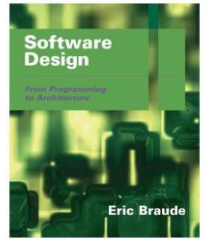
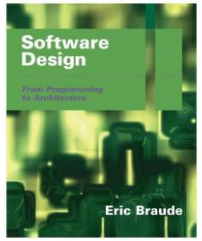


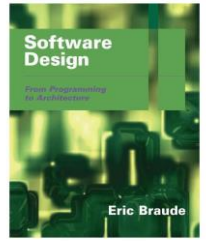
Lecture 3- The Unified Modeling Language

Topic covered



- ✧ Review of the relationships between classes.
- ✧ UML models and notations.

The Unified Modeling Language (UML)

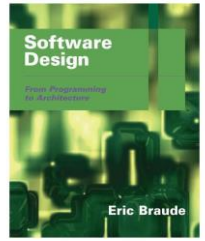


✧ UML is a graphical notation for expressing object-oriented design

✧ UML is managed by the Object Managed Group (OMG)

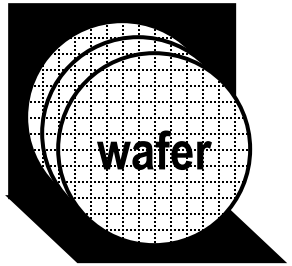
✧

Classes at Detailed Design



canister

+:Visible from without




```
Canister
```

```
+numCanisters: int
-numWafers: int
-size: float
```

```
+display()
-getNumOpenSlots()
+setStatus()
```

```
Responsibilities:
-- describes each
canister undergoing
fabrication
```

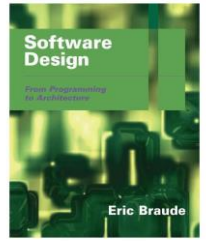
Class name

Attribute: type

Operations

Place for comments

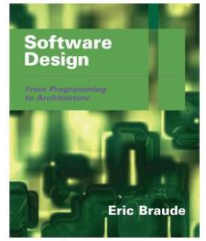
Class Relationships in UML



✧ Class relationships
- Inheritance and interface
- Aggregation
  - Composition
- Dependency
- Association

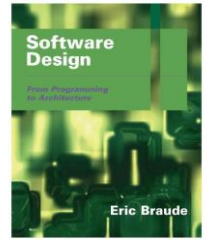
✧ UML uses the term *package* for collecting design elements such as classes.

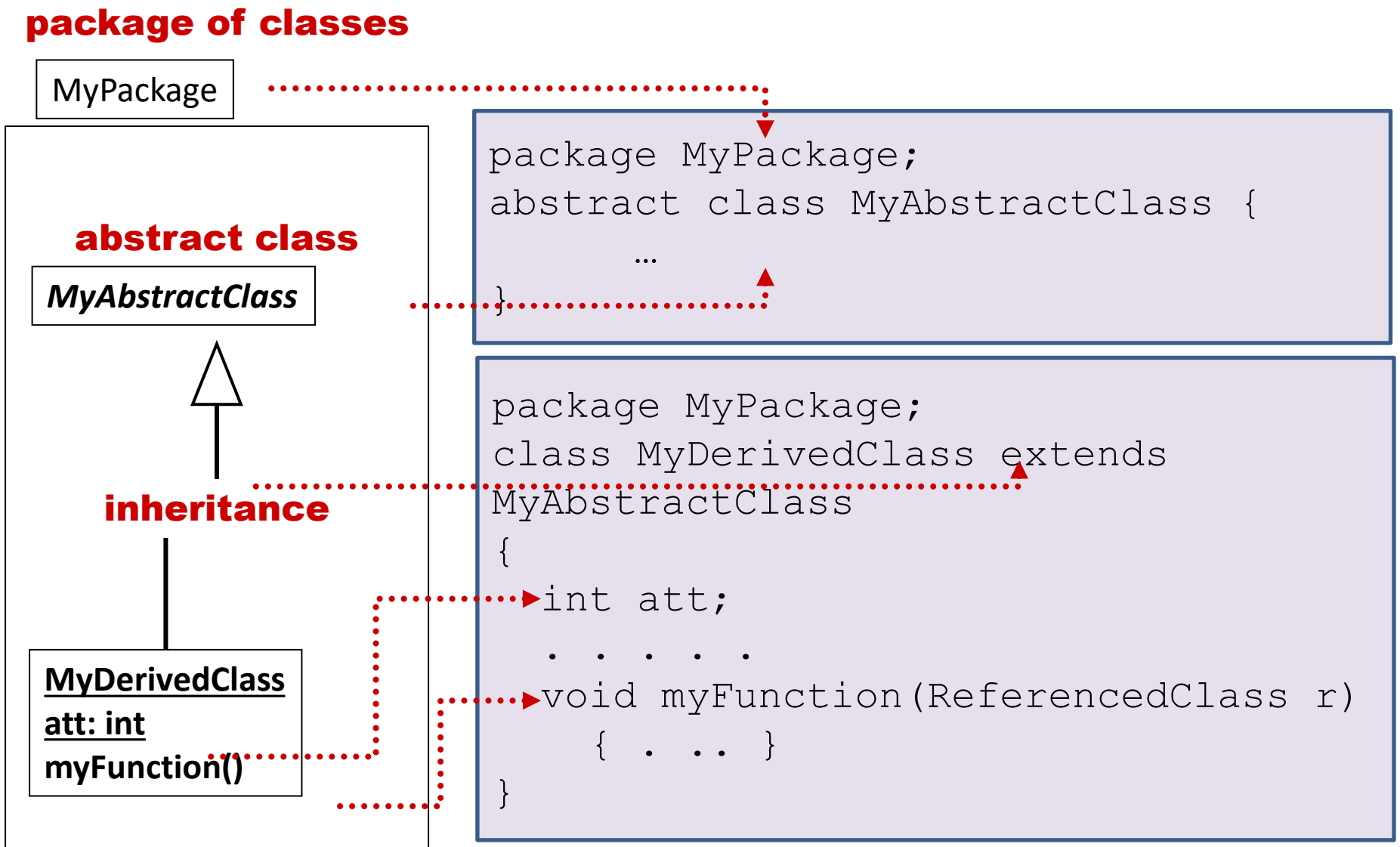
Representing a Class in UML



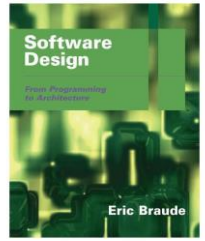
- UML represents a class with a rectangle containing the class name. We display additional information within the rectangle as needed: Variables, methods, etc.

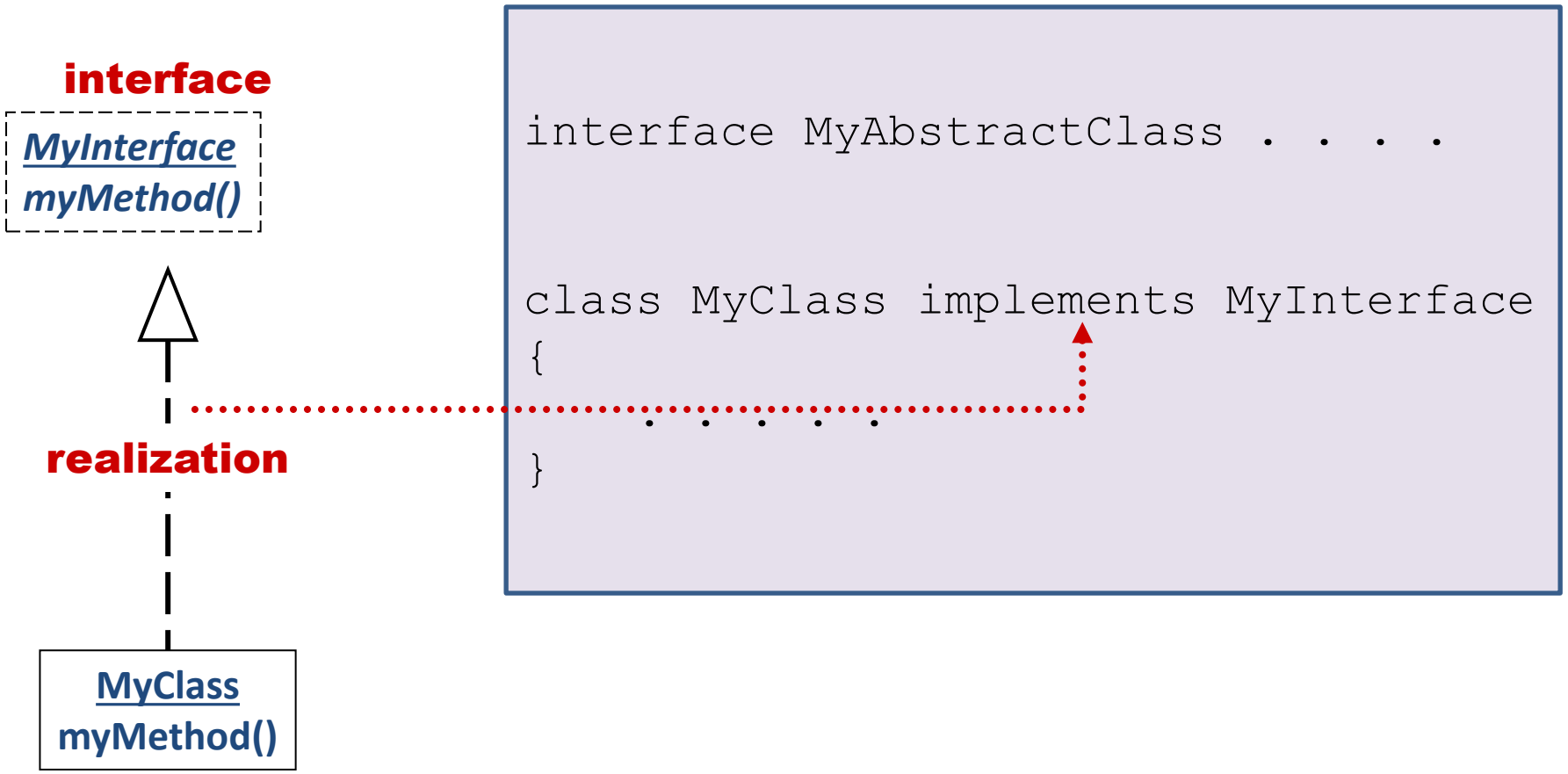
UML Notation and Typical Implementation



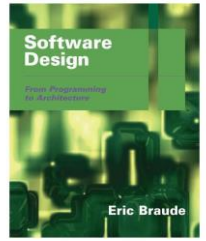


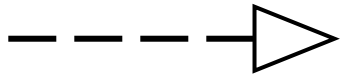
Interfaces UML Notation/Java Implementation



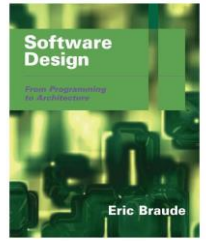


Representing Inheritance in UML



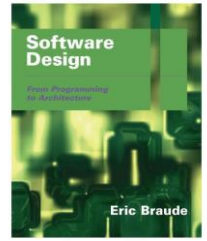
- ✧ UML represents inheritance and interface realization with an open triangle.
- ✧ Inheritance: with solid line 
- ✧ Realization: with dotted line 

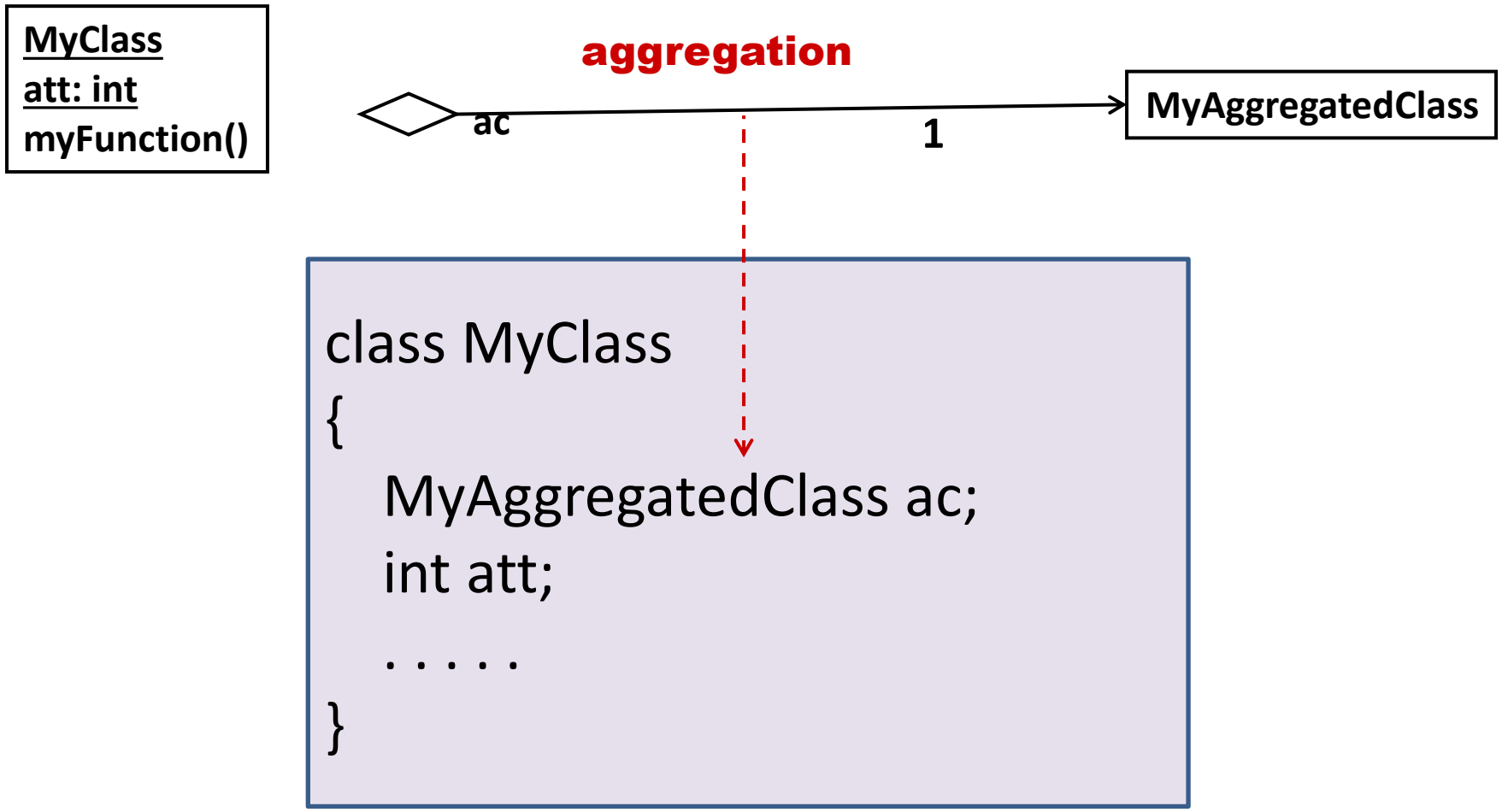
Class Relationships in UML: Aggregation



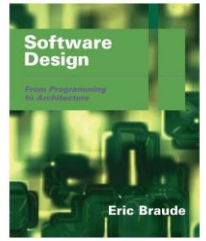
- ✧ Class A aggregates class B if A objects require B objects in a structural sense
- ✧ It includes the structural inclusion of objects of one class by another
- ✧ It models “whole – part” relationship
- ✧ It is denoted with a diamond
- ✧ Composition is a stronger form of aggregation in which the aggregated object exists only during the lifetime of the composing object

Aggregation : UML Notation / Implementation



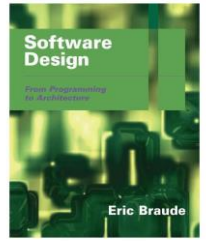


Representing Aggregation in UML



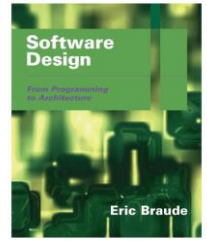
- Class A aggregates class B if A objects require B objects in a structural sense – typically with an instance variable.
- UML symbol is an open diamond.

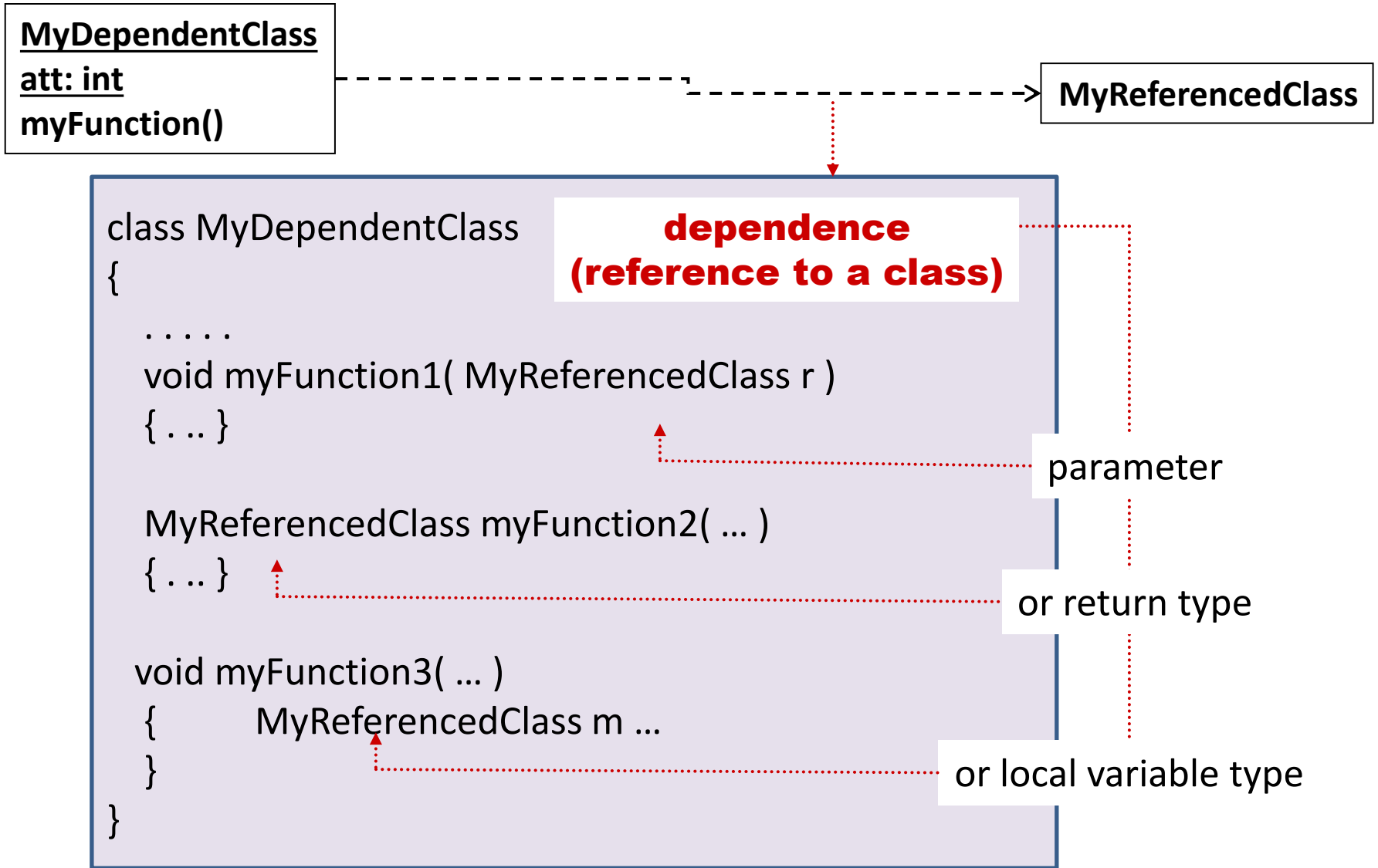
Class Relationships in UML: Dependency



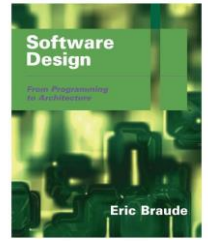
- ✧ Class A depends on class B if objects of A require objects of B for their definition.
- ✧ It is denoted by dotted line arrow
- ✧ Dependency includes inheritance and aggregation
- ✧ Dependency in UML indicates that a method of one class utilizes another class

Dependence : UML Notation … and …Typical Implementation



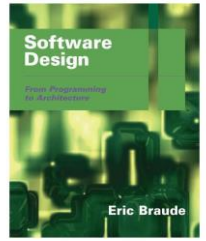


Representing Dependency



- ✧ Class A depends on class B if A objects require B objects for their definition. In practice, this means that B appears in at least one method of A.
- ✧ UML representation: a dotted arrow.

Class Relationships in UML: Association

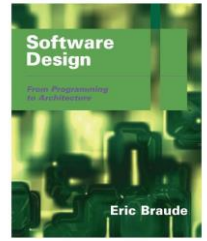


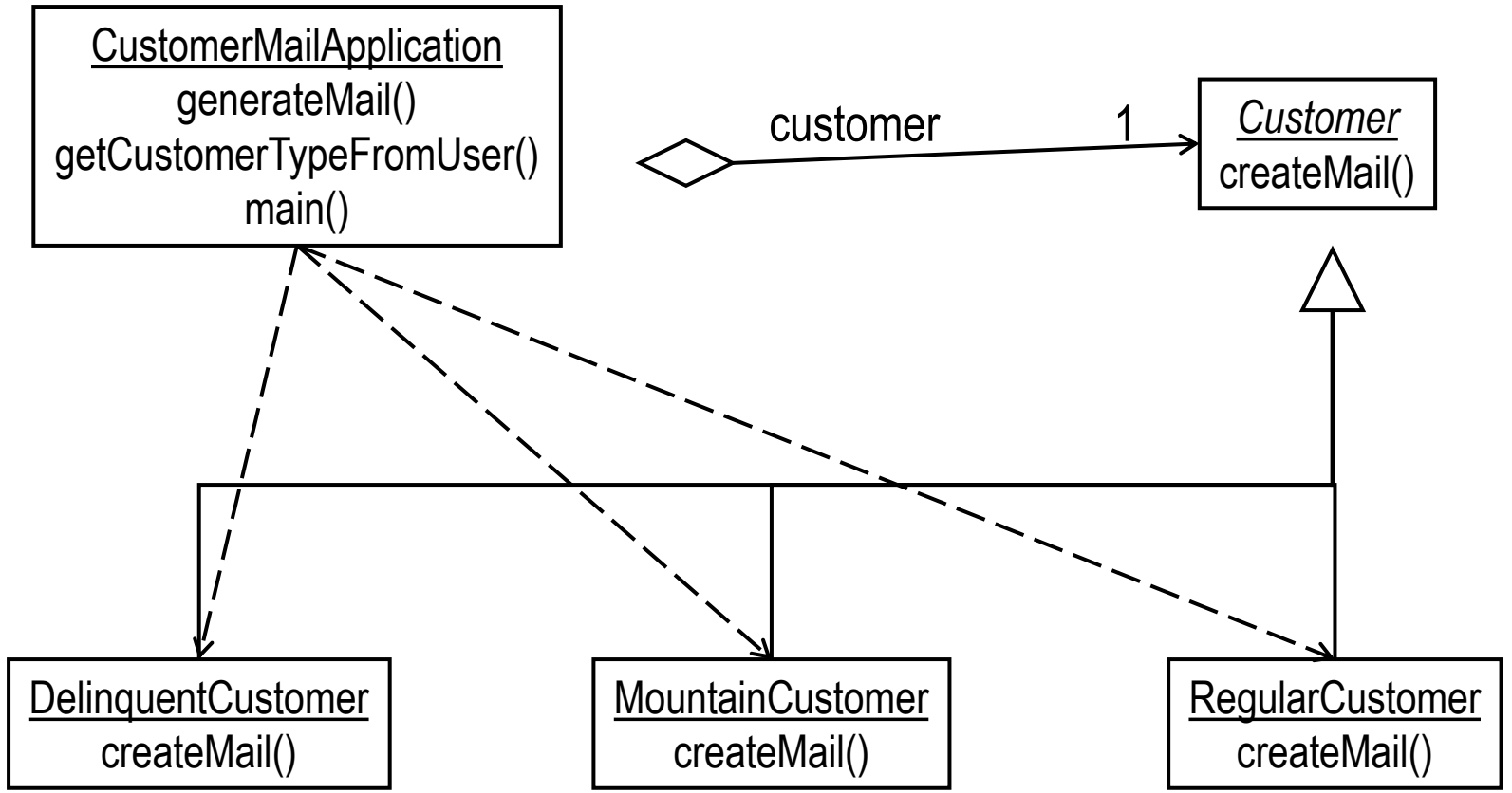
- ✧ Objects of one class depends on objects of the other class in structural sense.
- ✧ It is denoted with **solid** line between two classes

Association : UML Notation /Implementation

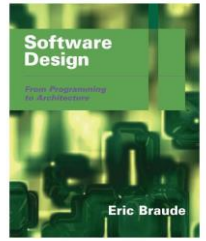

```
class Employer
{
   Employee[ ] employees;
   . . . . .
}

class Employee
{
   Employer[ ] employers;
   . . . . .
}
```

Customer Mail Application



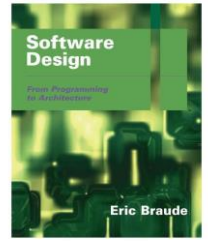


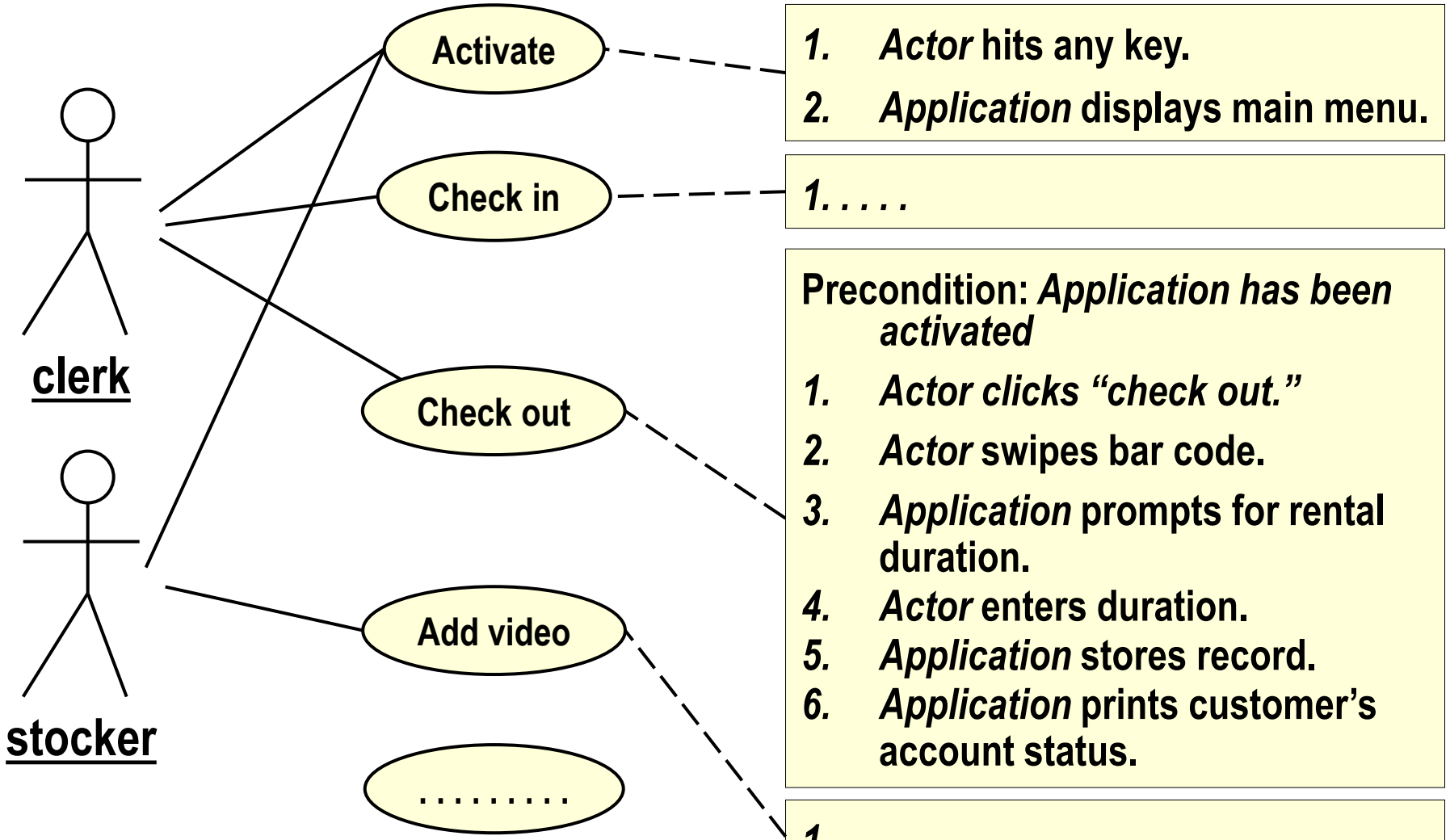
Use Cases



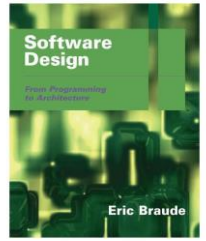
- ✧ A use case is a sequence of actions taken by an application and its user
- ✧ A use case has three components
  - Name
  - Actor: the type of user of the application
  - Interaction between the actor and application
- ✧ Use cases are particularly useful for requirements analysis

Use Cases For Video Store Application





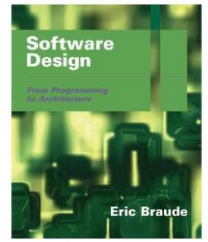
Use Cases (cont...)

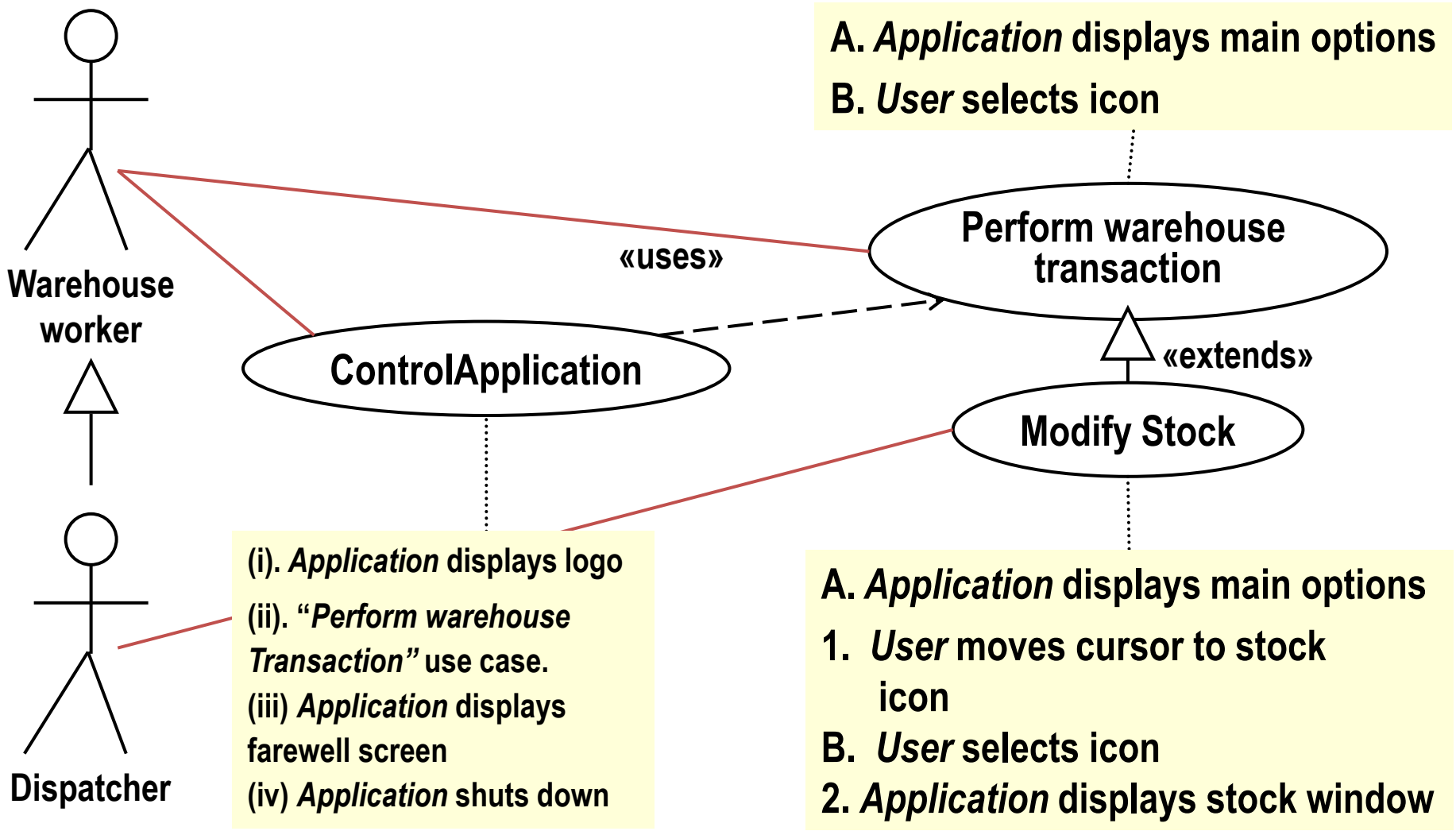


✧ Combining use cases

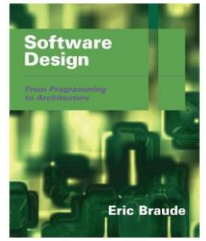
- Use cases can be dependent on other use cases
- *extends* relationship: like inheritance relationship
- *uses* relationship: like aggregation relationship

Use Case Generalization & Usage



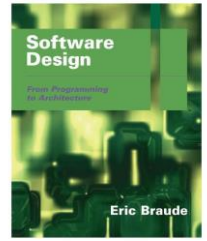


Sequence Diagrams

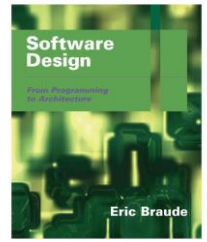


- Sequence diagrams are graphical representations of control flow
- Sequence diagrams are useful for describing executions that involve several classes
- Sequence diagrams require us to think in terms of objects

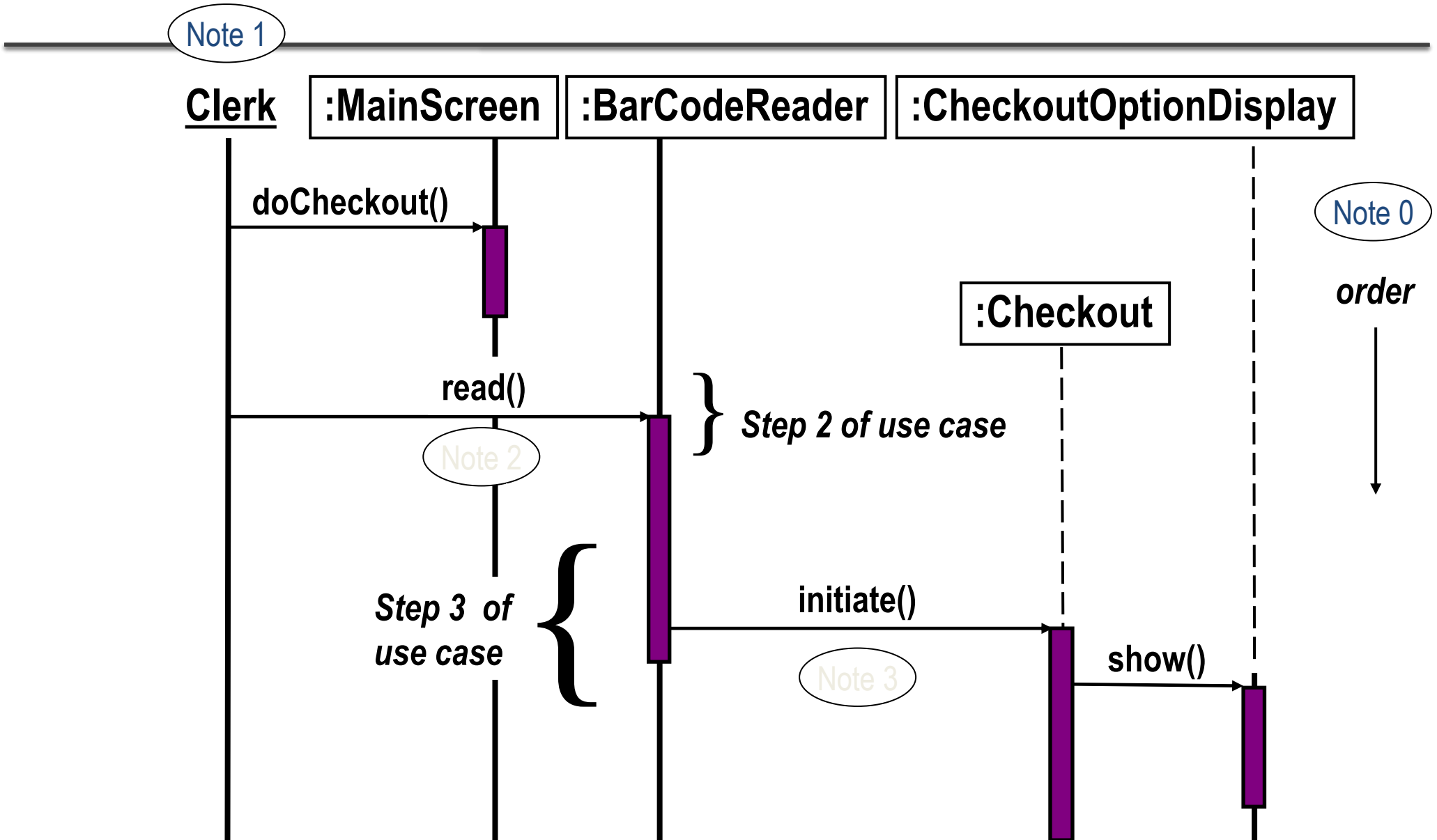
Sequence Diagrams



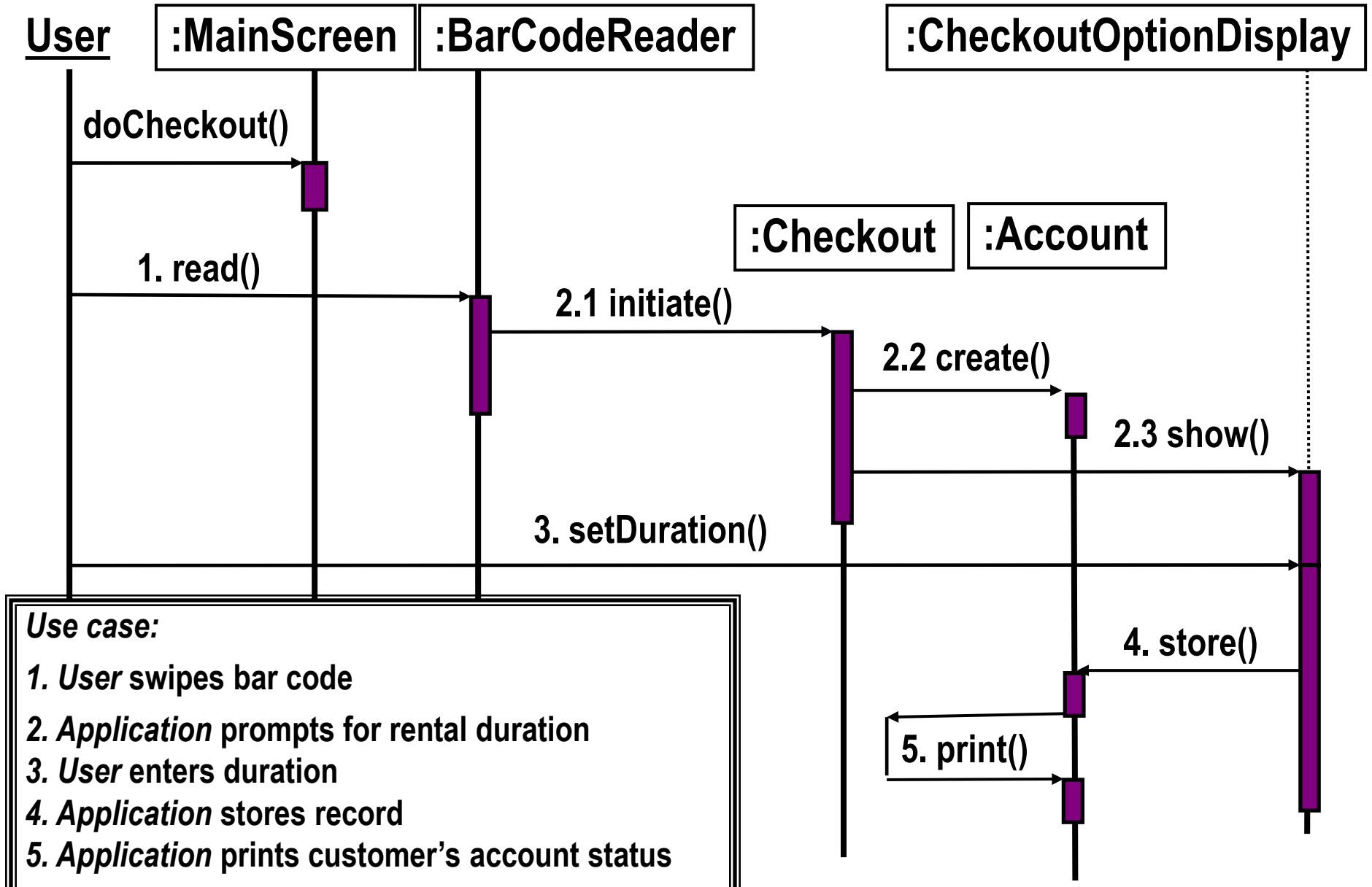
- A sequence diagram shows the order in which methods of various objects execute.



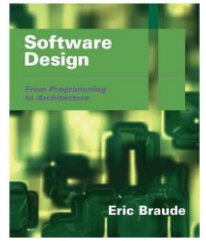
Beginning of Sequence Diagram for *Check Out* Use Case



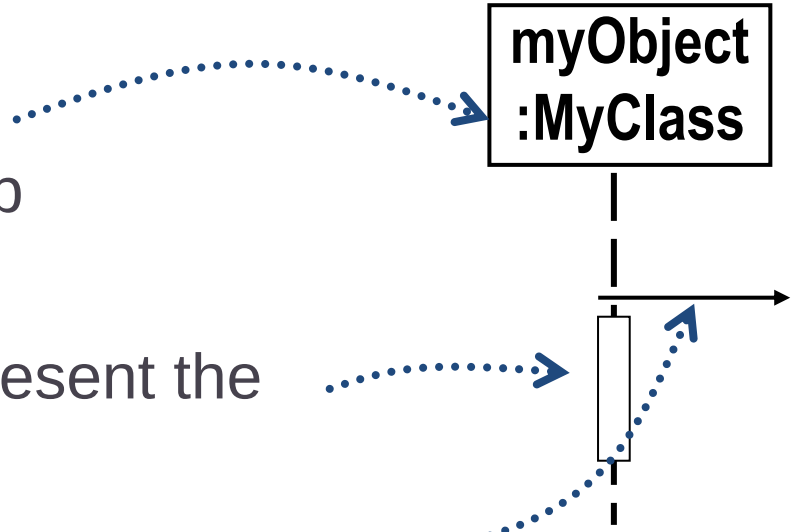
Beginning of Sequence Diagram for *Check Out* Use Case



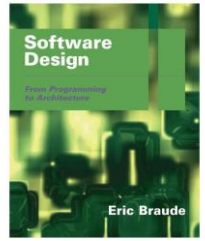
Building a Sequence Diagram 1



1. Identify the use case whose sequence diagram you will build (if applicable).
2. Identify which entity initiates the use case
   - the user, or
   - an object of a class
     - name the class
     - name the object
3. Draw a rectangle to represent this object at left top
   - use UML *object:Class* notation
4. Draw an elongated rectangle beneath this to represent the execution of an operation
5. Draw an arrow pointing right from its top



Building a Sequence Diagram 2

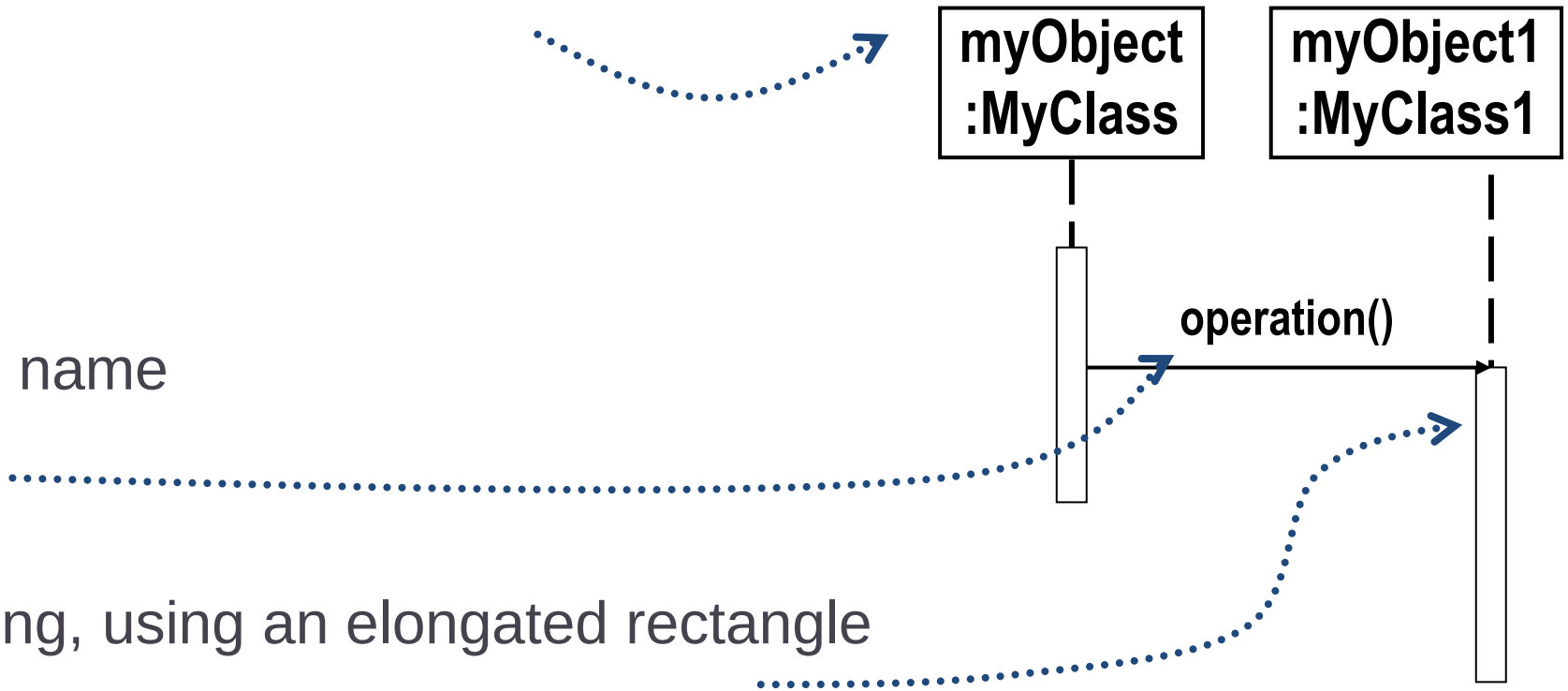


6. Identify which entity handles the operation initiated
   - an object of a class
     - name the class
     - name the object

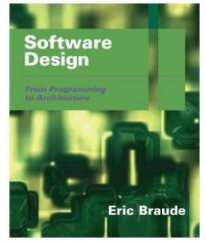
7. Label the arrow with the name of the operation

8. Show a process beginning, using an elongated rectangle

9…… Continue with each new statement of the use case.

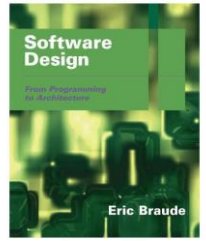


State Models



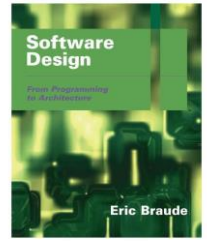
- ✧ The state of an application is its situation or status
- ✧ A state of an object is defined by the values of the object's variables
- ✧ States and substates are denoted by rounded rectangles
- ✧ Event --- something whose occurrence is sensed directly by objects of the class in question.
- ✧ Transitions
  - An event may cause an object to transition from its current state to another state
  - Transition is denoted by a labeled arrow
- ✧ State transition diagram (also known as state charts)

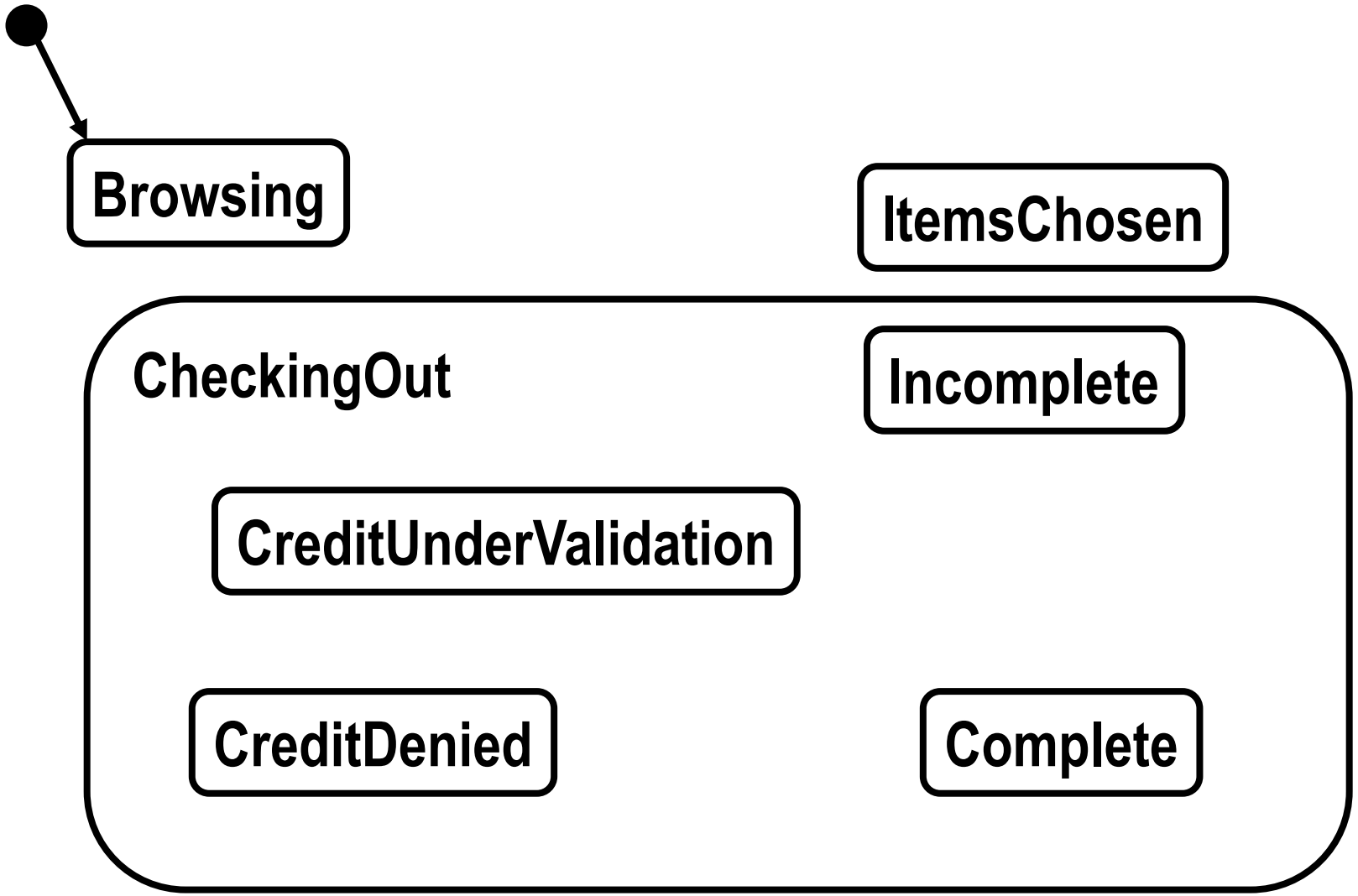
State Diagrams



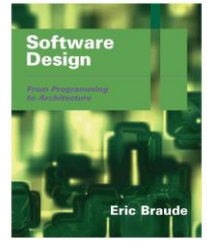
- Some applications or parts thereof are conveniently thought of as being in one of several possible states.
- UML state diagrams help us visualize these, and the events that cause transitions among them.

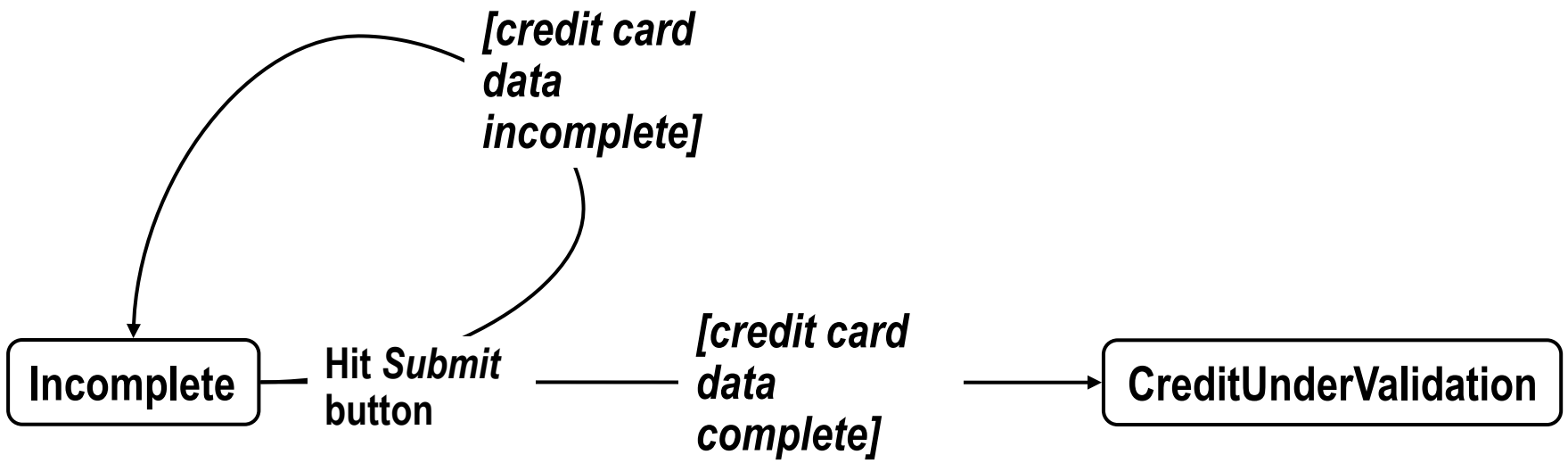
States for OnlineShopper Class



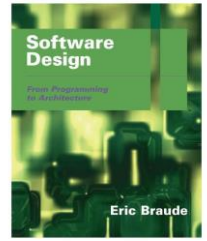


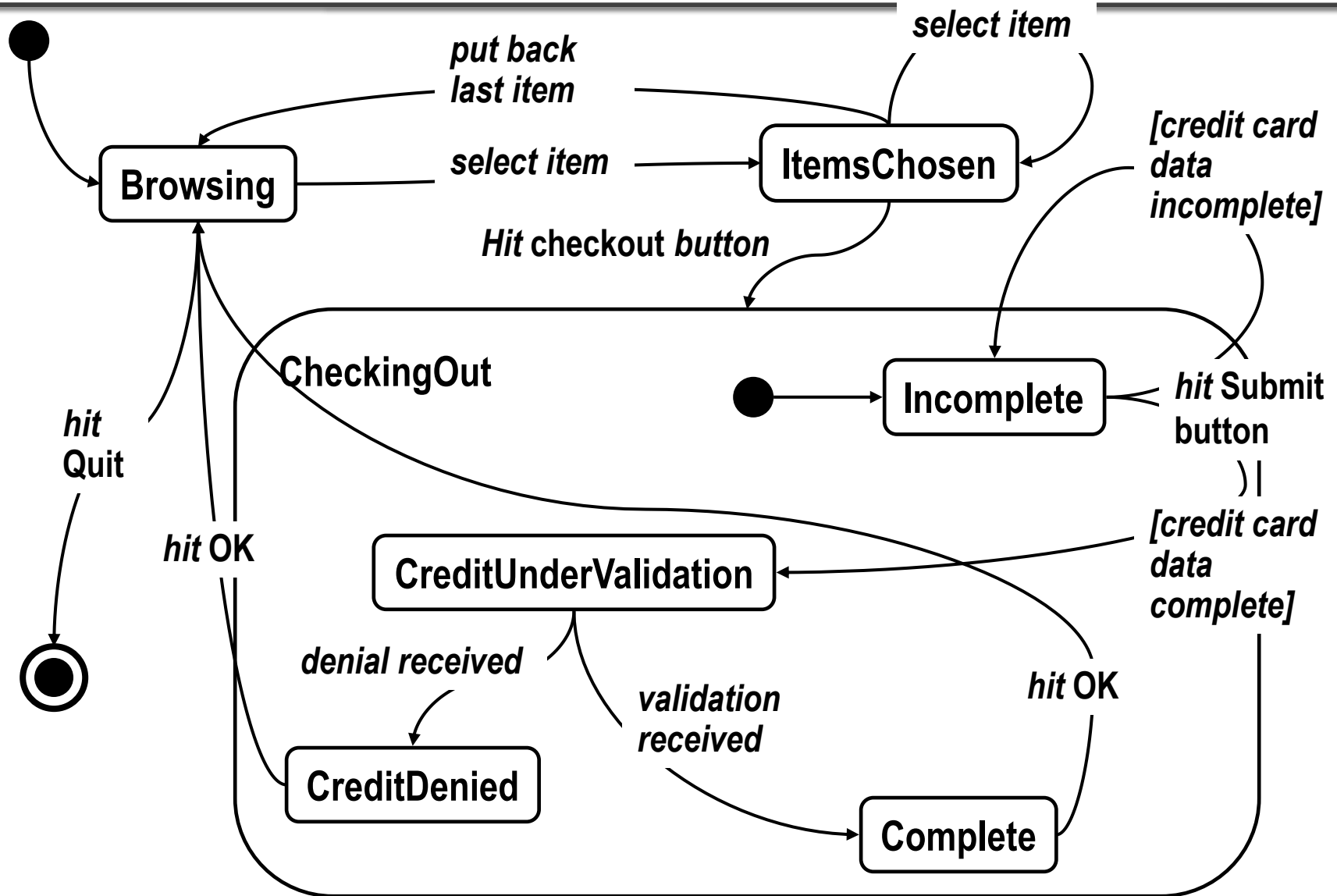
Conditions on Events



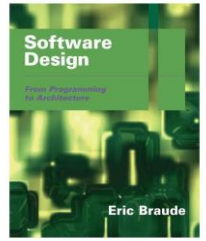


State/Transition Diagram for OnlineShopper Class

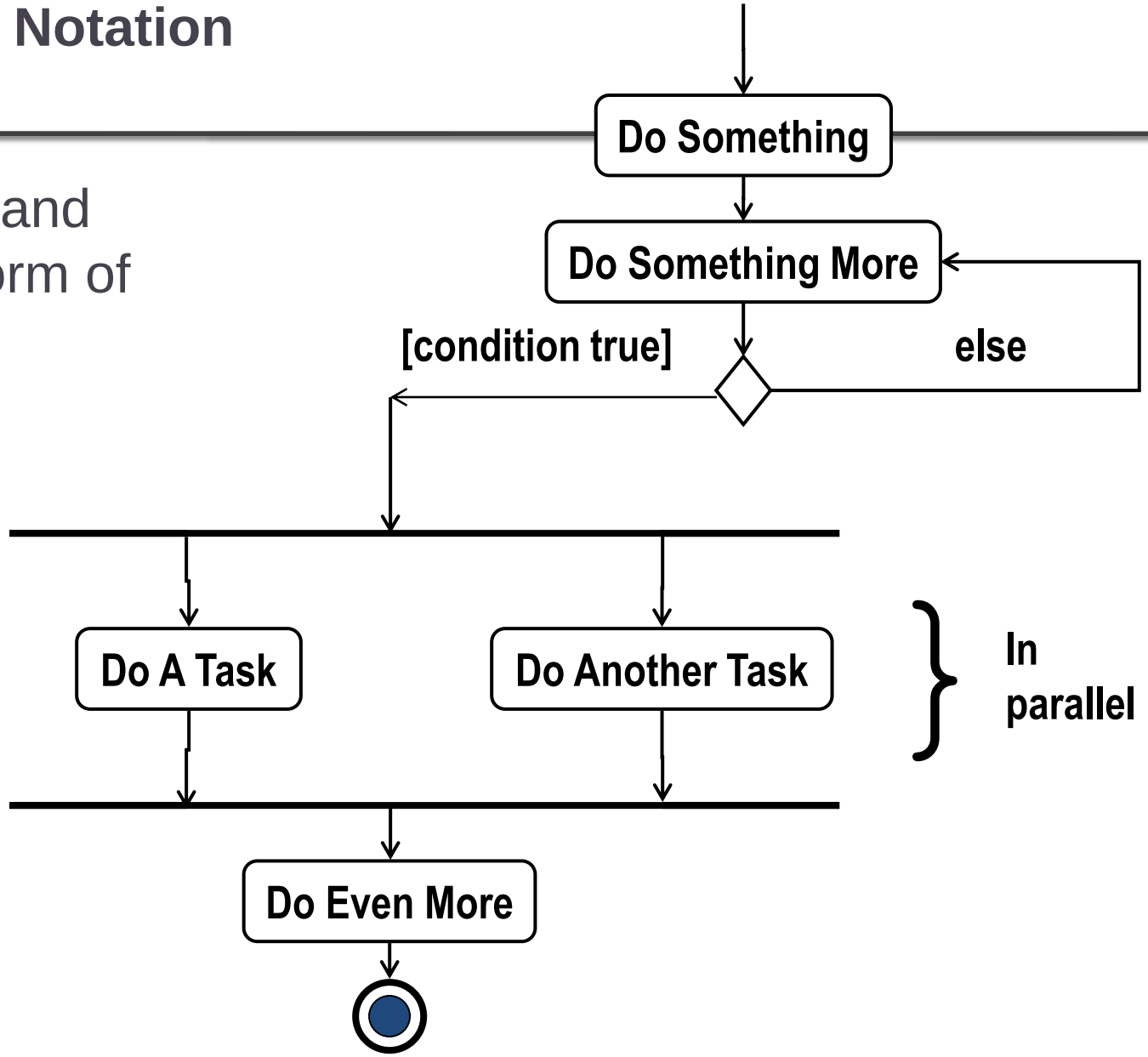




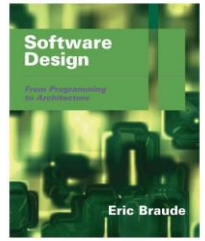
Activity Chart Notation

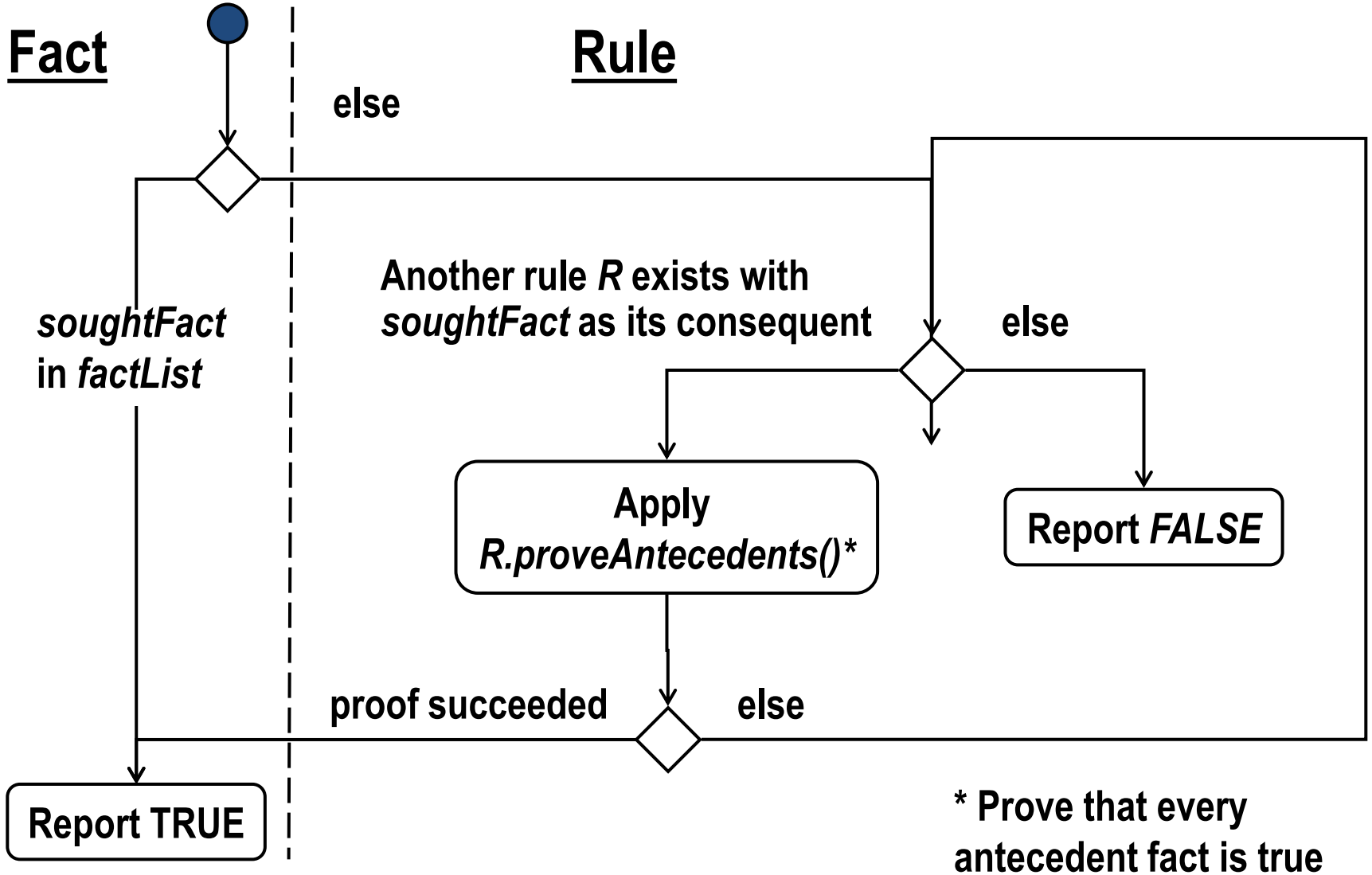


- An updated and enhanced form of flowcharts

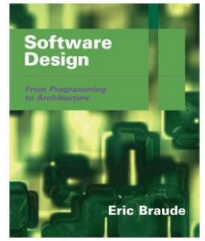


Flowchart for soughtFact.proveBack()



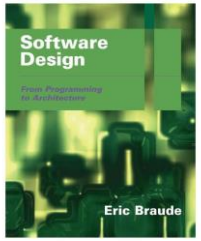


Summary of UML Models



- Use Cases
  - Actor / application interactions
- Sequence Diagrams
  - Objects
  - Sequence of methods
    - calling methods among objects
- Class Models
- Activity Diagrams
  - Flow of control
- State Diagrams
  - States
  - Transitions among states
    - caused by events

Summary :Relationships Between Classes



- Dependency
  - member method mentions another class
- Association
  - structural
  - e.g., *sale / receipt*
  - Aggregation
    - common kind of association
    - e.g. *airplane / wing*
    - *one-way*
    - “has-a”
- Inheritance
  - “is-a”