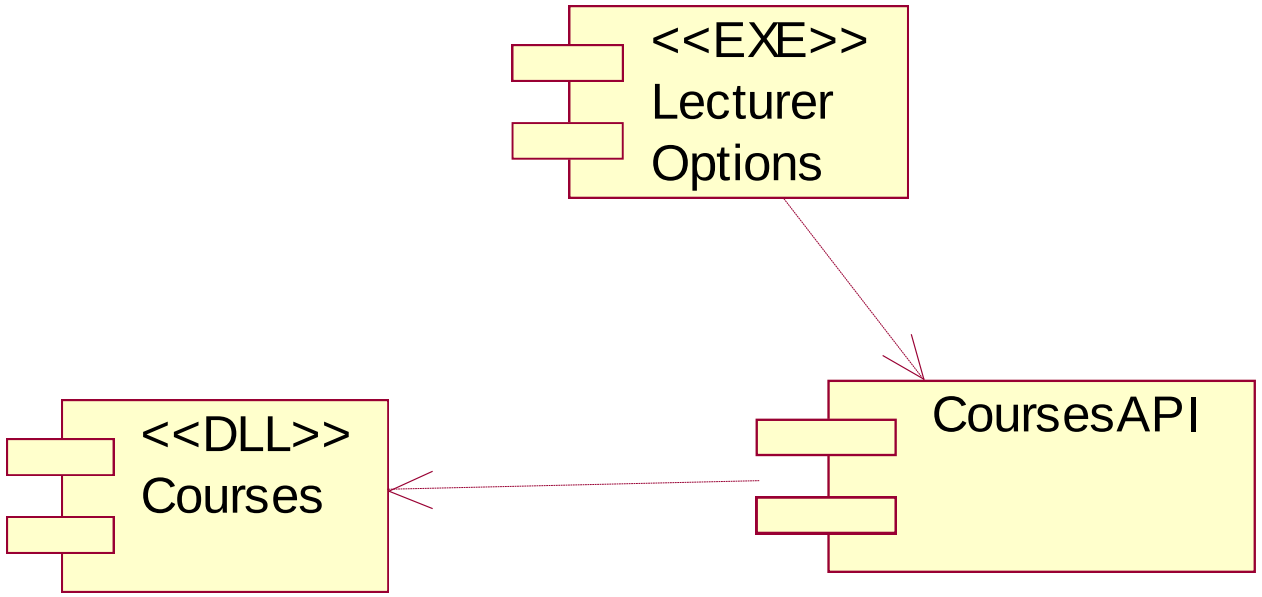
- All transitions contained in a behavior state machine must be associated with a message being sent on a sequence and communication diagram, and it must be classified as a (C)reate, (U)pdate, or (D)elete message in a CRUDE matrix.
- All entries in a CRUDE matrix imply a message being sent from an actor or object to another actor or object. If the entry is a (C)reate, (U)pdate, or (D)elete, then there must be an associated transition in a behavioral state machine that represents the instances of the receiving class.

8.4 Other UML Diagrams

Component Diagrams

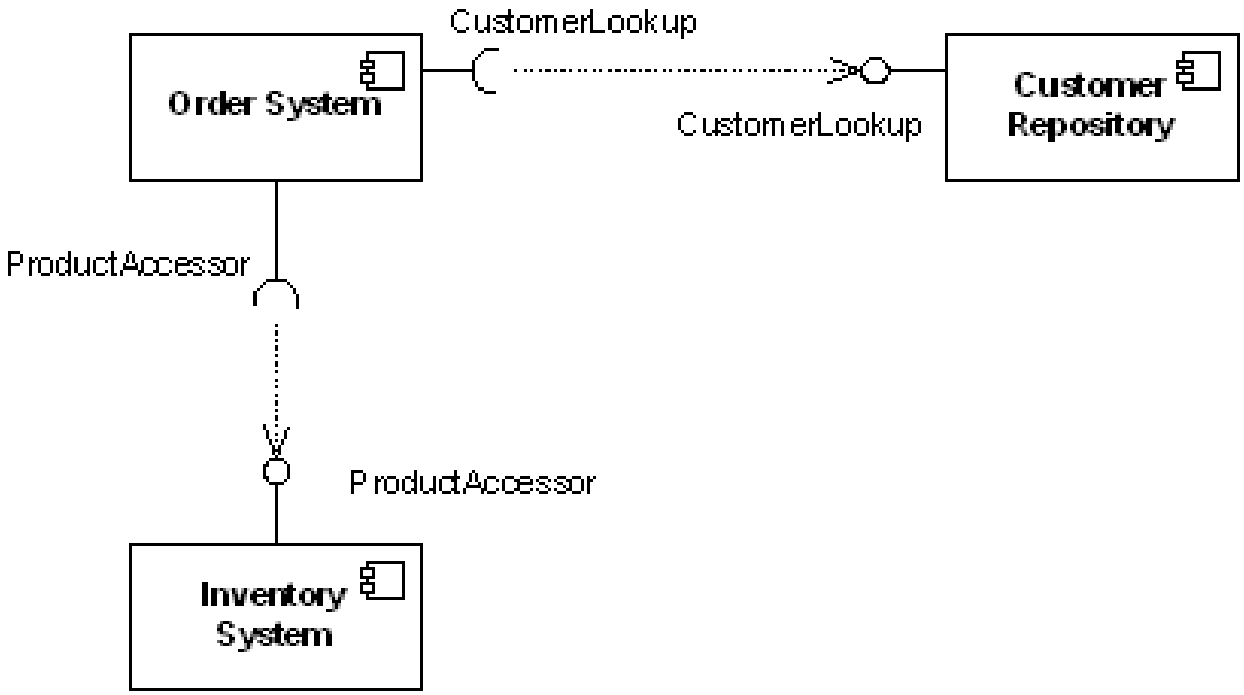
- Component diagrams are used to
  - visualize the physical components in a system. These components are libraries, packages, files, etc.
  - depict how components are wired together to form larger components or software systems.
  - illustrate the structure of arbitrarily complex systems.
- Component diagrams can also be described as a static implementation view of a system.
- Component diagrams contain mainly *components*, *interfaces* and *relationships*.
  - Components are related via dependency relationships.

Component diagram



(UML 1.X notation)

Component diagram



A Component diagram (UML 2.0) that shows how the Order System component depends on other components

8.4 Other UML Diagrams cont.

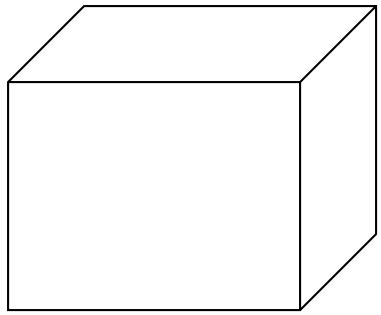
Deployment Diagram

- The deployment diagram shows the physical architecture of a computer-based system.
- It can show
  - the computers and devices,
  - their connections with one another
  - the software that sits on each machine
- It maps the software architecture to the hardware architecture.

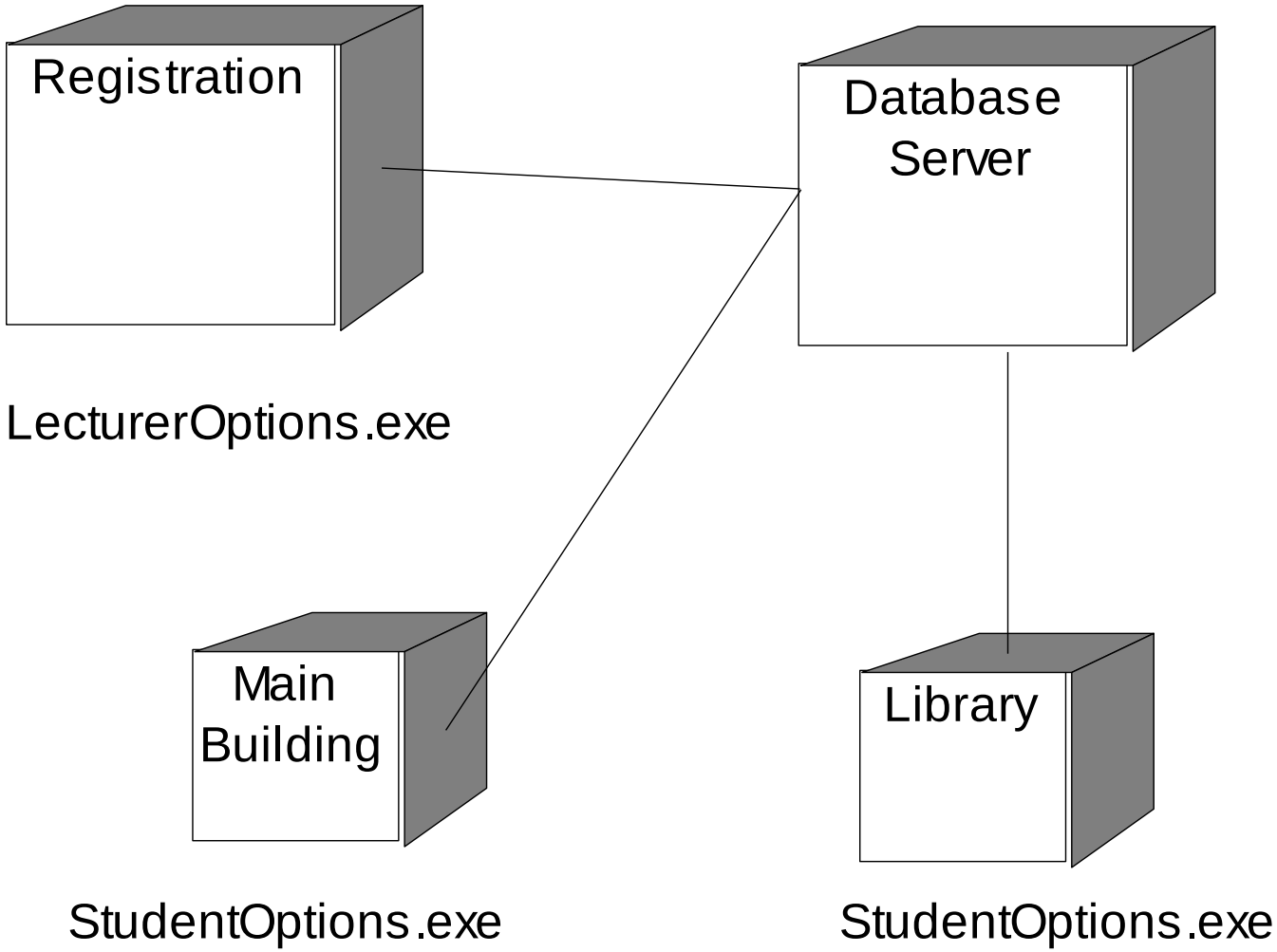
Deployment Diagram

- Deployment diagrams are used only for distributed systems.
- The main hardware item is a *node*, a generic name for any kind of computing resource.

UML node icon



Deployment Diagram

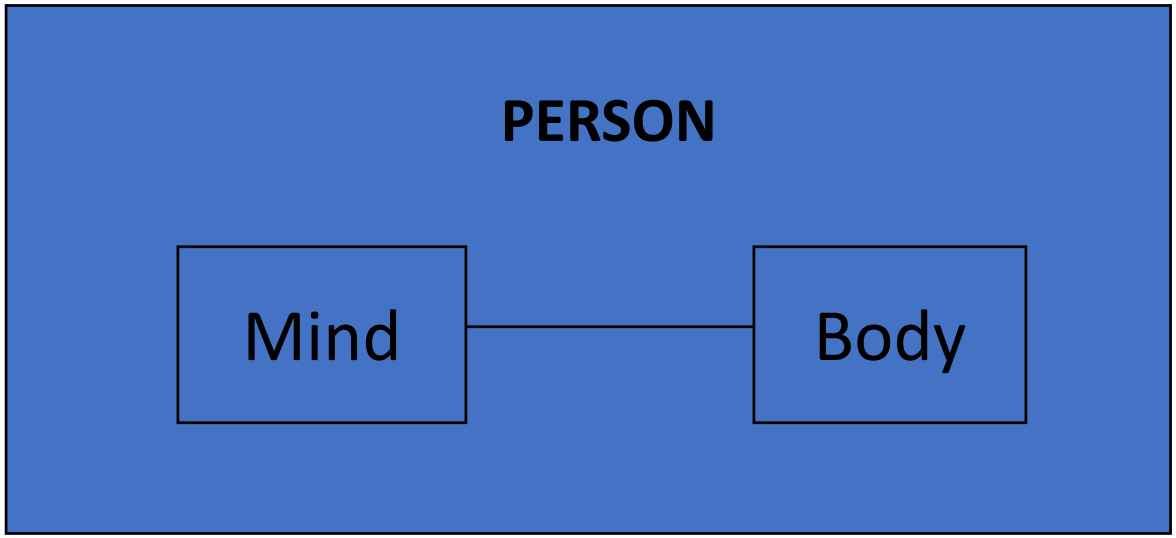


8.4 Other UML Diagrams cont.

What is New in UML 2.0

- New Diagrams: Composite Structure Diagram
  - Show classes internal structure

EXAMPLE



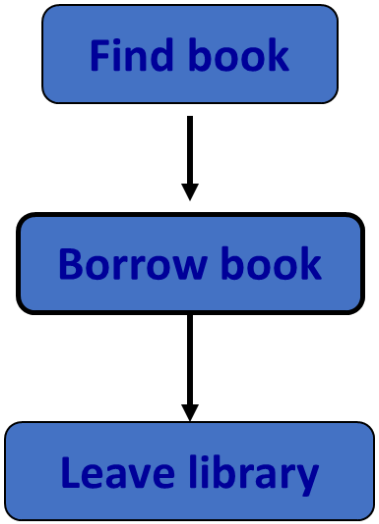
8.4 Other UML Diagrams cont.

What is New in UML 2.0

- New Diagrams: Interaction Overview Diagram
  - Expands the Activity Diagram

Example:

Consider three activities in visiting a library



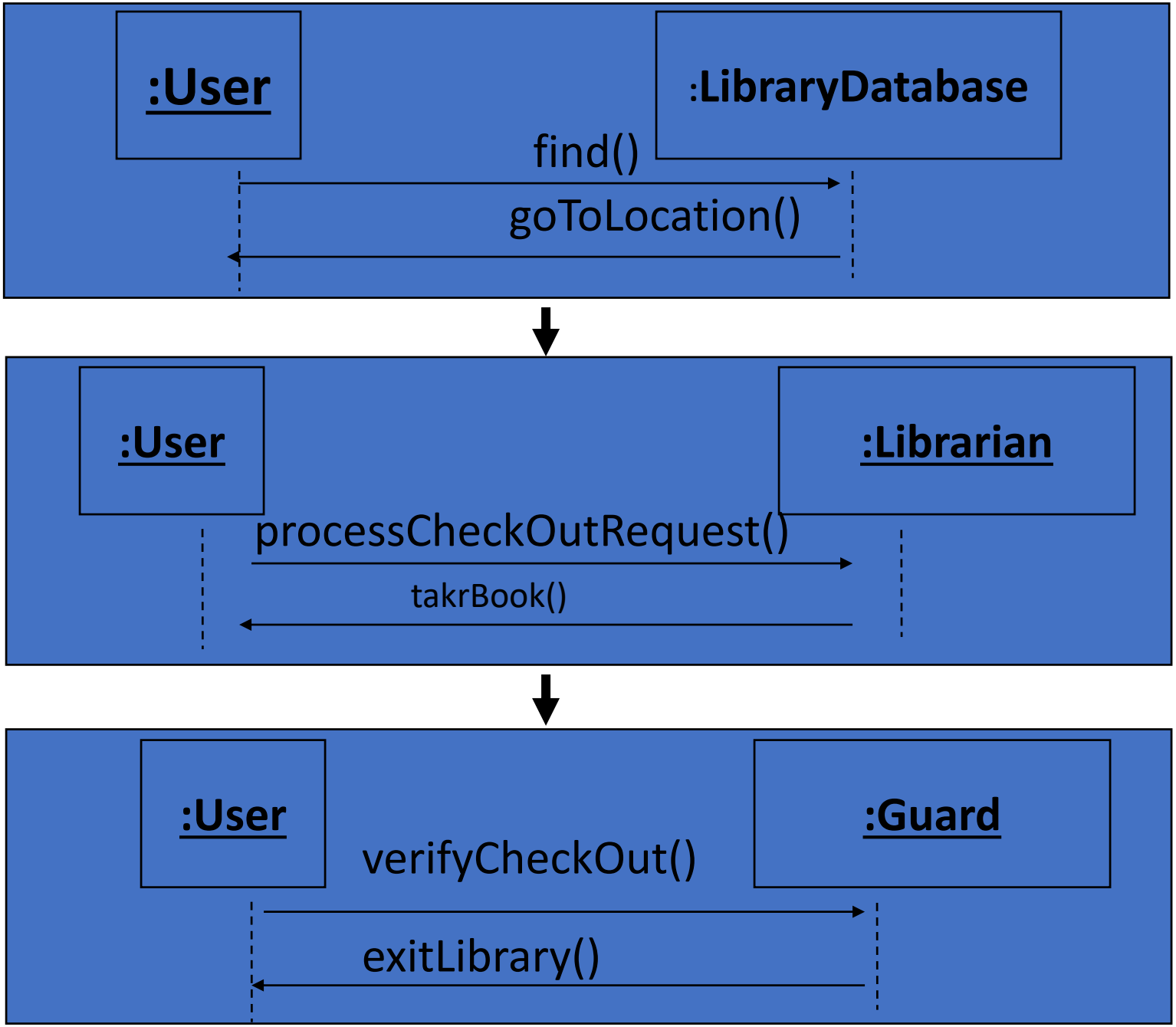
Interaction Overview Diagram

- Let's analyze each activity

Find Book : Ask library database to locate the book. Database responds by telling you to go to the books location.

Borrow Book : Ask the librarian to check the book out to you. After checkout, the librarian tells you to take the book.

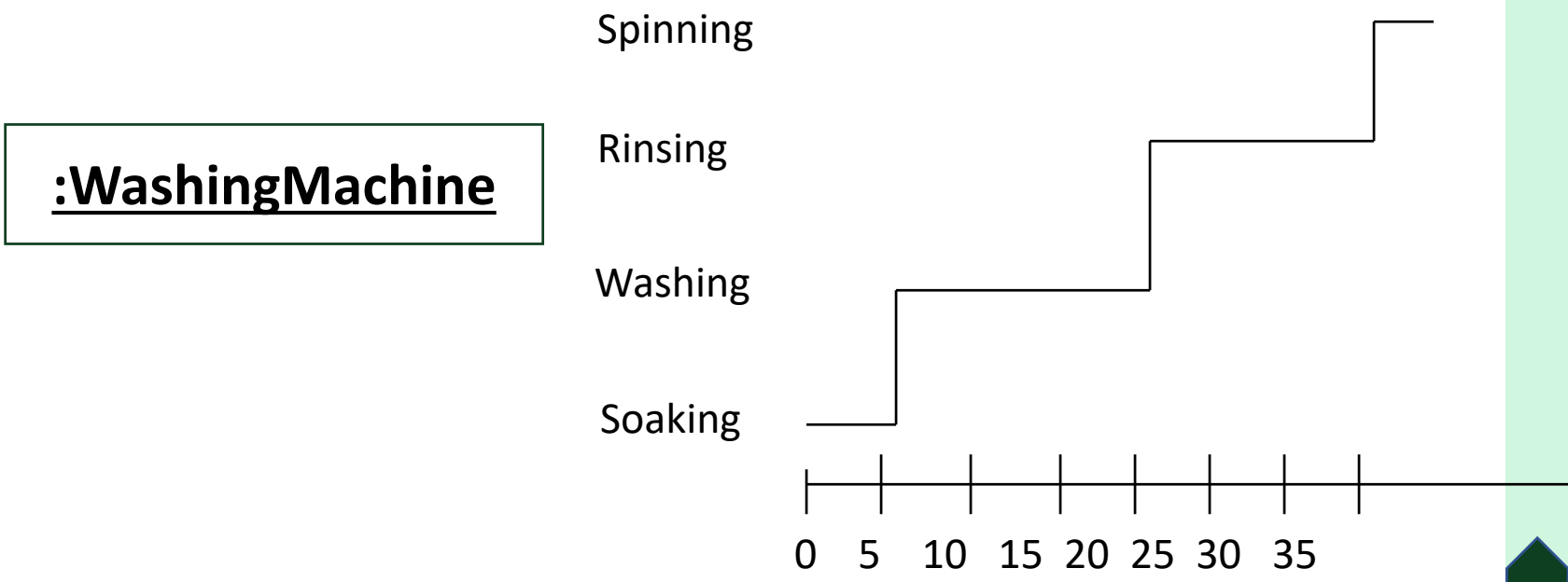
Leave Library : You can leave the library only if a guard verifies that you have checked out the book.



8.4 Other UML Diagrams cont.

What is New in UML 2.0

- New Diagrams: Timing Diagram
  - Design to show how long an object is in a state.
  - Sequence diagrams does not show the durations explicitly.



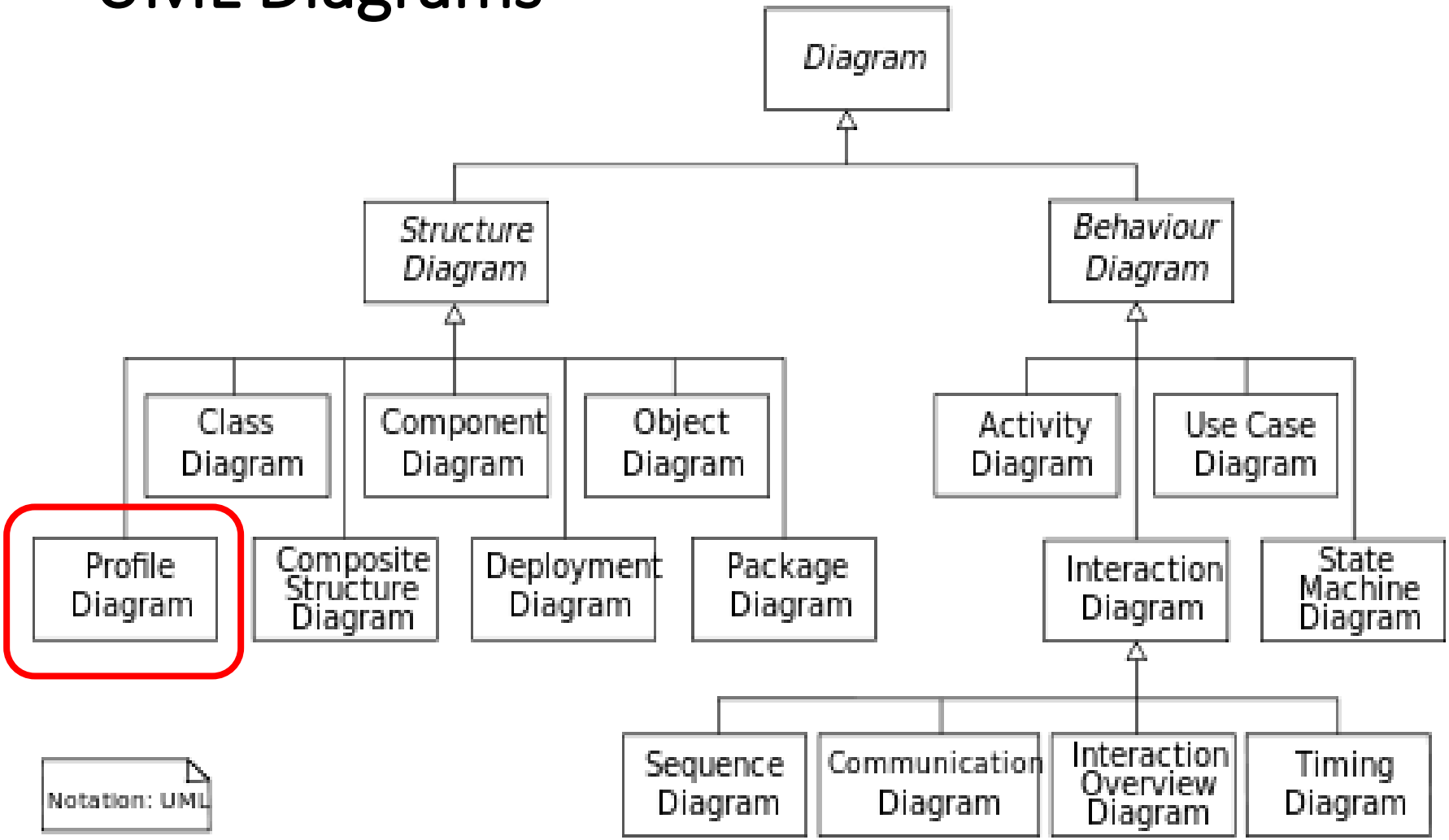
UML Profiles

- A lightweight extension mechanism for UML
- Concepts partially present in earlier versions
  - Stereotypes. <<entity>>
  - Tagged Values {author=Siman Silva}
- Established as a specific meta-modeling technique in UML 2.0
  - Contains mechanisms that allow meta classes from existing meta models to be extended.
  - Ability to tailor the UML meta model for different platforms or domains.

UML Profile Diagrams

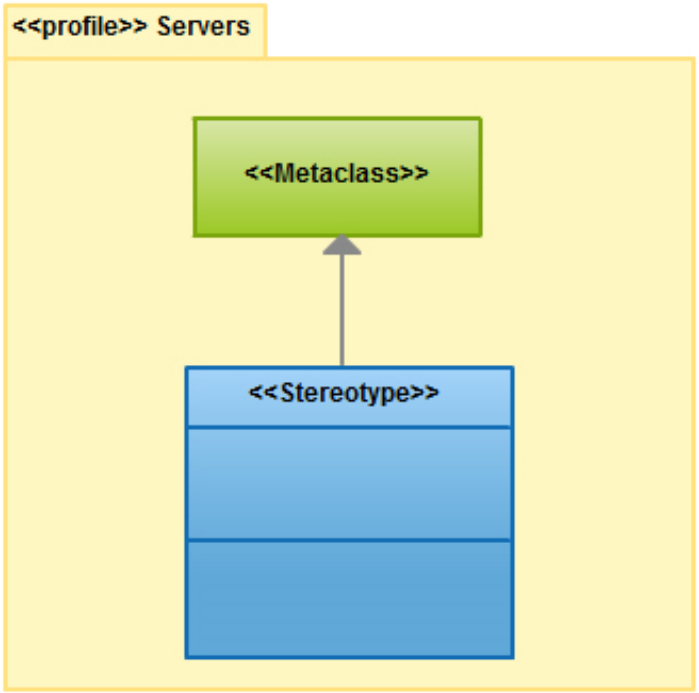
- **Profile diagram** is **structure diagram**
- Describes **lightweight extension mechanism** to the UML by defining custom **stereotypes**, **tagged values**, and constraints.
- Profiles allow adaptation of the UML metamodel for different:
  - **platforms** (such as J2EE or .NET), or
  - **domains** (such as real-time or business process modeling).

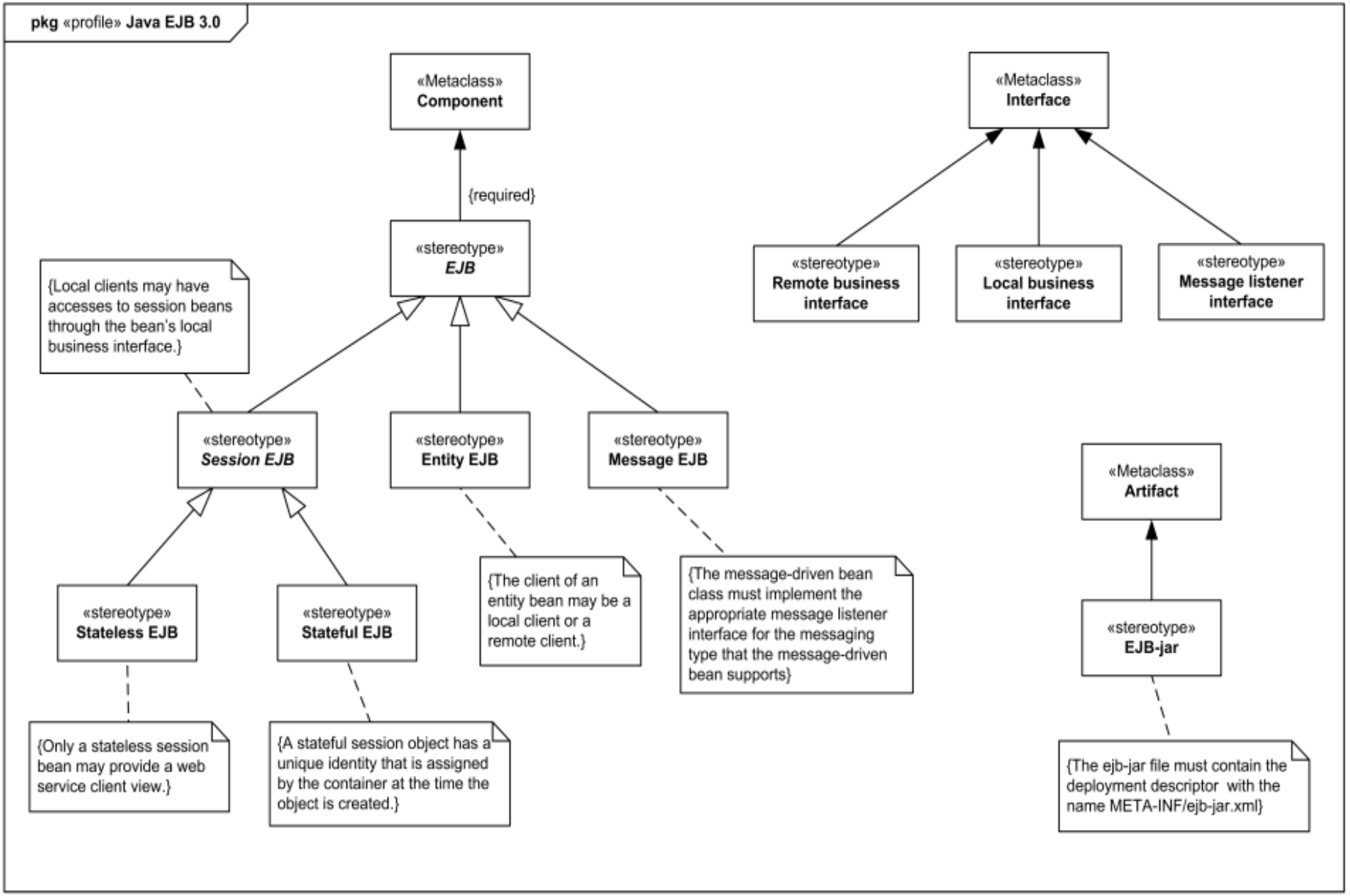
UML Diagrams



Basic UML Profile Diagram Structure

UML Profiles provide a generic extension mechanism for building UML models in particular domains.





Ref. http://www.uml-diagrams.org/profile-diagrams.html

Summary

- A behavioral state machine is a dynamic model that shows the different states through which a single object passes during its life in response to events, along with its responses and actions.
- State Diagrams are created only for classes with significant dynamic behaviour.
- State diagrams mainly depict states and transitions. States are represented with rectangles with rounded corners that are labeled with the name of the state. Transitions are marked with arrows that flow from one state to another, showing how the states change.
- A State Transition may have the following associated with:
  - an action: behaviour that occurs when the state transition occurs, and/or
  - a guard condition: allows state transition only if it is true.

Summary cont..

- One useful technique to identify how the underlying objects in the problem domain work together to collaborate in support of the use cases is *CRUDE analysis.*
- CRUDE analysis uses a *CRUDE matrix*, in which each interaction among objects is labeled with a letter for the type of interaction:
  - C for Create, R for Read or Reference, U for Update, D for Delete, and E for Execute.
- We can combine walkthroughs with CRUDE matrices to more completely verify and validate the behavioral models.

Summary cont..

- A Component diagram shows how different elements of your system have been grouped together and the link between these components.
- A Deployment diagram takes you one step further and describes on which hardware elements do these components reside.
- Composite Structure Diagram shows the internal structure of the class, whereas Timing diagram is design to show how long an object is in a state.
- An interaction overview diagram is a form of activity diagram in which the nodes represent interaction diagrams.
- Profile diagram is a kind of structural diagram in the UML, that provides a generic extension mechanism for customizing UML models for particular domains and platforms.

*****