Modeling Behavior

Alexandre Bergel
abergel@dcc.uchile.cl
22/04/2010

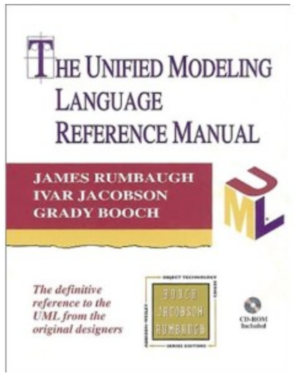
Roadmap

1. Use Case Diagrams
2. Sequence Diagrams
3. Collaboration (Communication) Diagrams
4. Statechart Diagrams
5. Using UML

Source

The Unified Modeling Language Reference Manual

James Rumbaugh, Ivar Jacobson and Grady Booch, Addison Wesley, 1999.



Roadmap

1. **Use Case Diagrams**
2. Sequence Diagrams
3. Collaboration (Communication) Diagrams
4. Statechart Diagrams
5. Using UML

Use Case Diagrams

A use case is a *generic description of an entire transaction* involving several actors.

A use case diagram presents a *set of use cases* (ellipses) and the external actors that interact with the system. *Dependencies* and *associations* between use cases may be indicated.

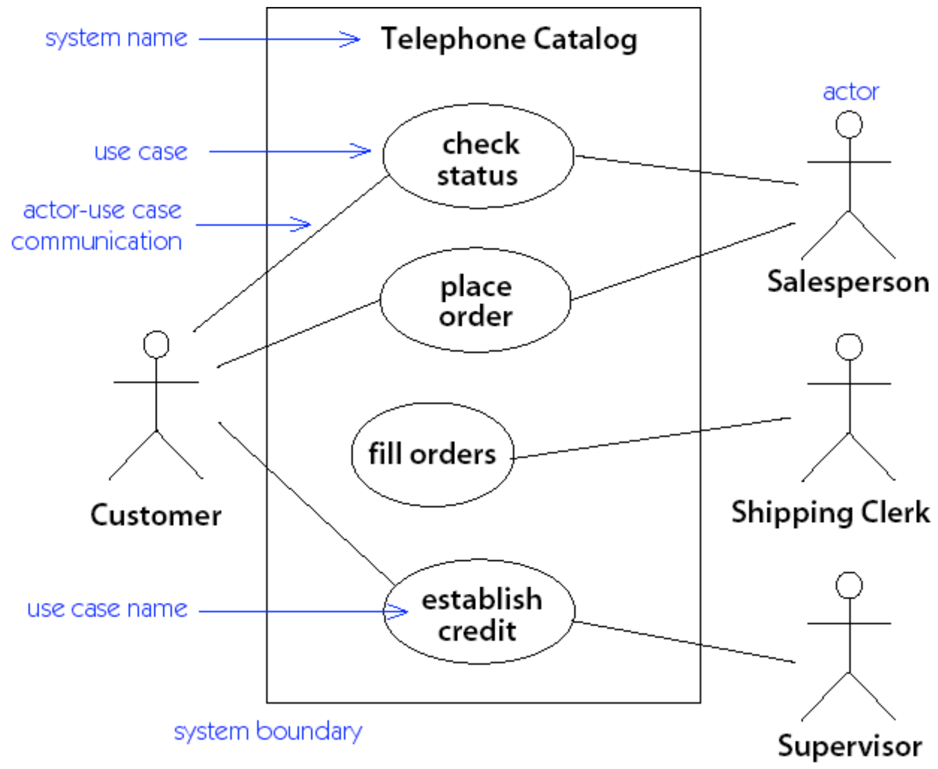


Figure 5-1. *Use case diagram*

Using Use Case Diagrams

"A use case is a *snapshot of one aspect* of your system. The sum of all use cases is *the external picture* of your system …"

UML Distilled

"As use cases appear, assess their impact on the domain model."

Use cases can *drive domain modeling* by highlighting the important concepts.

Roadmap

1. Use Case Diagrams
2. **Sequence Diagrams**
3. Collaboration (Communication) Diagrams
4. Statechart Diagrams
5. Using UML

Scenarios

A scenario is an *instance of a use case* showing a typical example of its execution

Scenarios can be presented in UML using either *sequence diagrams or collaboration diagrams*

Note that a scenario only describes an example of a use case, so conditionality cannot be expressed!

Sequence Diagrams

A sequence diagram depicts a scenario by showing the interactions among a set of objects in *temporal order*.

Objects (not classes!) are shown as *vertical bars*. *Events* or message dispatches are shown as horizontal (or slanted) *arrows* from the sender to the receiver.

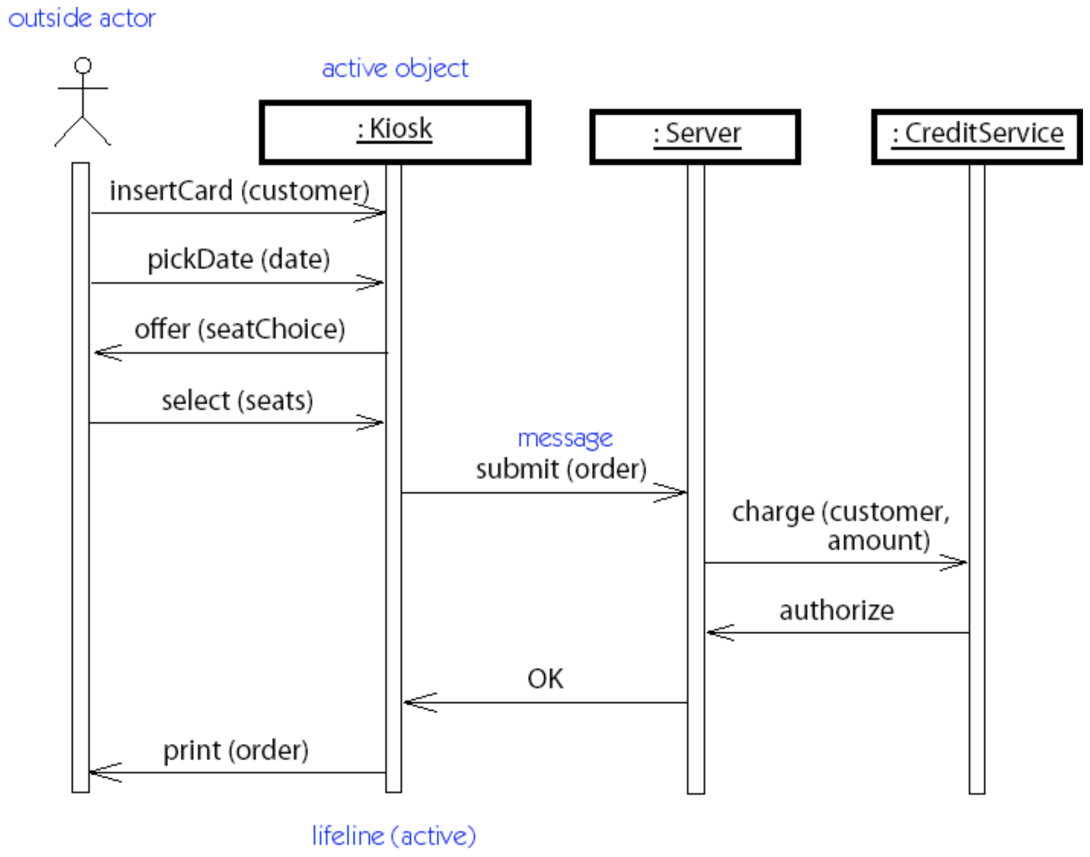


Figure 8-1. *Sequence diagram*

Activations

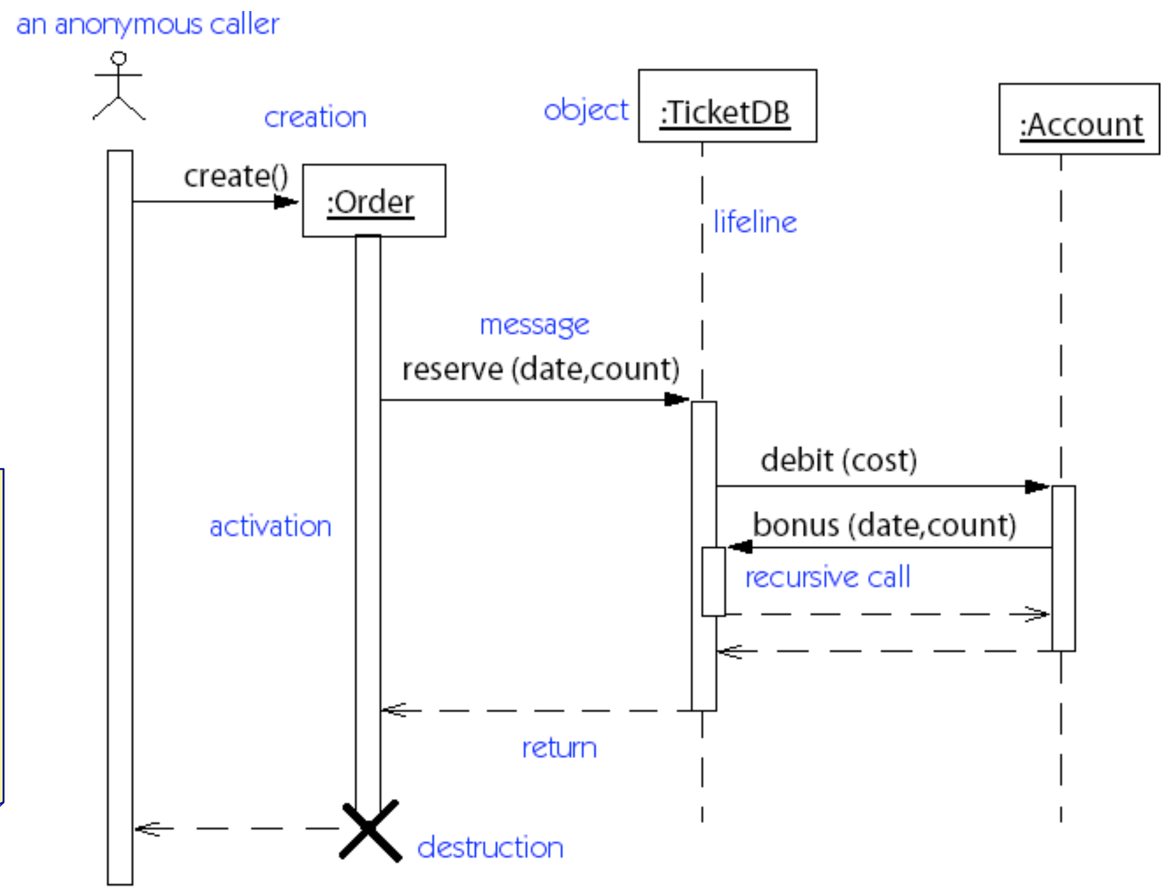


Figure 8-2. *Sequence diagram with activations*

Avoid returns in sequence diagrams unless they add clarity.

Asynchrony and Constraints

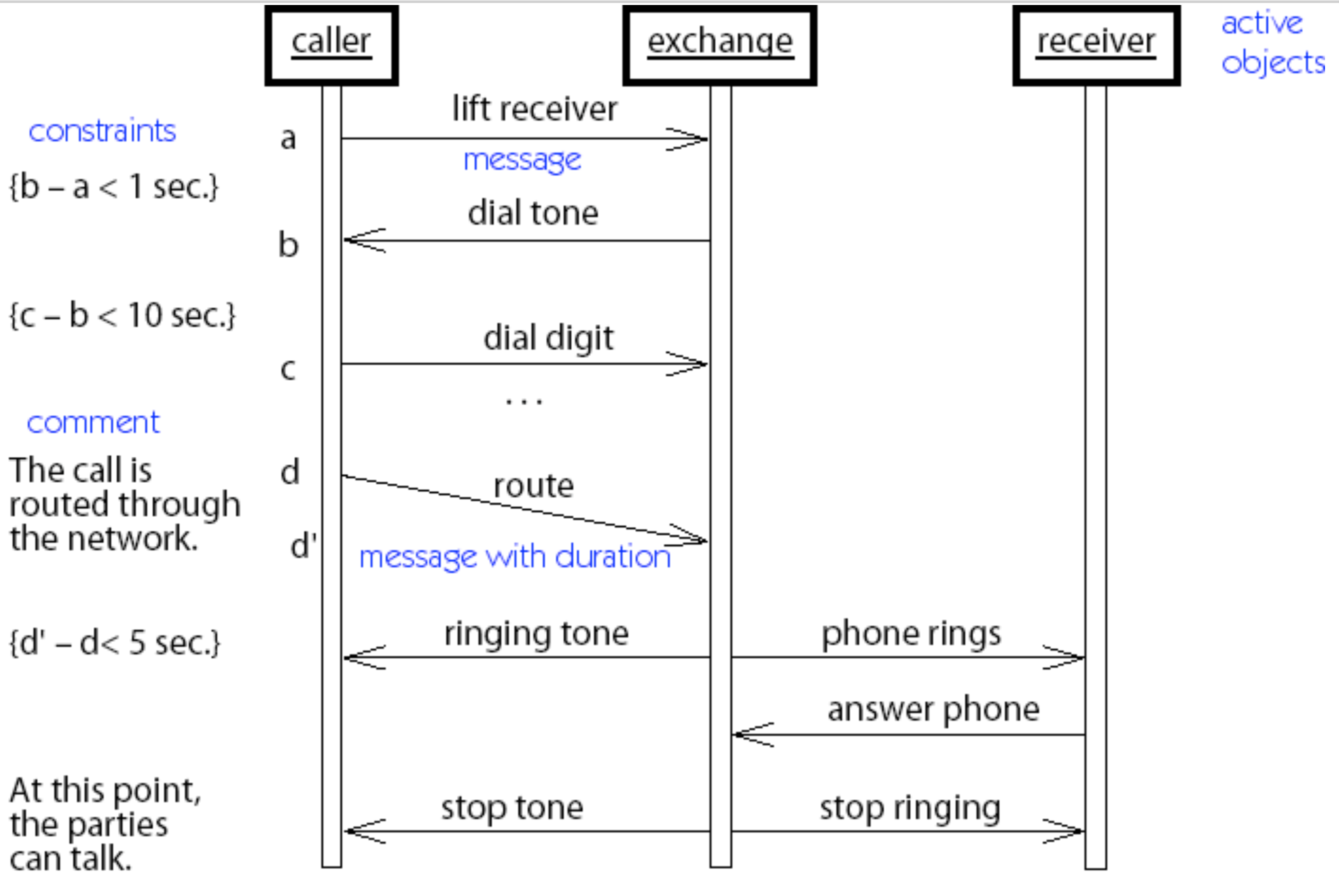


Figure 13-161. *Sequence diagram with asynchronous control*

Roadmap

1. Use Case Diagrams
2. Sequence Diagrams
3. **Collaboration (Communication) Diagrams**
4. Statechart Diagrams
5. Using UML

Collaboration Diagrams

Collaboration diagrams (called Communication diagrams in UML 2.0) depict scenarios as *flows of messages* between objects:

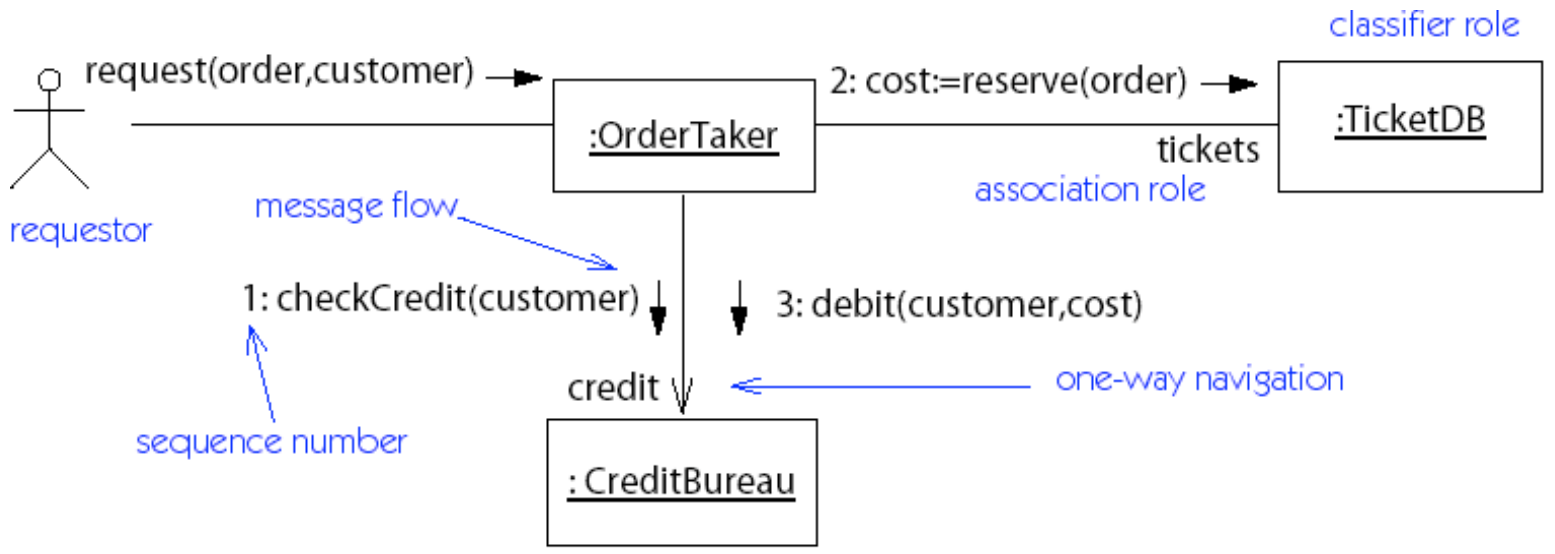


Figure 8-3. *Collaboration diagram*

Message Labels...

Messages from one object to another are labelled with text strings showing the *direction* of message flow and information indicating the message *sequence*.

Prior messages from other threads (e.g. “[A1.3, B6.7.1]”) -- only needed with concurrent flow of control

- Dot-separated list of sequencing elements

Message Labels...

- Dot-separated list of sequencing elements
  - sequencing integer (e.g., “3.1.2” is invoked by “3.1” and follows “3.1.1”)
  - letter indicating concurrent threads (e.g., “1.2a” and “1.2b”)
  - iteration indicator (e.g., “1.1*[i=1..n]”)
  - conditional indicator (e.g., “2.3 [#items = 0]”)
- Return value binding (e.g., “status :=”)
- Message name (event or operation name)
- Argument list

Nested Message Flows

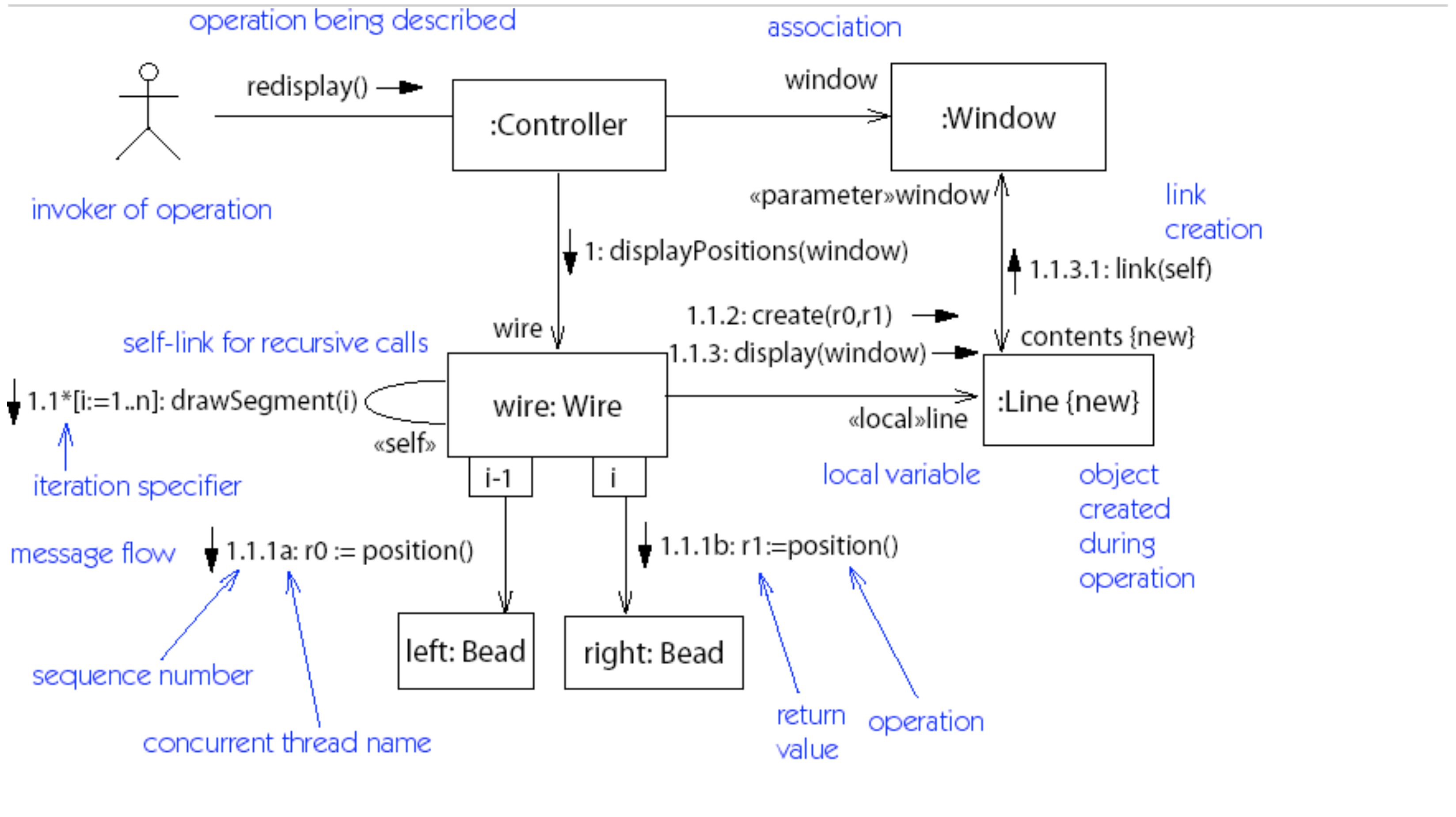


Figure 13-51. *Collaboration diagram with message flows*

Roadmap

1. Use Case Diagrams
2. Sequence Diagrams
3. Collaboration (Communication) Diagrams
4. **Statechart Diagrams**
5. Using UML

Statechart Diagrams

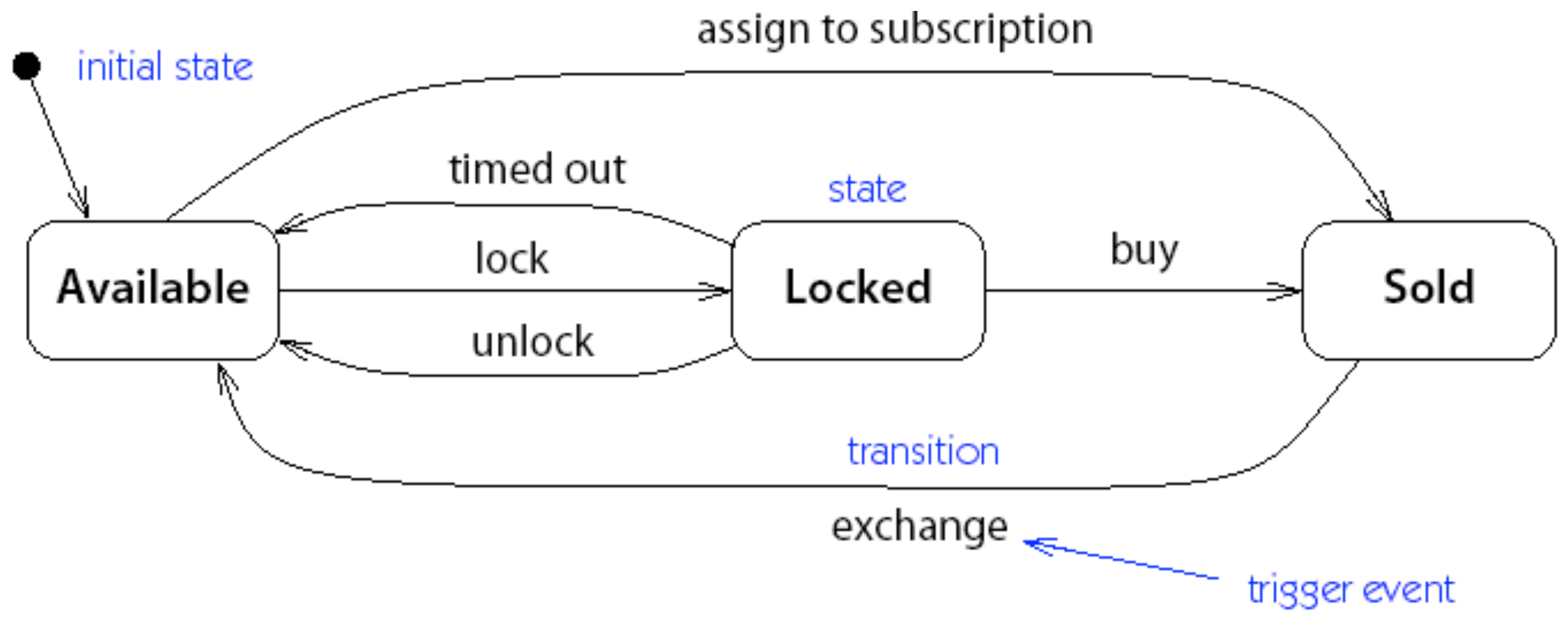


Figure 3-5. *Statechart diagram*

Statechart Diagram Notation...

A Statechart Diagram describes the *temporal evolution* of an object of a given class in response to *interactions* with other objects inside or outside the system.

An event is a one-way (asynchronous) communication from one object to another:

- *atomic* (non-interruptible)
- includes events from *hardware* and real-world objects e.g., message receipt, input event, elapsed time, ...
- notation: eventName(parameter: type, ...)
- may cause object to make a *transition* between states

Statechart Diagram Notation

A state is a period of time during which an object is waiting for an event to occur:

- depicted as rounded box with (up to) three sections:
  - 1 - name — optional
  - 2 - state variables — name: type = value (valid only for that state)
  - 3 - triggered operations — internal transitions and ongoing operations
- may be nested

State Box with Regions

The *entry event* occurs whenever a transition is made into this state, and the *exit operation* is triggered when a transition is made out of this state.

The *help* and *character* events cause internal transitions with no change of state, so the entry and exit operations are not performed.

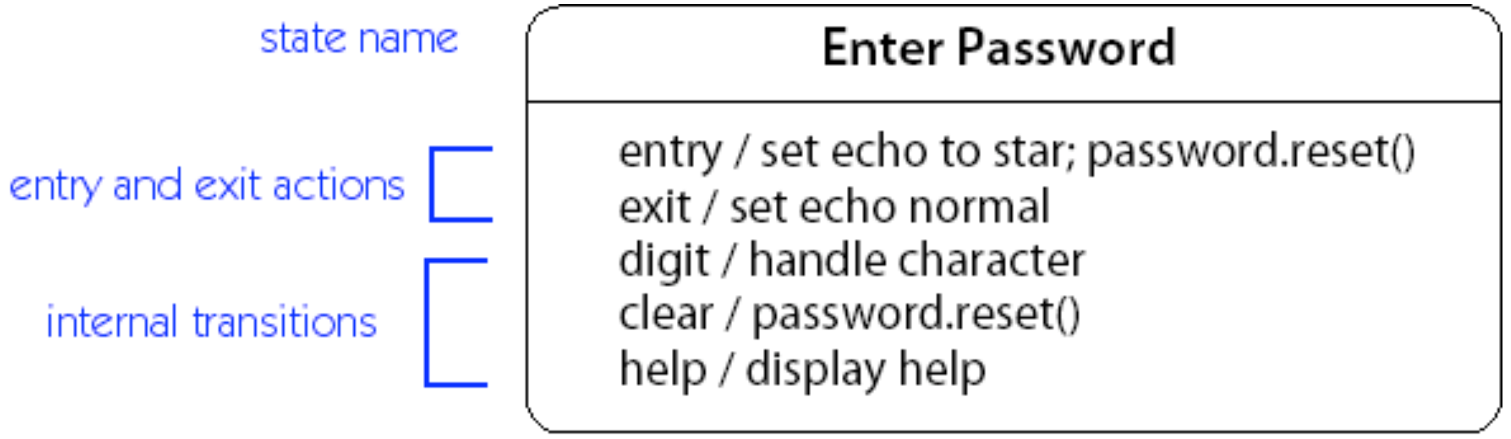


Figure 6-4. *Internal transitions, and entry and exit actions*

Transitions

A transition is an response to an external event received by an object in a given state

May *invoke* an operation, and cause the object to change state

May *send* an event to an external object

Transition syntax (each part is optional):

```
event(arguments) [condition]
/ ^target.sendEvent operation(arguments)
```

External transitions label arcs between states

Internal transitions are part of the triggered operations of a state

Operations and Activities

An operation is an *atomic action* invoked by a transition

- *Entry and exit operations* can be associated with states

An activity is an *ongoing operation* that takes place while object is in a given state

Modeled as “internal transitions” labelled with the pseudo-event do

Roadmap

1. Use Case Diagrams
2. Sequence Diagrams
3. Collaboration (Communication) Diagrams
4. Statechart Diagrams
5. **Using UML**

Perspectives

Three perspectives in drawing UML diagrams:

1 - Conceptual

- Represent domain concepts
- Ignore software issues

2 - Specification

- Focus on visible interfaces and behaviour
- Ignore internal implementation

3 - Implementation

- Document implementation choices
- Most common, but least useful perspective(!)

Using the Notations

The diagrams introduced here complement class and object diagrams.

During Analysis:

- Use case, sequence and collaboration diagrams document use cases and their scenarios during requirements specification

During Design:

- Sequence and collaboration diagrams can be used to document implementation scenarios or refine use case scenarios
- State diagrams document internal behaviour of classes and must be validated against the specified use cases

What you should know!

What is the purpose of a use case diagram?

Why do scenarios depict objects but not classes?

How can timing constraints be expressed in scenarios?

How do you specify and interpret message labels in a scenario?

How can you model interaction between state diagrams for several classes?

Can you answer the following questions?

Can a sequence diagram always be translated to an collaboration diagram?

Or vice versa?

Why are arrows depicted with the message labels rather than with links?

When should you use concurrent substates?

License