Analysis, Design, and Software Architecture (BDSA)

Paolo Tell

Introduction to UML

IT University
of Copenhagen

Recap

What did we see so far … I

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Software engineering is a collection of techniques, methodologies, and tools that help with the production of a high quality software system developed with a given budget before a given deadline while change occurs.
- Requirements engineering is the process of developing a software specification.
- Design and implementation processes are concerned with transforming a requirements specification into an executable software system.
- Software validation is the process of checking that the system conforms to its specification and that it meets the real needs of the users of the system.
- Software evolution takes place when you change existing software systems to meet new requirements. The software must evolve to remain useful.

What did we see so far … II

- Functional requirements statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations.

- Non-functional requirements constrain the system being developed and the development process being used. Often relate to the emergent properties of the system and therefore apply to the system as a whole—they concern the architecture.

- Domain requirements are constraints on the system from the domain of operation—the application domain.

- The software requirements document is an agreed statement of the system requirements that should be organized so that both system customers and software developers can use it.

- The requirements engineering process an iterative process that includes requirements elicitation, specification, and validation.

- Requirements elicitation and analysis is an iterative process that can be represented as a spiral of activities—requirements discovery, requirements classification and organization, requirements negotiation, and requirements documentation.

What did we see so far … III

- Requirement elicitation concepts
  - Completeness, Consistency, Clarity, and Correctness
  - Realism, Verifiability, and Traceability
  - Greenfield Engineering, Reengineering, and Interface Engineering

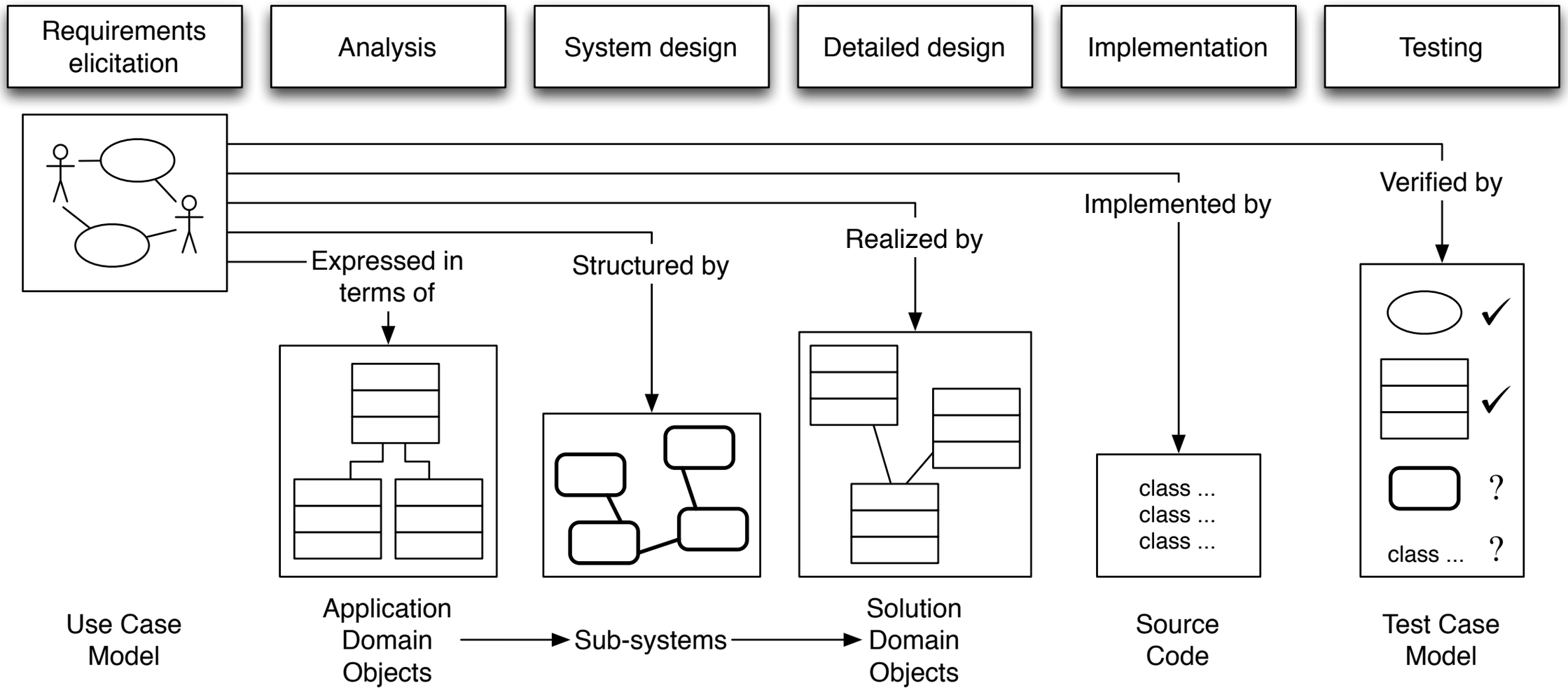
- Requirements elicitation activities
  - Identify Actors, Scenarios, and Use Cases
  - Identify Relationships between Actors and Use Cases
  - Identify Initial Analysis Objects
  - Identify Non-functional Requirements

- Documenting requirements elicitation
  - Requirements analysis document (RAD)

Requirements Analysis Document

1. Introduction
   - 1.1 Purpose of the system
   - 1.2 Scope of the system
   - 1.3 Objectives and success criteria of the project
   - 1.4 Definitions, acronyms, and abbreviations
   - 1.5 References
   - 1.6 Overview
2. Current system
3. Proposed system
   - 3.1 Overview
   - 3.2 Functional requirements
   - 3.3 Nonfunctional requirements
     - 3.3.1 Usability
     - 3.3.2 Reliability
     - 3.3.3 Performance
     - 3.3.4 Supportability
     - 3.3.5 Implementation
     - 3.3.6 Interface
     - 3.3.7 Packaging
     - 3.3.8 Legal
   - 3.4 System models
     - 3.4.1 Scenarios
     - 3.4.2 Use case model
     - *3.4.3 Object model*
     - *3.4.4 Dynamic model*
     - 3.4.5 User interface—navigational paths and screen mock-ups
4. Glossary

What did we see so far … IV



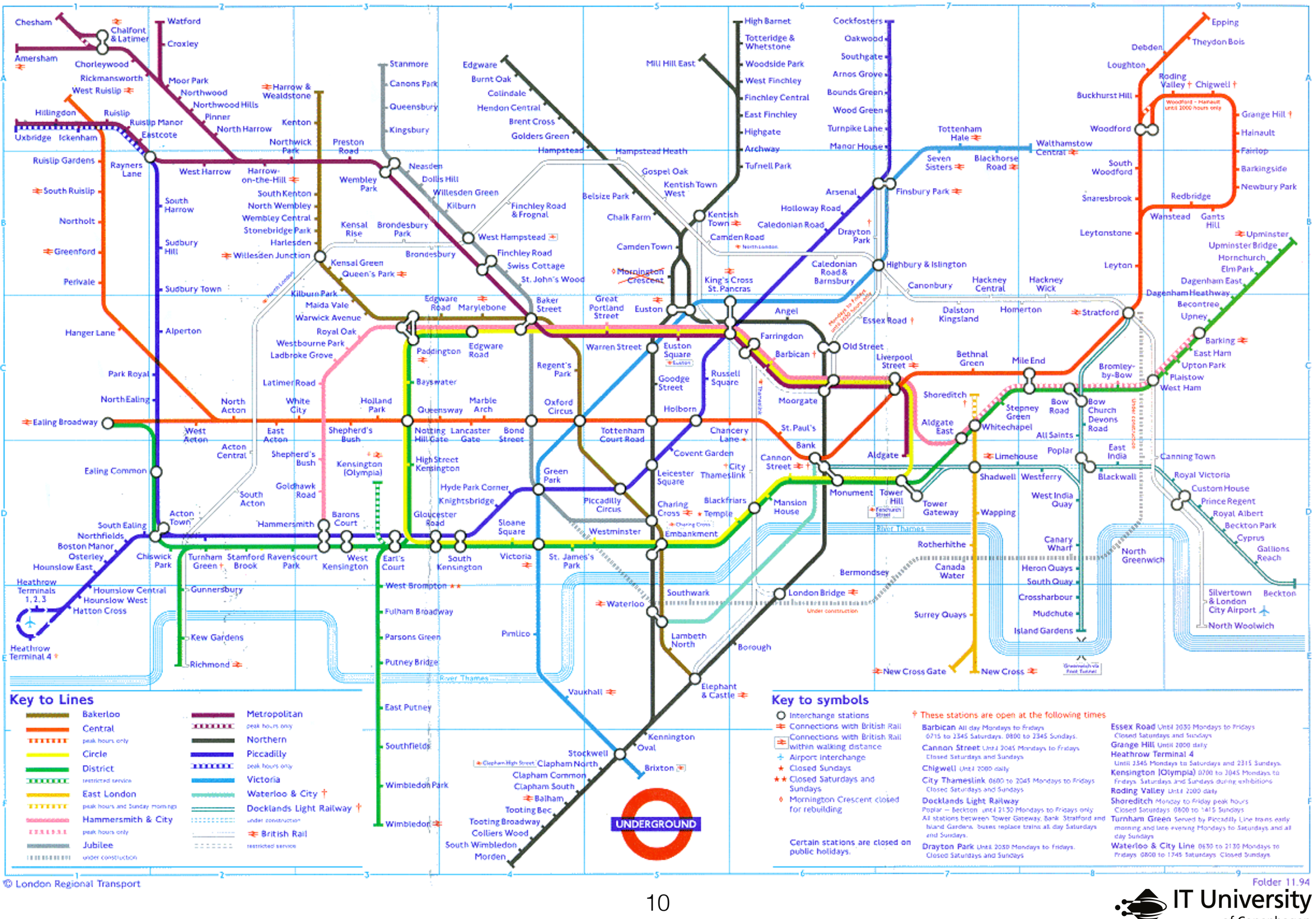
- Software lifecycle

Outline

- Literature
  - [OOSE] ch. 2
- Topics covered:
  - Modeling
  - UML and its history
  - UML behaviour diagrams
    - Brief description
    - Example(s)
    - Comment (or another example)
    - Template from [uml-diagrams.org]
- Next lecture: UML structure diagrams

Modeling


GREEN PARK
←Lift
to Piccadilly line
1 STRATFORD
2 NORTH GREENWICH
fly to hungary & poland
london luton→
budapest
warsaw . gdansk
katowice krakow



Key to Lines

- Bakerloo
- Central (peak hours only)
- Circle
- District (restricted service)
- East London (peak hours and Sunday mornings)
- Hammersmith & City (peak hours only)
- Jubilee (under construction)
- Metropolitan (peak hours only)
- Northern
- Piccadilly (peak hours only)
- Victoria
- Waterloo & City †
- Docklands Light Railway † (under construction)
- British Rail (restricted service)

Key to symbols

- Interchange stations
- Connections with British Rail
- Connections with British Rail within walking distance
- Airport interchange
- ★ Closed Sundays
- ★★ Closed Saturdays and Sundays
- ◊ Mornington Crescent closed for rebuilding

Certain stations are closed on public holidays.

† These stations are open at the following times

Barbican All day Mondays to Fridays 0715 to 2345 Saturdays, 0800 to 2345 Sundays.

Cannon Street Until 2045 Mondays to Fridays Closed Saturdays and Sundays

Chigwell Until 2000 daily

City Thameslink 0600 to 2045 Mondays to Fridays Closed Saturdays and Sundays

Docklands Light Railway Poplar – Beckton until 2130 Mondays to Fridays only. All stations between Tower Gateway, Bank, Stratford and Island Gardens. Buses replace trains all day Saturdays and Sundays.

Drayton Park Until 2030 Mondays to Fridays. Closed Saturdays and Sundays

Essex Road Until 2030 Mondays to Fridays Closed Saturdays and Sundays

Grange Hill Until 2000 daily

Heathrow Terminal 4 Until 2345 Mondays to Saturdays and 2315 Sundays.

Kensington (Olympia) 0700 to 2045 Mondays to Fridays. Saturdays and Sundays during exhibitions

Roding Valley Until 2000 daily

Shoreditch Monday to Friday peak hours Closed Saturdays 0800 to 1415 Sundays

Turnham Green Served by Piccadilly Line trains early morning and late evening Mondays to Saturdays and all day Sundays

Waterloo & City Line 0630 to 2130 Mondays to Fridays 0800 to 1745 Saturdays Closed Sundays

Folder 11.94

Modeling concepts

- System
  - an organized set of communicating parts.
  - a systems is composed of a set of sub-systems.
  - models are used to handle complex systems.
    - often several models are used to describe a system.

- Model
  - an abstraction describing a system or sub-system (divide-et-impera).

- Modeling
  - Modeling means constructing an abstraction of a system that focuses on interesting aspects and ignores irrelevant details.
  - A rule of thumb is that each entity should contain at most 7 ± 2 parts [Miller, 1956].

- View
  - depicts selected aspects of a model.

- Notation
  - is a set of graphical or textual rules for depicting models and views.

Systems, Models, and Views

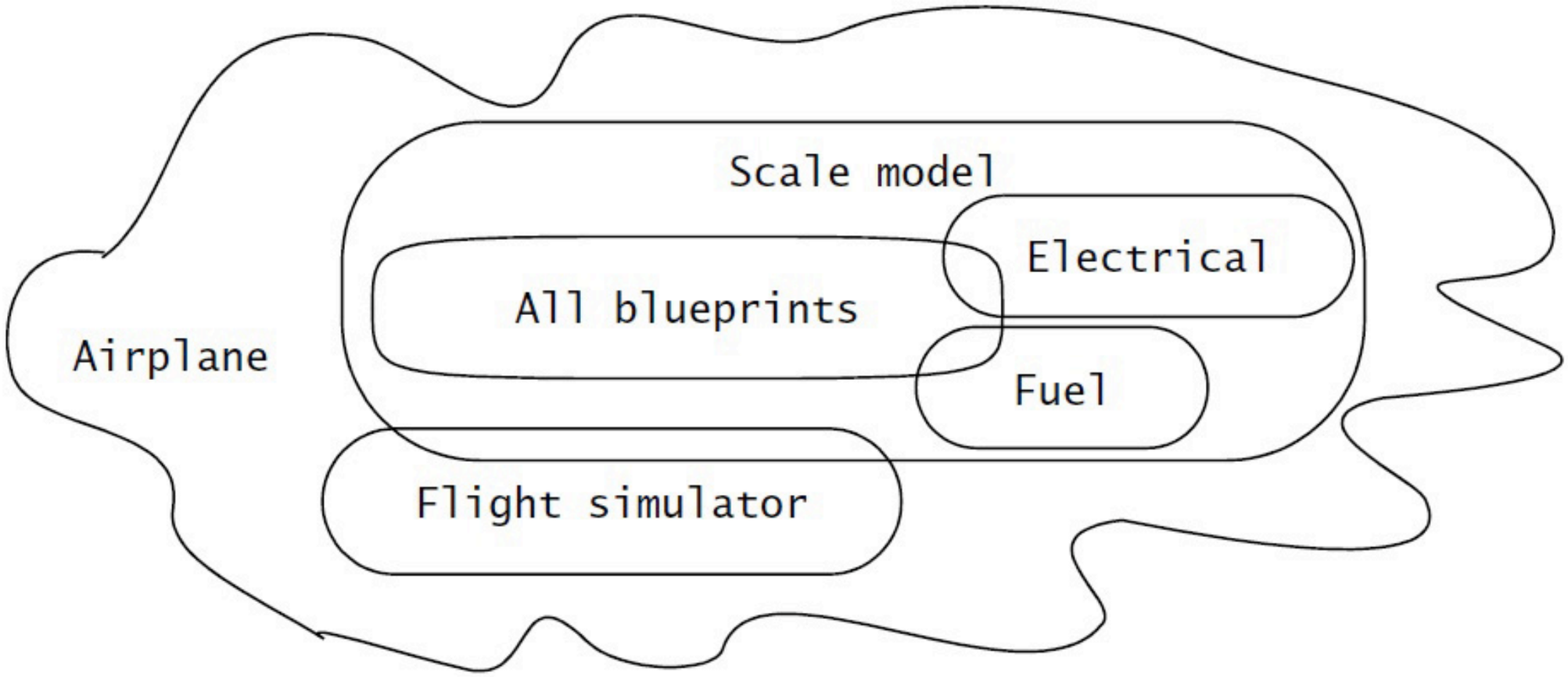
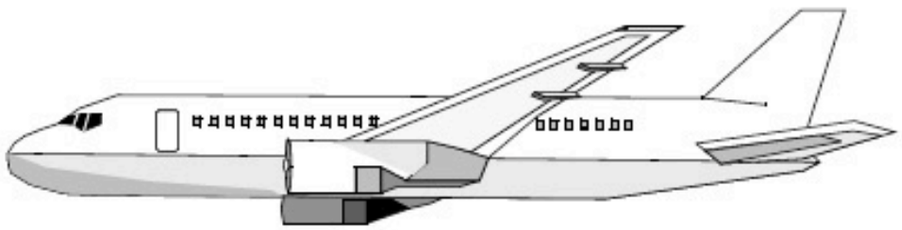


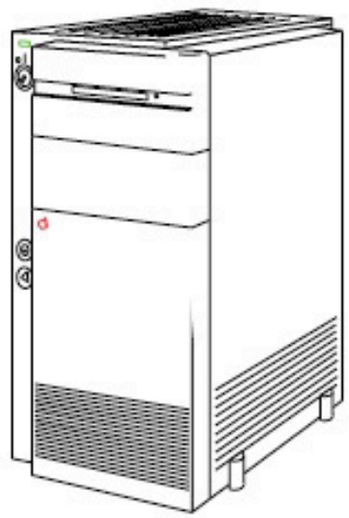
Figure 2-6 A model is an abstraction describing a subset of a system. A view depicts selected aspects of a model. Views and models of a single system may overlap each other.

Object-oriented modeling

- Application Domain (Analysis)
  - The environment in which the system is operating.
  - The user's problem.
  - In OO we model this domain through object-oriented analysis (OOA).
- Solution Domain (Design, Implementation)
  - The technologies used to build the system.
  - The software engineer's problem.
  - In OO we model this domain through object-oriented design (OOD).
- Both domains contain abstractions that we can use for the construction of the system model.

Object-oriented modeling



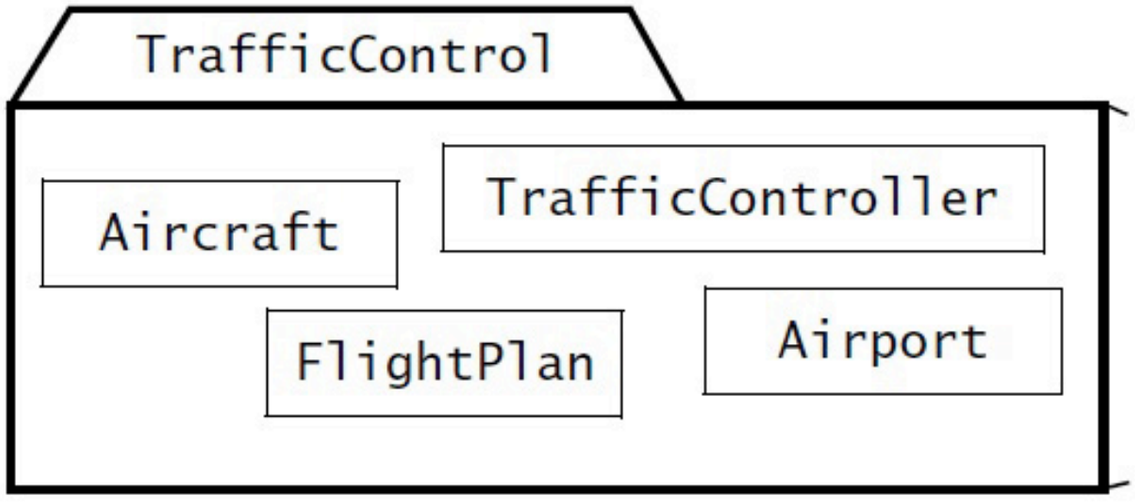


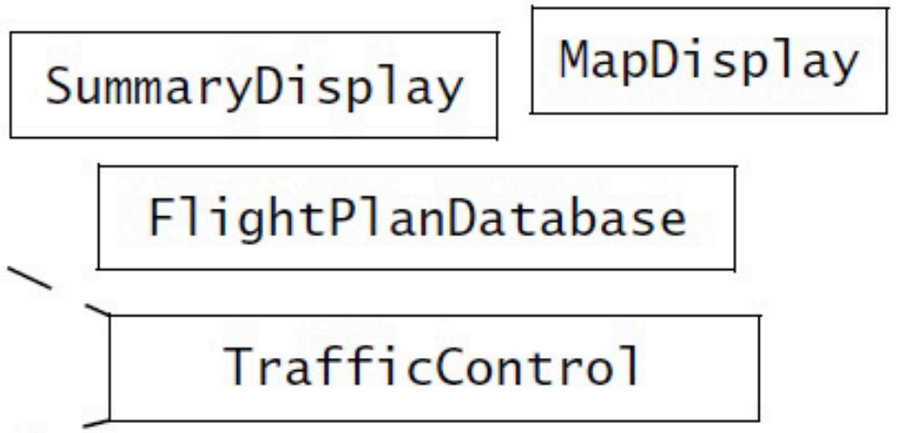
Application Domain

Solution Domain

Application Domain Model

System Model





The Unified Modeling Language (UML) and its history

Unified Modeling Language (UML)

- Object Management Group (OMG) definition
  - "The Unified Modeling Language is a visual language for specifying, constructing and documenting the artifacts of systems" [OMG].
- Adopted in 1997 as a standard by OMG.
- Published in 2005 by the International Organization for Standardization (ISO) as an approved ISO standard.

The 3 amigos

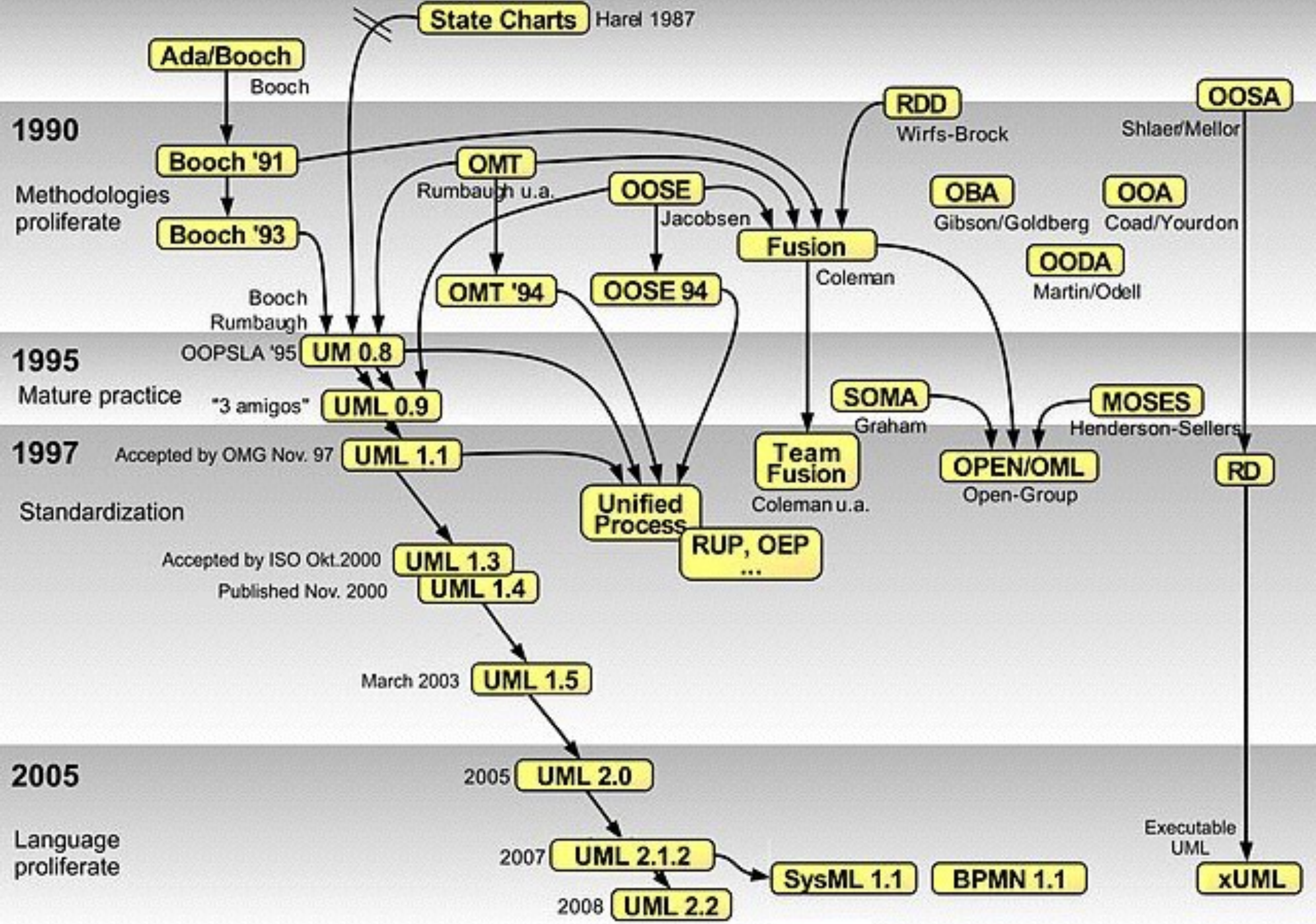
25 year at General Electric Research, where he developed OMT, joined (IBM) Rational in 1994, CASE tool OMTool

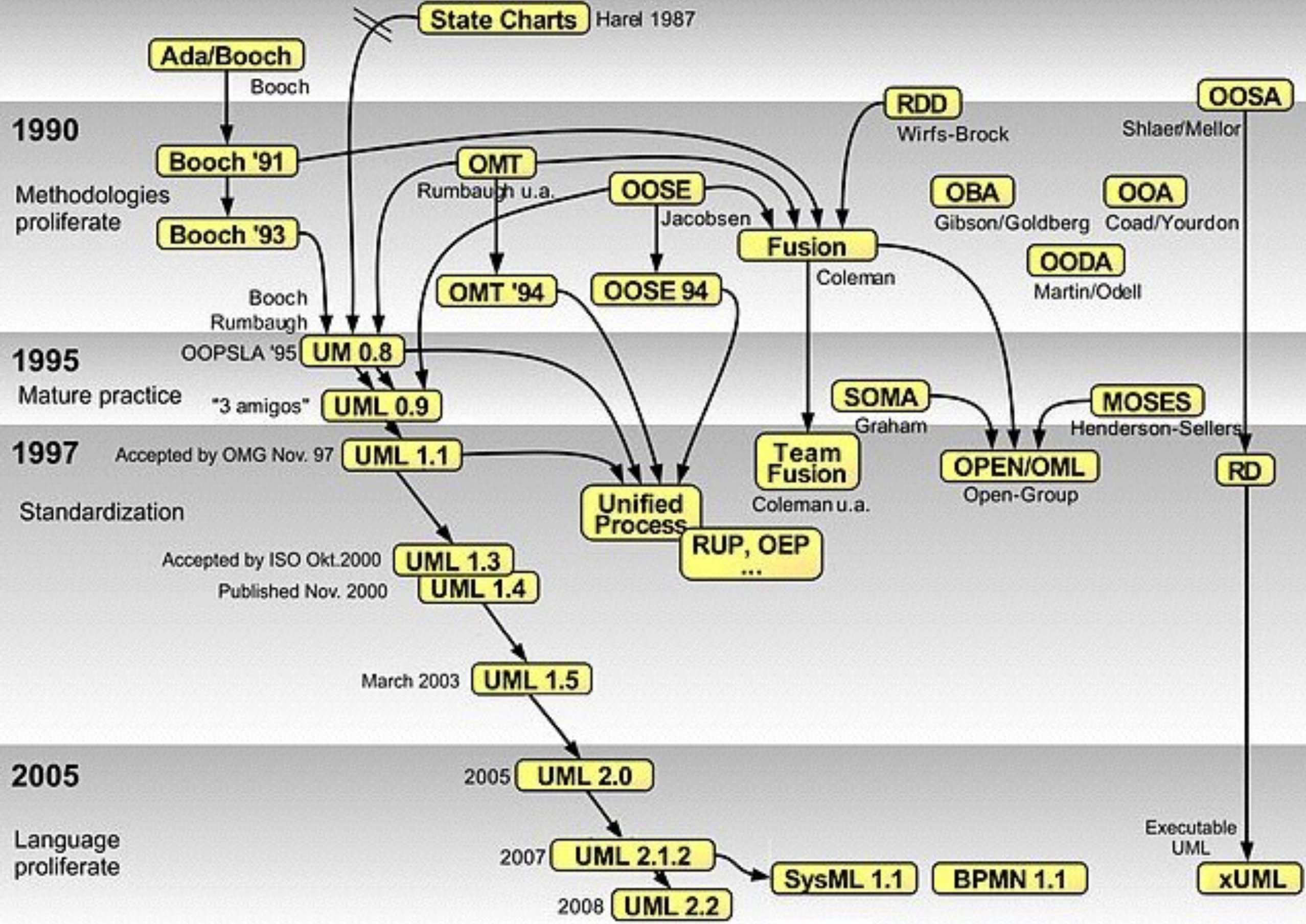
At Ericsson until 1994, developed use cases and the CASE tool Objectory, at IBM Rational since 1995

Developed the Booch method ("clouds"), ACM Fellow 1995, and IBM Fellow 2003

- Convergence of different notations used in object-oriented methods, mainly
  - OMT (James Rumbaugh and collegues), OOSE (Ivar Jacobson), Booch (Grady Booch)

- They also developed the Rational Unified Process, which became the Unified Process in 1999
  - Iterative process (analysis, design, implementation, and testing at each iteration)


State Charts
Harel 1987
Ada/Booch
Booch
1990
Booch '91
Booch '93
Methodologies
proliferate
OMT
Rumbaugh u.a.
OOSE
Jacobsen
RDD
Wirfs-Brock
OOSA
Shlaer/Mellor
OBA
Gibson/Goldberg
OOA
Coad/Yourdon
Fusion
Coleman
OODA
Martin/Odell
Booch
Rumbaugh
OOPSLA '95
UM 0.8
OMT '94
OOSE 94
1995
Mature practice
"3 amigos"
UML 0.9
SOMA
Graham
MOSES
Henderson-Sellers
1997
Accepted by OMG Nov. 97
UML 1.1
Team
Fusion
OPEN/OML
Open-Group
RD
Unified
Process
Coleman u.a.
RUP, OEP
...
Standardization
Accepted by ISO Okt.2000
UML 1.3
Published Nov. 2000
UML 1.4
March 2003
UML 1.5
2005
2005
UML 2.0
Language
proliferate
2007
UML 2.1.2
SysML 1.1
BPMN 1.1
Executable
UML
xUML
2008
UML 2.2


State Charts
Harel 1987
Ada/Booch
Booch
1990
Booch '91
Booch '93
Methodologies
proliferate
OMT
Rumbaugh u.a.
OOSE
Jacobsen
RDD
Wirfs-Brock
OOSA
Shlaer/Mellor
OBA
Gibson/Goldberg
OOA
Coad/Yourdon
Fusion
Coleman
OODA
Martin/Odell
Booch
Rumbaugh
OOPSLA '95
UM 0.8
OMT '94
OOSE 94
1995
Mature practice
"3 amigos"
UML 0.9
SOMA
Graham
MOSES
Henderson-Sellers
1997
Accepted by OMG Nov. 97
UML 1.1
Team
Fusion
OPEN/OML
Open-Group
RD
Unified
Process
Coleman u.a.
RUP, OEP
...
Standardization
Accepted by ISO Okt.2000
Published Nov. 2000
UML 1.3
UML 1.4
March 2003
UML 1.5
2005
2005
UML 2.0
Language
proliferate
2007
UML 2.1.2
SysML 1.1
BPMN 1.1
Executable
UML
xUML
2008
UML 2.2

2015 - UML 2.5

Why UML?

- Provide users a expressive visual modeling language.
- Facilitate communication among developers. UML became the lingua franca in modeling.
- Incorporate the object-oriented community's consensus on core modeling concepts.
- Provide extensibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development processes.
- Provide a formal basis for understanding the modeling language.
- Support the creation of object-oriented CASE tools.
- Support higher-level development concepts such as collaborations, frameworks, patterns, and components.
- Win the 'war of notations'.

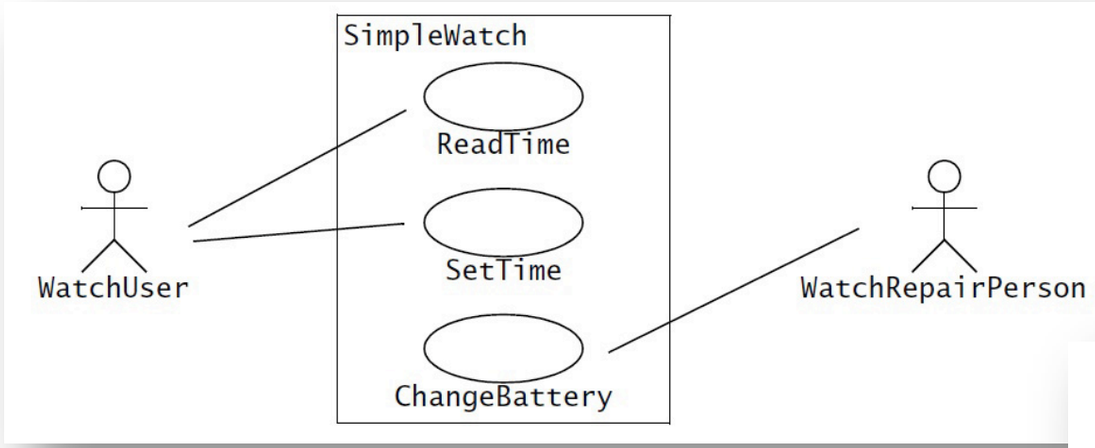
UML

- 3 views on UML
  - UML as sketch.
  - UML as blueprint.
    - Before ~ specification.
    - After ~ documentation.
  - UML as a programming language.

- In the OOSE book
  - "Just" a diagramming notation standard.
  - Trivial and relatively unimportant.
  - Not a method, process, or design guide.

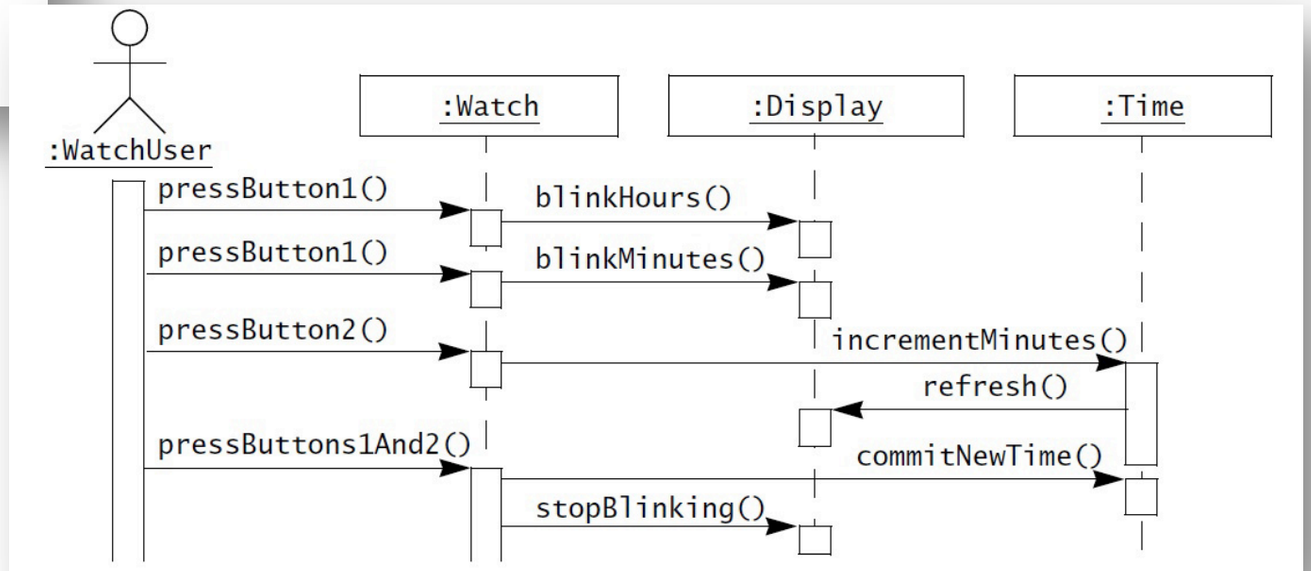
- In this course
  - You need to be able to draw syntactically correct UML diagrams – important part of the exam.

Examples

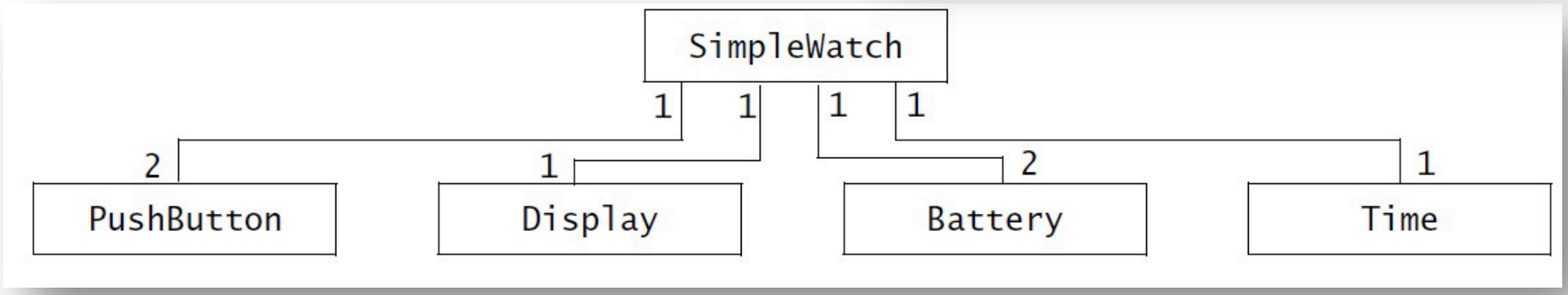


Use Case Diagram

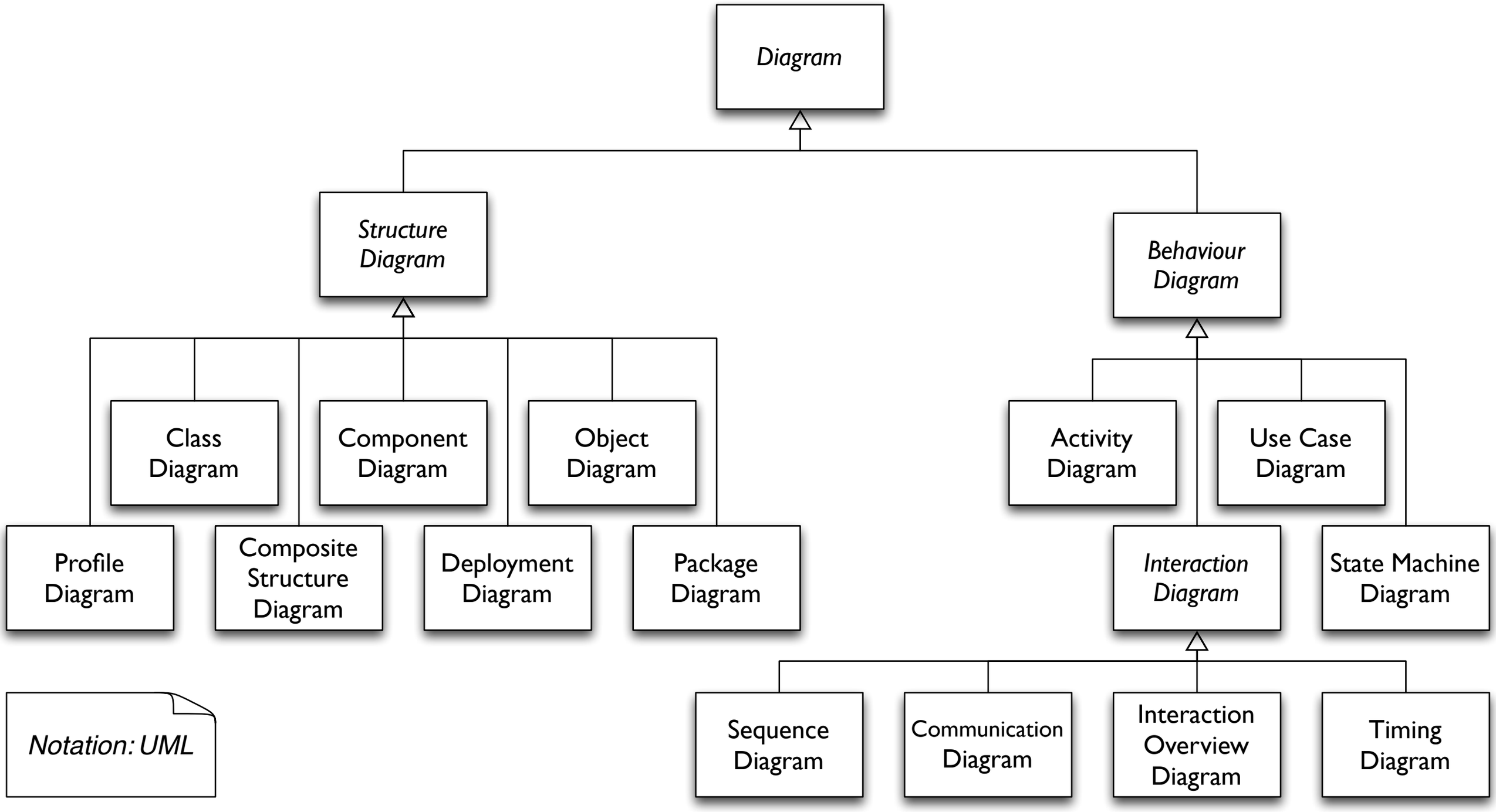
Interaction Diagram

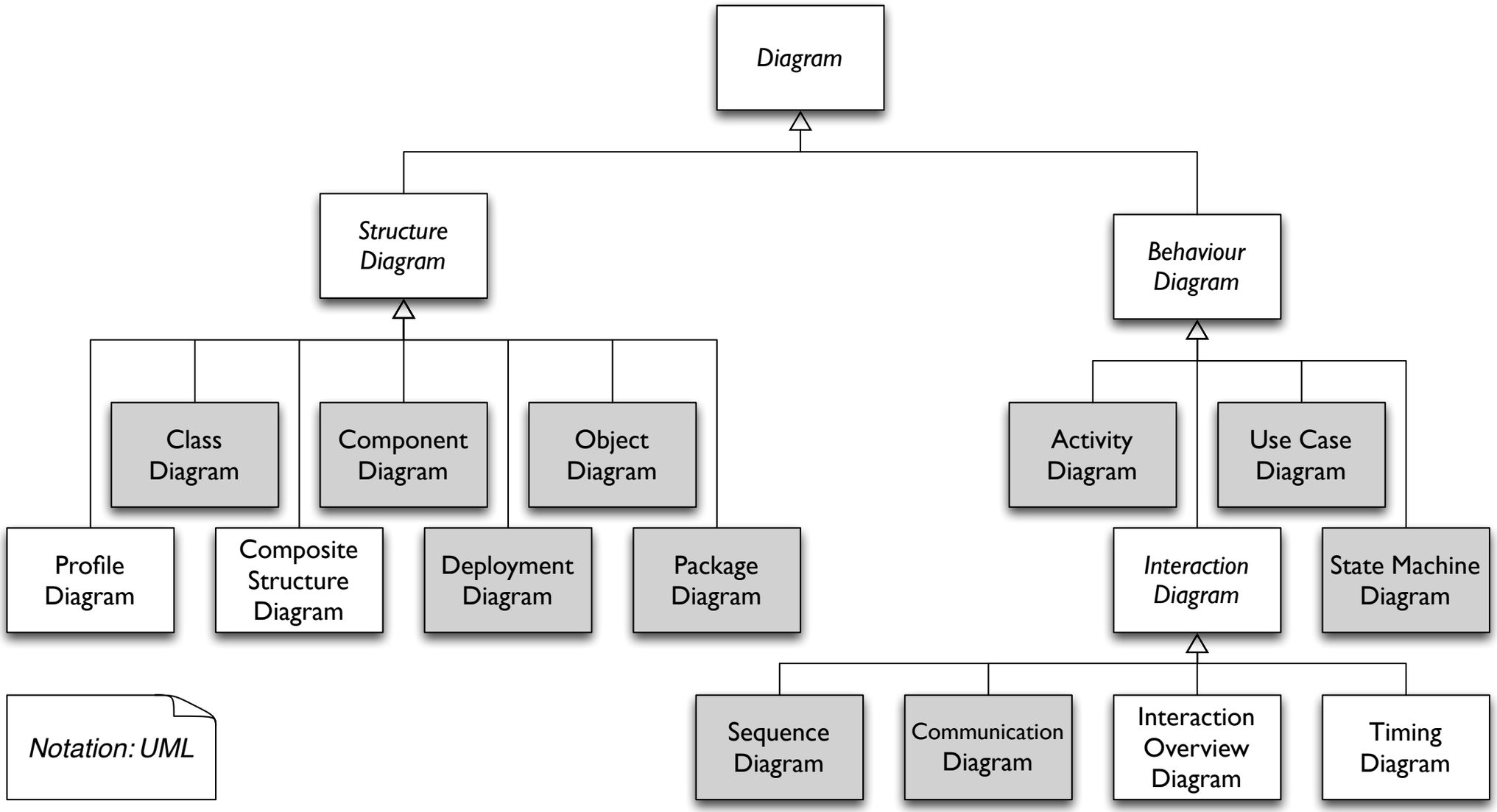


Object Diagram

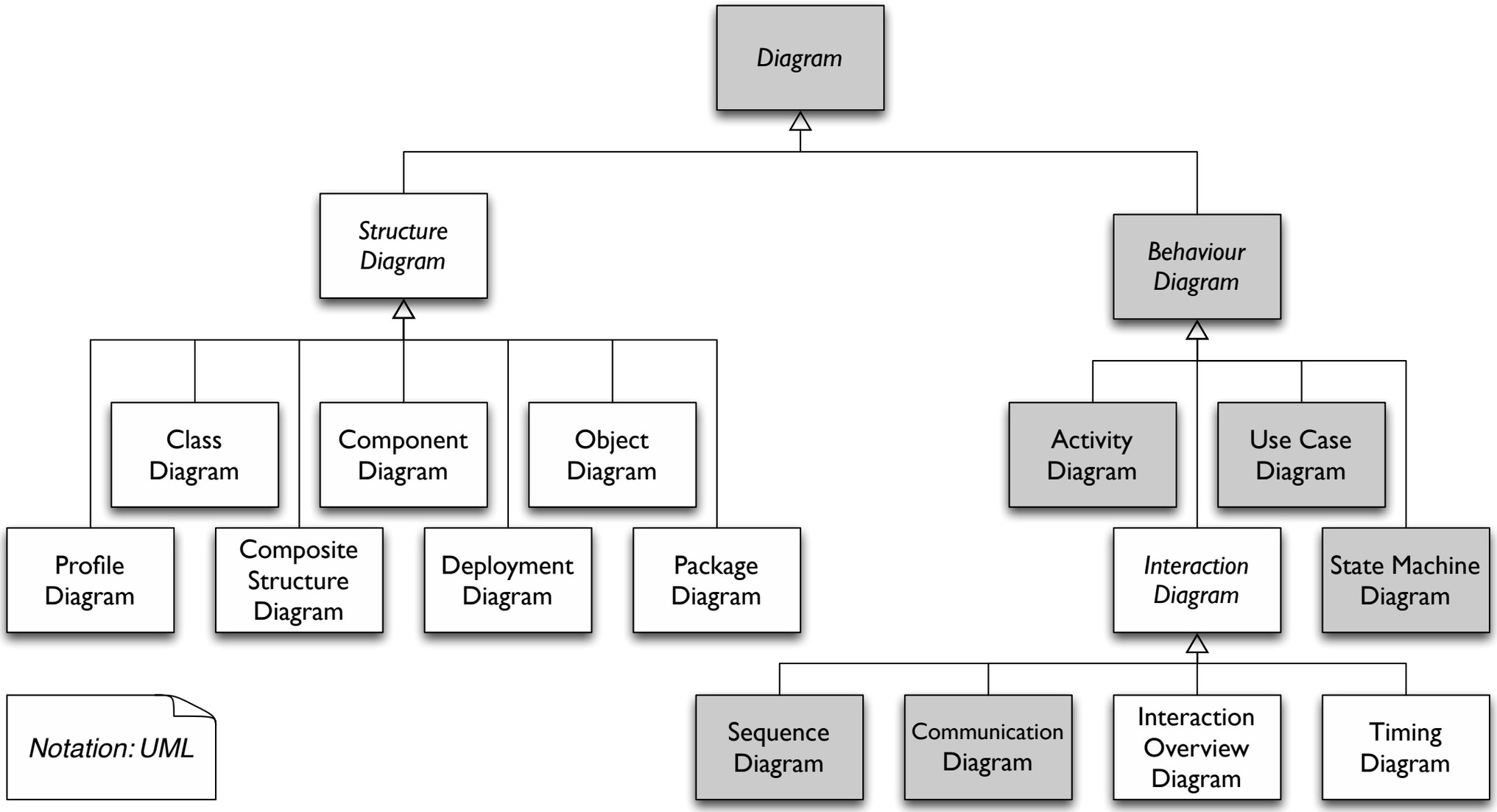


Hierarchy of diagrams in UML 2.x





Diving into UML



Diving into UML

Core conventions

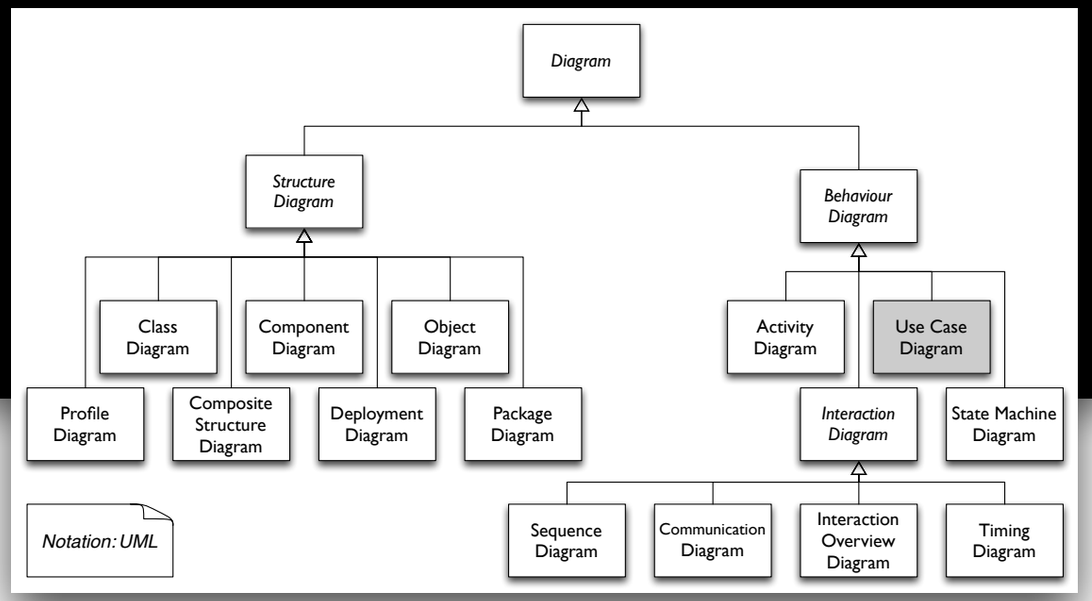
- All UML Diagrams denote graphs of nodes and edges
  - Nodes are entities and drawn as rectangles or ovals
  - Rectangles denote classes or instances
  - Ovals denote functions

- Names of Classes are not underlined
  - SimpleWatch
  - Firefighter

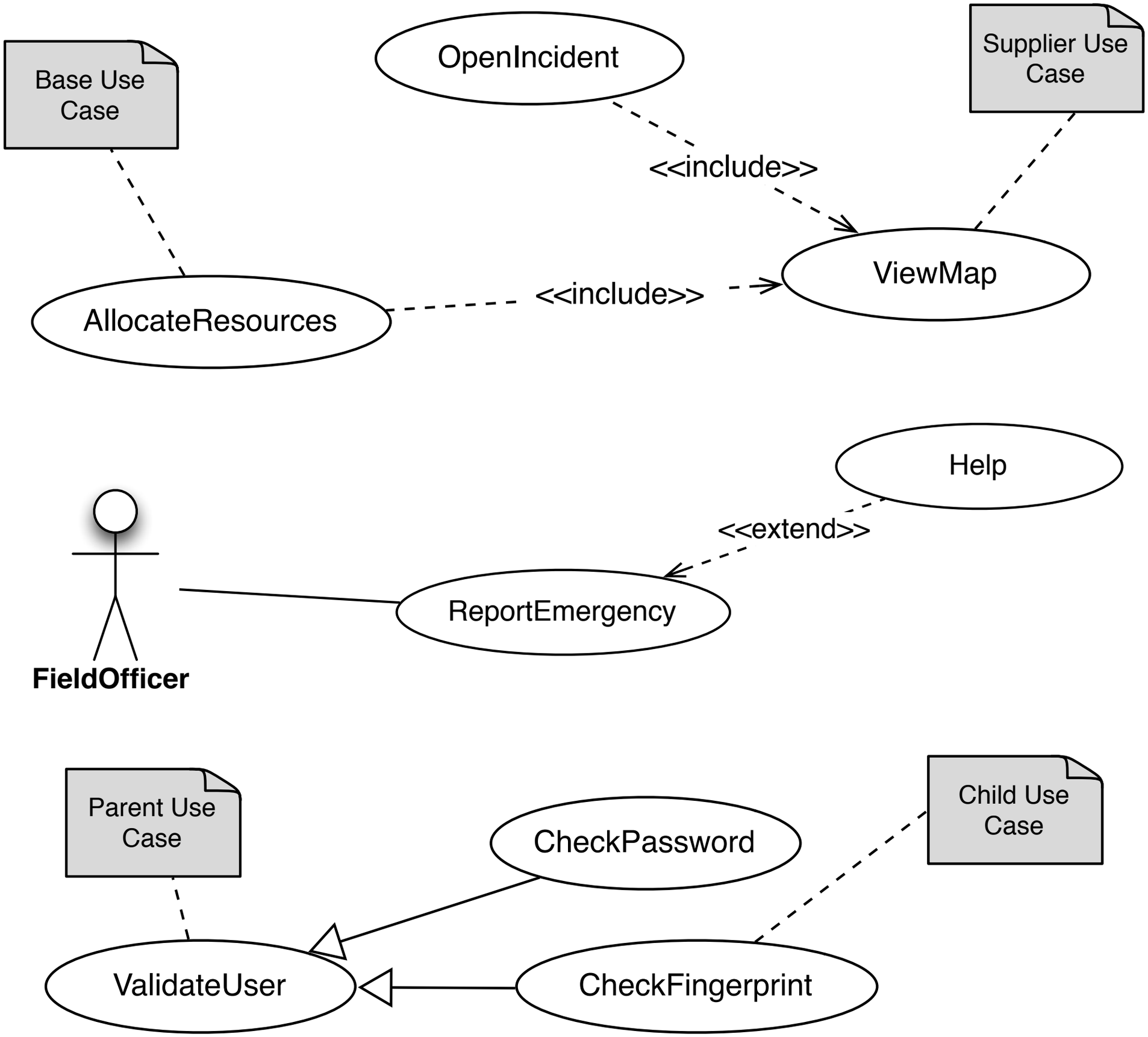
- Names of Instances are underlined
  - myWatch:SimpleWatch
  - Joe:Firefighter

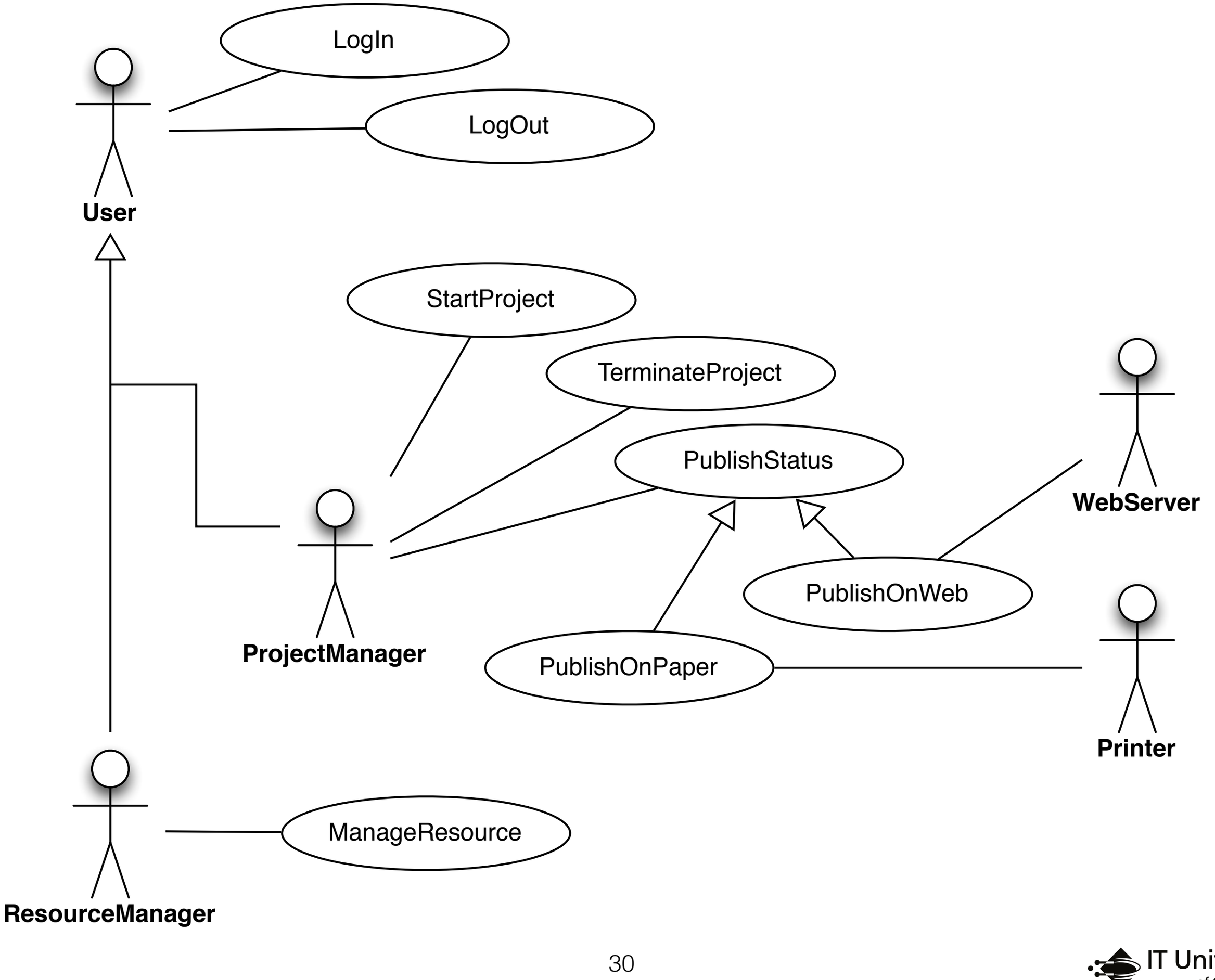
- An edge between two nodes denotes a relationship between the corresponding entities

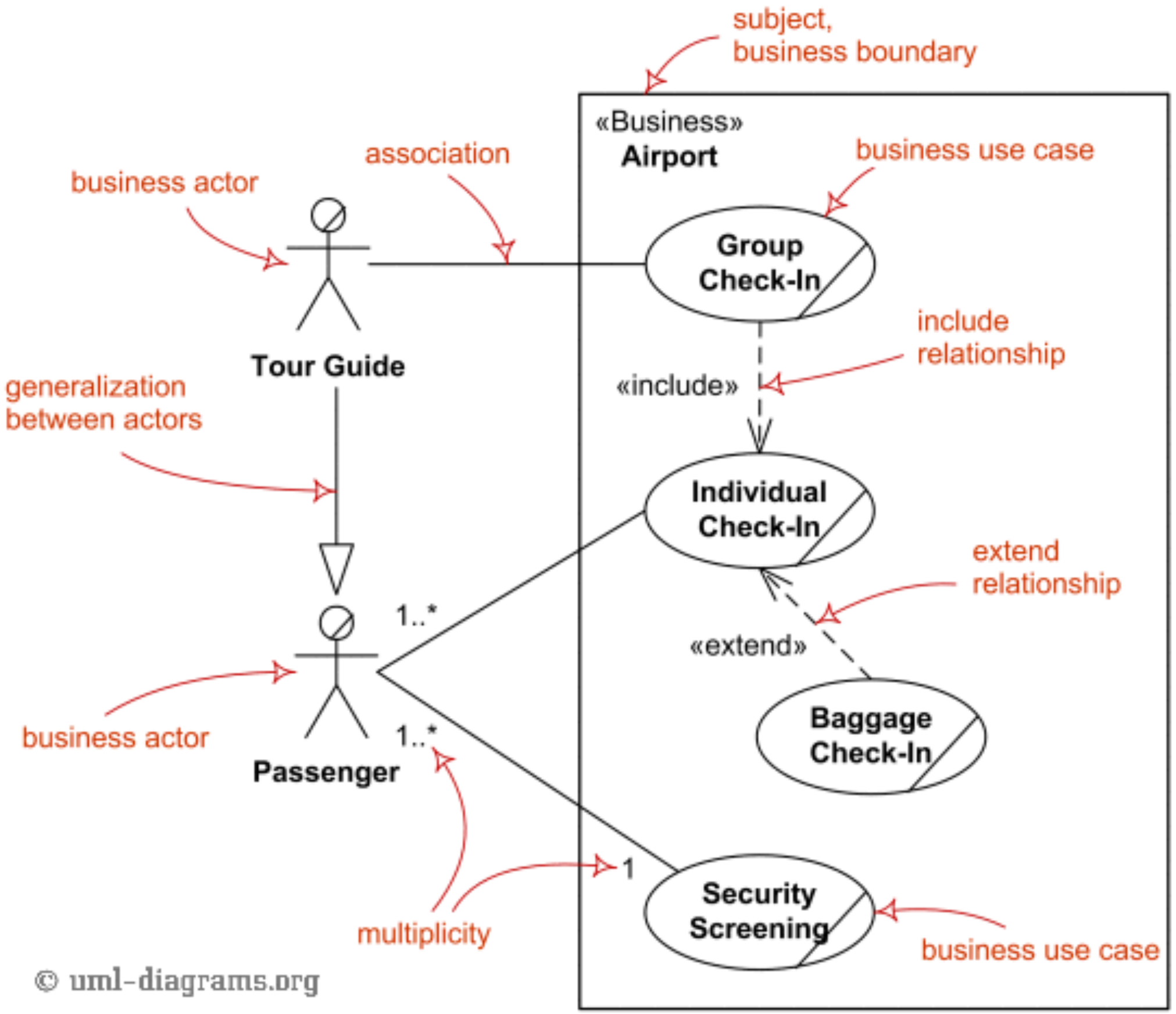
Use case diagram



- Behaviour diagrams.
- They capture requirements.
- A use case captures a specific functionality offered by the system.
- They abstract scenarios.
- They comprise of two parts:
  - the use case diagram (the "sticky man" diagram).
    - actors (roles, not only humans), use cases, and associations (includes, extends, generalization).
  - the use case text.


Base Use Case
OpenIncident
Supplier Use Case
<<include>>
ViewMap
AllocateResources
<<include>>
Help
<<extend>>
ReportEmergency
FieldOfficer
Parent Use Case
CheckPassword
Child Use Case
ValidateUser
CheckFingerprint


LogIn
LogOut
User
StartProject
TerminateProject
PublishStatus
WebServer
ProjectManager
PublishOnWeb
PublishOnPaper
Printer
ManageResource
ResourceManager


subject,
business boundary
«Business»
Airport
association
business actor
business use case
Group
Check-In
include
relationship
«include»
Tour Guide
generalization
between actors
Individual
Check-In
extend
relationship
«extend»
Baggage
Check-In
1..*
business actor
1..*
Passenger
1
Security
Screening
multiplicity
business use case
© uml-diagrams.org

Use case name	`ReportEmergency`
Participating actors	Initiated by `FieldOfficer` Communicates with `Dispatcher`
Flow of events	1. The `FieldOfficer` activates the "Report Emergency" function of her terminal. 2. `FRIEND` responds by presenting a form to the officer. *The form includes an emergency type menu (general emergency, fire, transportation) and location, incident description, resource request, and hazardous material fields.* 3. The `FieldOfficer` completes the form by *specifying minimally the emergency type and description fields*. The `FieldOfficer` may also describe possible responses to the emergency situation *and request specific resources*. Once the form is completed, the `FieldOfficer` submits the form. 4. `FRIEND` receives the form and notifies the `Dispatcher` *by a pop-up dialog*. 5. The `Dispatcher` reviews the submitted information and creates an `Incident` in the database by invoking the `OpenIncident` use case. *All the information contained in the `FieldOfficer's` form is automatically included in the `Incident`. The `Dispatcher` selects a response by allocating resources to the `Incident` (with the `AllocateResources` use case) and acknowledges the emergency report by sending a short message to the `FieldOfficer`.* 6. `FRIEND` displays the acknowledgment and the selected response to the `FieldOfficer`.
Entry condition	• ...

Figure 4-10 Refined description for the `ReportEmergency` use case. Additions emphasized in *italics*.

Beware!

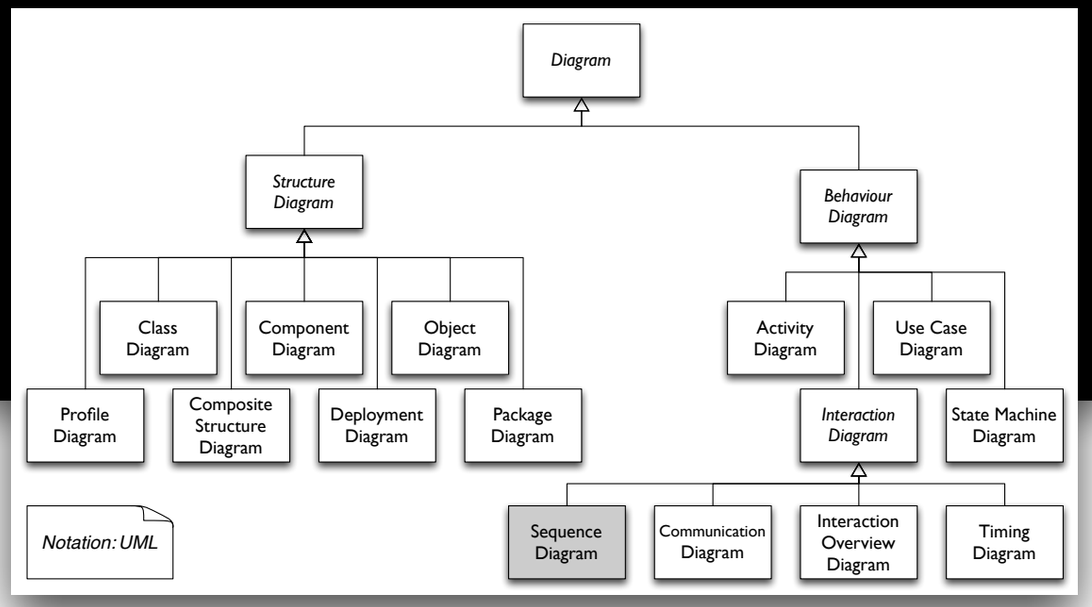
- Use case diagrams are dangerous, they are a double-edged sword as the functional decomposition they facilitate can be endless.

- Use cases must include also the text:
  - Name of Use Case
  - Actors: description of Actors involved in use case.
  - Entry condition: "This use case starts when…".
  - Flow of Events: free form, informal natural language.
  - Exit condition: "This use cases terminates when…".
  - Exceptions: describe what happens if things go wrong.
  - Quality Requirements: nonfunctional requirements, constraints.

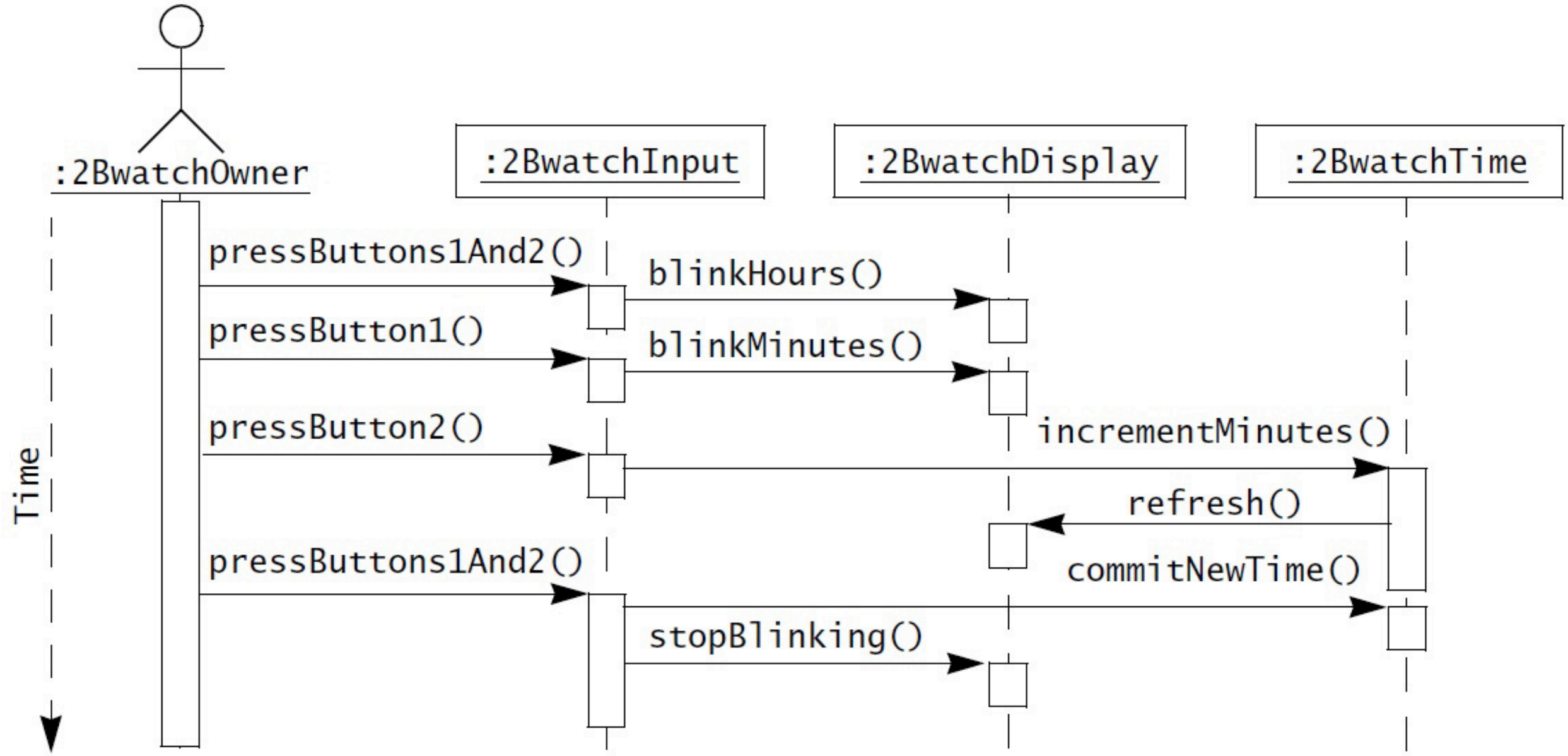
Exercise

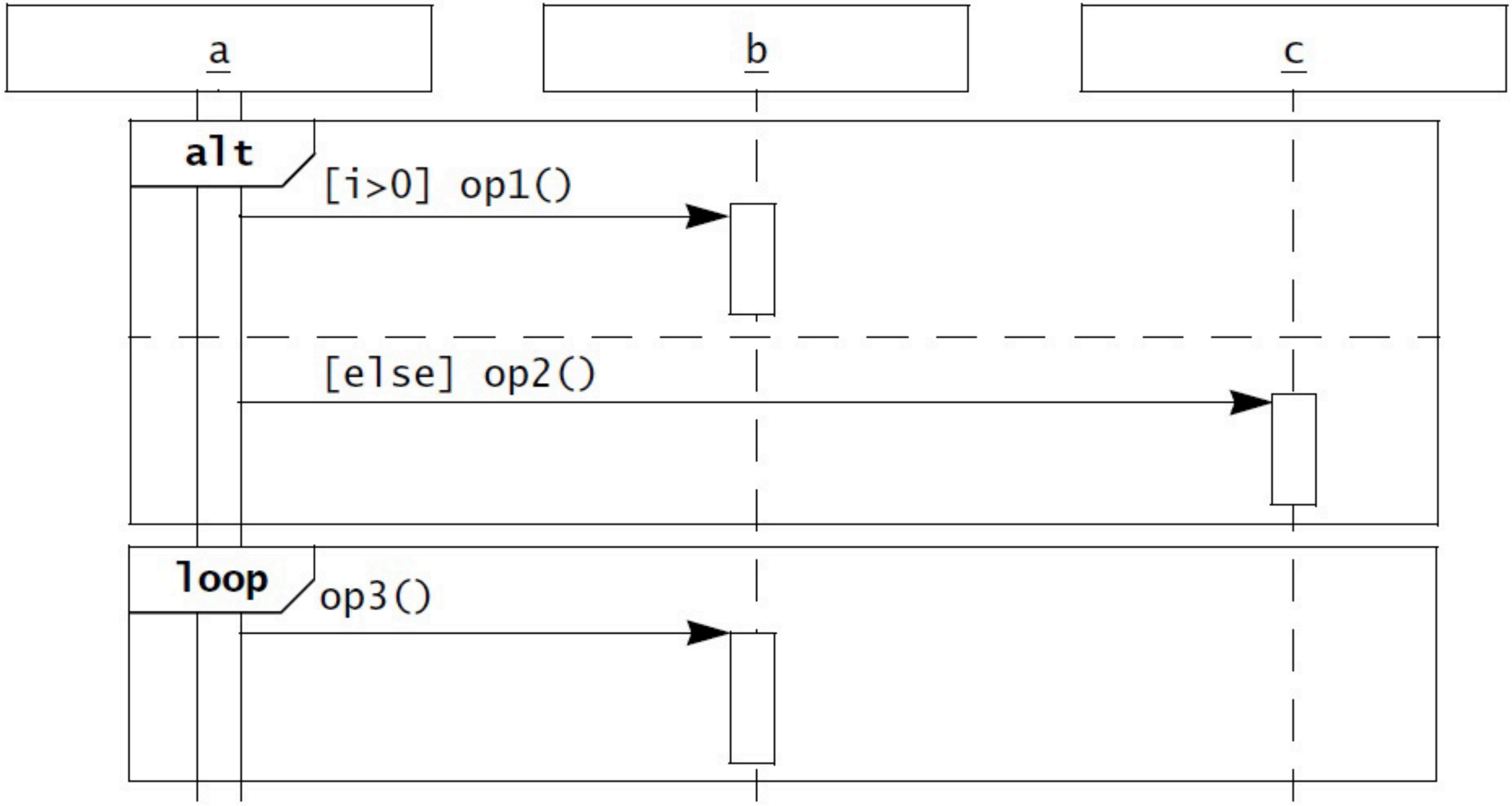
- Based on the10 functional requirements you identified about the Project for today, draw the sticky-man part of the use case diagram.

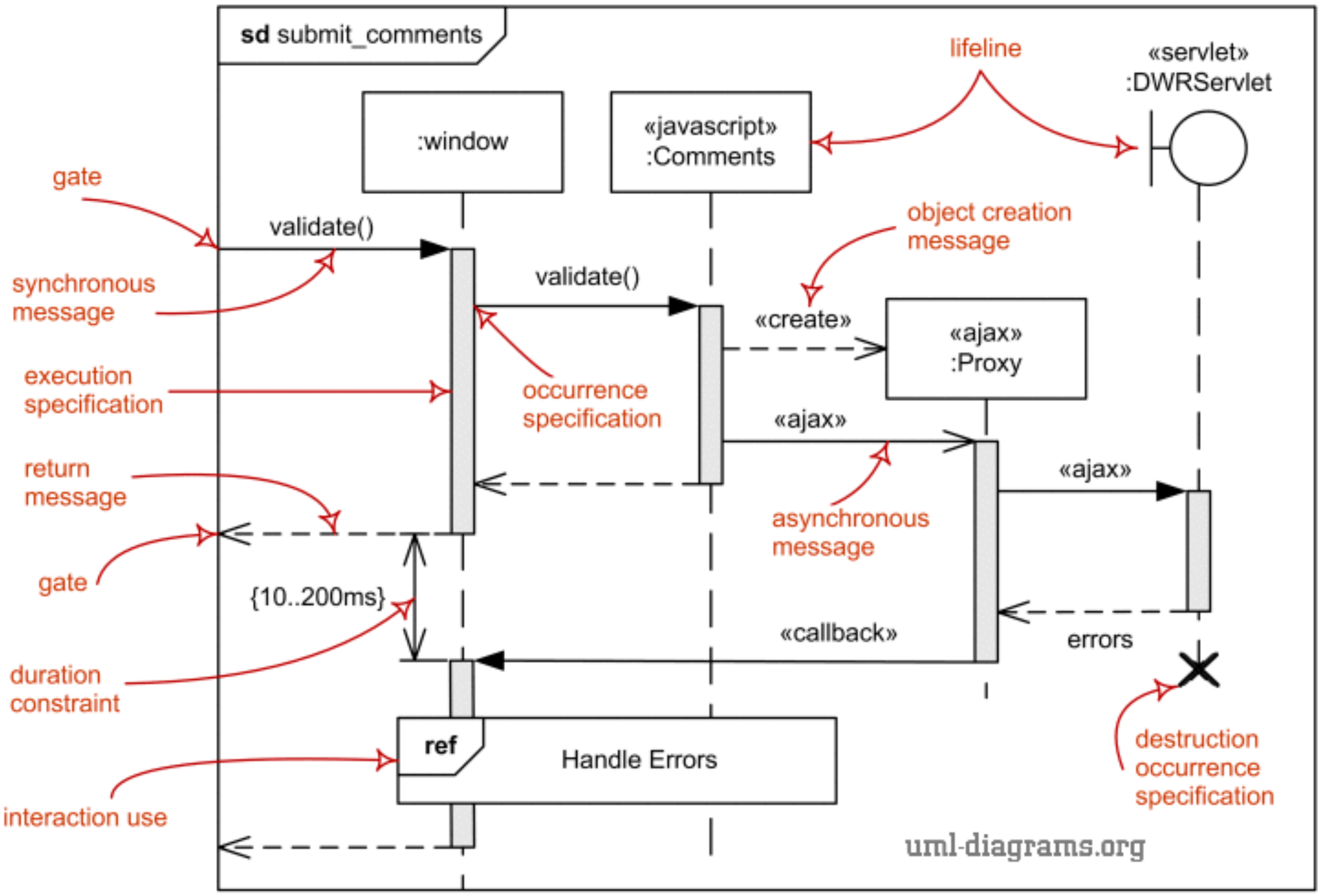
Sequence diagram



- Behaviour interaction diagrams.
- They specify the dynamic behaviour of a system as interactions between different objects.
- They capture the events that happen at runtime.
- They are "formal" and are used for specification and/or documentation.
- They focus on time and flow (e.g., concurrency) between objects.


:2BwatchOwner
:2BwatchInput
:2BwatchDisplay
:2BwatchTime
pressButtons1And2()
blinkHours()
pressButton1()
blinkMinutes()
pressButton2()
incrementMinutes()
refresh()
pressButtons1And2()
commitNewTime()
stopBlinking()
Time

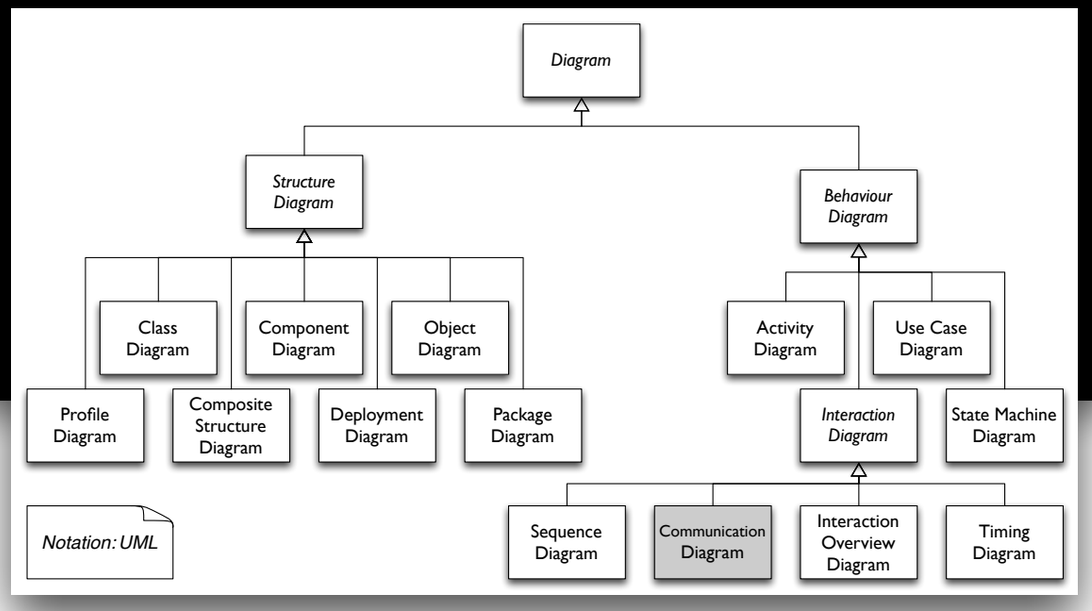

a
b
c
alt
[i>0] op1()
[else] op2()
loop
op3()


sd submit_comments
lifeline
«servlet»
:DWRServlet
:window
«javascript»
:Comments
gate
validate()
object creation
message
synchronous
message
validate()
«create»
«ajax»
:Proxy
execution
specification
occurrence
specification
«ajax»
return
message
asynchronous
message
«ajax»
gate
{10..200ms}
«callback»
errors
duration
constraint
ref
Handle Errors
destruction
occurrence
specification
interaction use
uml-diagrams.org

Other details

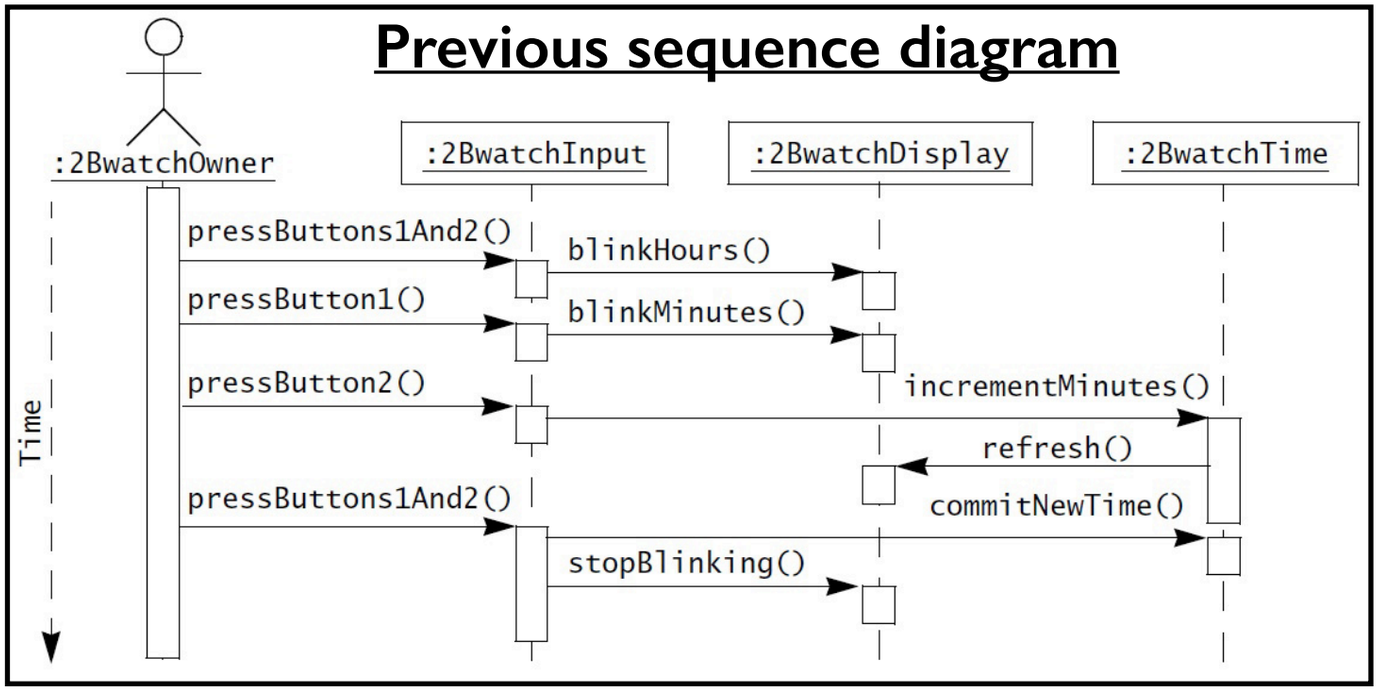
- object:Class
- Life line = the object has been created (new).
- Execution specification or activation box = a method of the object has its activation record on the stack.
- The “return” is not always necessary. If it is obvious, it can be omitted.
- The sequence diagrams provide an immediate overview of where the methods are distributed across the object/classes.

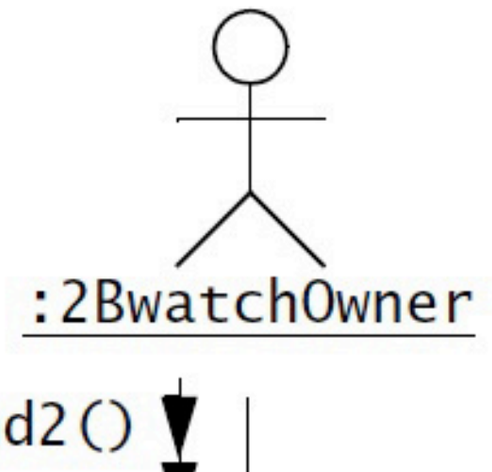
Communication diagram

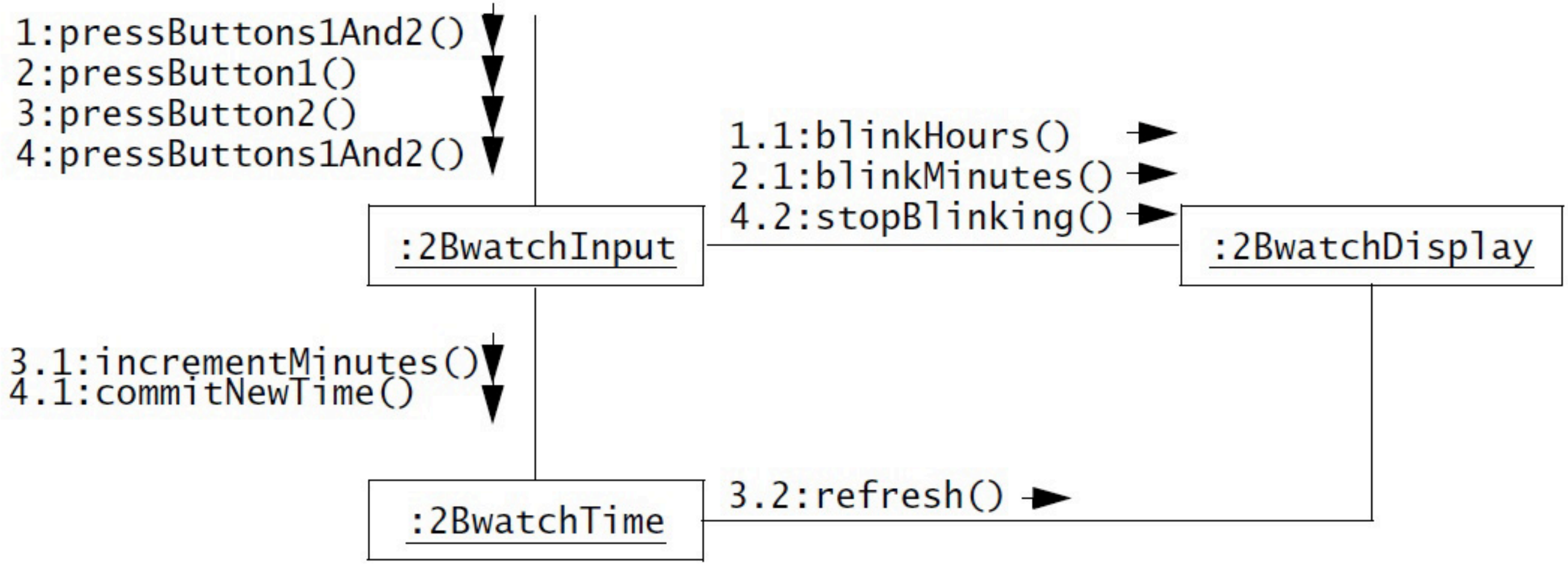


- Behaviour interaction diagrams.
- They specify the dynamic behaviour of a system as interactions between different objects.
- They contain the same information as the sequence diagrams.
- They are object diagrams + informations on the method invocations.
- They focus on the communication rather then the temporal order.
- They are more "informal" and used for sketching.

Previous sequence diagram







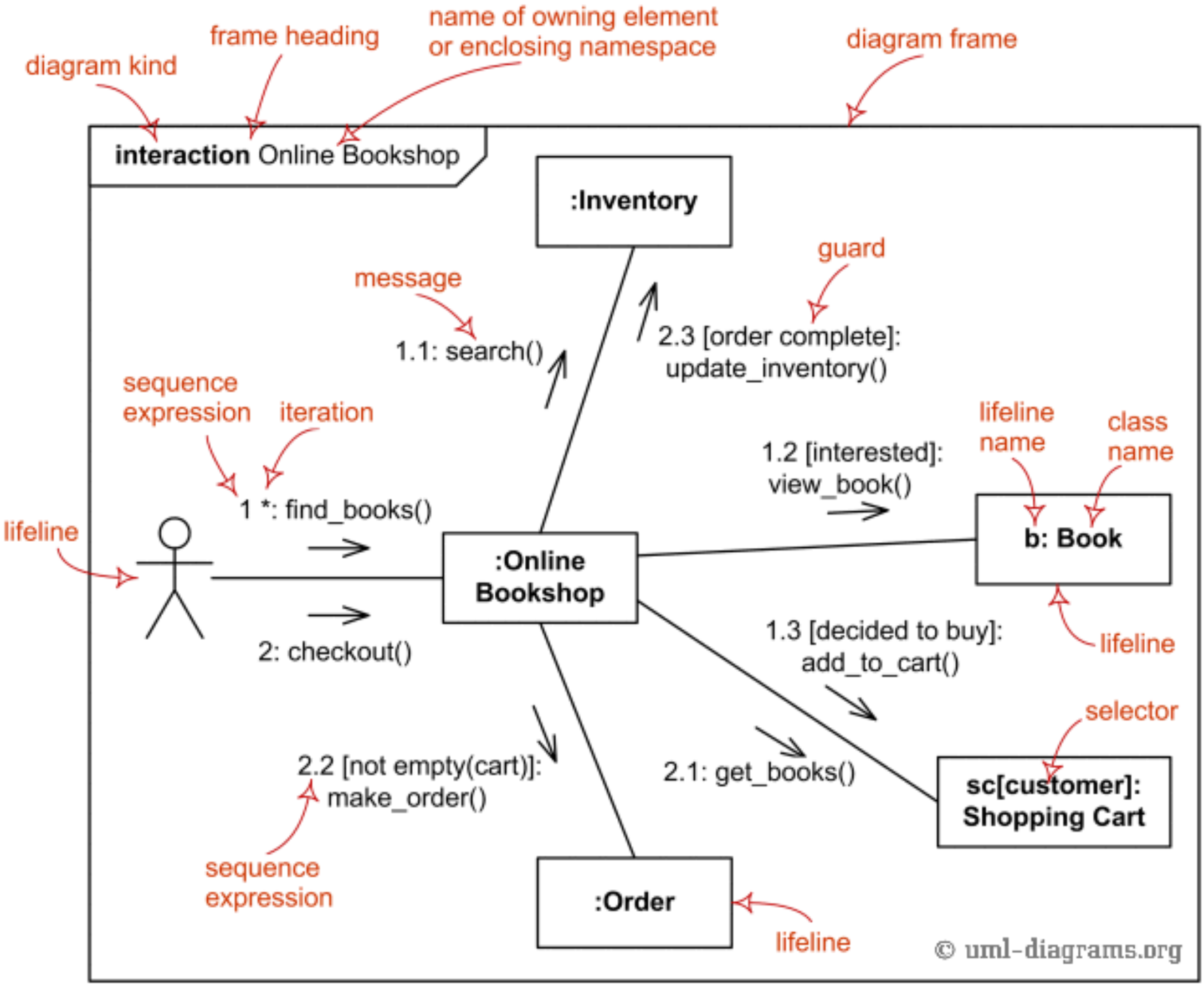
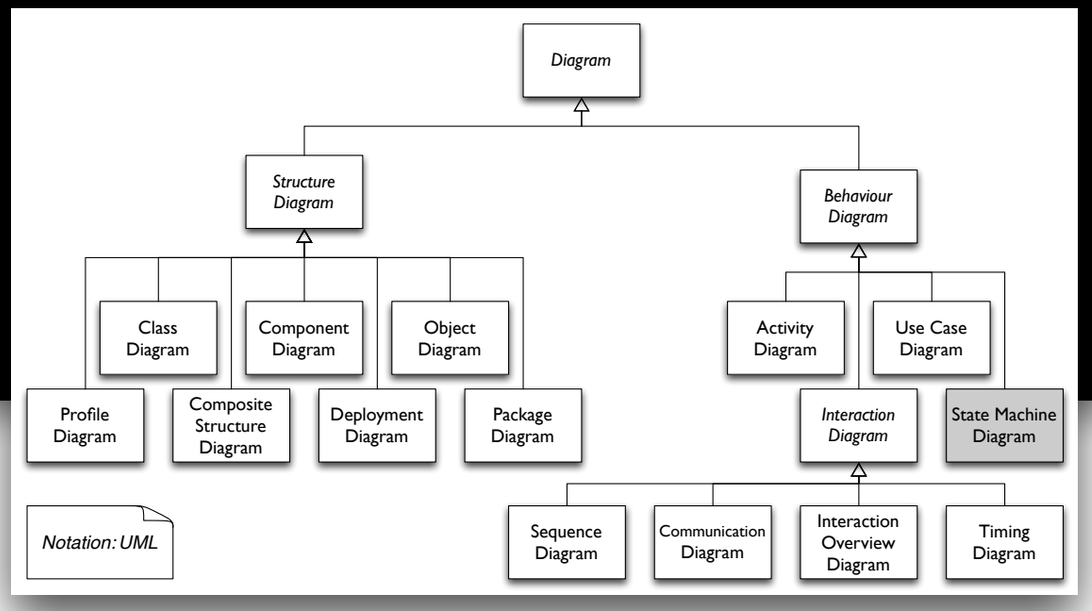
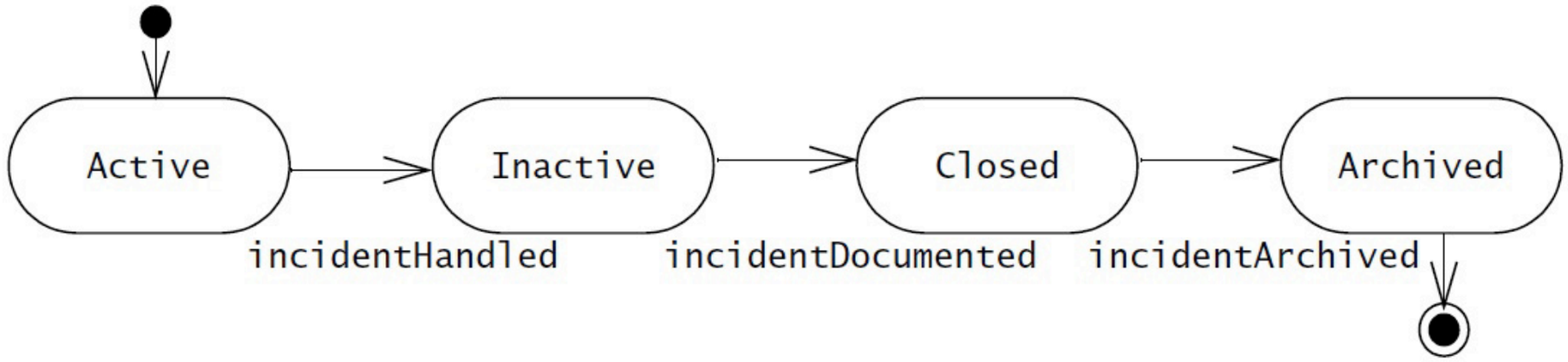


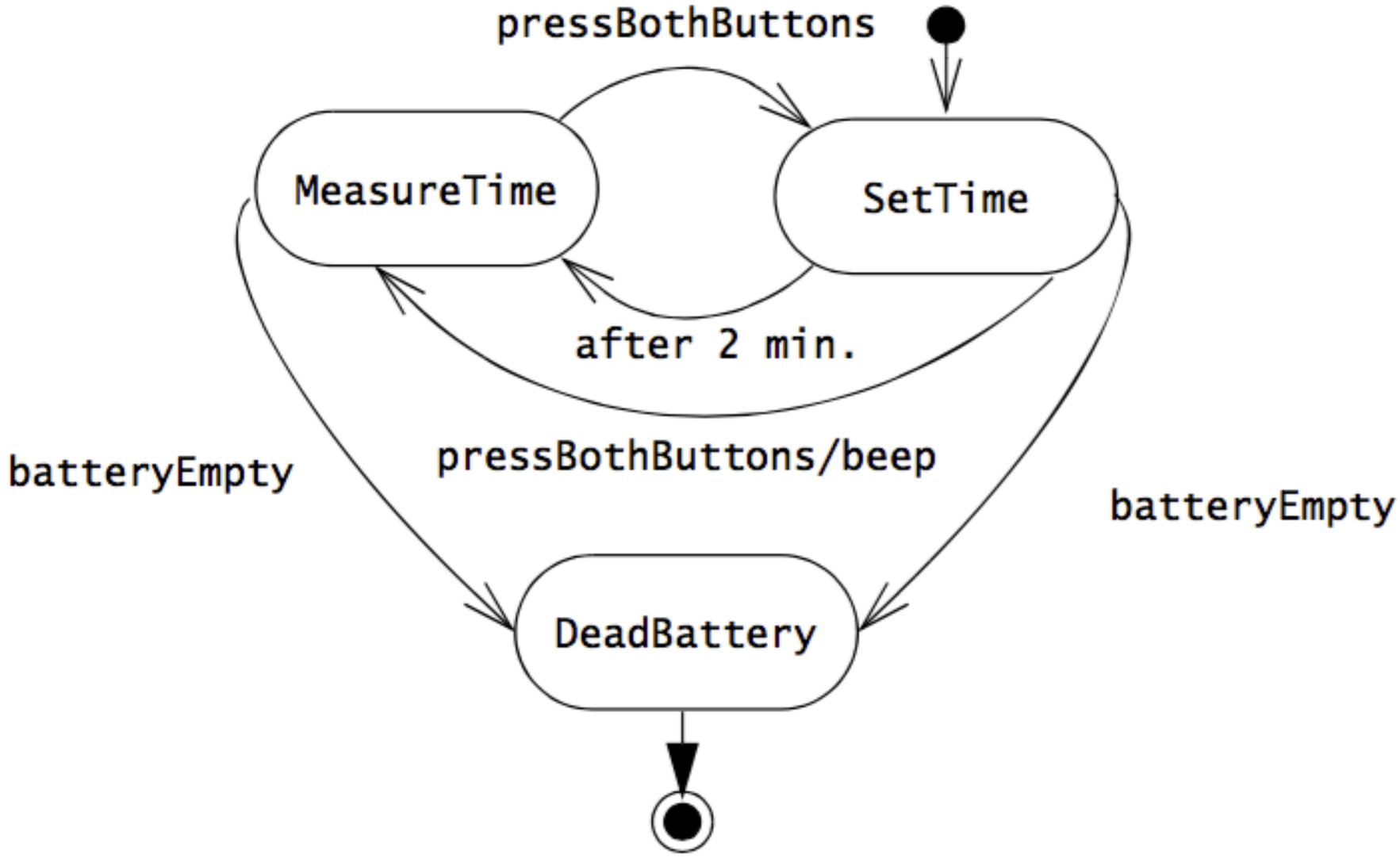
diagram kind
frame heading
name of owning element
or enclosing namespace
diagram frame
interaction Online Bookshop
:Inventory
message
1.1: search()
guard
2.3 [order complete]:
update_inventory()
sequence
expression
iteration
1 *: find_books()
1.2 [interested]:
view_book()
lifeline
name
class
name
lifeline
:Online
Bookshop
b: Book
2: checkout()
1.3 [decided to buy]:
add_to_cart()
lifeline
selector
2.2 [not empty(cart)]:
make_order()
2.1: get_books()
sc[customer]:
Shopping Cart
sequence
expression
:Order
lifeline
© uml-diagrams.org

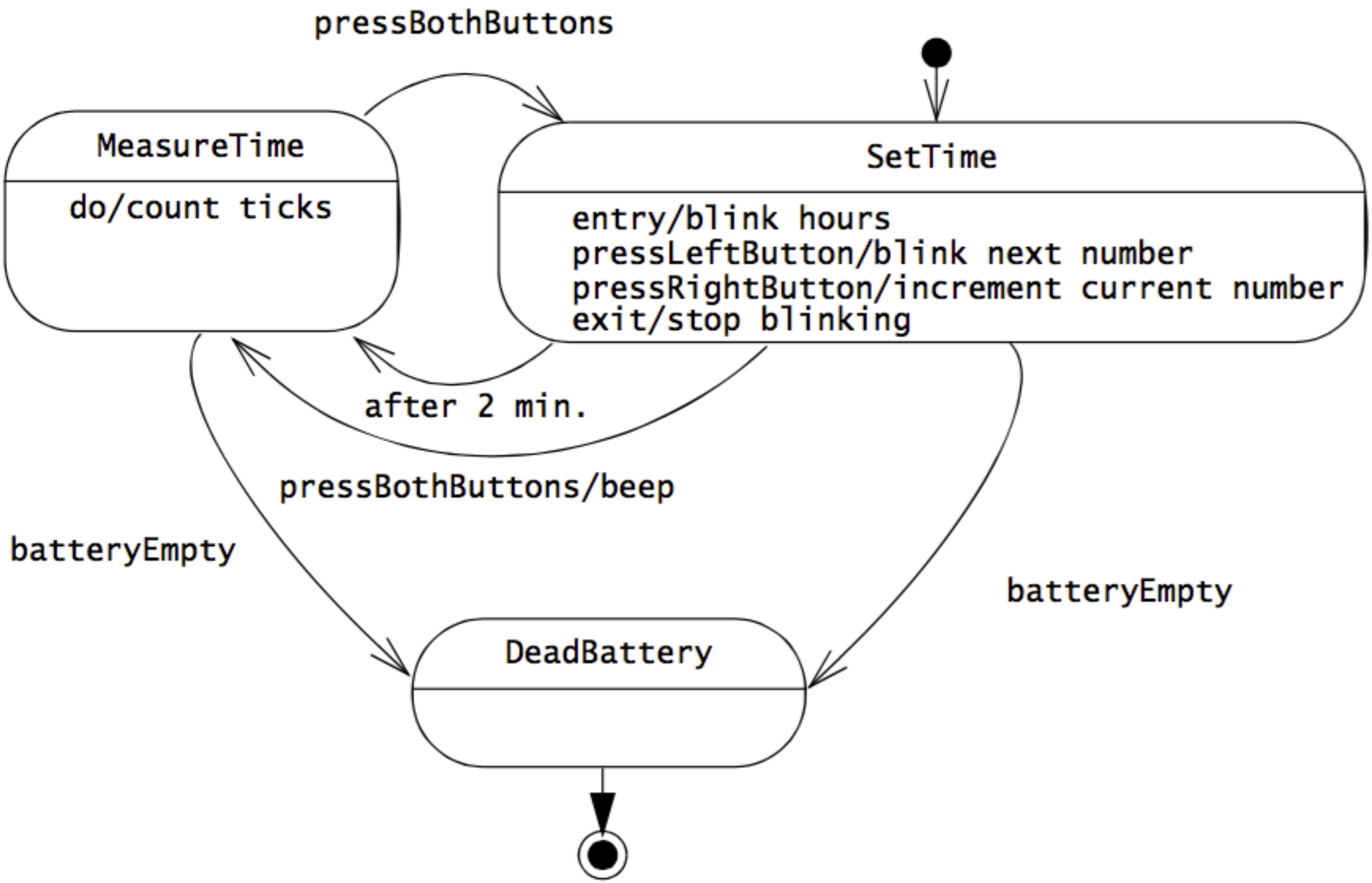
State machine diagram

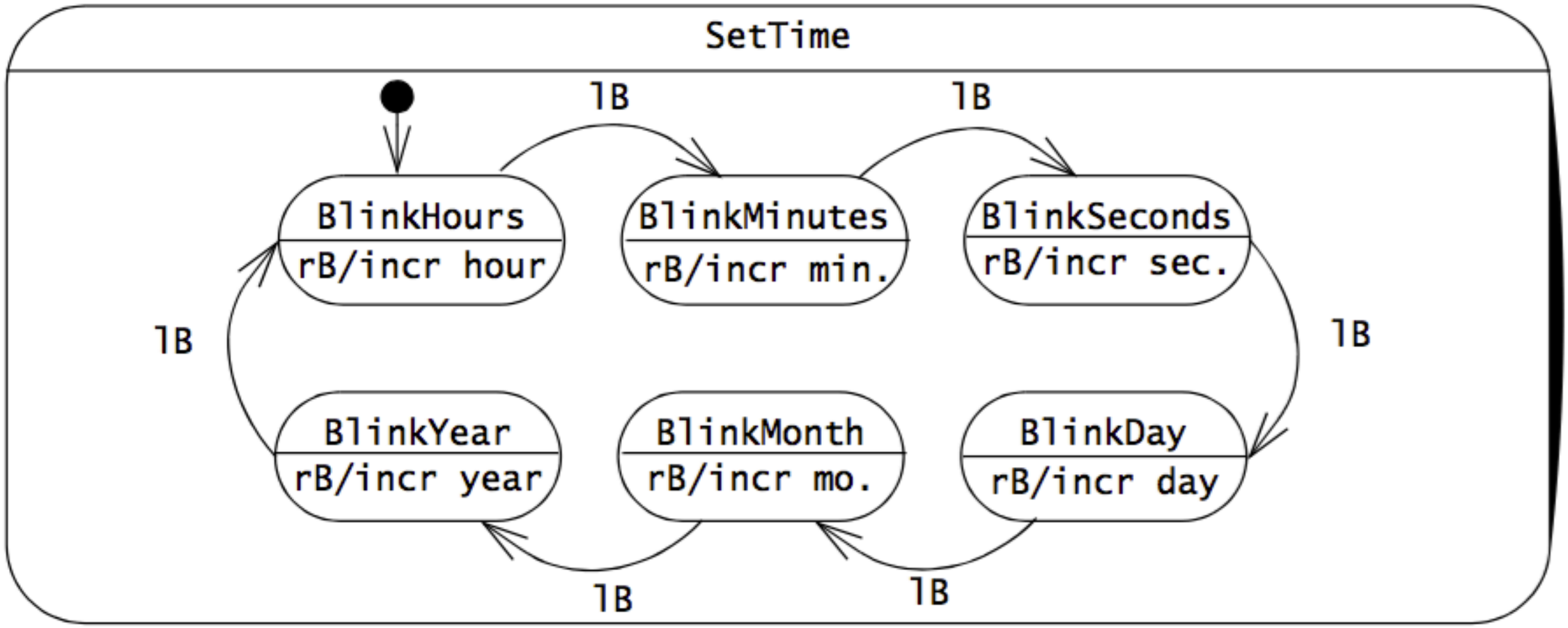


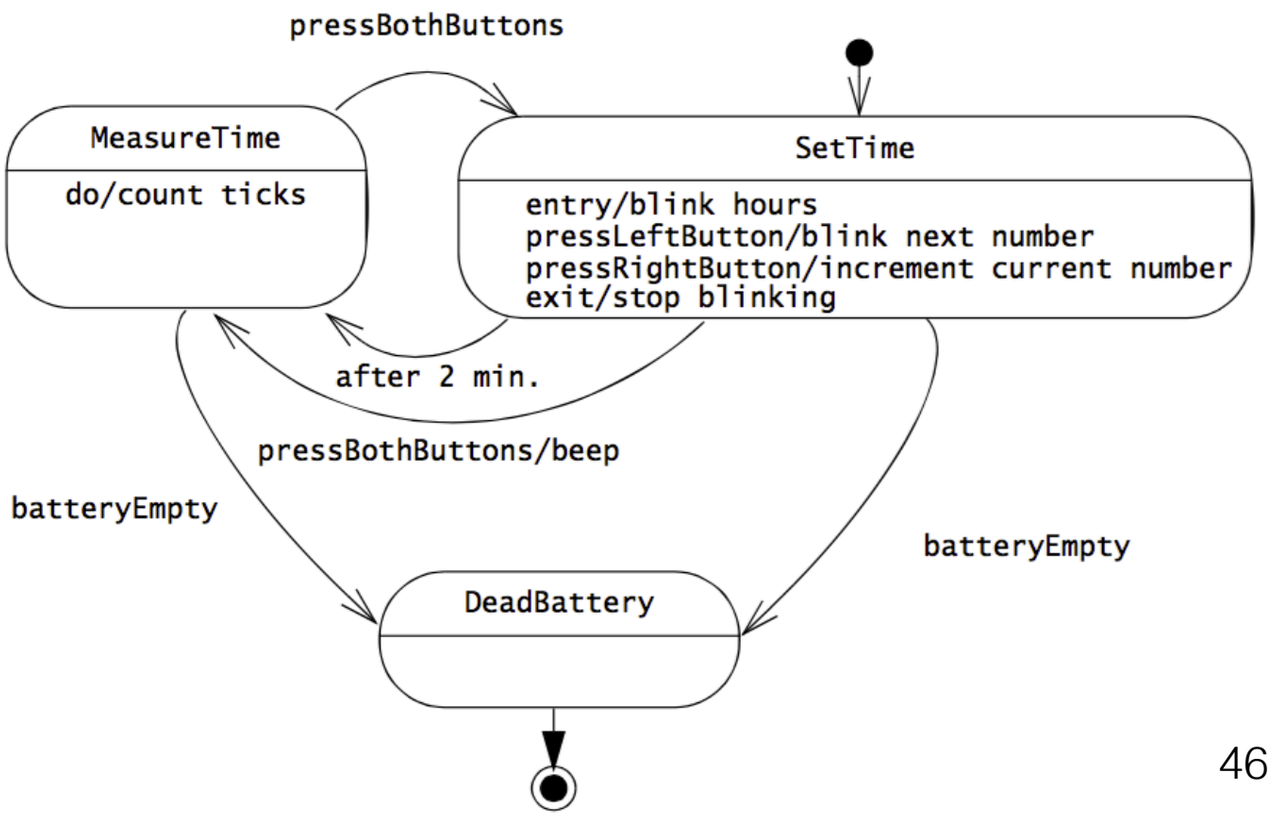
- Behaviour diagrams.
- They specify the dynamic behaviour of a single object.
- They model the sequence of states an object goes through at runtime in reaction to external events.
- They include an initial and final state.
- They show states and transactions.
- Substates are permitted.
- Transaction are labeled with events, guards, etc.

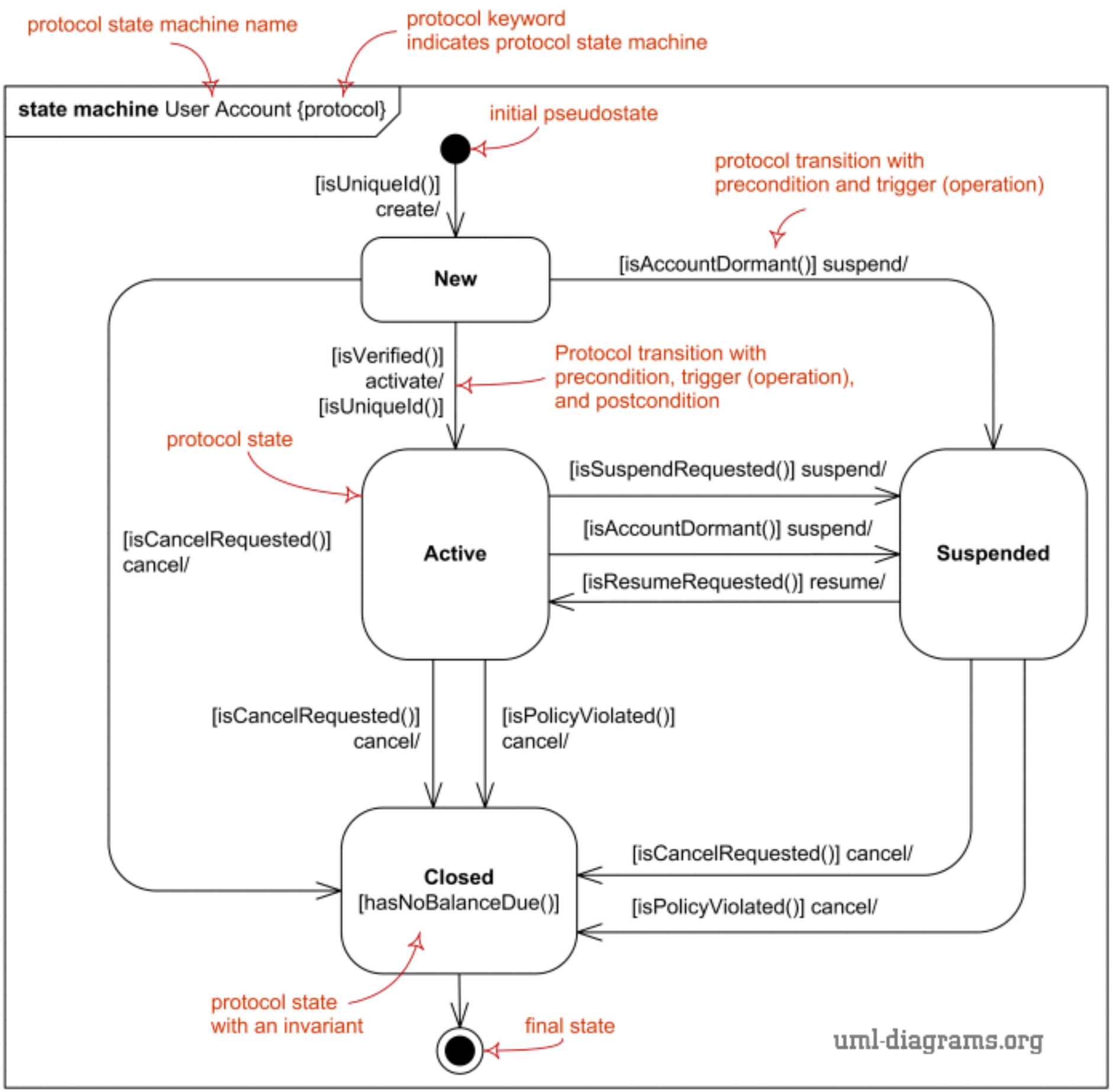

Active
Inactive
Closed
Archived
incidentHandled
incidentDocumented
incidentArchived


pressBothButtons
MeasureTime
SetTime
after 2 min.
batteryEmpty
pressBothButtons/beep
batteryEmpty
DeadBattery

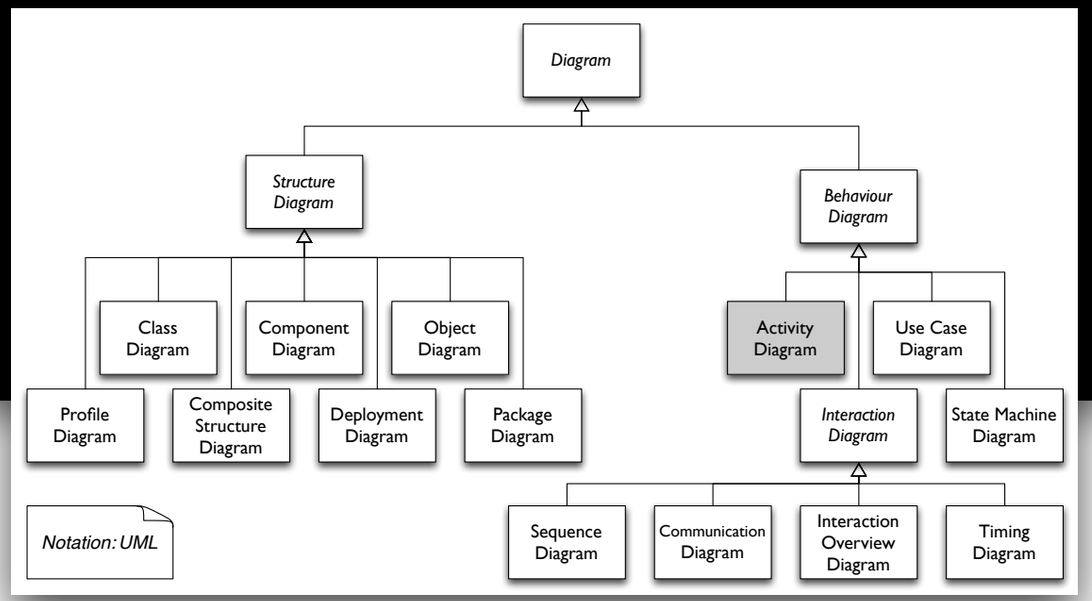

pressBothButtons
MeasureTime
do/count ticks
SetTime
entry/blink hours
pressLeftButton/blink next number
pressRightButton/increment current number
exit/stop blinking
after 2 min.
pressBothButtons/beep
batteryEmpty
batteryEmpty
DeadBattery


SetTime
lB
lB
BlinkHours
rB/incr hour
BlinkMinutes
rB/incr min.
BlinkSeconds
rB/incr sec.
lB
BlinkYear
rB/incr year
BlinkMonth
rB/incr mo.
BlinkDay
rB/incr day
lB
lB

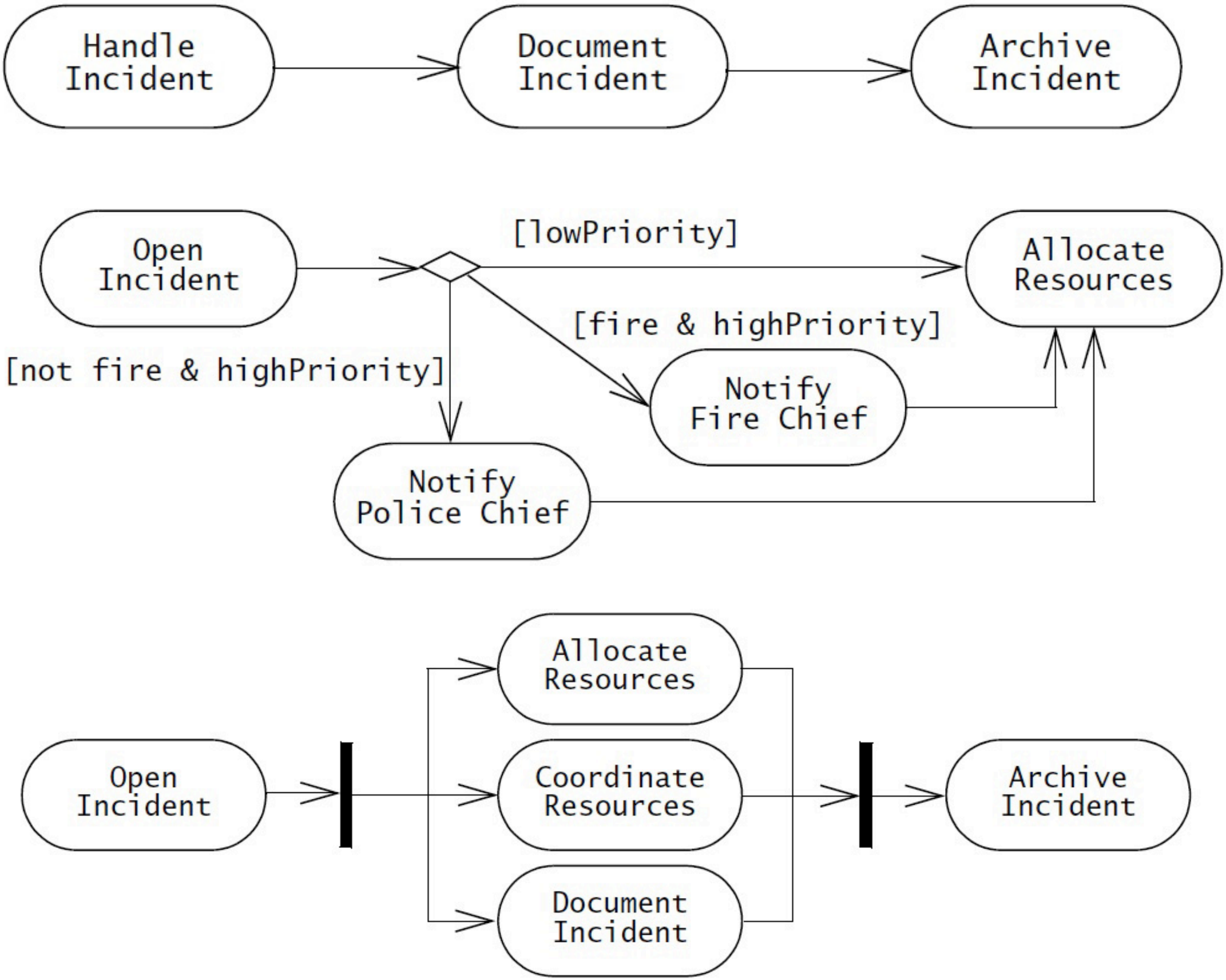

pressBothButtons
MeasureTime
do/count ticks
SetTime
entry/blink hours
pressLeftButton/blink next number
pressRightButton/increment current number
exit/stop blinking
after 2 min.
pressBothButtons/beep
batteryEmpty
batteryEmpty
DeadBattery

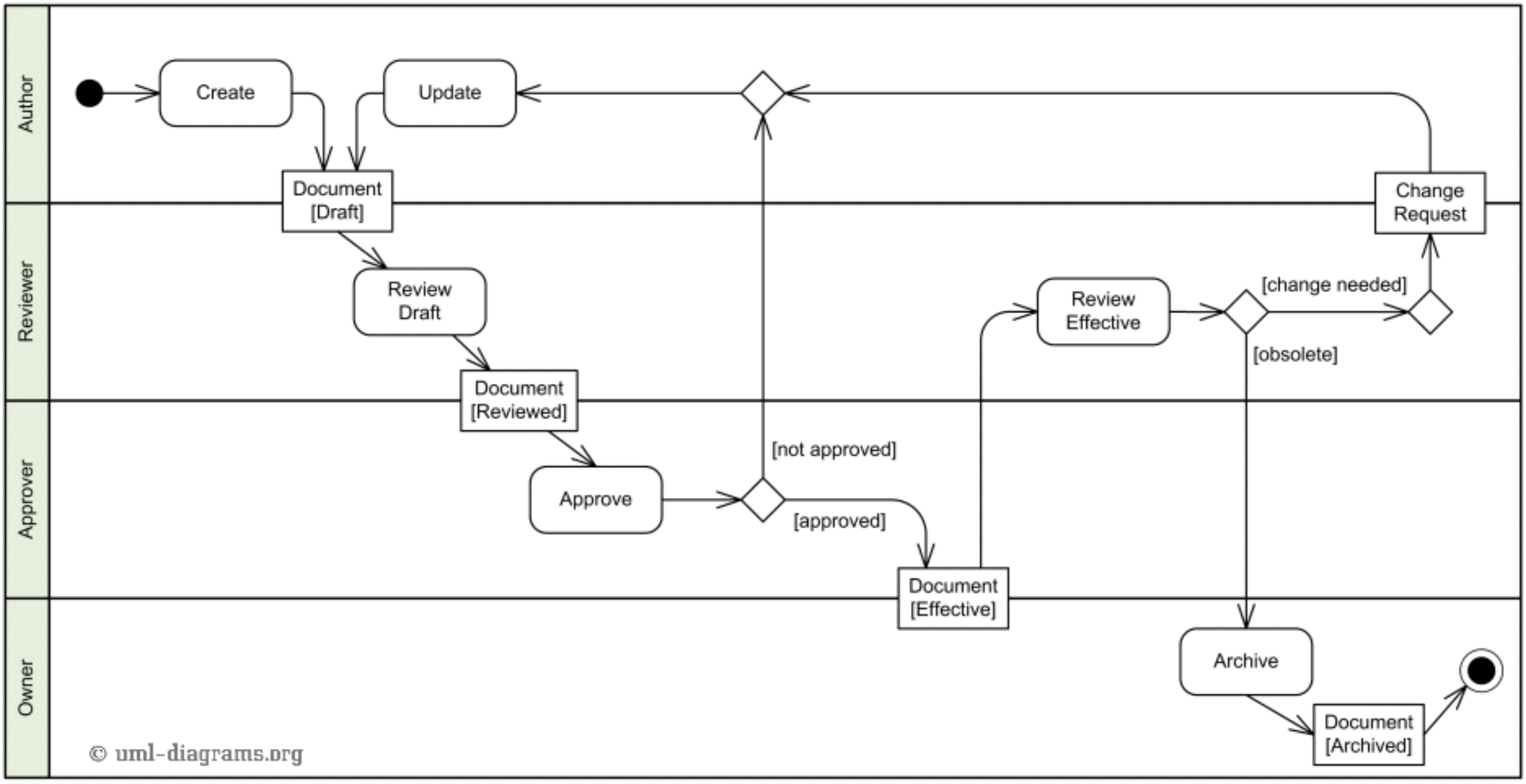

protocol state machine name
protocol keyword
indicates protocol state machine
state machine User Account {protocol}
initial pseudostate
[isUniqueId()]
create/
protocol transition with
precondition and trigger (operation)
New
[isAccountDormant()] suspend/
[isVerified()]
activate/
[isUniqueId()]
Protocol transition with
precondition, trigger (operation),
and postcondition
protocol state
[isSuspendRequested()] suspend/
[isCancelRequested()]
cancel/
Active
[isAccountDormant()] suspend/
Suspended
[isResumeRequested()] resume/
[isCancelRequested()]
cancel/
[isPolicyViolated()]
cancel/
[isCancelRequested()] cancel/
Closed
[hasNoBalanceDue()]
[isPolicyViolated()] cancel/
protocol state
with an invariant
final state
uml-diagrams.org

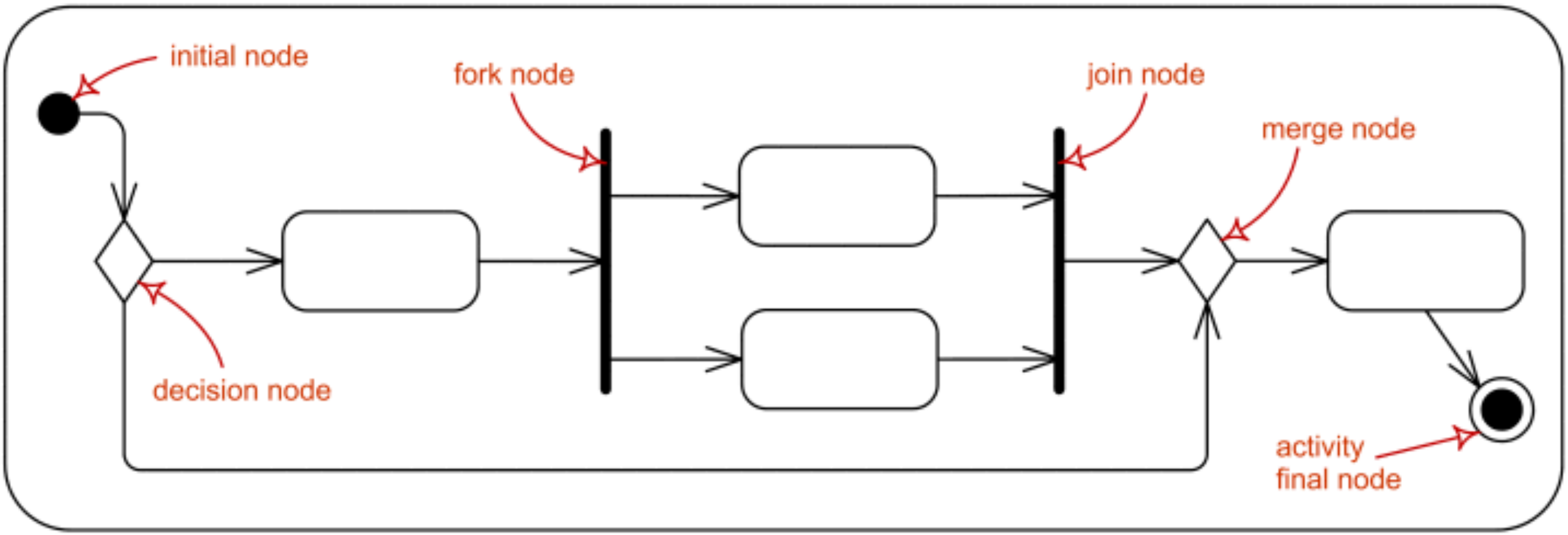
Activity diagram

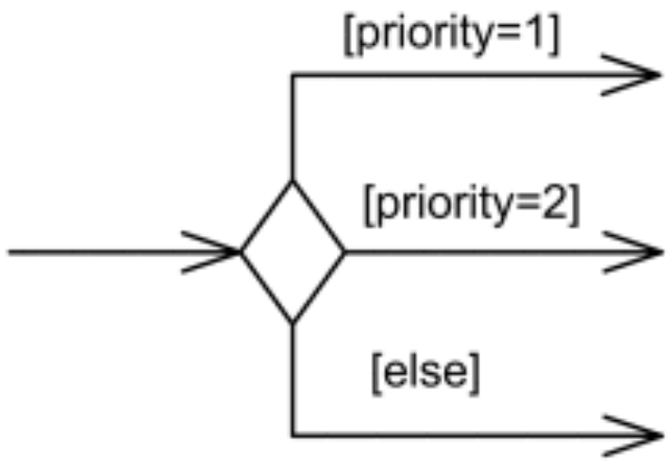


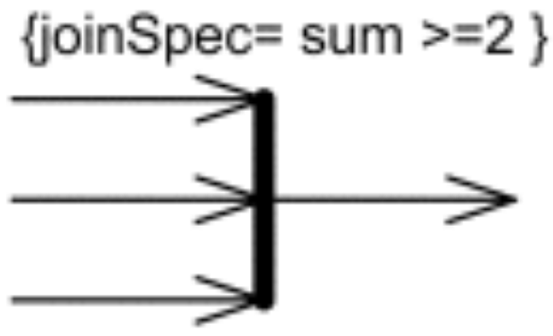
- Behaviour diagrams.
- They model the dynamic behaviour of a sub-system.
- They focus on lower level behaviour.
- They are realized on terms of one or several sequences of activities.
- Also known as flowchart.
- They include an initial and final state.
- They show decisions/merges and forks/joins.
- They can be partitioned in swimlanes to highlight concerns.


Handle Incident
Document Incident
Archive Incident
Open Incident
[lowPriority]
Allocate Resources
[fire & highPriority]
[not fire & highPriority]
Notify Fire Chief
Notify Police Chief
Open Incident
Allocate Resources
Coordinate Resources
Document Incident
Archive Incident


Author
Reviewer
Approver
Owner
Create
Update
Document [Draft]
Review Draft
Document [Reviewed]
Approve
[not approved]
[approved]
Document [Effective]
Review Effective
[change needed]
[obsolete]
Change Request
Archive
Document [Archived]
© uml-diagrams.org


initial node
fork node
join node
merge node
decision node
activity
final node


[priority=1]
[priority=2]
[else]


{joinSpec= sum >=2 }

Exercise

- Based on the10 functional requirements you identified about the Project for today and the Project description you received during the last exercise lecture, draw:

- either an activity diagram representing an in-depth decomposition of an interesting activity you identified in the project;

- or a state machine diagram describing the specification of the behaviour of an interesting object you identified in the project.

Summing up

Outline

- Literature
  - [OOSE] ch. 2
- Topics covered:
  - Modeling
  - UML and its history
  - UML behaviour diagrams
    - Brief description
    - Example(s)
    - Comment (or another example)
    - Template from [uml-diagrams.org]
- Next lecture: UML structure diagrams