UML: Class Diagram

Dependency Relationship

 In UML, a dependency relationship is a relationship in which one element, the client, uses or depends on another element, the supplier.

 a change to the supplier might require a change to the client.

 Dependency A dependency relationship is represented by a dotted arrow (see in the figure) that is drawn from the dependent class to the independent class.

Example: In an e-commerce application, a Cart class depends on a Product class because the Cart class uses the Product class as a parameter for an add operation. In a class diagram, a dependency relationship points from the Cart class to the Product class. As the following figure illustrates, the Cart class is, therefore, the client, and the Product class is the supplier.

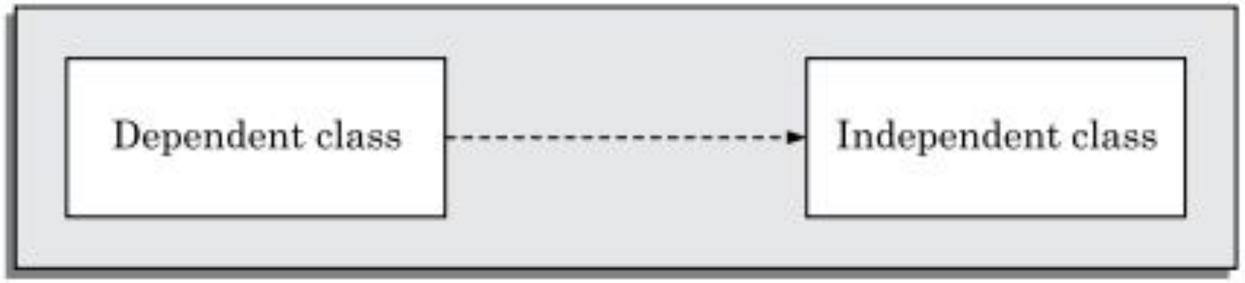
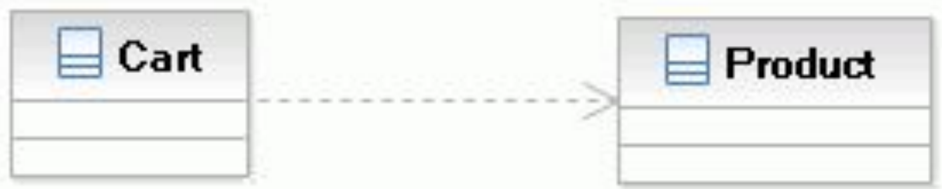


Figure: Representation of dependence between classes



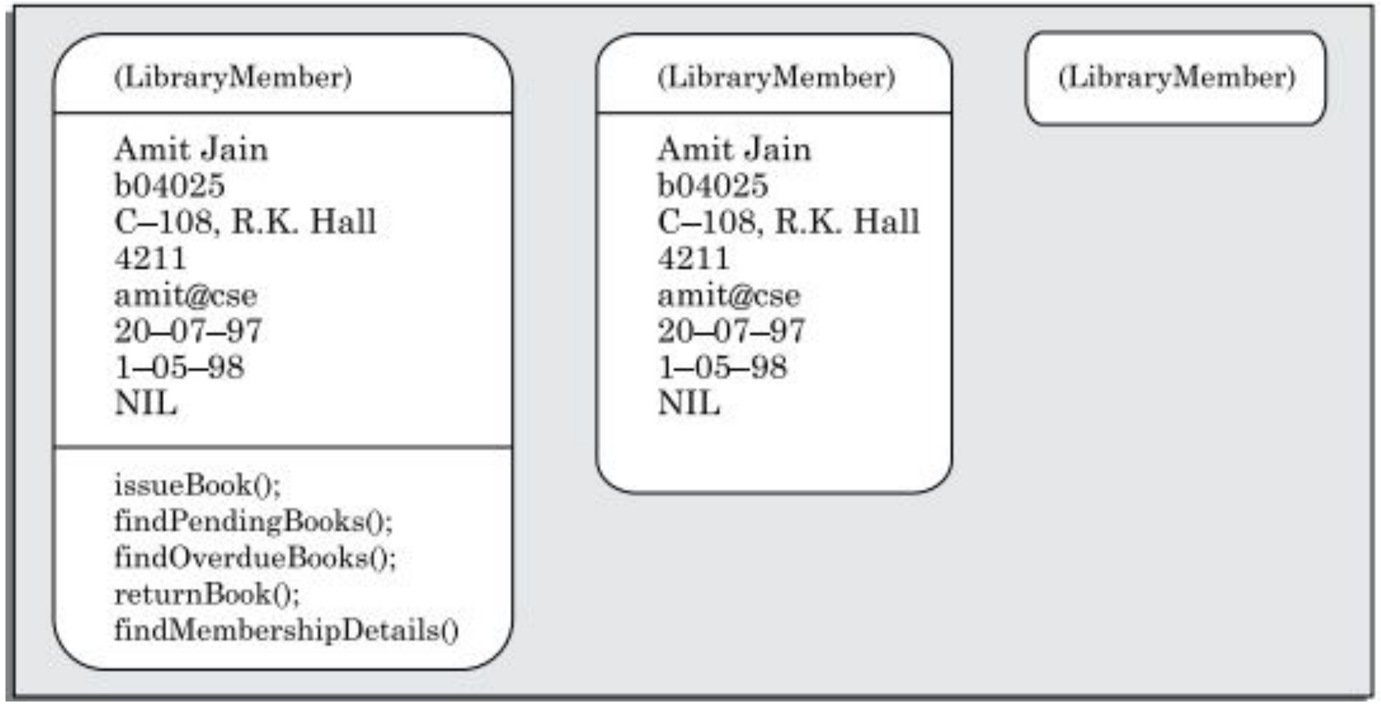
Example of dependency Relationship

Constraints

- A constraint describes either a condition that needs to be satisfied or an integrity rule for some entity.
- Constraints are typically used to describe aspects such as: permissible set of values of an attribute, specification of the pre- and post-conditions for operations, and definition of certain ordering of items.
- Example: To denote that the books in a library are maintained sorted on ISBN number, we can annotate the book class with the constraint {sorted}.
- UML allows the flexibility to use any set of words to describe the constraints. The only rule is that they are to be enclosed within braces.

Object Diagram

- During the execution of a program, objects may dynamically get created and also destroyed.
- An object diagrams shows a snapshot of the objects in a system at a point in time. Since an object diagram shows instances of classes, rather than the classes themselves, it is often called as instance diagram.
- The objects are drawn using rounded rectangles (as shown in the figure).
- As the class diagrams, the object diagrams may just indicate the name of the object or more details.



UML: Interaction Diagrams

Interaction Diagram

- When a user invokes any one of the use cases of a system, the required behavior is realized through the interaction of several objects in the system.
- An interaction diagram describes how groups of objects interact among themselves through message passing to realize some behavior (execution of a use case).
- For complex use cases, more than one interaction diagram may be necessary to capture the behavior.
- There are two kinds of interaction diagrams—
  - Sequence diagrams
  - Collaboration diagrams.
- Both Sequence and Collaboration diagrams are equivalent in the sense that any one diagram can be derived automatically from the other.

Sequence Diagram

A sequence diagram shows the interactions among objects as a two dimensional chart.

The chart is read from top to bottom.

The objects participating in the interaction are drawn at the top of the chart as boxes attached to a vertical dashed line.

Inside the box the name of the object is written with a colon separating it from the name of the class and both the name of the object and the class are underlined.

When no name is specified, it indicates that we are referring any arbitrary instance of the class. For example, in the given figure, Book represents any arbitrary instance of the Book class.

Sequence Diagram

- An object appearing at the top of a sequence diagram signifies that the object existed even before the time the use case execution was initiated.

- If some object is created during the execution of a use case and participates in the interaction (e.g., a method call), then the object should be shown at the appropriate place on the diagram where it is created.

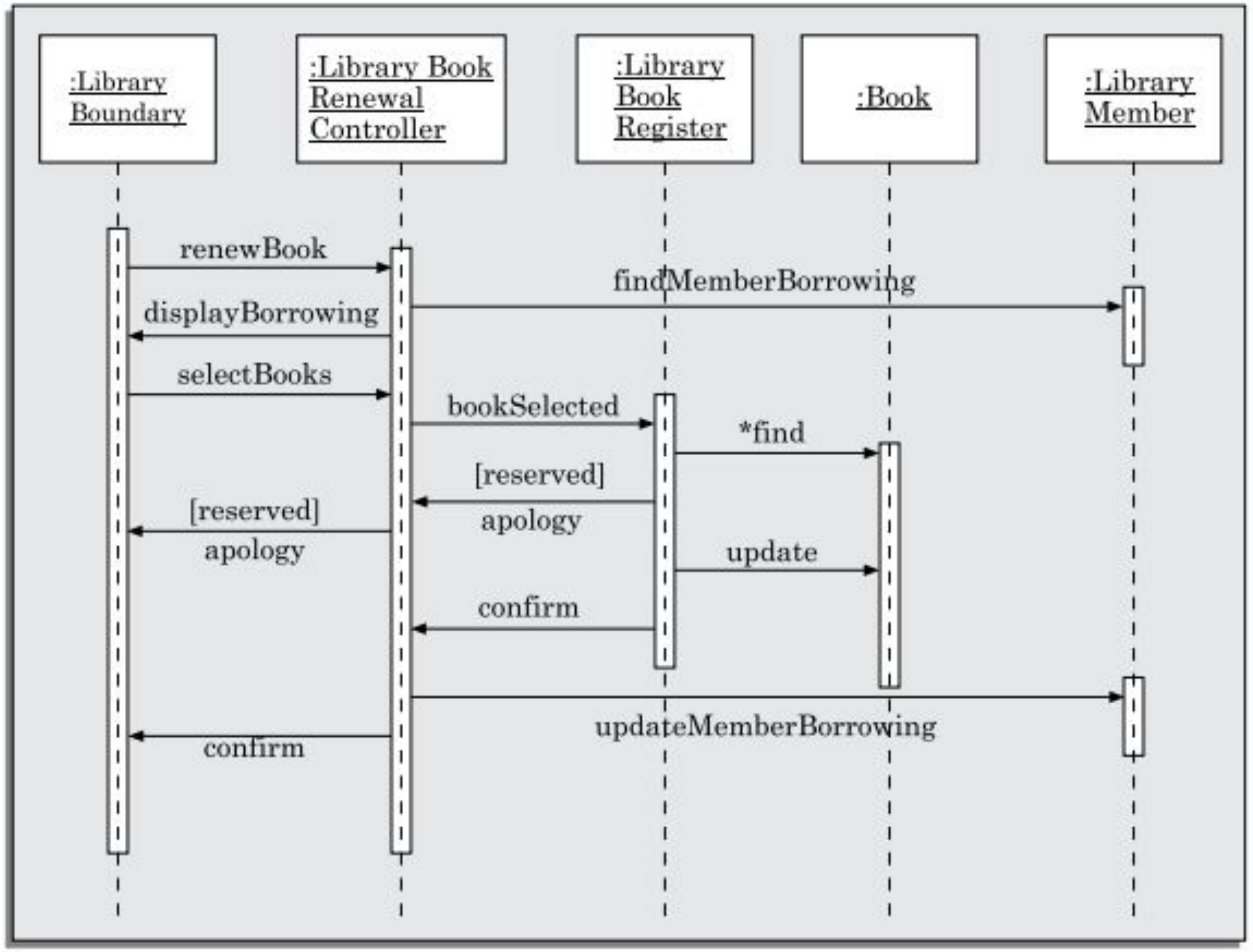


Figure: Sequence diagram for the renew book use case

Sequence Diagram

- The vertical dashed line attached to an object is called the object’s lifeline. An object exists as long as its lifeline exists. Absence of lifeline after some point indicates that the object ceases to exist after that point in time.
- Normally, at a certain point if an object is destroyed, the lifeline of the object is crossed at that point and the lifeline for the object is not drawn beyond that point.
- When an object receives a message, a rectangle called the *activation symbol* is drawn on the lifeline of an object to indicate the points of time at which the object is active.
- An activation symbol indicates that an object is active as long as the symbol (rectangle) exists on the lifeline.
- Each message is indicated as an arrow between the lifelines of two objects. The messages are drawn in chronological order from the top to the bottom.
- Each message is labelled with the message name, and optionally some control information can also be included along with the message name.

Collaboration Diagram

- A collaboration diagram shows both structural and behavioral aspects explicitly. This is unlike a sequence diagram which shows only the behavioral aspects.
- The structural aspect of a collaboration diagram consists of objects and links among them indicating association between the corresponding classes.
- In this diagram, each object is also called a collaborator.
- The behavioral aspect is described by the set of messages exchanged among the different collaborators.
- The messages are numbered to indicate the order in which they occur in the corresponding sequence diagram. The link between objects is shown as a solid line and messages are annotated on the line.
- A message is drawn as a labelled arrow and is placed near the link. Messages are prefixed with sequence numbers because they are the only way to describe the relative sequencing of the messages in this diagram.

Collaboration Diagram

The collaboration diagram for the example of use-case sequence ‘renew book’ is shown in figure. A collaboration diagram explicitly shows which class is associated with other classes. Therefore, we can say that the collaboration diagram shows structural information more clearly than the sequence diagram.

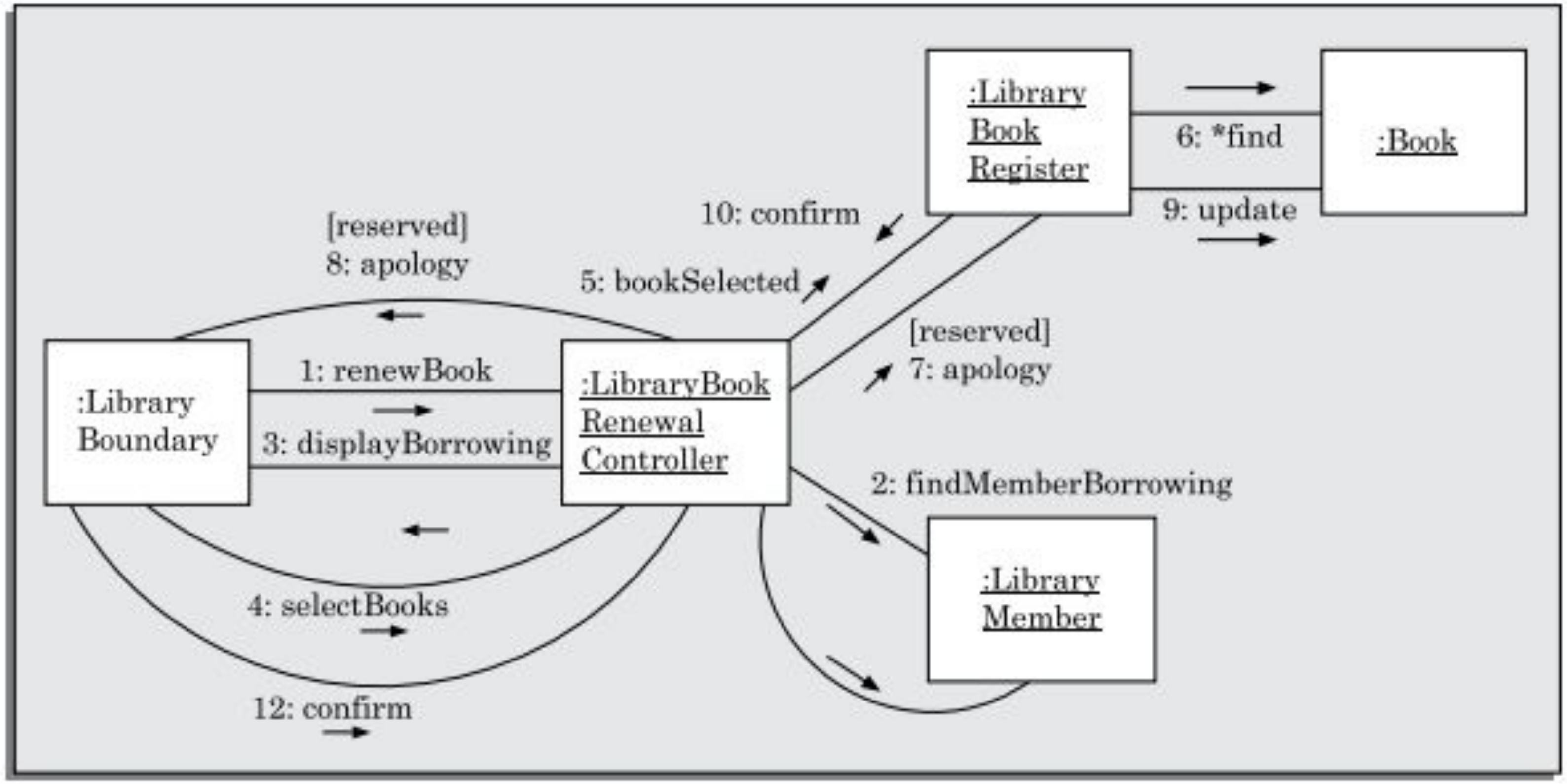


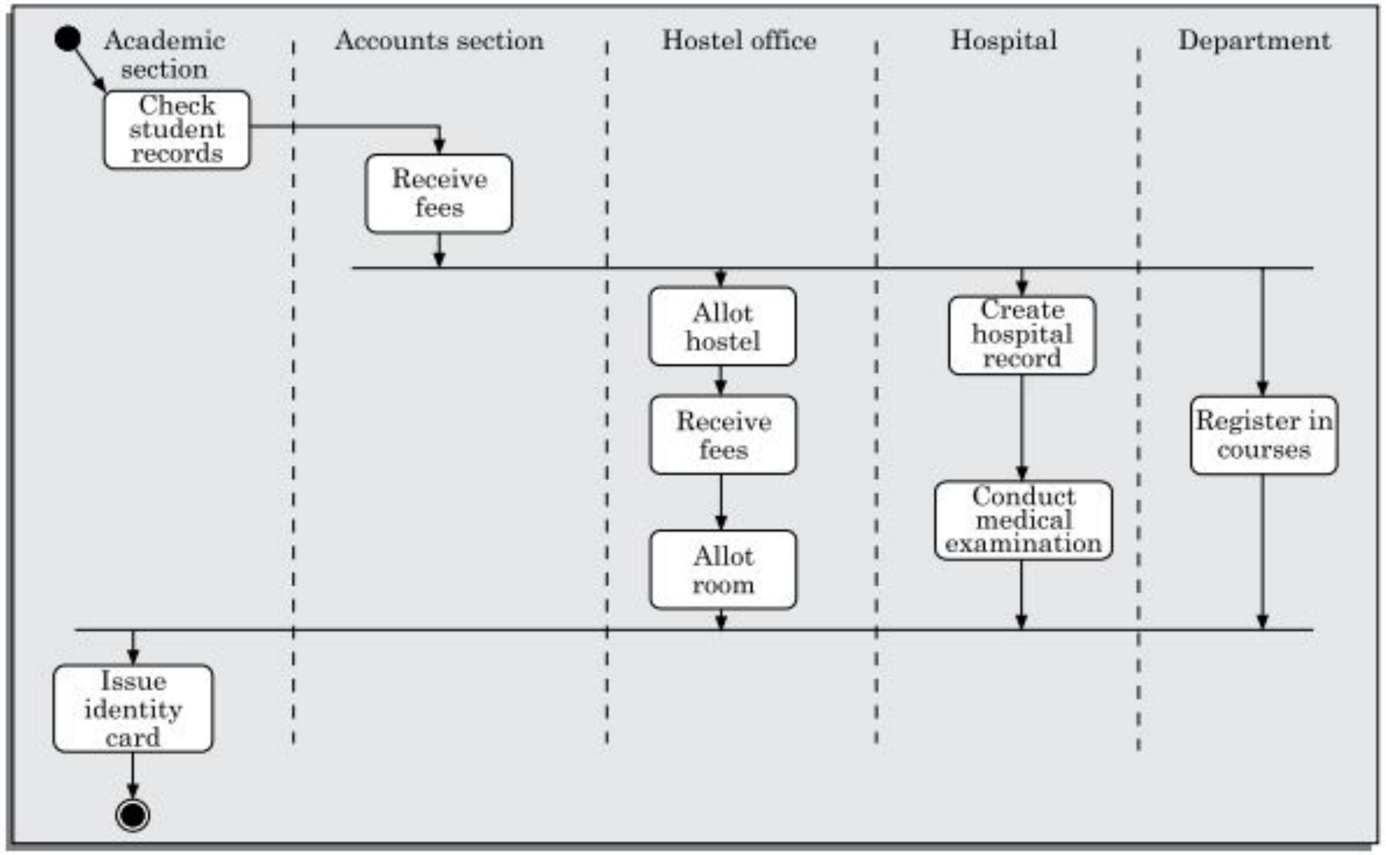
Figure: Collaboration diagram for the renew book use case

Activity Diagram

- An activity diagram can be used to represent various activities (or chunks of processing) that occur during execution of the software and their sequence of activation.
- An activity is a state with an internal action and one or more outgoing transitions. On termination of the internal activity, the appropriate transition is taken. If an activity has more than one outgoing transition, then the exact conditions under which each is executed must be identified through use of appropriate conditions.
- Activity diagrams are to some extent similar to flow charts. The main difference is that activity diagrams support description of parallel activities and synchronization aspects involved in different activities.
- Activity diagrams incorporate swim lanes to indicate which components of the software are responsible for which activities.

Activity Diagram

- Parallel activities are represented on an activity diagram by using swim lanes.
- Swim lanes make it possible to group activities based on who is performing them.
- Swim lanes subdivide activities based on the responsibilities of various components.
- For each component participating in the use case execution, it becomes clear as to the specific activities for which it is responsible.



- The swim lane corresponding to the academic section, the activities that are carried out by the academic section (check student record and issue identity card) and the specific situation in which these are carried out are shown in the figure above.

State Chart Diagram

- A state chart diagram is normally used to model how the state of an object may change over its life time.
- State chart diagrams are good at describing how the behavior of an object changes across several use case executions.
- State chart diagrams are based on the finite state machine (FSM) formalism which consists of a finite number of states corresponding to those that the object being modeled can take. The object undergoes state changes when specific events occur.

State Chart Diagram

Basic elements of a state chart

Initial state: This represented by a filled circle.

Final state: This is represented by a filled circle inside a larger circle.

State: These are represented by rectangles with rounded corners.

Transition: A transition is shown as an arrow between two states. In general, the name of the event which causes the transition is placed alongside the arrow. A guard can be assigned to the transition. A guard is a Boolean logic condition. The transition can take place only if the guard evaluates to true. Transition having no event or guard annotated with it is called pseudo transition. The syntax for the label of the transition is shown in 3 parts—[guard]event/action.

State Chart Diagram

An example state chart model for the order object of the Trade House Automation software is shown in the figure. Observe that from the Rejected order state, there is an automatic and implicit transition to the end state. Such transitions which do not have any event or guard annotated with it are called pseudo transitions.

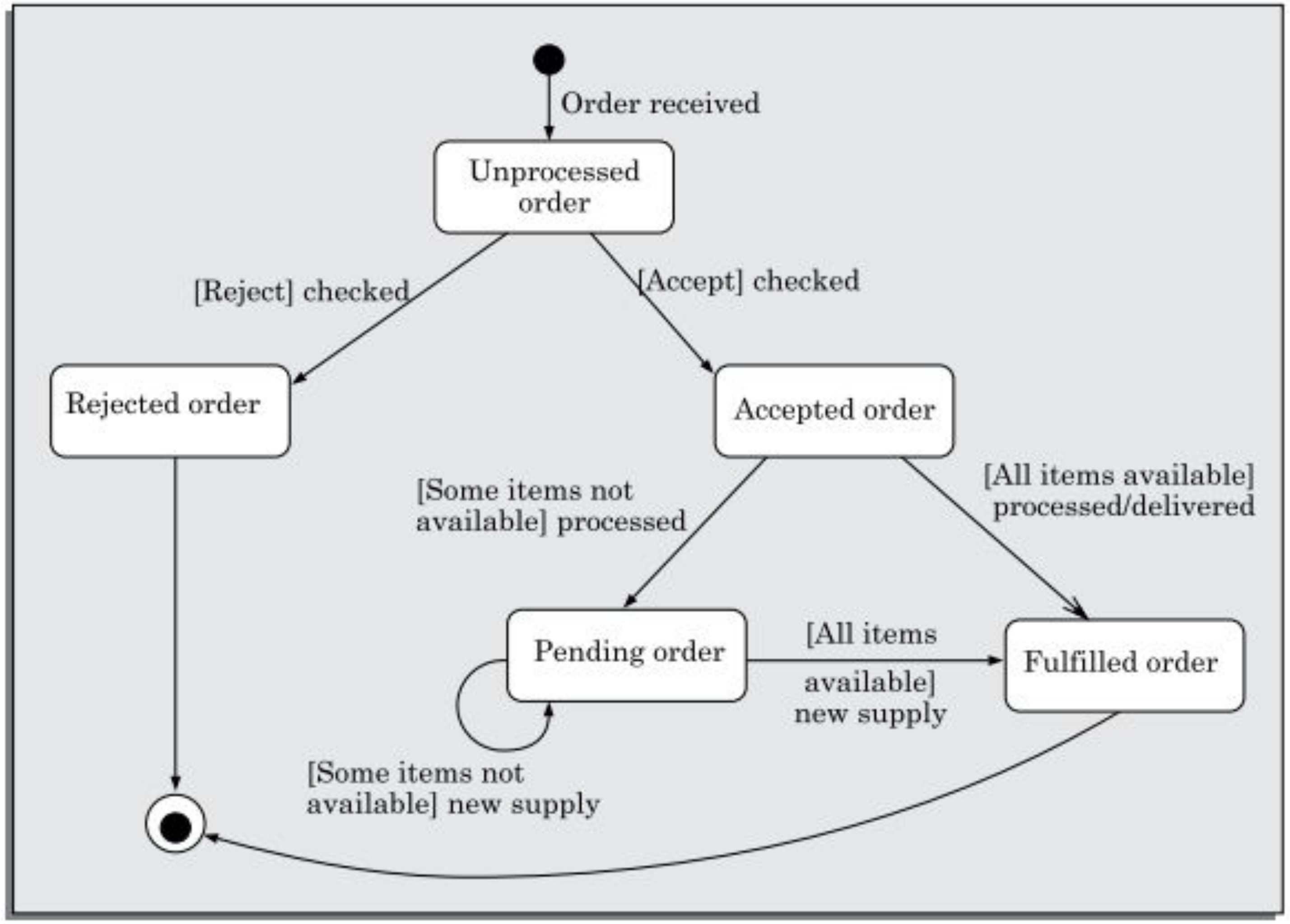

Order received
Unprocessed order
[Reject] checked
[Accept] checked
Rejected order
Accepted order
[Some items not available] processed
[All items available] processed/delivered
Pending order
[All items available] new supply
Fulfilled order
[Some items not available] new supply

Figure: State chart diagram for an order object