S	W	Name ↓	Last Success	Last Failure	Last Duration	
		ese-2013-team1	N/A	7 hr 17 min - #7	3 ms	
		ese-2013-team2	N/A	18 hr - #9	4 ms	
		ese-2013-team3	14 hr - #15	2 days 16 hr - #7	2 min 14 sec	
		ese-2013-team4	1 day 18 hr - #13	18 hr - #14	2 min 3 sec	
		ese-2013-team5	N/A	5 hr 3 min - #7	4 ms	
		ese-2013-team6	20 min - #14	3 days 0 hr - #9	5 min 0 sec	
		ese-2013-team7	N/A	23 hr - #9	41 ms	
		ese-2013-team8	13 hr - #15	2 days 13 hr - #13	2 min 16 sec	
		ese-2013-team9	N/A	21 hr - #10	2 min 11 sec	
		PomodoroBox	6 hr 45 min - #36	N/A	2 min 18 sec	

UNIVERSITÄT BERN

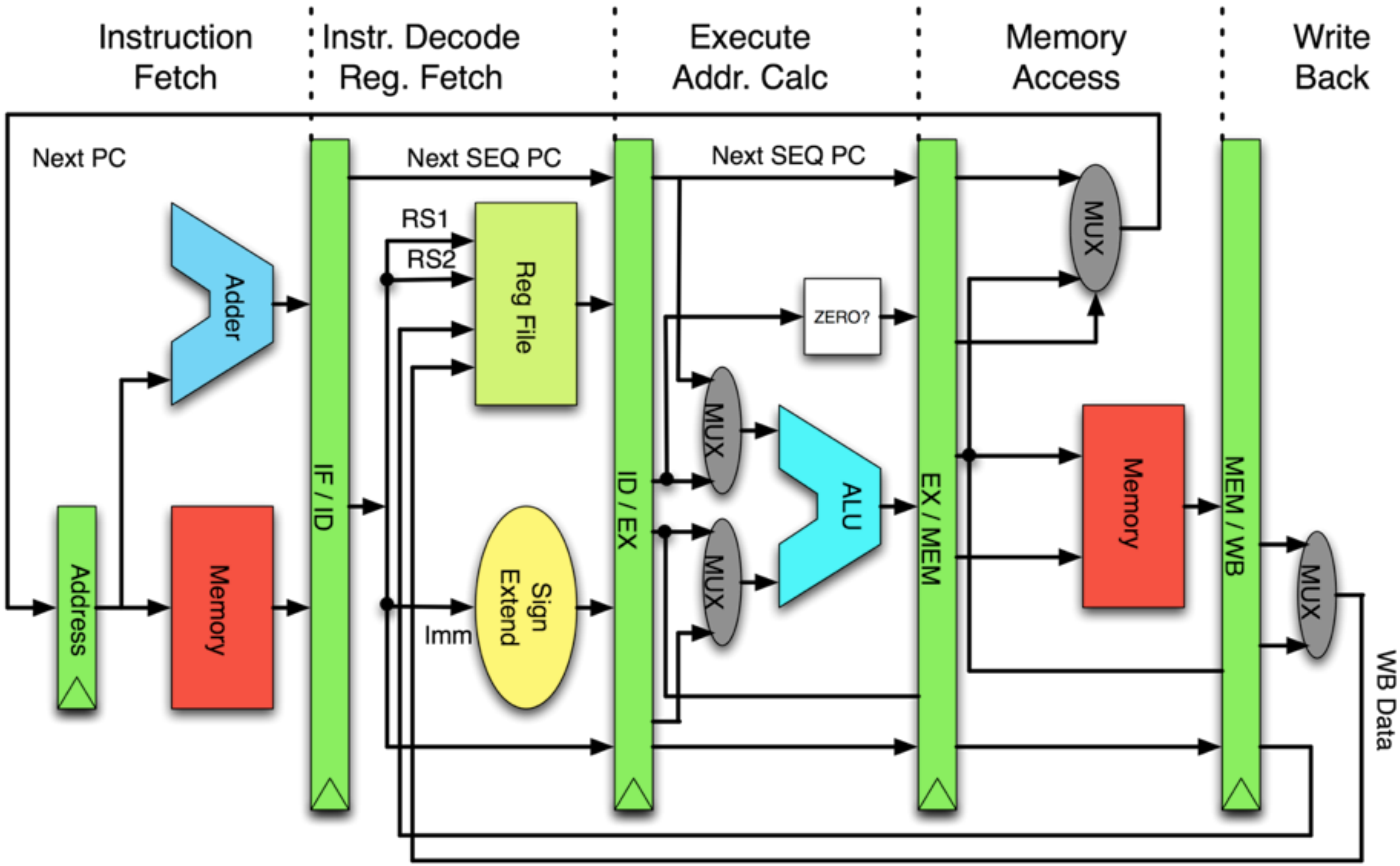
Introduction to Software Engineering

8. UML

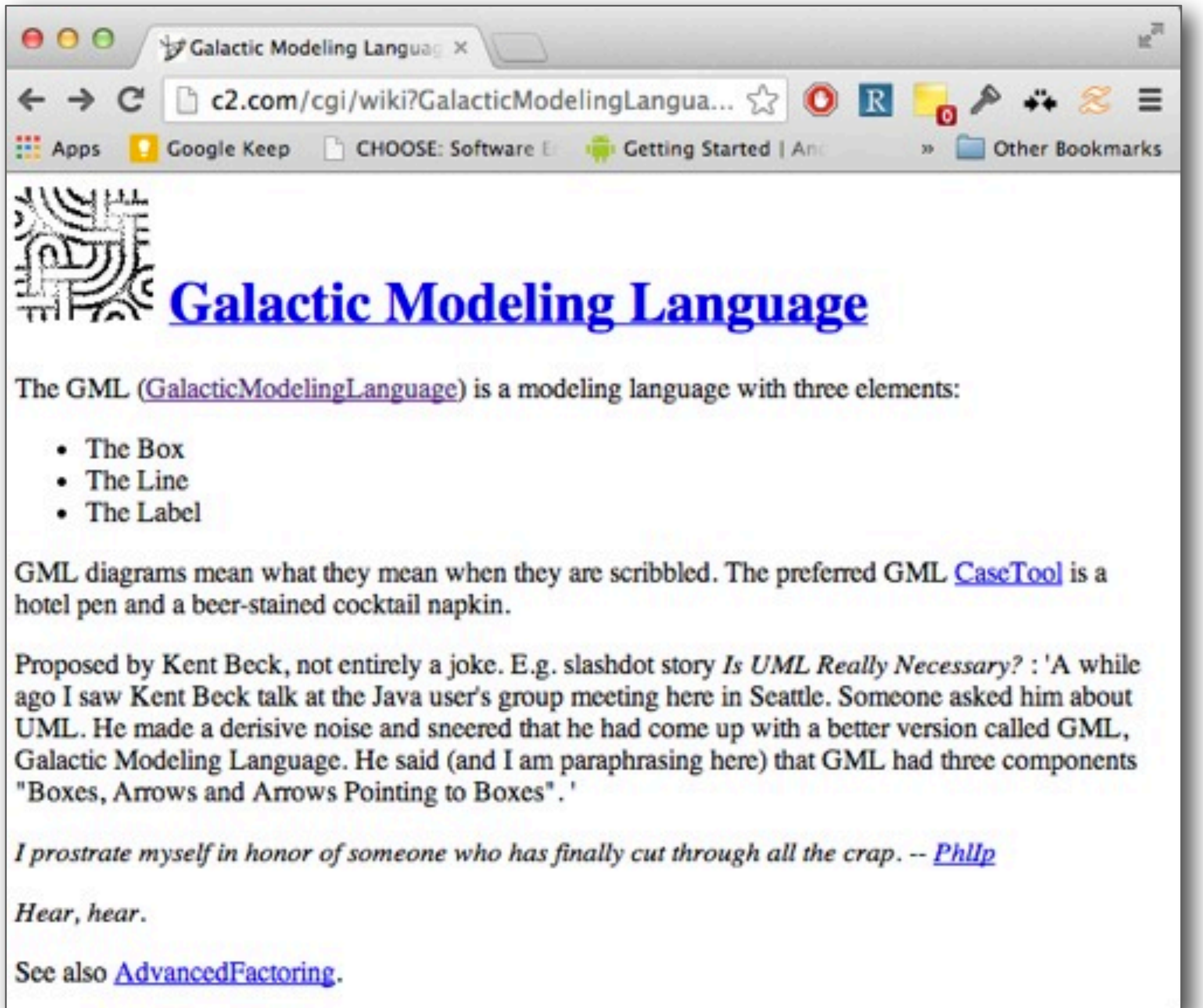
Mircea F. Lungu

Based on a lecture by Oscar Nierstrasz.


FRANCE
Chauvet Cave



Instruction Fetch
Instr. Decode Reg. Fetch
Execute Addr. Calc
Memory Access
Write Back
Next PC
Next SEQ PC
Next SEQ PC
RS1
RS2
Reg File
Adder
MUX
ZERO?
MUX
MUX
ALU
Memory
Memory
Address
IF / ID
ID / EX
EX / MEM
MEM / WB
MUX
Sign Extend
Imm
WB Data


Galactic Modeling Languag ×

c2.com/cgi/wiki?GalacticModelingLangua...

Apps | Google Keep | CHOOSE: Software E | Getting Started | An | » | Other Bookmarks

Galactic Modeling Language

The GML (GalacticModelingLanguage) is a modeling language with three elements:

- The Box
- The Line
- The Label

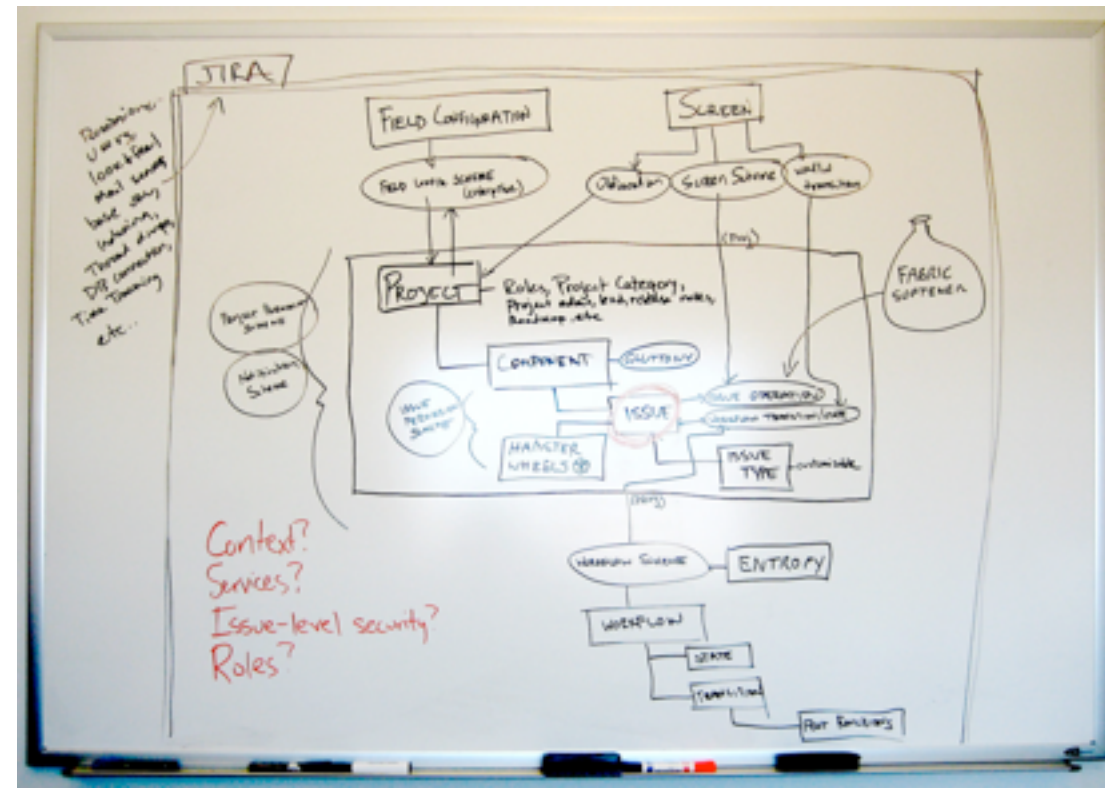
GML diagrams mean what they mean when they are scribbled. The preferred GML CaseTool is a hotel pen and a beer-stained cocktail napkin.

Proposed by Kent Beck, not entirely a joke. E.g. slashdot story *Is UML Really Necessary?* : 'A while ago I saw Kent Beck talk at the Java user's group meeting here in Seattle. Someone asked him about UML. He made a derisive noise and sneered that he had come up with a better version called GML, Galactic Modeling Language. He said (and I am paraphrasing here) that GML had three components "Boxes, Arrows and Arrows Pointing to Boxes".'

I prostrate myself in honor of someone who has finally cut through all the crap. -- PhlIp

Hear, hear.

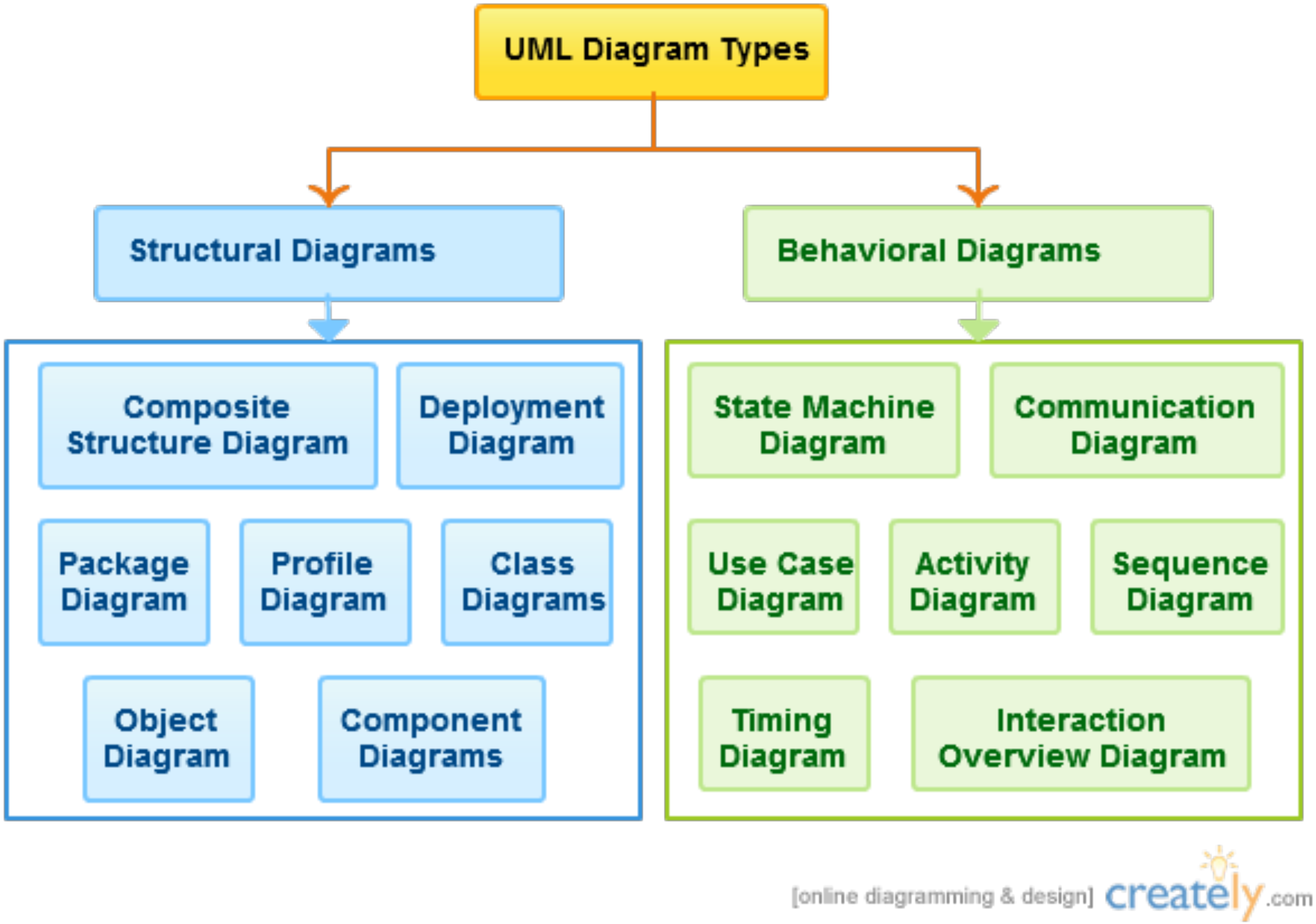
See also AdvancedFactoring.



Who knows UML?

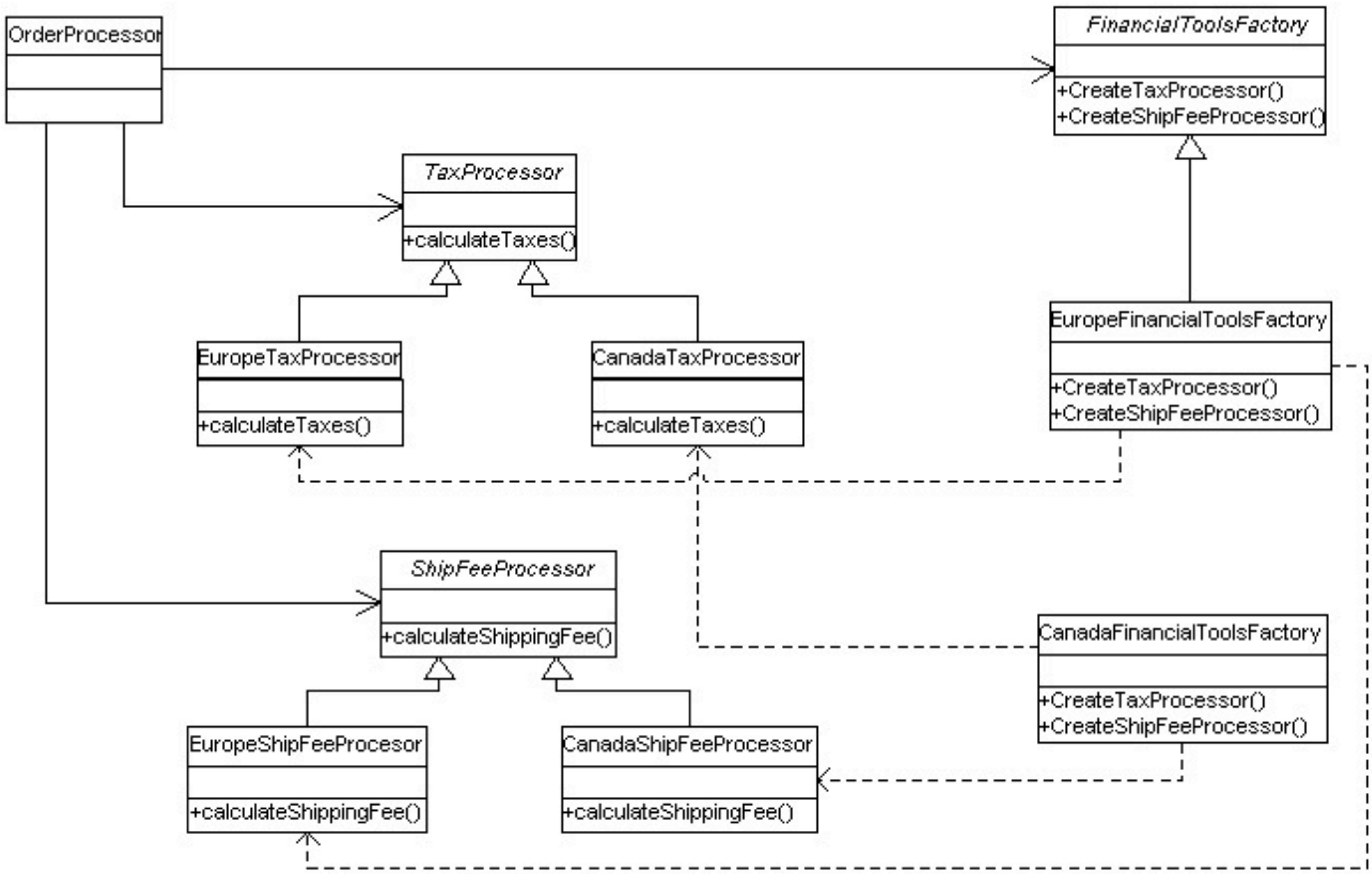
Who can name more than 3 types of diagrams?

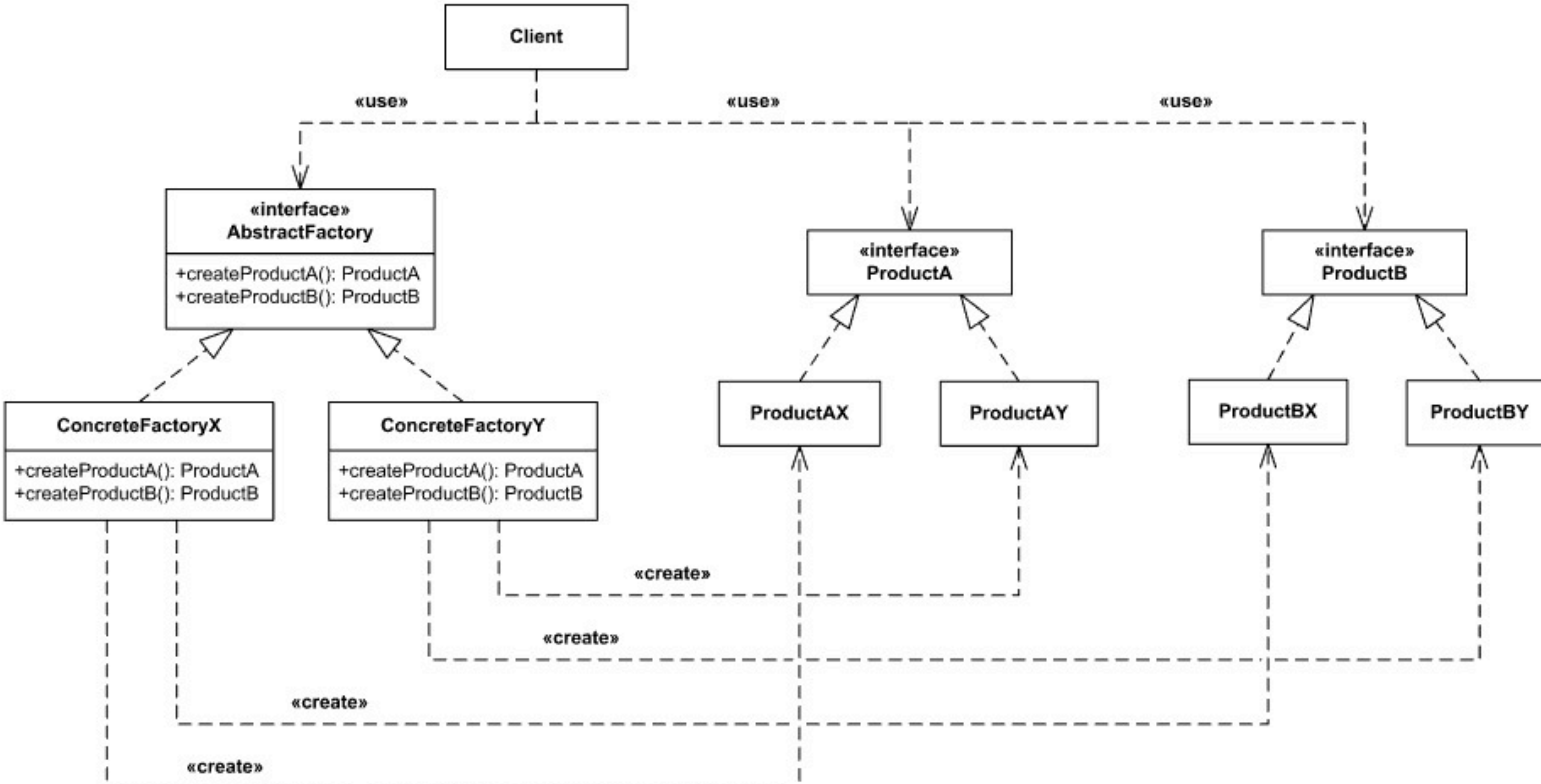
1. ...
2.

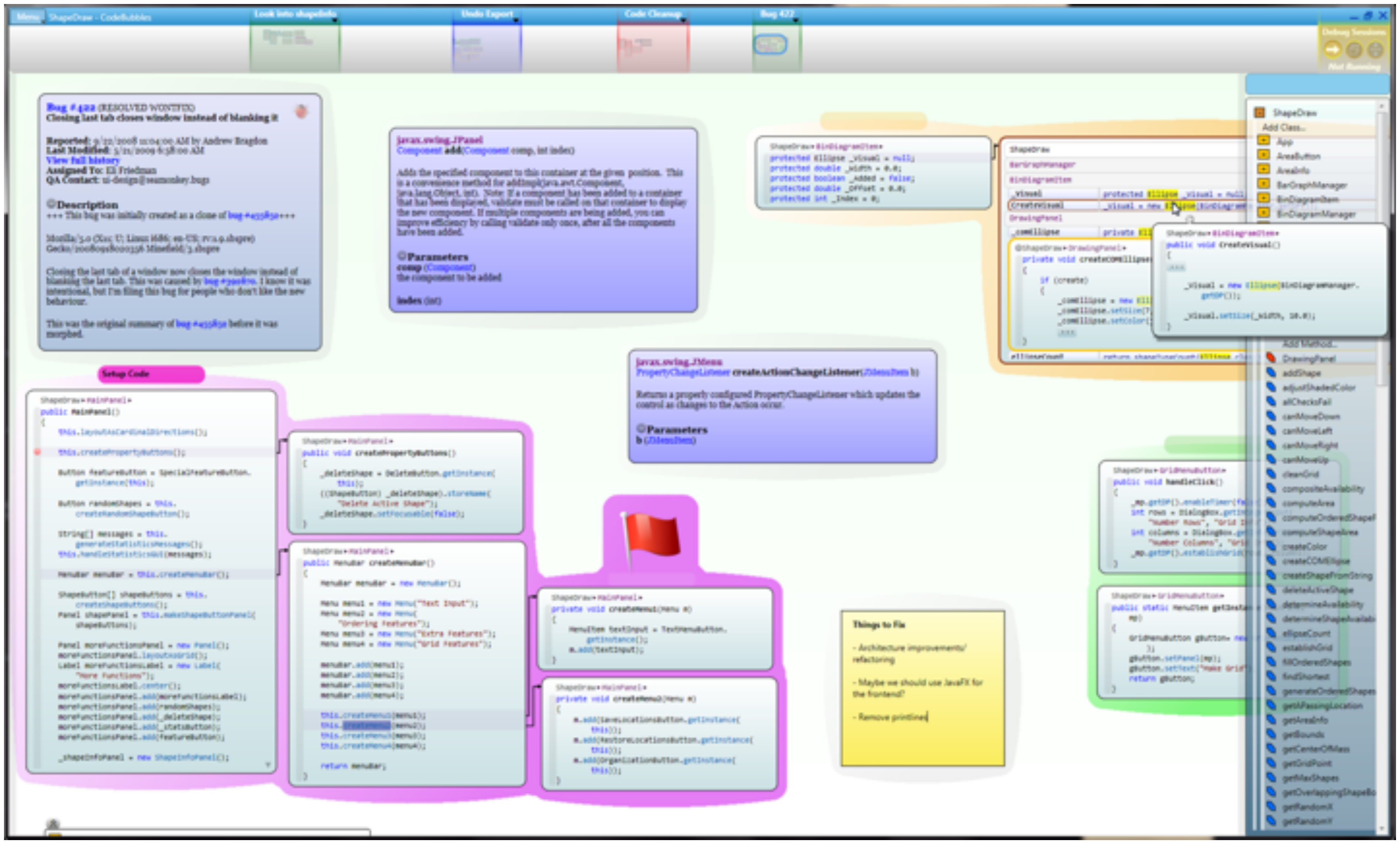

UML Diagram Types
Structural Diagrams
Behavioral Diagrams
Composite Structure Diagram
Deployment Diagram
Package Diagram
Profile Diagram
Class Diagrams
Object Diagram
Component Diagrams
State Machine Diagram
Communication Diagram
Use Case Diagram
Activity Diagram
Sequence Diagram
Timing Diagram
Interaction Overview Diagram
[online diagramming & design] creately.com

UML Class Diagram Quiz!

CD-Q1. What design pattern is this?




Client
«use»
«use»
«use»
«interface»
AbstractFactory
+createProductA(): ProductA
+createProductB(): ProductB
«interface»
ProductA
«interface»
ProductB
ConcreteFactoryX
+createProductA(): ProductA
+createProductB(): ProductB
ConcreteFactoryY
+createProductA(): ProductA
+createProductB(): ProductB
ProductAX
ProductAY
ProductBX
ProductBY
«create»
«create»
«create»
«create»



CodeBubbles also allows an overview of a system. And it’s executable.

What is UML?

UML

3-in-1

Sketching
Blueprint
Language

UML

What?

> uniform notation: Booch + OMT + Use Cases (+ state charts)

Why?

> Software projects are carried out in *team*

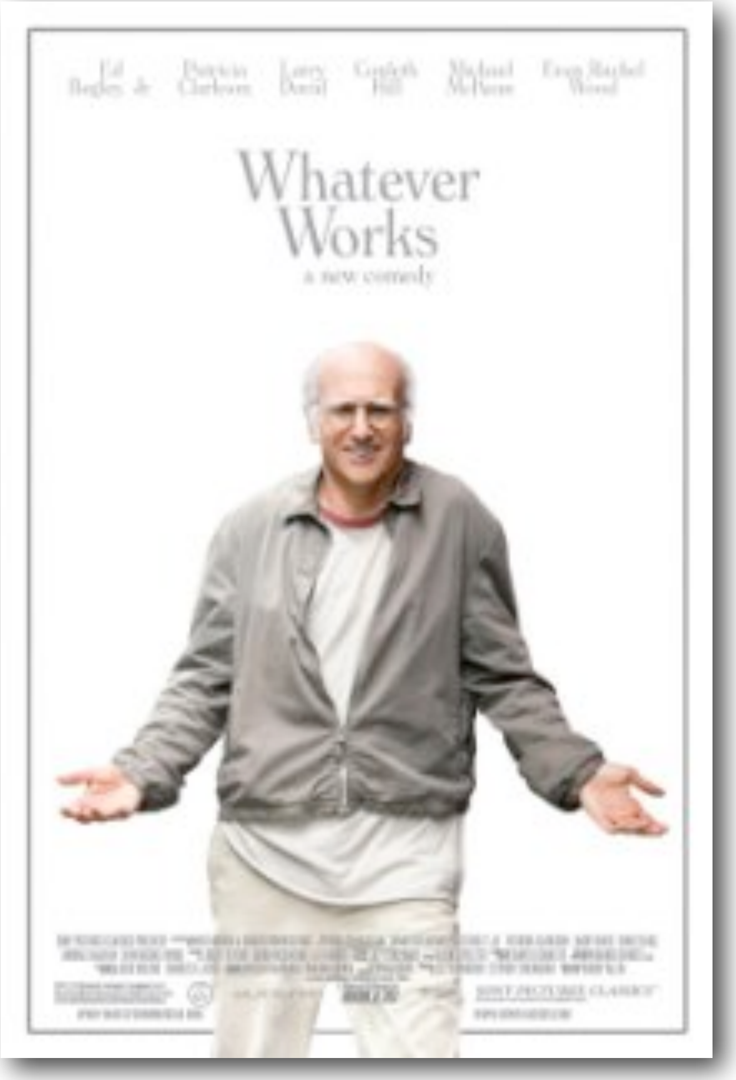
> Team members need to *communicate*
  — ... sometimes even with the end users

> "One picture conveys a thousand words"
  — the question is only *which words*
  — Need for *different views* on the same software artifact

UML in Practice

Marian Petre
Centre for Research in Computing
The Open University
Milton Keynes, UK
m.petre@open.ac.uk

I. Introduction: "Where's the UML?"



Class diagrams 7
Sequence diagrams 6
Activity diagrams 6
State machine diagrams 3

thinking tool
communication tool (with stakeholders)
collaboration tool (dialog with designers)
adaptation (i.e., using a homegrown variant of the "real" notation)
selective traction (i.e., using it just as long as is useful, then moving on)

Roadmap

UML Overview

Structural Diagrams

➡ Classes, attributes and operations

Objects, Associations

Behavioral Diagrams

Sequence

Communication

Activity

State

Further Discussion

Class Diagrams

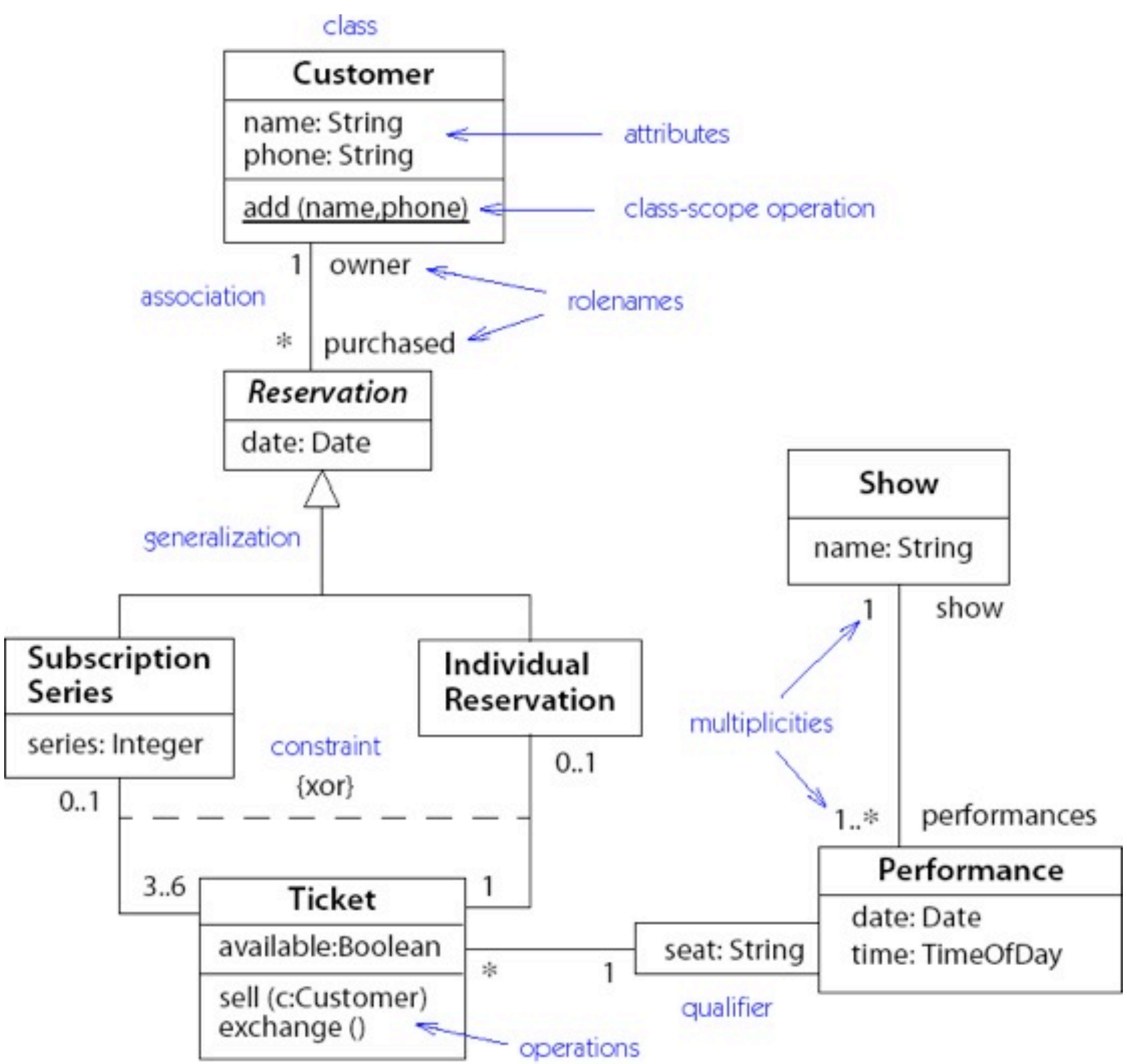
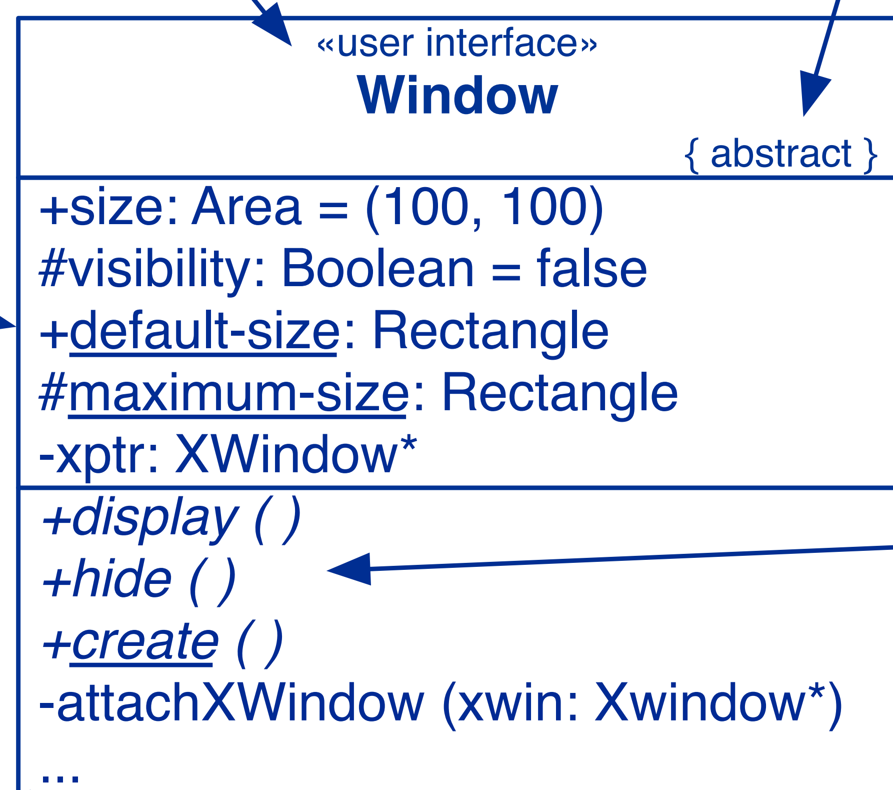


Figure 3-1. *Class diagram*

Visibility and Scope of Features

Stereotype
(what "kind" of class is it?)

User-defined properties
(e.g., readonly, owner = "Pingu")



underlined
attributes have
class scope

italic attributes
are *abstract*

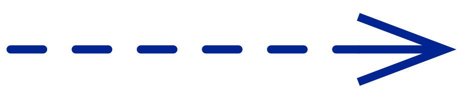
+ = "public"
# = "protected"
- = "private"

An ellipsis signals that further entries are not shown

UML Lines and Arrows

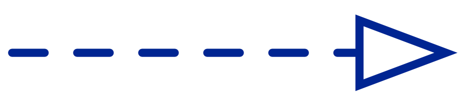
Constraint
(usually annotated)

Association
e.g., «uses»



Dependency
e.g., «requires»,
«imports» ...

Navigable
association
e.g., part-of



Realization
e.g., class/template,
class/interface

"Generalization"
i.e., specialization (!)
e.g., class/superclass,
concrete/abstract class

Aggregation
i.e., "consists of"

"Composition"
i.e., containment

Parameterized Classes

Parameterized (aka "template" or "generic") classes are depicted with their parameters shown in a *dashed box*.

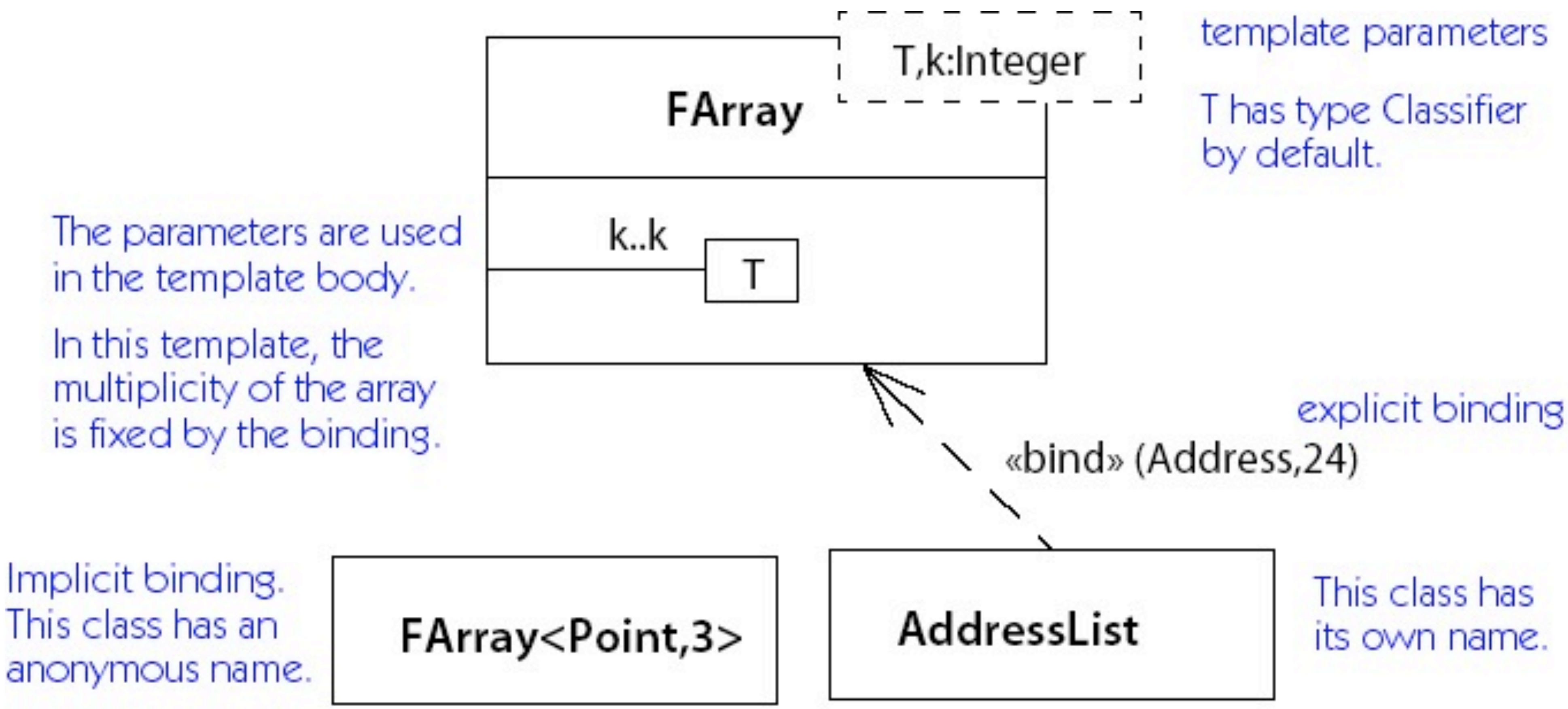


Figure 13-180. *Template notation with use of parameter as a reference*

Interfaces

Interfaces, equivalent to abstract classes with no attributes, are represented as classes with the stereotype «interface» or, alternatively, with the “Lollipop-Notation”:

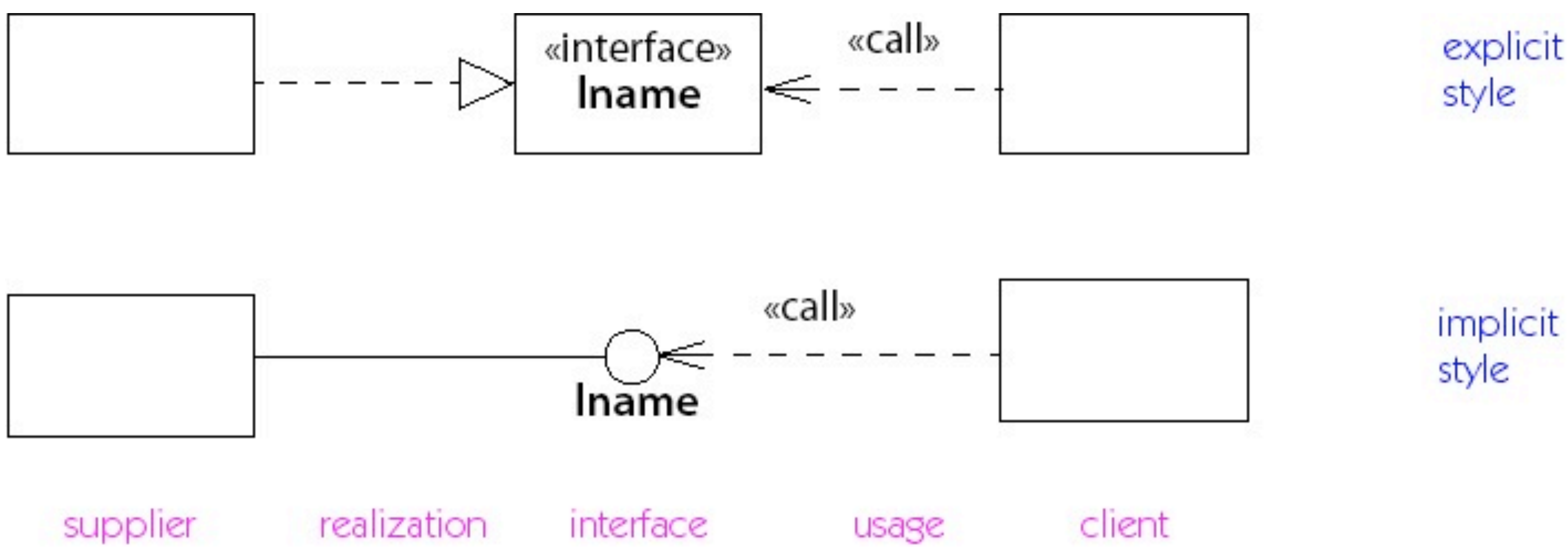


Figure B-5. *Realization of an interface*

Generalization

A subclass specializes its superclass:

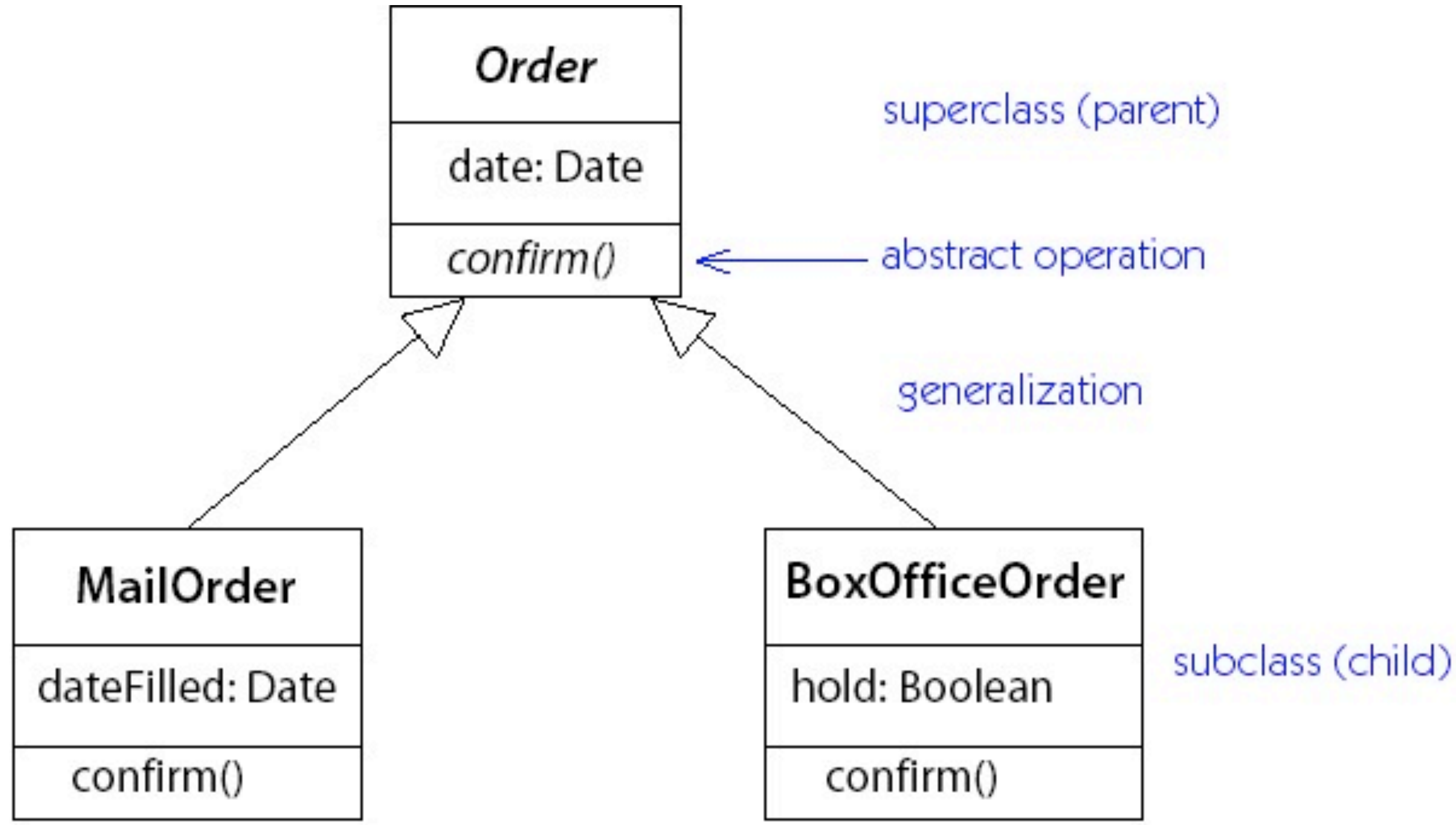
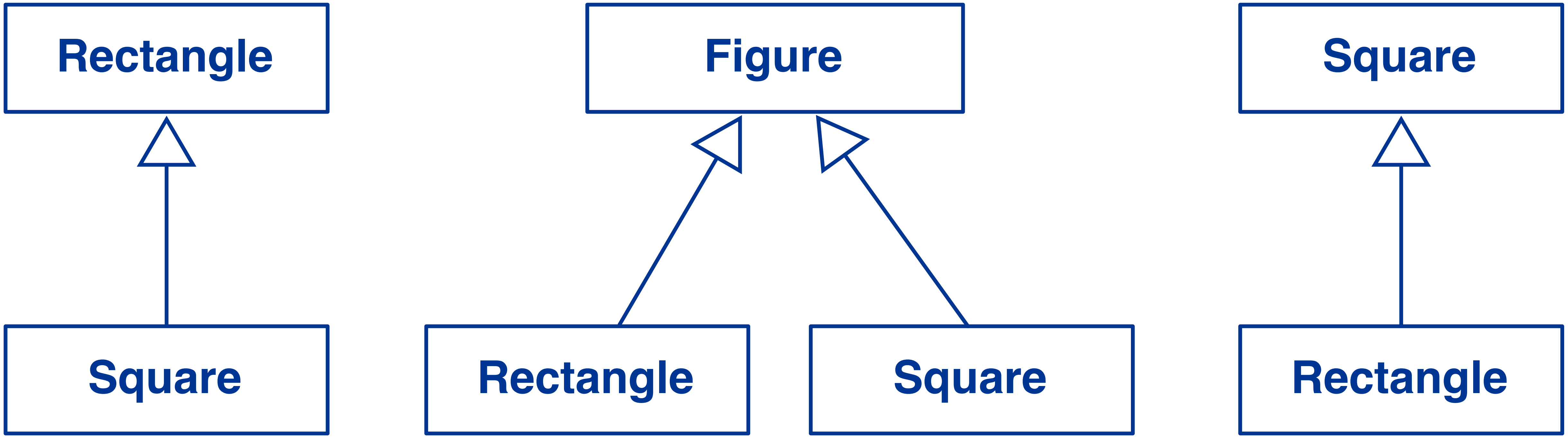


Figure 4-7. *Generalization notation*

The different faces of inheritance

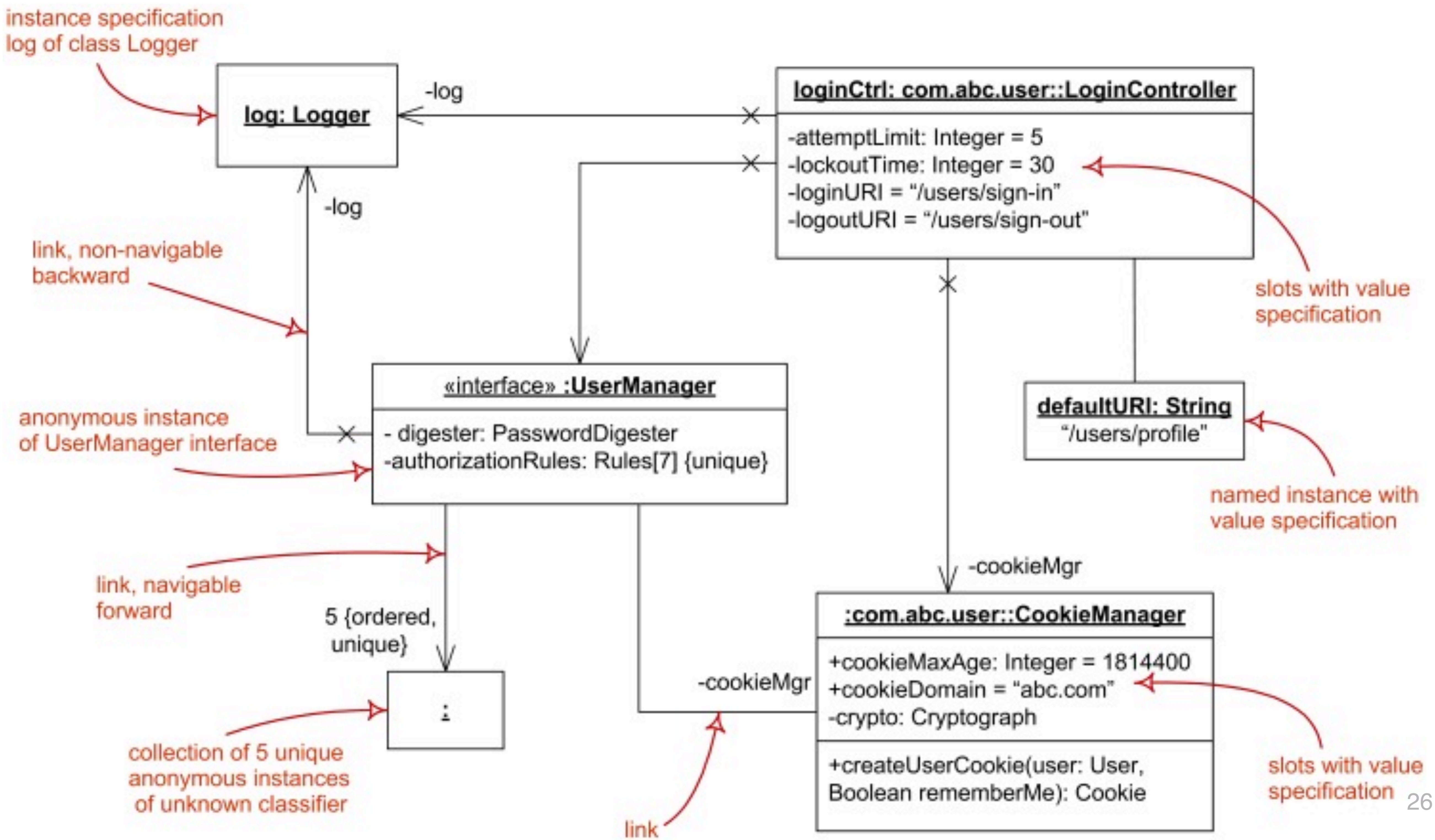


Is-a *Polymorphism* *Reuse*

Roadmap

- UML Overview
- Structural Diagrams
  - Classes, attributes and operations
  - ➡ Objects, Associations
- Behavioral Diagrams
  - Sequence
  - Communication
  - Activity
  - State
- Further Discussion

Objects


instance specification
log of class Logger
log: Logger
-log
loginCtrl: com.abc.user::LoginController
-attemptLimit: Integer = 5
-lockoutTime: Integer = 30
-loginURI = "/users/sign-in"
-logoutURI = "/users/sign-out"
-log
link, non-navigable
backward
slots with value
specification
«interface» :UserManager
- digester: PasswordDigester
-authorizationRules: Rules[7] {unique}
defaultURI: String
"/users/profile"
anonymous instance
of UserManager interface
named instance with
value specification
link, navigable
forward
-cookieMgr
5 {ordered,
unique}
:com.abc.user::CookieManager
+cookieMaxAge: Integer = 1814400
+cookieDomain = "abc.com"
-crypto: Cryptograph
+createUserCookie(user: User,
Boolean rememberMe): Cookie
-cookieMgr
:
collection of 5 unique
anonymous instances
of unknown classifier
link
slots with value
specification

Associations

Associations represent *structural relationships* between objects

- —usually *binary* (but may be ternary etc.)
- —optional *name* and *direction*
- —(unique) *role names* and *multiplicities* at end-points

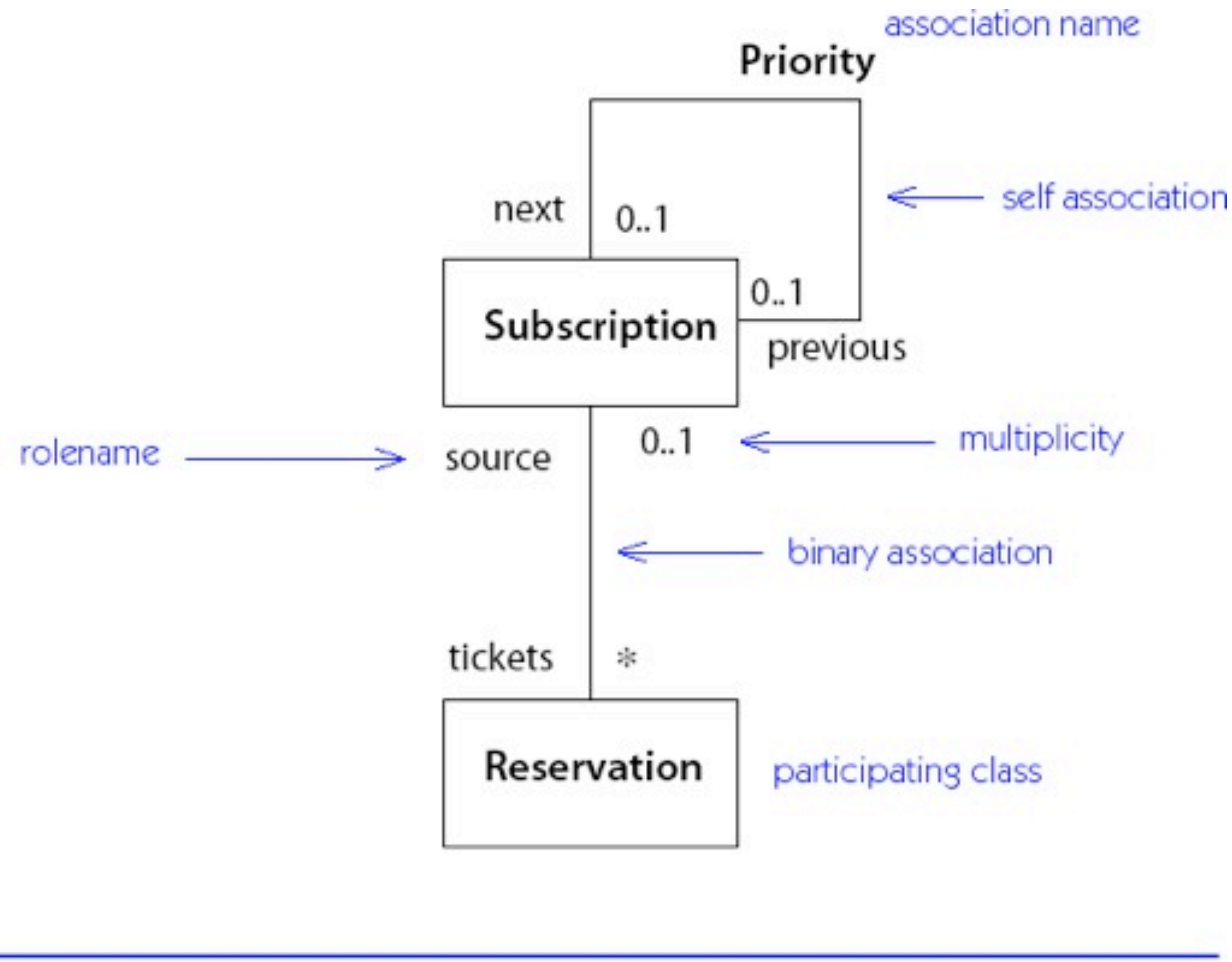


Figure 4-2. *Association notation*

Multiplicity

0..1	Zero or one entity
1	Exactly one entity
*	Any number of entities
1..*	One or more entities
1..n	One to n entities
	And so on …

Aggregation and Composition

Aggregation is denoted by a *diamond* and indicates a *part-whole dependency*:

A *hollow diamond* indicates a *reference*; a *solid diamond* an *implementation* (i.e., ownership).

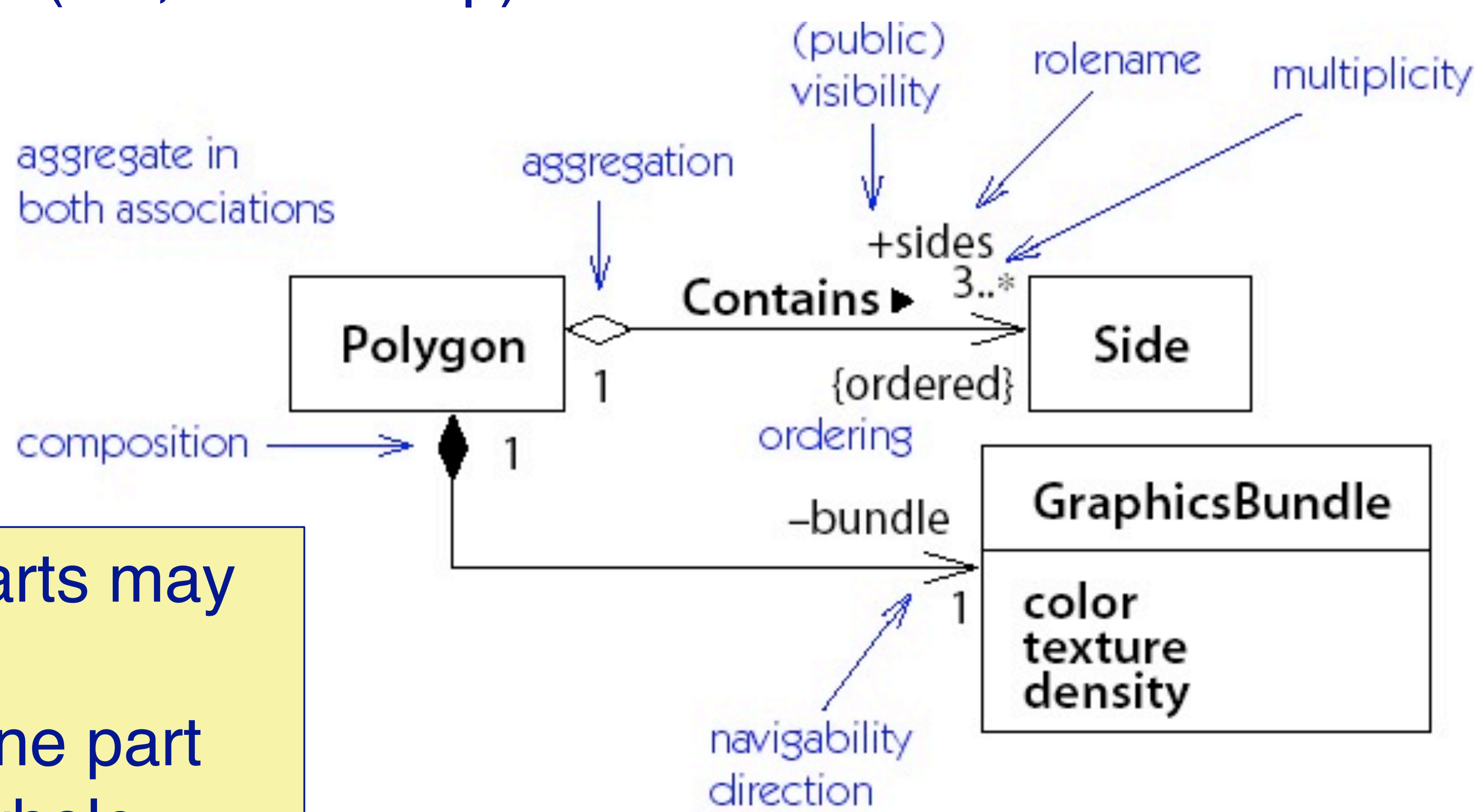


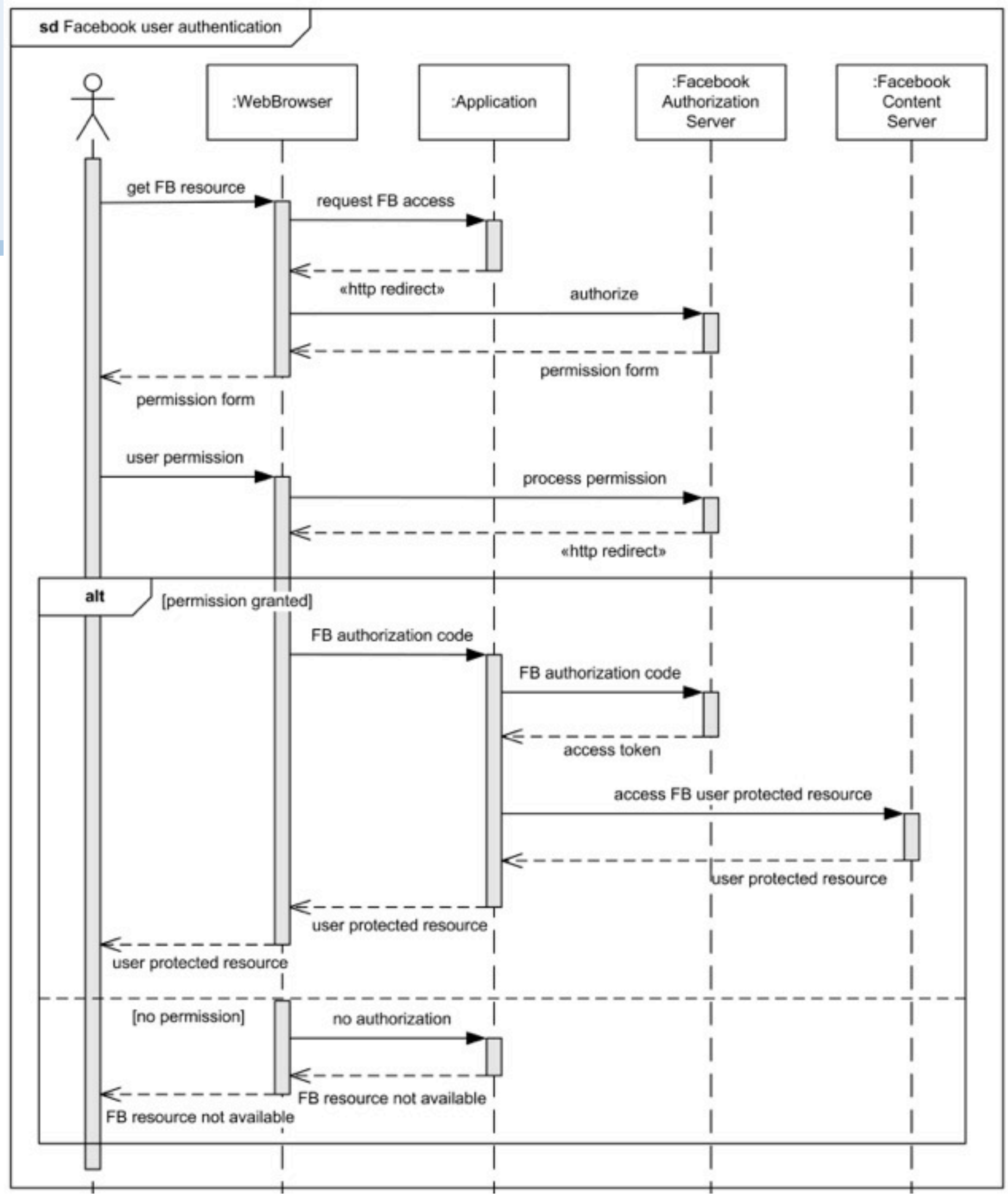
Figure 13-29. *Various adornments on association ends*

Aggregation: parts may be shared.
Composition: one part belongs to one whole.

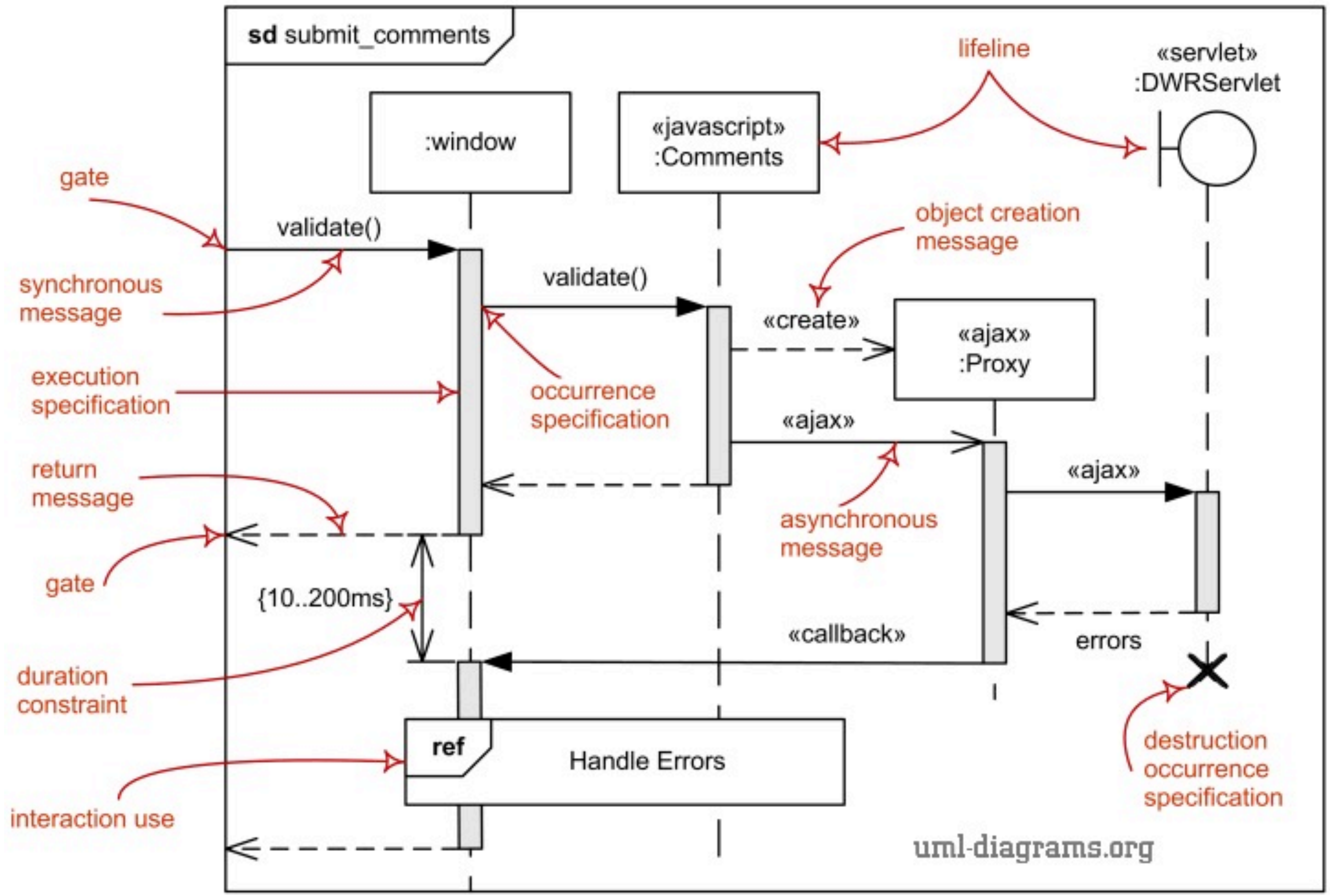
Roadmap

- UML Overview
- Structural Diagrams
  - Classes, attributes and operations
  - Objects, Associations
- Behavioral Diagrams
  - ➡ Sequence
  - Communication
  - Activity
  - State
- Further Discussion

Sequence Diagram, e.g.



The Elements of a Sequence Diagram



Activations

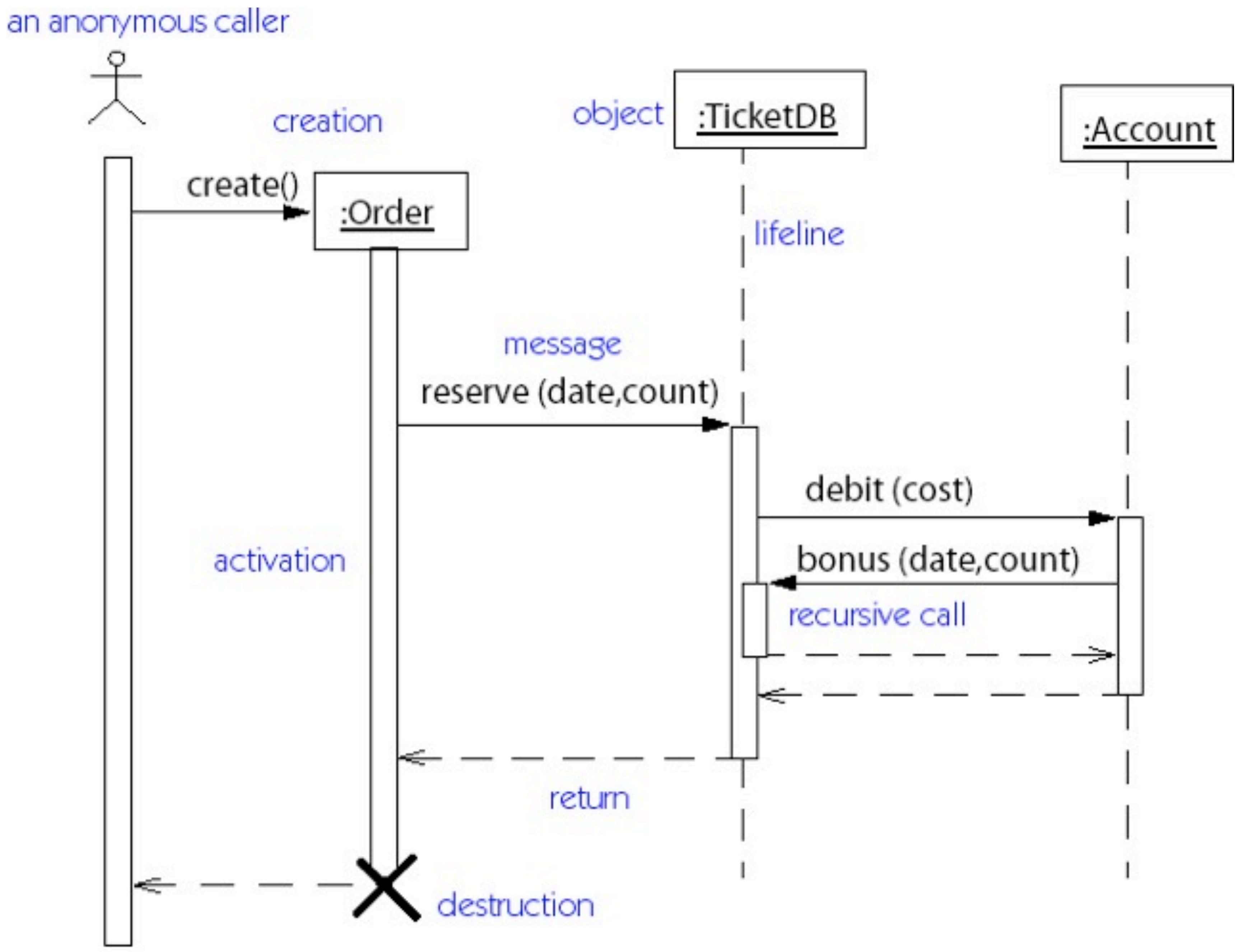


Figure 8-2. *Sequence diagram with activations*

Asynchrony and Constraints

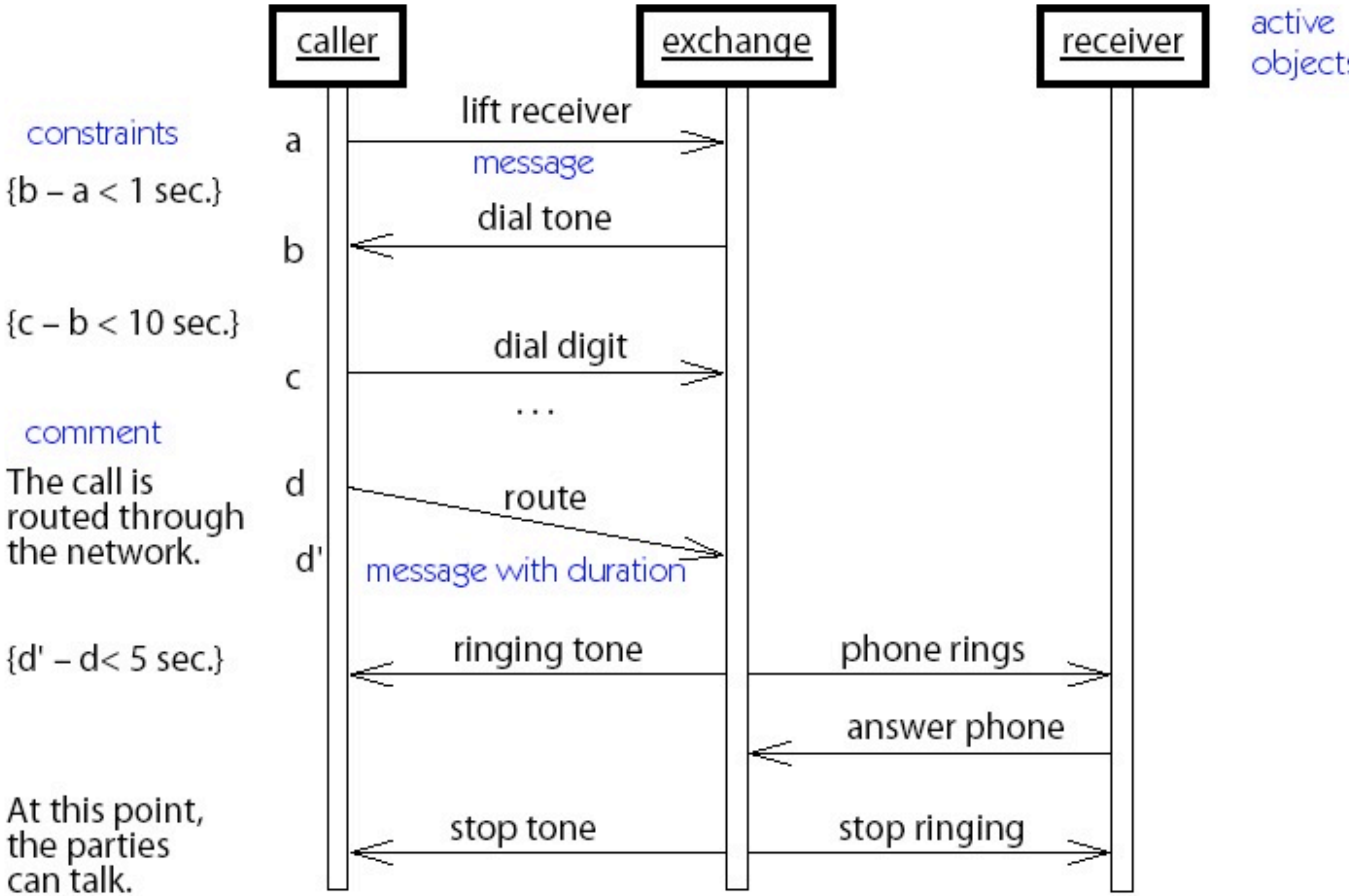


Figure 13-161. *Sequence diagram with asynchronous control*

Roadmap

UML Overview

Structural Diagrams

- Classes, attributes and operations
- Objects, Associations

Behavioral Diagrams

- Sequence
- ➡ Communication
- Activity
- State

Further Discussion

Communication Diagrams

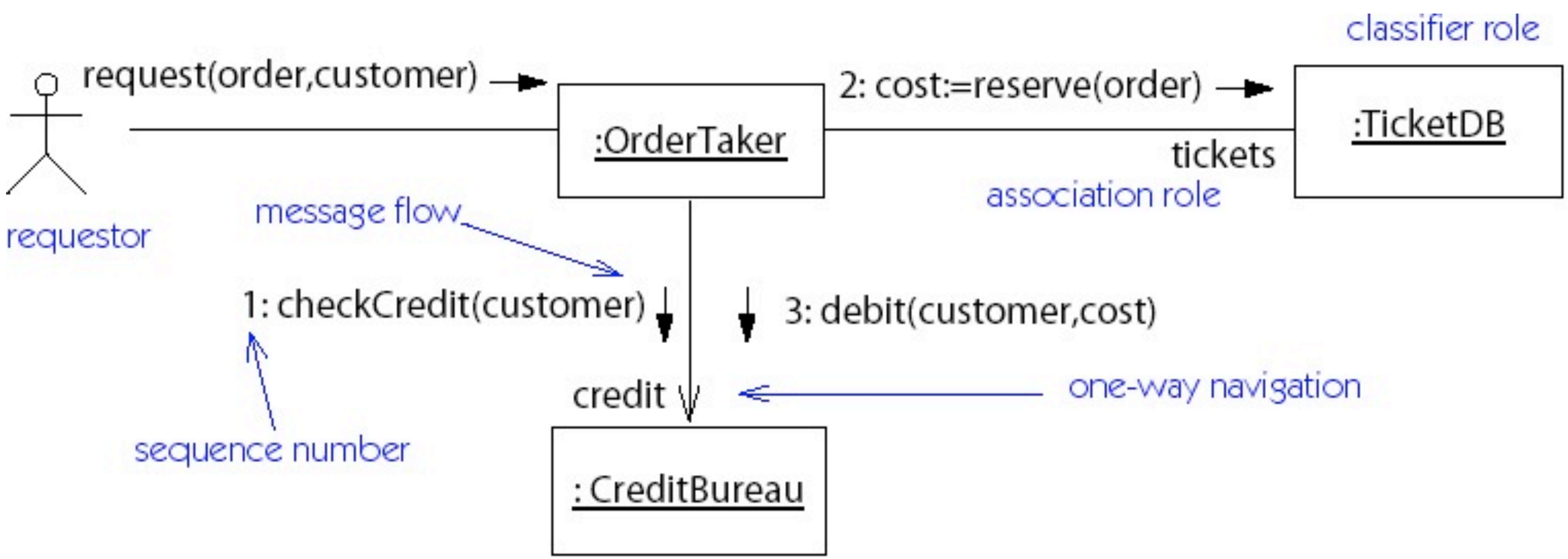


Figure 8-3. *Collaboration diagram*

Roadmap

UML Overview

Structural Diagrams

- Classes, attributes and operations
- Objects, Associations

Behavioral Diagrams

- Sequence
- Communication
- ➡ Activity
- State

Further Discussion

Activity Diagram

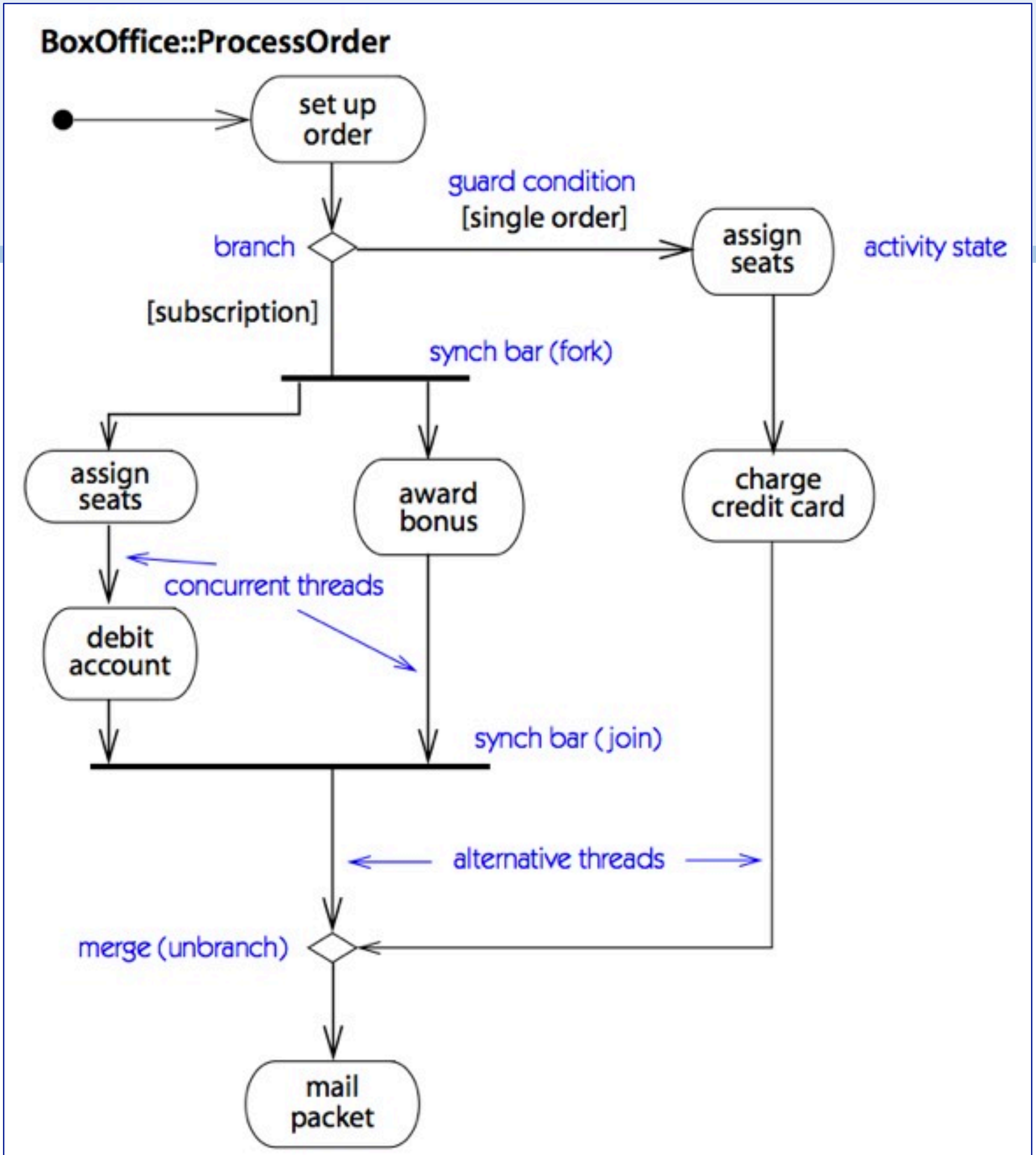


Figure 7-1. *Activity diagram*

Roadmap

UML Overview

Structural Diagrams

- Classes, attributes and operations
- Objects, Associations

Behavioral Diagrams

- Sequence
- Communication
- Activity
- ➡ State

Further Discussion

Statechart Diagrams

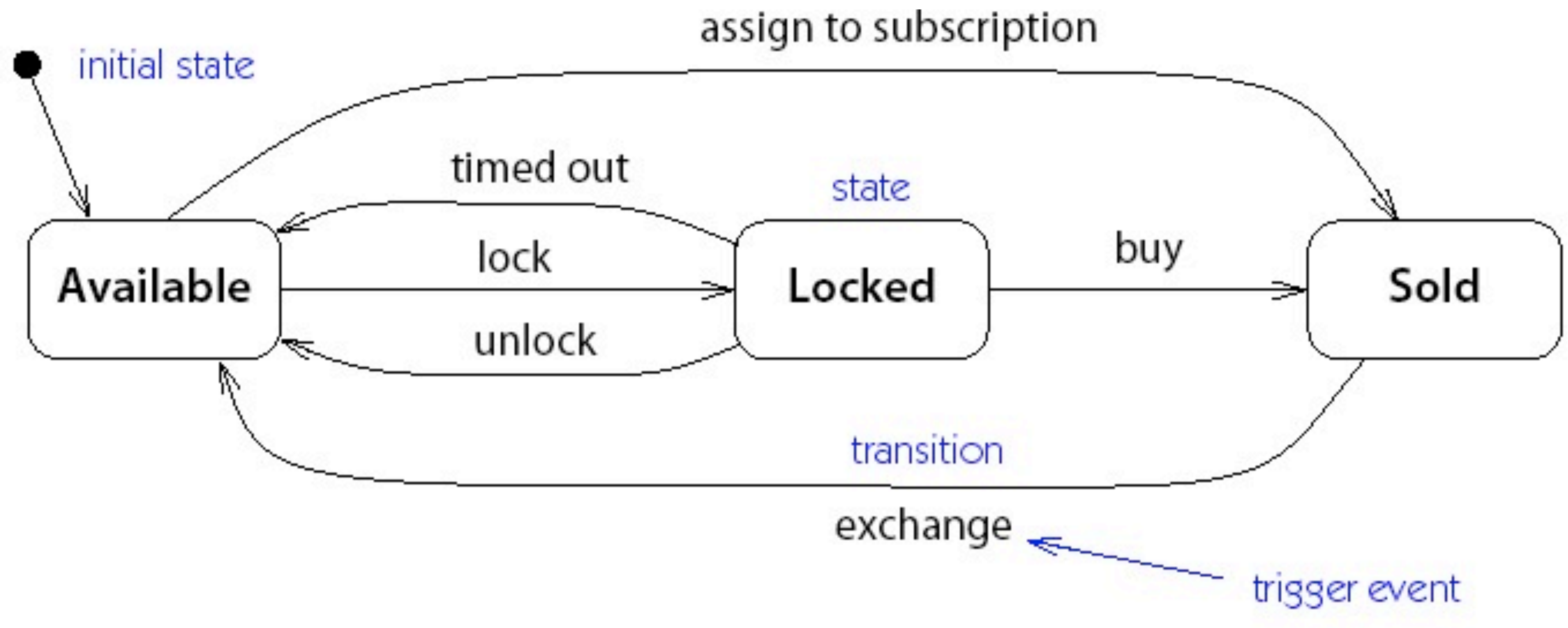


Figure 3-5. *Statechart diagram*

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 <u>state</u> 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:
  - name — *optional*
  - state *variables — name: type = value (valid only for that state)*
  - 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.

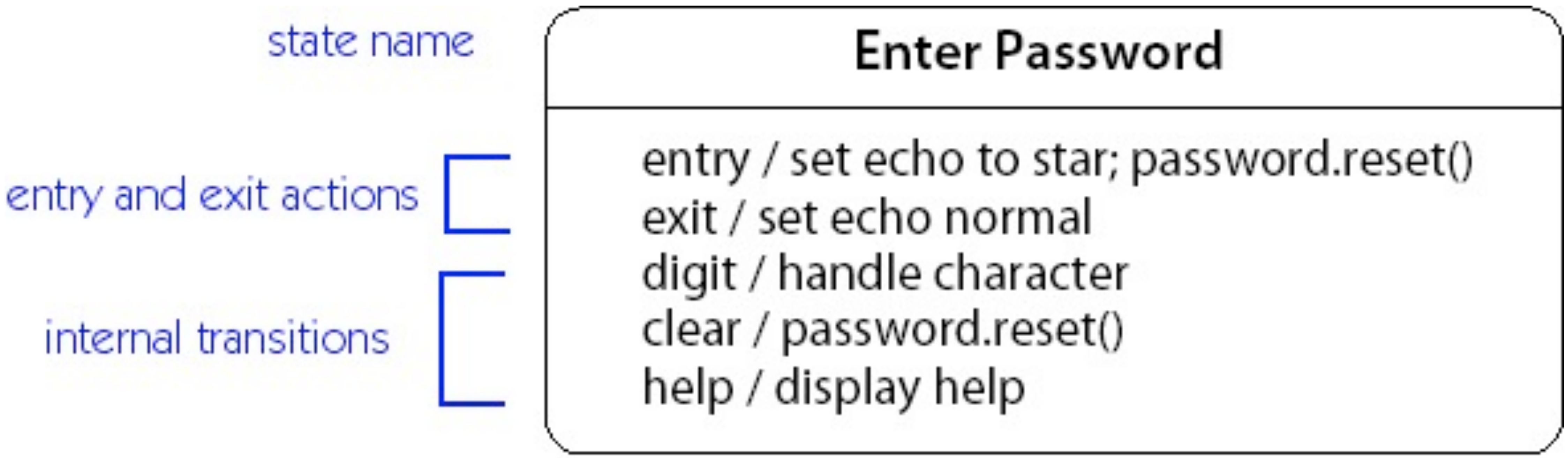


Figure 6-4. *Internal transitions, and entry and exit actions*

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 <u>operation</u> is an *atomic action* invoked by a transition

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

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

— Modelled as "internal transitions" labelled with the pseudo-event ***do***

Nested Statecharts

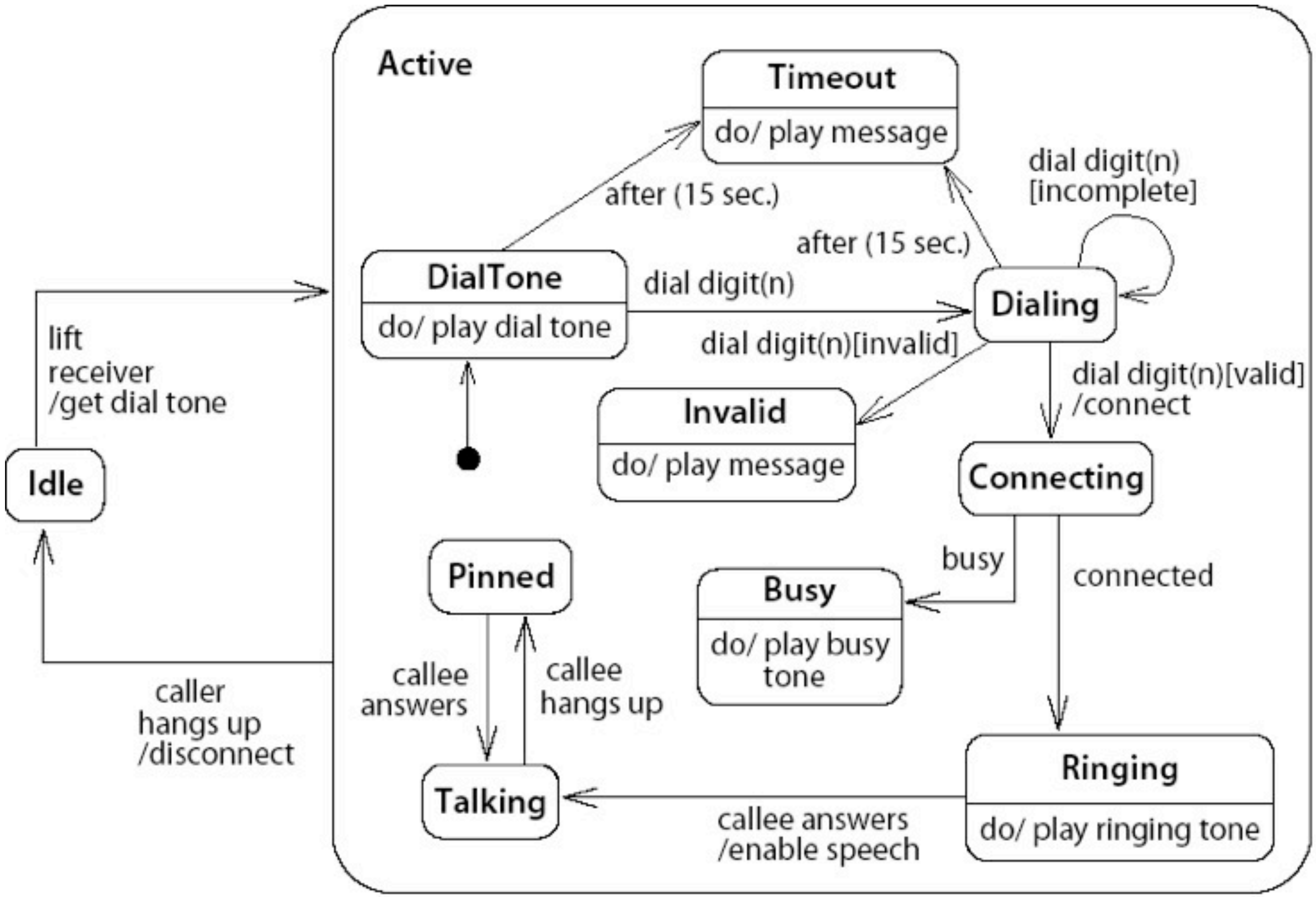


Figure 13-169. *State diagram*

Composite States

Composite states may depicted either as high-level or low-level views.

“Stubbed transitions” indicate the presence of internal states:

Initial and terminal substates are shown as black spots and “bulls-eyes”

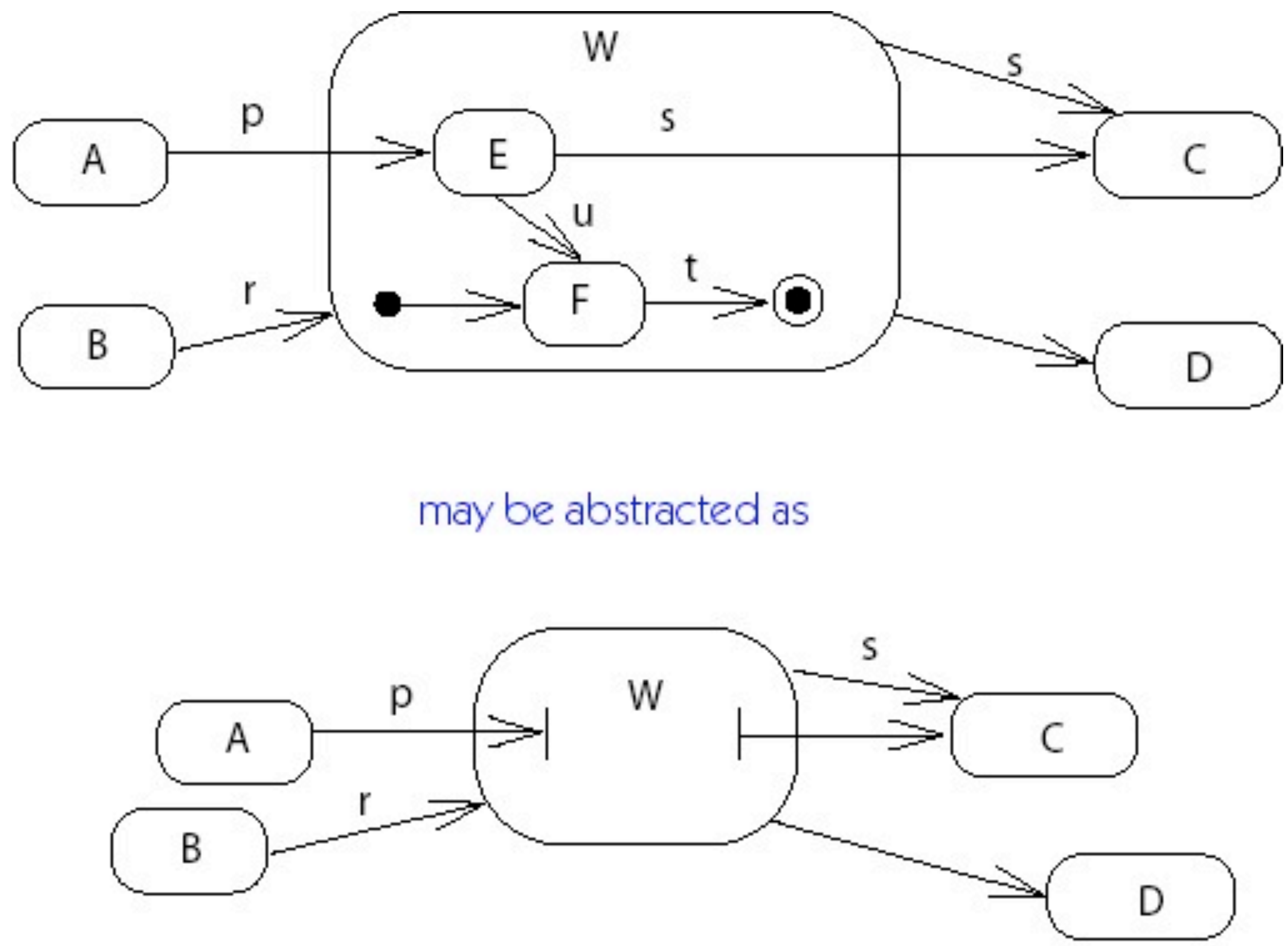


Figure 13-172. *Stubbed transition*

Sending Events between Objects

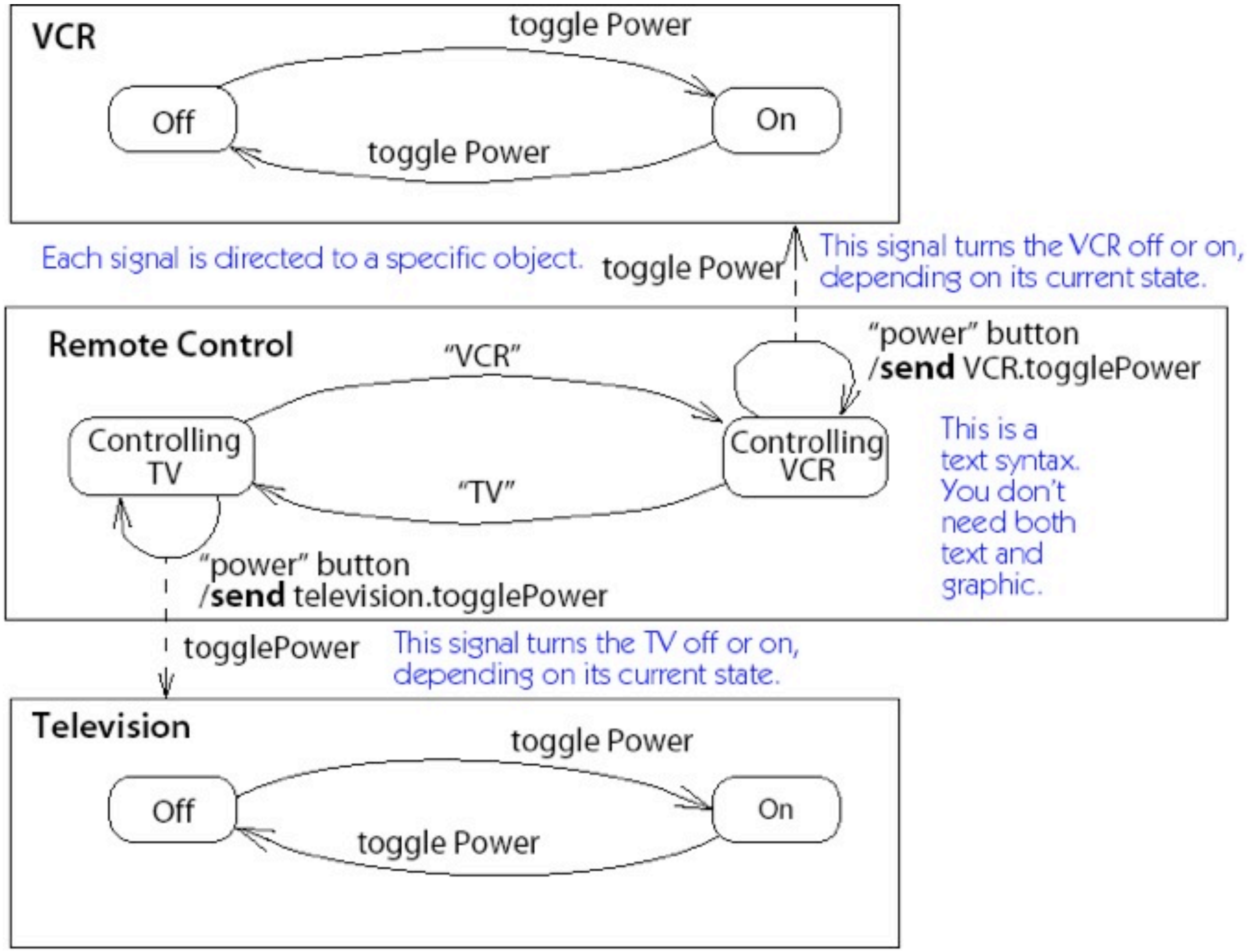


Figure 13-160. *Sending signals between objects*

Concurrent Substates

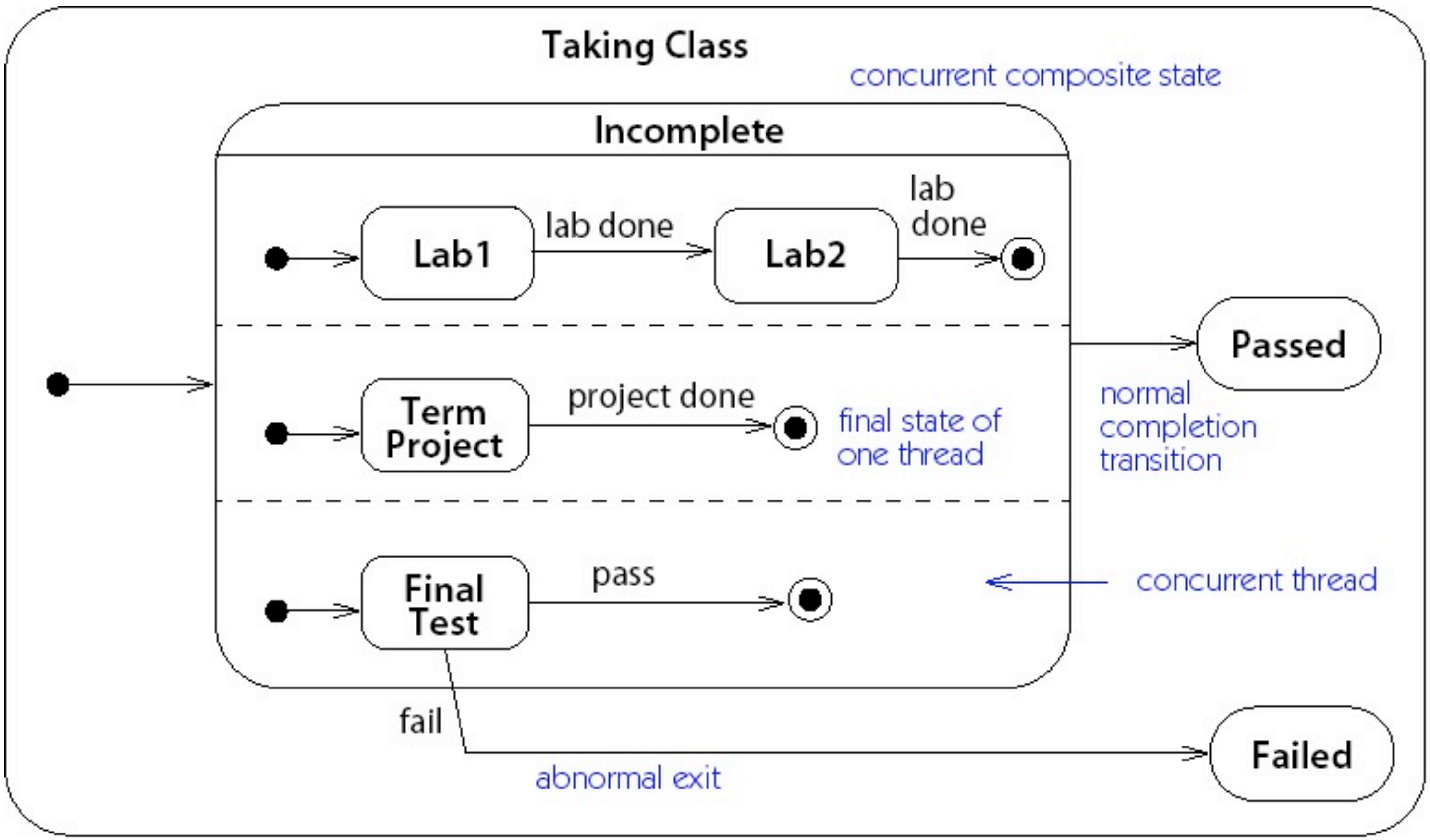


Figure 6-6. *State machine with concurrent composite state*

Branching and Merging

Entering concurrent states:

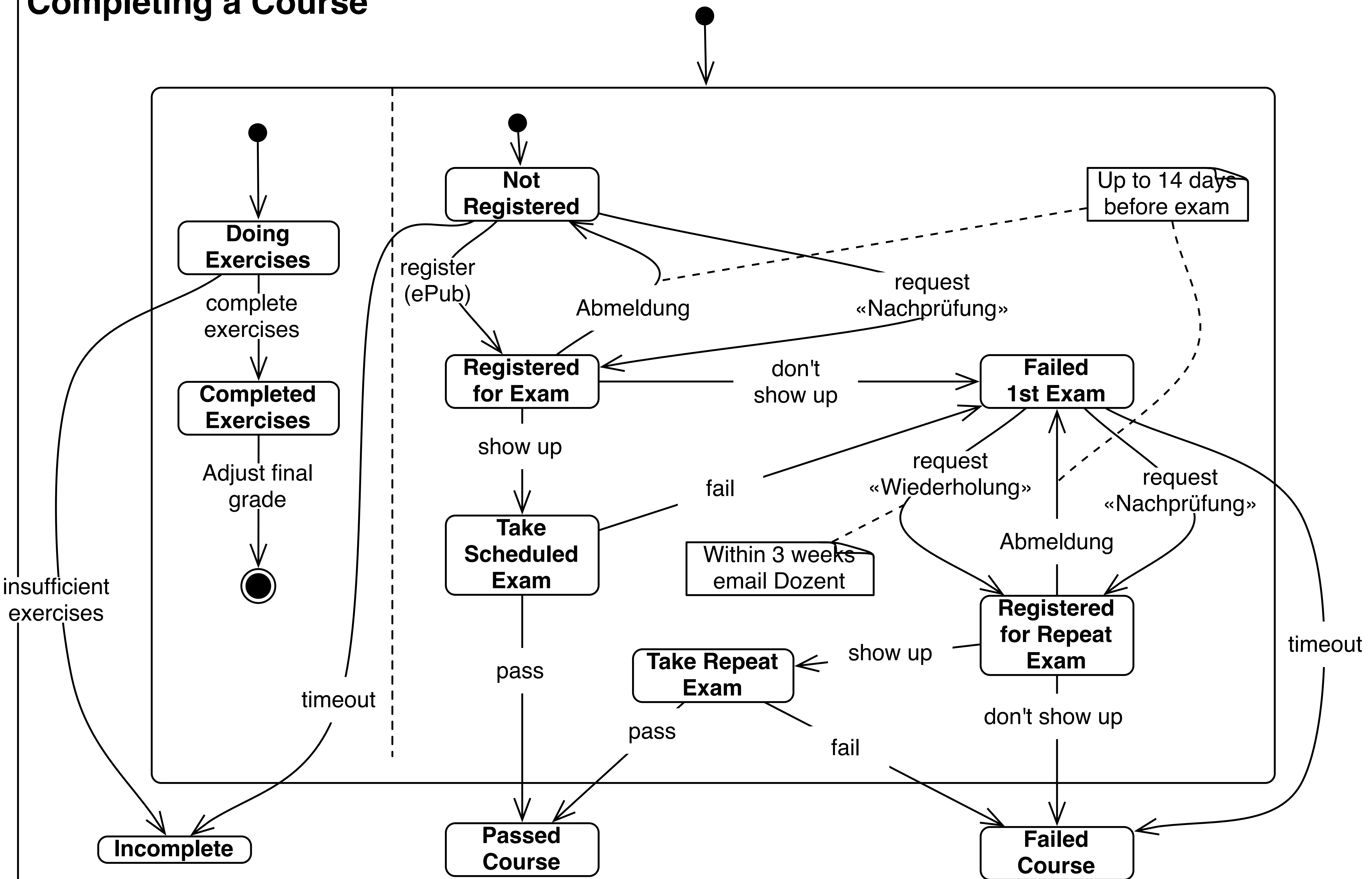
Entering a state with concurrent substates means that *each of the substates is entered concurrently* (one logical thread per substate).

Leaving concurrent states:

A *labelled transition* out of any of the substates *terminates all of the substates.*

An *unlabelled transition* out of the overall state *waits* for all substates to terminate.

Completing a Course



Is it correct?

Roadmap

- UML Overview
- Structural Diagrams
  - Classes, attributes and operations
  - Objects, Associations
- Behavioral Diagrams
  - Sequence
  - Communication
  - Activity
  - State
- ➡ Further Discussion

Constraints

Constraints are *restrictions* on values attached to classes or associations.

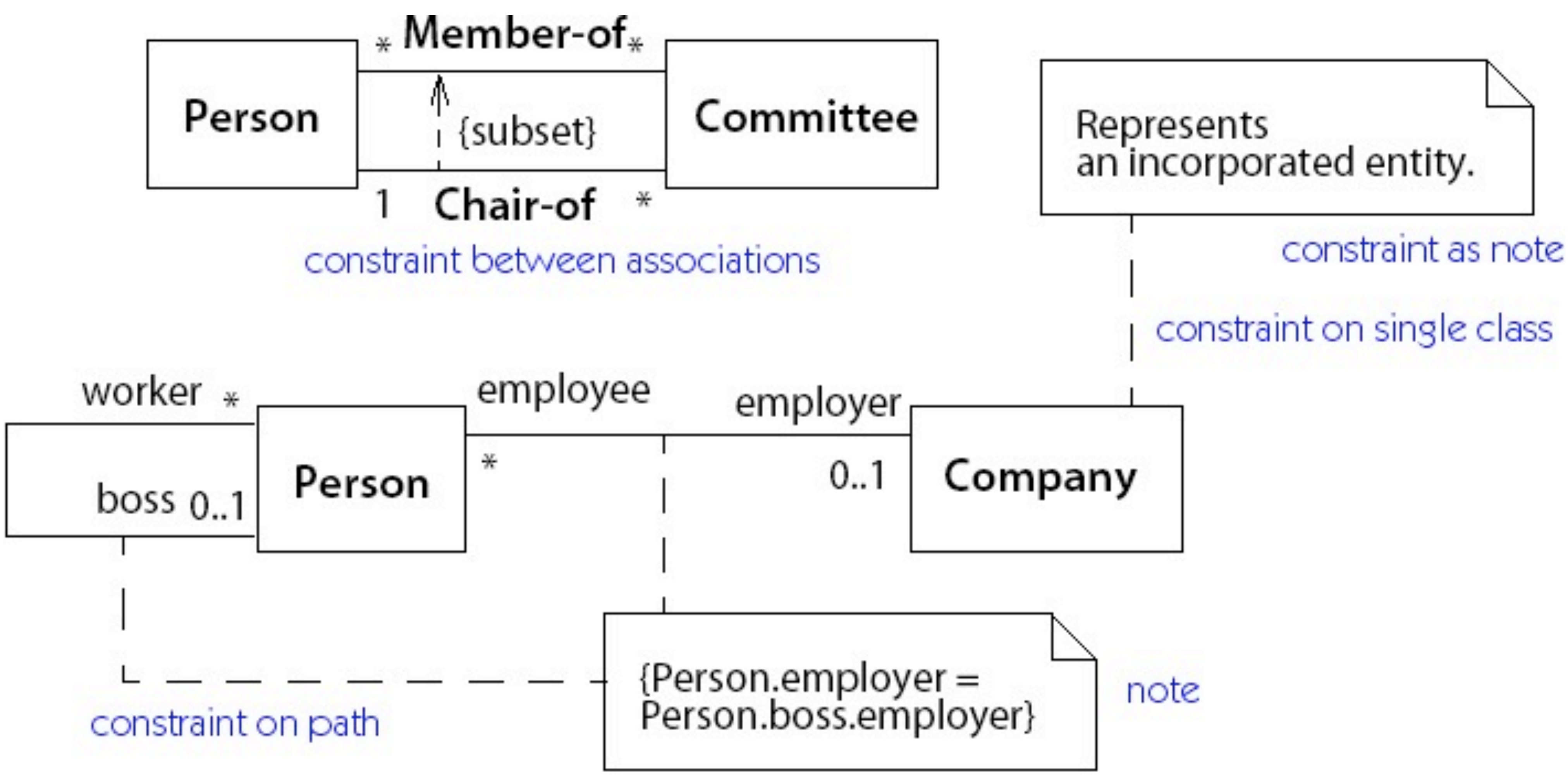
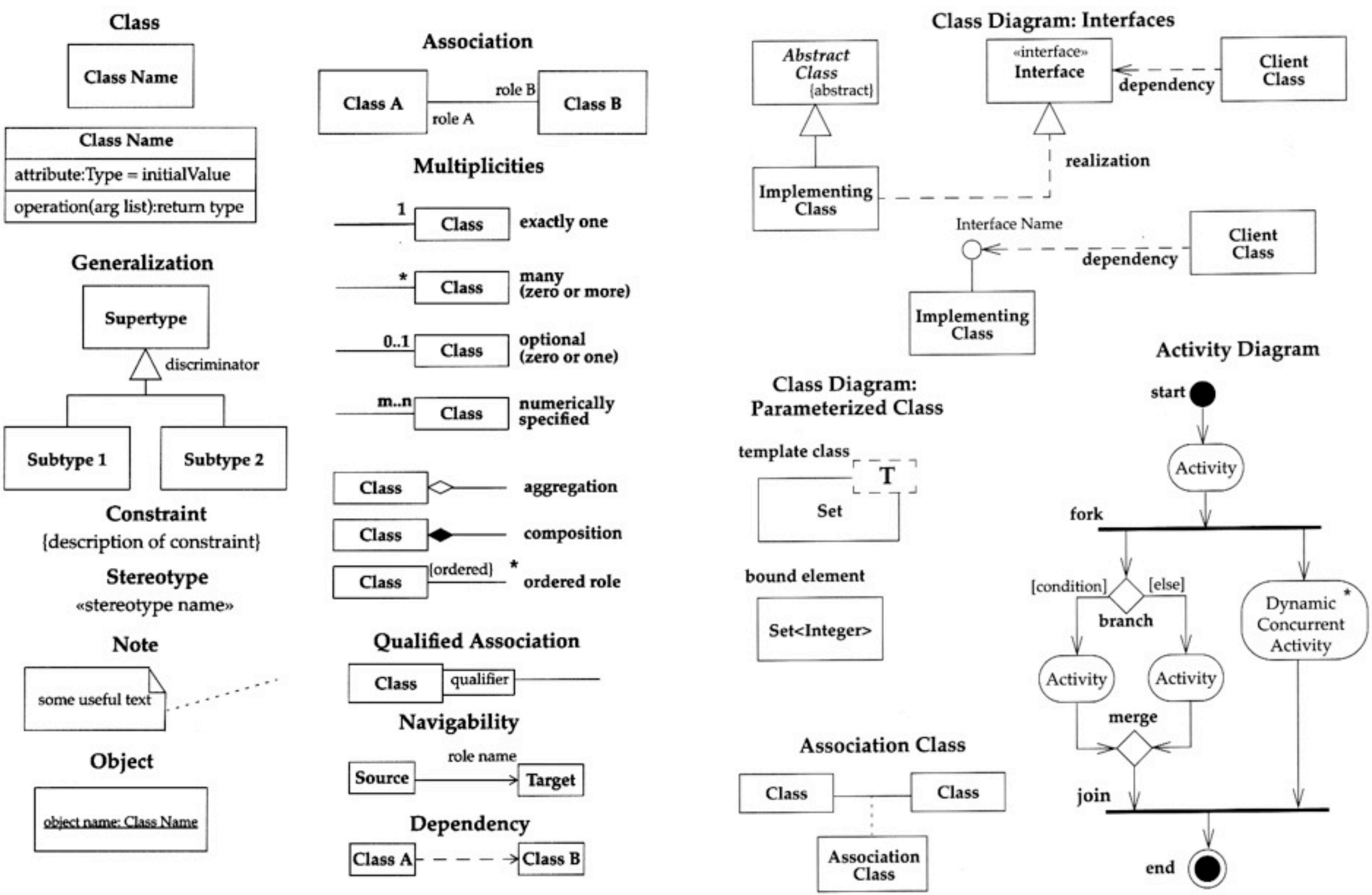
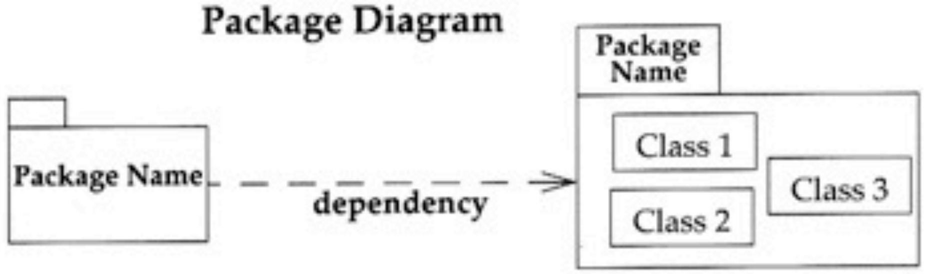


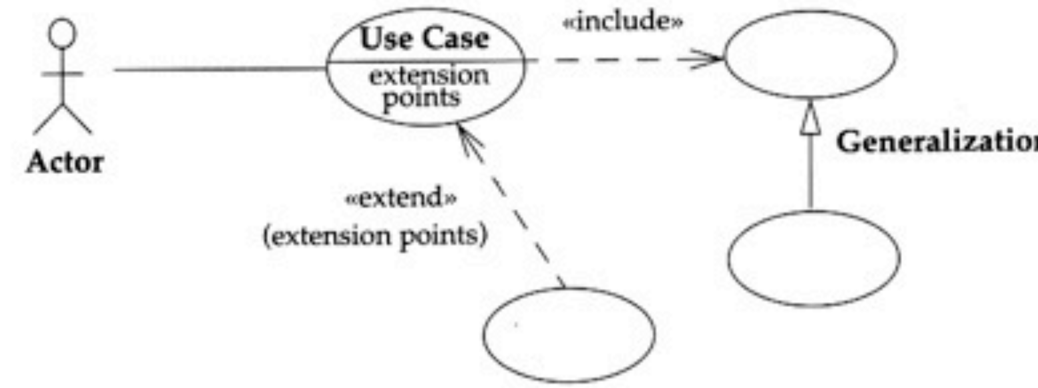
Figure 4-12. *Constraints*


Class
Class Name
Class Name
attribute:Type = initialValue
operation(arg list):return type
Generalization
Supertype
discriminator
Subtype 1
Subtype 2
Constraint
{description of constraint}
Stereotype
«stereotype name»
Note
some useful text
Object
object name: Class Name
Association
Class A
role B
role A
Class B
Multiplicities
1
Class
exactly one
*
Class
many
(zero or more)
0..1
Class
optional
(zero or one)
m..n
Class
numerically
specified
Class
aggregation
Class
composition
Class
{ordered}
*
ordered role
Qualified Association
Class
qualifier
Navigability
Source
role name
Target
Dependency
Class A
Class B
Class Diagram: Interfaces
Abstract
Class
{abstract}
«interface»
Interface
dependency
Client
Class
realization
Implementing
Class
Interface Name
dependency
Client
Class
Implementing
Class
Class Diagram:
Parameterized Class
template class
T
Set
bound element
Set<Integer>
Association Class
Class
Class
Association
Class
Activity Diagram
start
Activity
fork
[condition]
[else]
branch
Dynamic
Concurrent
Activity
*
Activity
Activity
merge
join
end

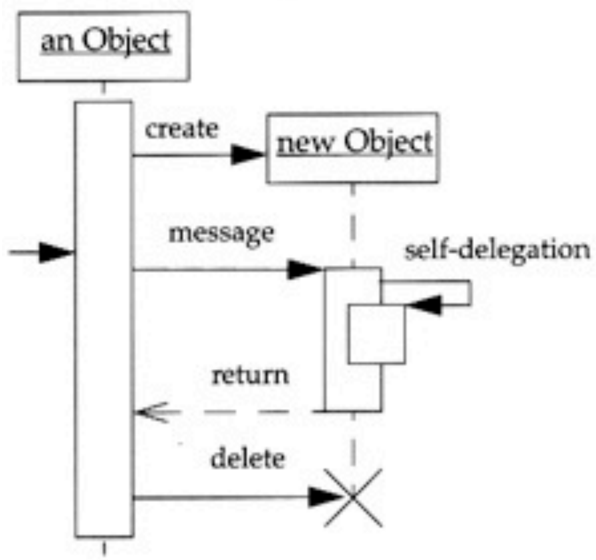
Package Diagram



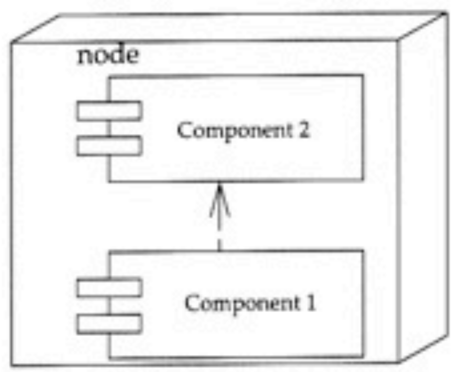
Use Case Diagram



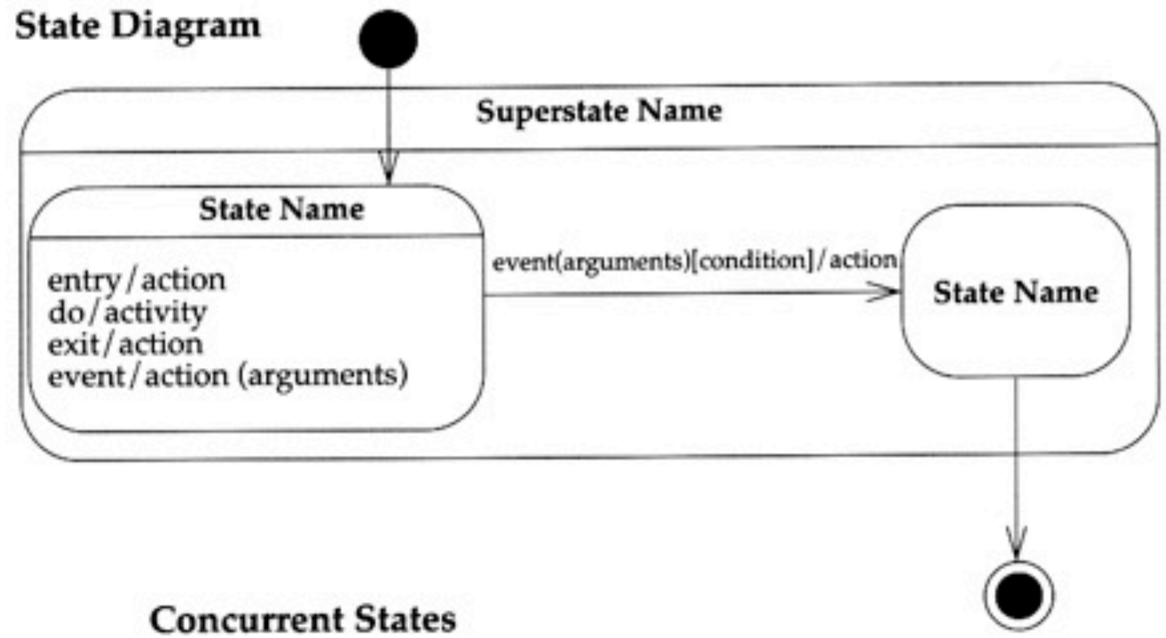
Sequence Diagram



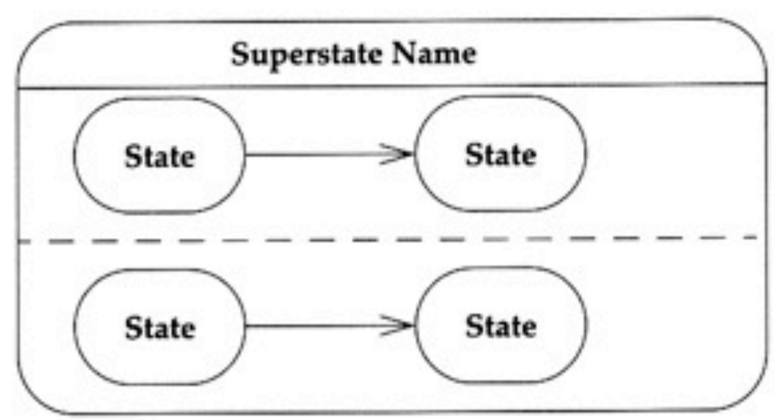
Deployment Diagram



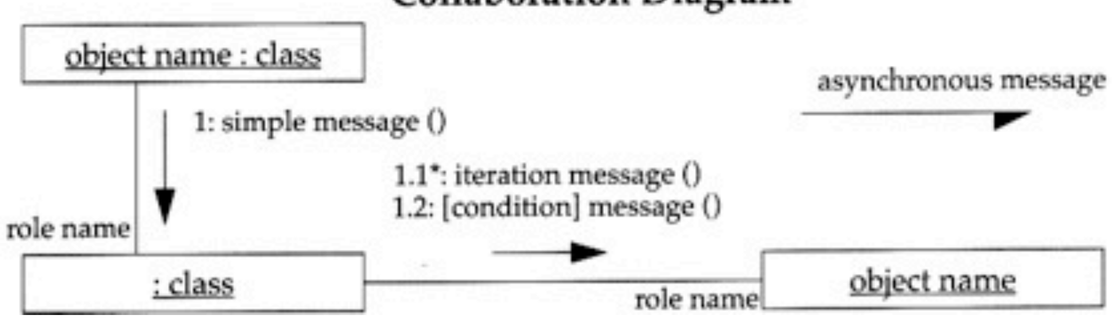
State Diagram



Concurrent States



Collaboration Diagram



Sources

- *The Unified Modeling Language Reference Manual*, James Rumbaugh, Ivar Jacobson and Grady Booch, Addison Wesley, 1999.
- *UML Distilled*, Martin Fowler, Kendall Scott, Addison-Wesley, Second Edition, 2000.
- *UML in Practice*, Marian Petre, ICSE 2013
- http://www.uml-diagrams.org/ by Kiril Fakrouthdinov

What you should know!

- Why do scenarios depict objects but not classes?
- How can timing constraints be expressed in scenarios?
- How do you use nested state diagrams to model object behavior?
- What is the difference between “external” and “internal” transitions?
- How can you model interaction between state diagrams for several classes?
- How do you represent classes, objects and associations?
- How do you specify the visibility of attributes and operations to clients?
- Why is inheritance useful in analysis? In design?

Can you answer the following questions?

- Can a sequence diagram always be translated to an communication diagram?
- Why are arrows depicted with the message labels rather than with links?
- How is aggregation different from any other kind of association?
- How are associations realized in an implementation language?

License