DevOps
Paolo Tell

Documentation of Systems and projects & Wrap-up

IT University
of Copenhagen

Schedule for today

1st Block

[15’] - Final evaluation
- *Please provide feedback to the final evaluation by rating your colleagues statements.*
- *http://64.225.103.230:9999/*

[30’] - Modeling for the report
- *The unified modeling language: an overview.*
- *A pragmatic view on the report requirements.*

[15’] - Break

2nd Block

[30’] - Modeling for the report
- *Presentation of specific UML diagrams for specific parts of the report.*

[15’] - A selected example
- *Christoffer will provide you with an example.*

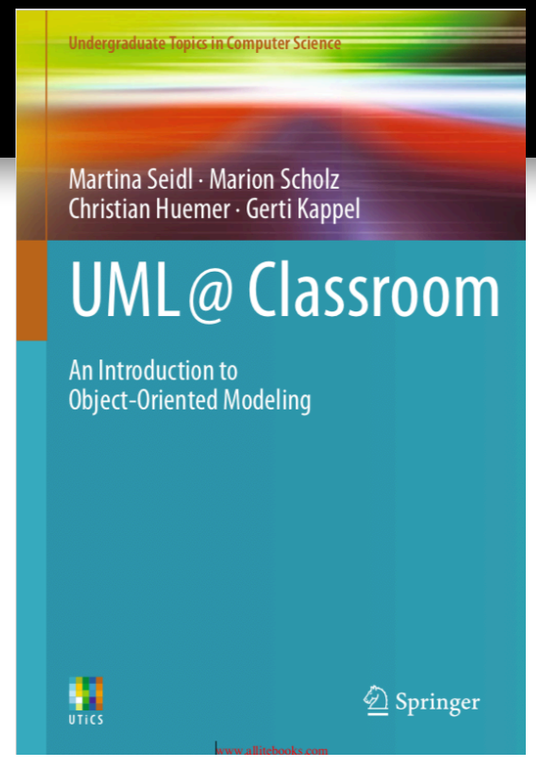
[15’] - Break

3rd Block

[?’] - Final evaluation
- *Helge will moderate this session in which we will discuss the main points emerging from the feedback you provided.*
- *This session will take as long as it is needed.*

[15’] - Simulation termination ceremony
- *Helge will moderate a short ceremony in which Christoffer will end the simulation officially closing the data collection and testing of your system.*

Sources



https://link.springer.com/book/10.1007/978-3-319-12742-2

- Seidl, Martina, Marion Brandsteidl, Christian Huemer, and Gerti Kappel.
  UML@ Classroom. Dpunkt, 2012.

- https://www.uml-diagrams.org/

- Bruegge, Bernd, and Allen H. Dutoit.
  Object-Oriented Software Engineering. 3rd edition. 2010.

Disclaimers

- I tried to compile these slides so that they can serve you as a reference, while showcasing examples I believe might be relevant to your task.
- I will go through the material with little interactions, if you have questions at any point, I need you to interrupt me.
- I expect you to indicate to me whether I should go faster or skip entirely a part, for you already dominate the concepts.

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 2000 by the International Organization for Standardization (ISO) as an approved ISO standard.

The 3 amigos

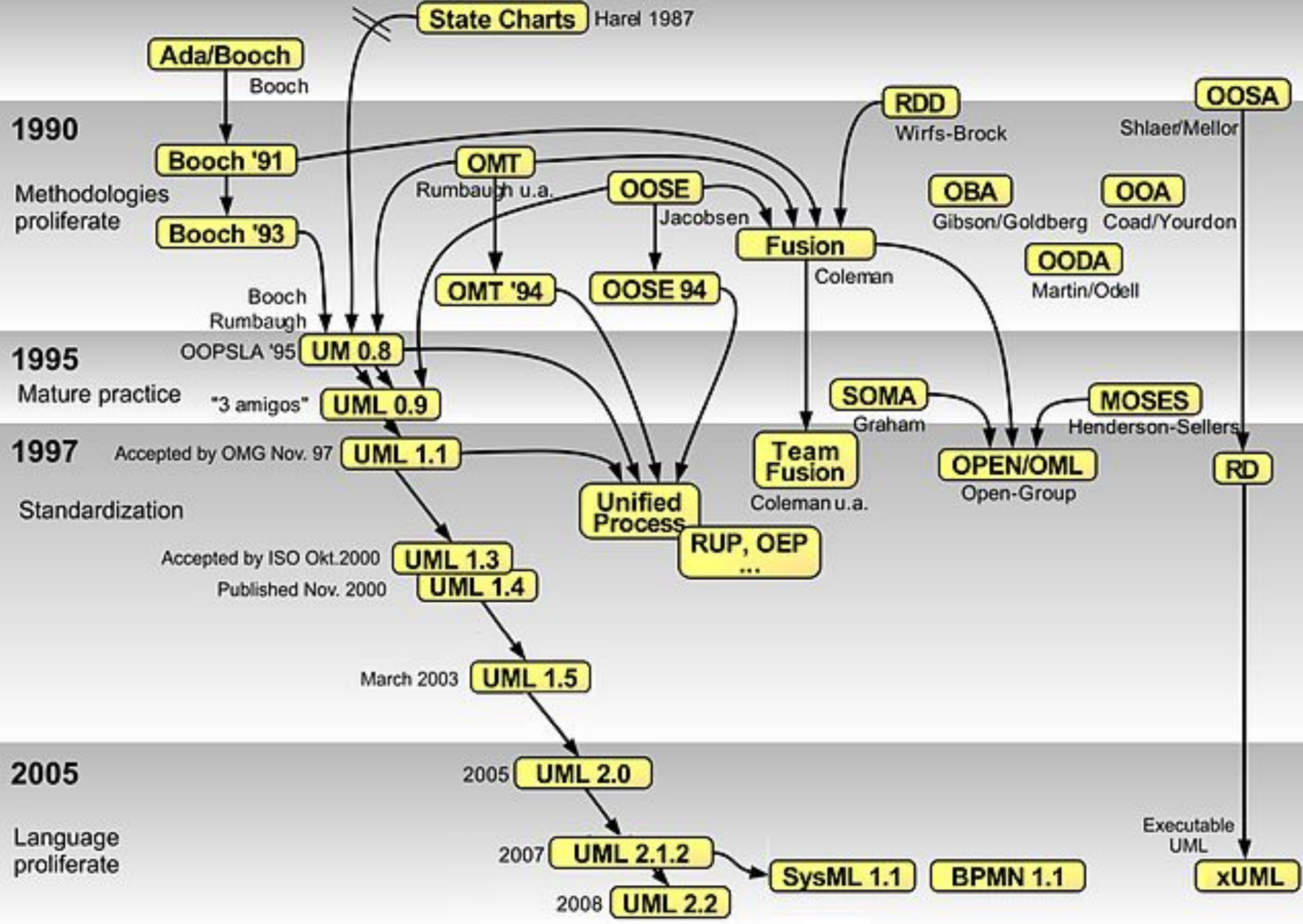
25 year at General Electric Research, where he developed OMT, joined (IBM) Rational in 1994, CASE tool OMTool

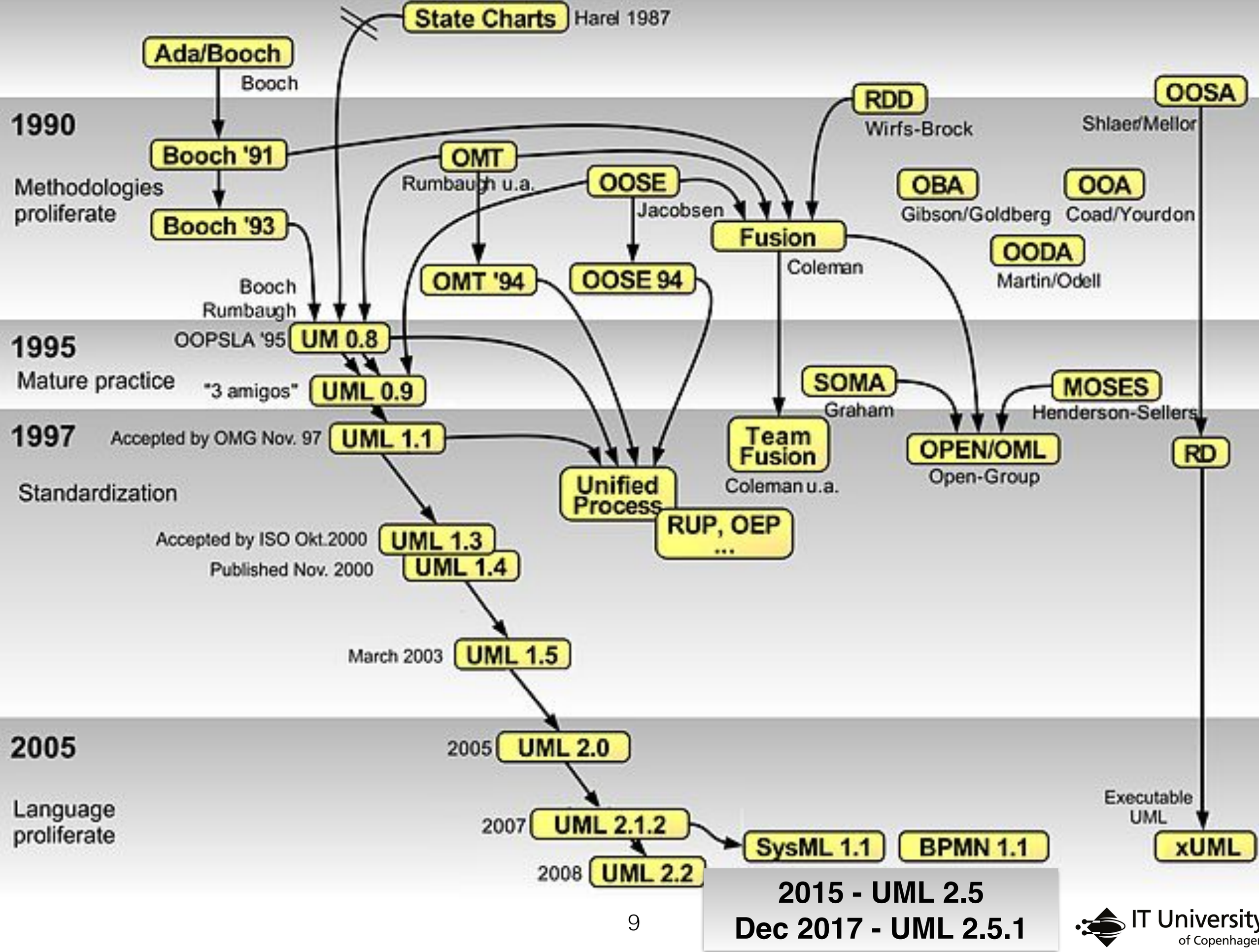
At Ericsson until 1994, developed use cases and the CASE tool Objectory, at IBM Rational since 1995

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

- Convergence of different notations used in object-oriented methods, mainly
  - Object-Modeling Technique (OMT) (James Rumbaugh and colleagues),
  - Object-Oriented Software Engineering (OOSE) (Ivar Jacobson),
  - Booch method (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)

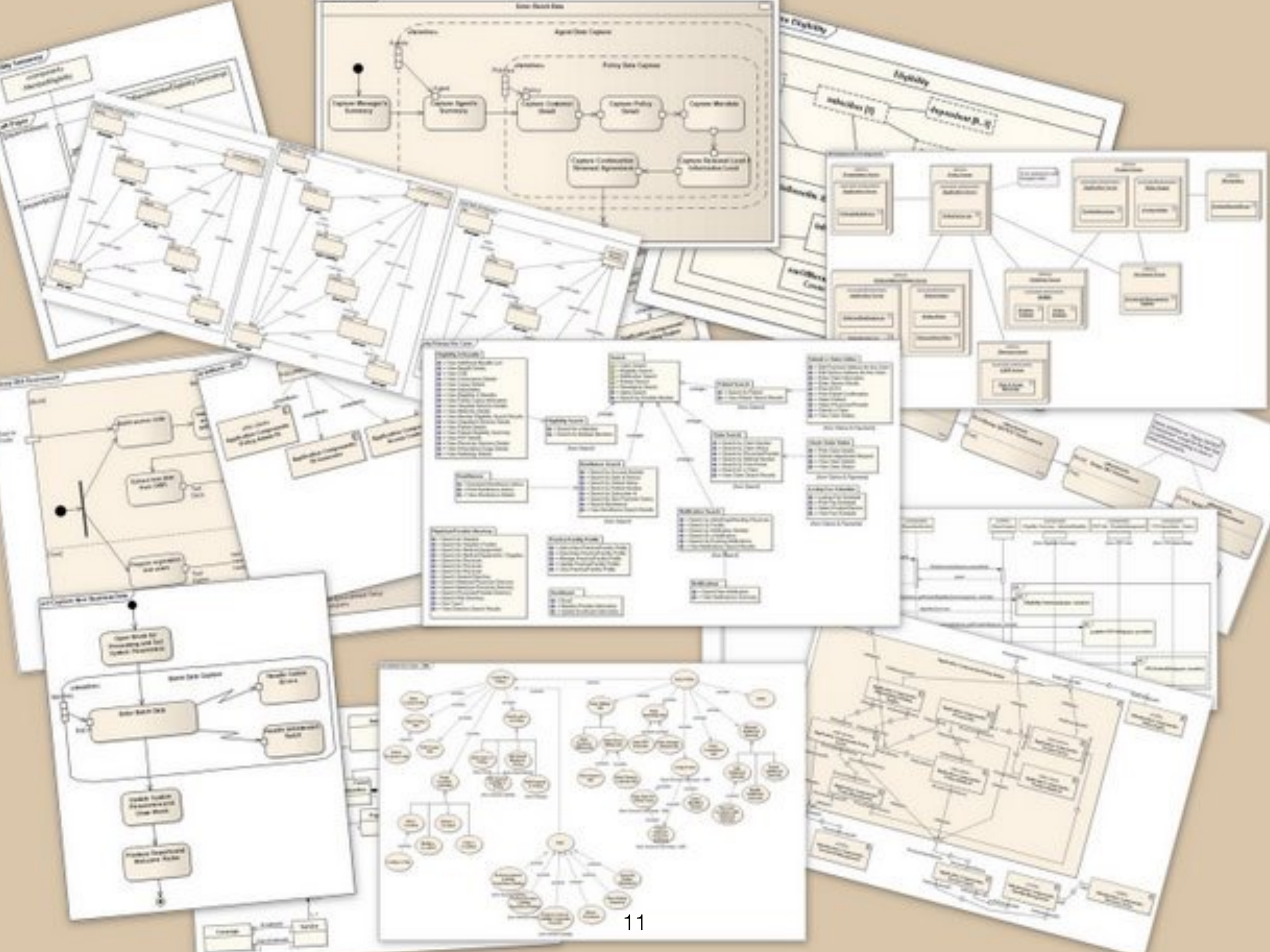

State Charts
Harel 1987
Ada/Booch
Booch
1990
Booch '91
Booch '93
Methodologies proliferate
OMT
Rumbaugh u.a.
OOSE
Jacobsen
RDD
Wirfs-Brock
OOSA
Shlaer/Mellor
OBA
Gibson/Goldberg
OOA
Coad/Yourdon
Fusion
Coleman
OODA
Martin/Odell
OMT '94
OOSE 94
Booch
Rumbaugh
OOPSLA '95
UM 0.8
1995
Mature practice
"3 amigos"
UML 0.9
SOMA
Graham
MOSES
Henderson-Sellers
1997
Accepted by OMG Nov. 97
UML 1.1
Team Fusion
Coleman u.a.
OPEN/OML
Open-Group
RD
Unified Process
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
Coleman u.a.
OPEN/OML
Open-Group
RD
Unified
Process
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
Dec 2017 - UML 2.5.1

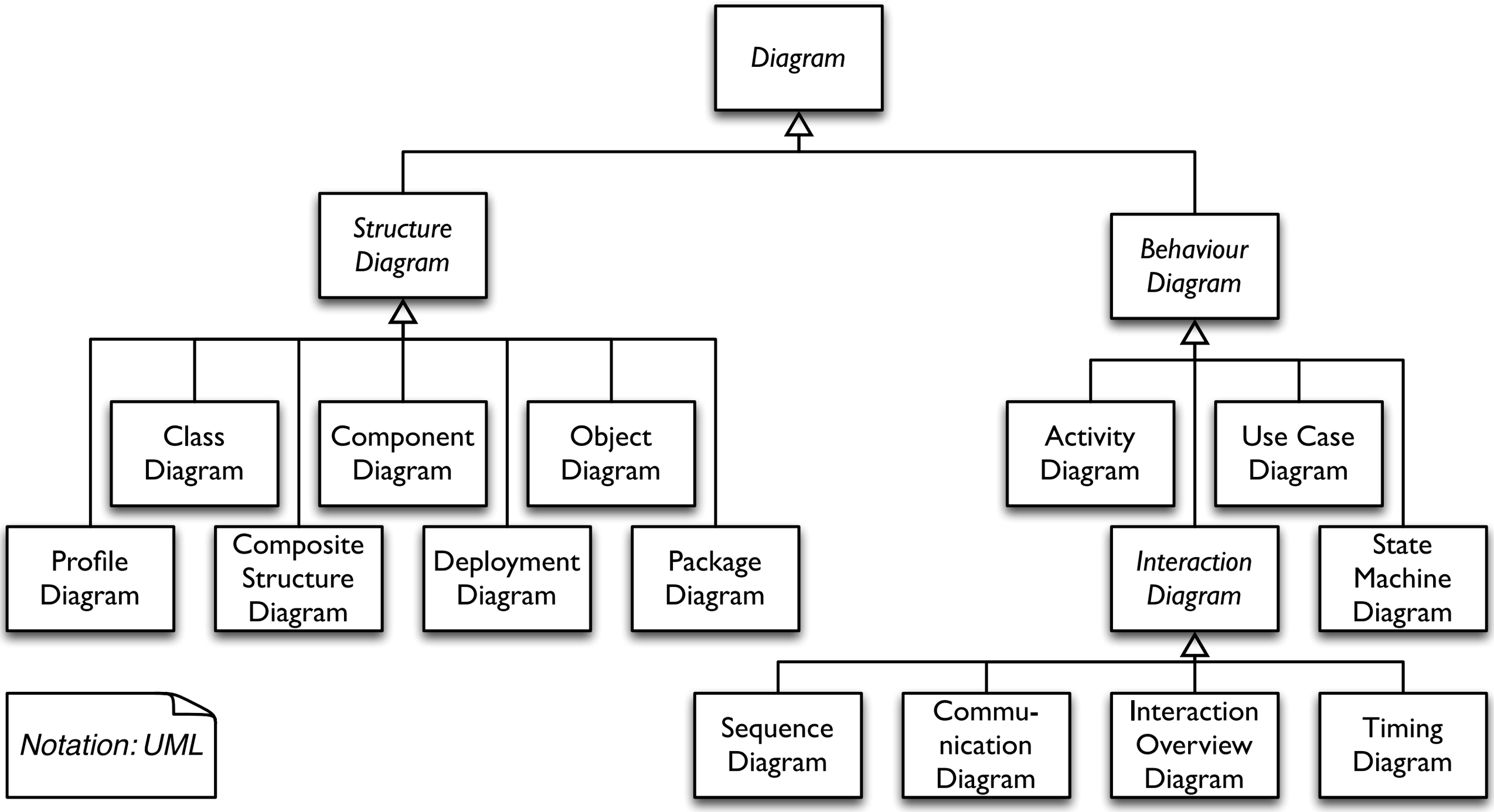
UML

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

- In this course
  - "Just" a diagramming notation standard.



Hierarchy of diagrams in UML 2.x



What to include in the report?
… with a selective focus on modeling

System's Perspective

A **description and illustration** of the:

- Design of your ITU-MiniTwit systems
- Architecture of your ITU-MiniTwit systems
- All dependencies of your ITU-MiniTwit systems on all levels of abstraction and development stages.
  - That is, list and briefly describe all technologies and tools you applied and depend on.
- Important interactions of subsystems

Process' Perspective

A **description and illustration** of:

- How do you interact as developers?
- How is the team organized?
- A complete description of stages and tools included in the CI/CD chains.
- Organization of your repository(ies).
- Applied branching strategy.
- Applied development process and tools supporting it
- How do you monitor your systems and what precisely do you monitor?
- What do you log in your systems and how do you aggregate logs?
- Brief results of the security assessment.
- Applied strategy for scaling and load balancing.

Lessons Learned Perspective

What to include in the report?
… with a selective focus on modeling

System's Perspective

Structure Diagrams
Class - Component
Package - Deployment

A **description and illustration** of the:

- Design of your ITU-MiniTwit systems
- Architecture of your ITU-MiniTwit systems
- All dependencies of your ITU-MiniTwit systems on all levels of abstraction and development stages.
  - That is, list and briefly describe all technologies and tools you applied and depend on.
- Important interactions of subsystems

Directed acyclic graphs

Process' Perspective

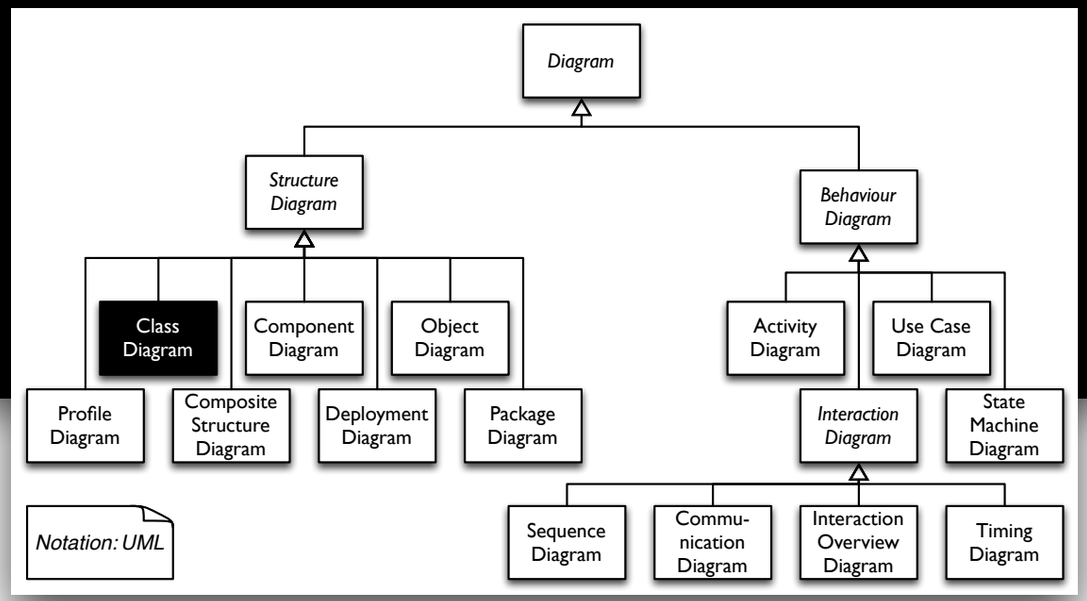
A **description and illustration** of:

Behaviour Diagrams
Sequence - Activity - …

- How do you interact as developers?
- How is the team organized?
- A complete description of stages and tools included in the CI/CD chains.
- Organization of your repository(ies).
- Applied branching strategy.
- Applied development process and tools supporting it
- How do you monitor your systems and what precisely do you monitor?
- What do you log in your systems and how do you aggregate logs?
- Brief results of the security assessment.
- Applied strategy for scaling and load balancing.

Lessons Learned Perspective

Class diagram

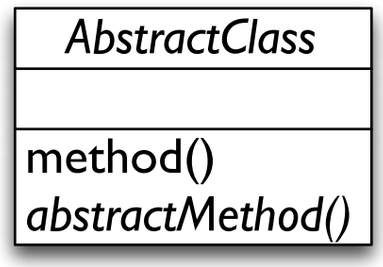


Name	Notation	Description
Class	A – a1: T1 – a2: T2 + o1(): void + o2(): void	Description of the structure and behavior of a set of objects
Abstract class	A {abstract} A	Class that cannot be instantiated
Association	A — B (a) A ⟷ B (b) A → B (c)	Relationship between classes: navigability unspecified (a), navigable in both directions (b), not navigable in one direction (c)
N-ary association	A, B, C	Relationship between N (in this case 3) classes
Association class	A, B, C	More detailed description of an association
xor relationship	B, {xor}, C, A	An object of A is in a relationship with an object of B or with an object of C but not with both
Strong aggregation = composition	A ◆— B	Existence-dependent parts-whole relationship (A is part of B; if B is deleted, related instances of A are also deleted)
Shared aggregation	A ◇— B	Parts-whole relationship (A is part of B; if B is deleted, related instances of A need not be deleted)
Generalization	A —▷ B	Inheritance relationship (A inherits from B)

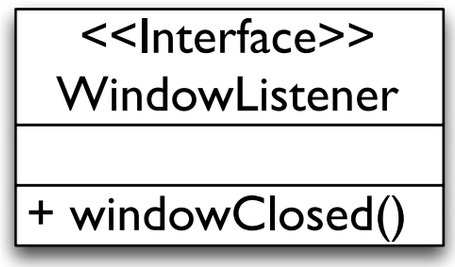
Source:
Seidl, Martina, Marion Brandsteidl, Christian Huemer, and Gerti Kappel. *UML@ Classroom*. Dpunkt, 2012.

Abstract classes and interfaces

- Abstract classes
  - Italic
  - or using the label {abstract}
  - holds also for methods

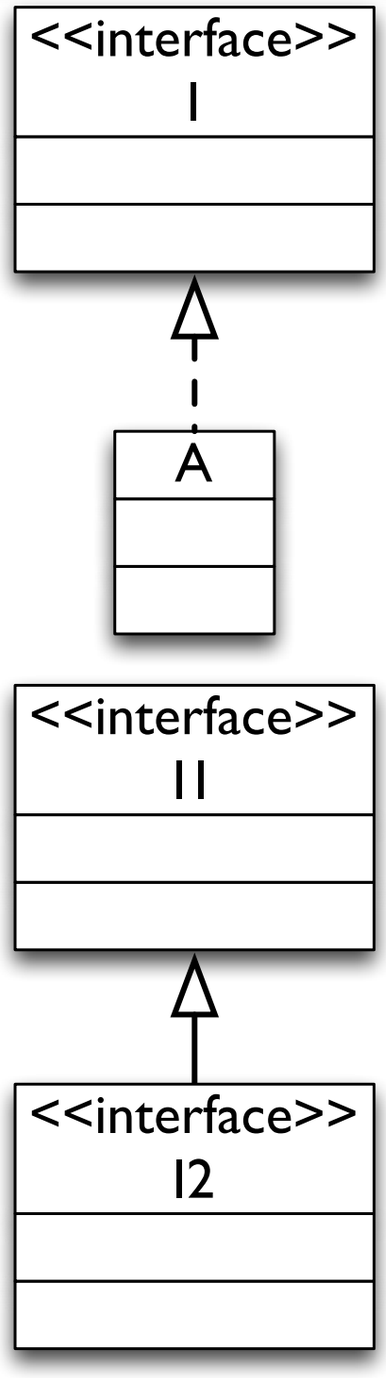


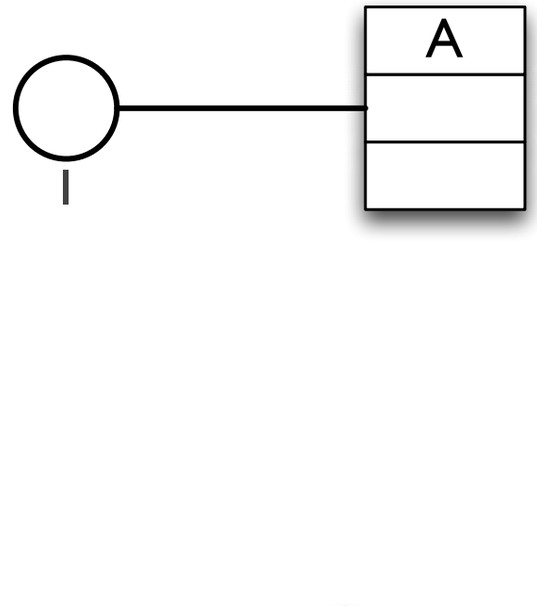
- Interfaces
  - Same as the classes
  - Use the stereotype <<Interface>>
  - or the lollipop notation



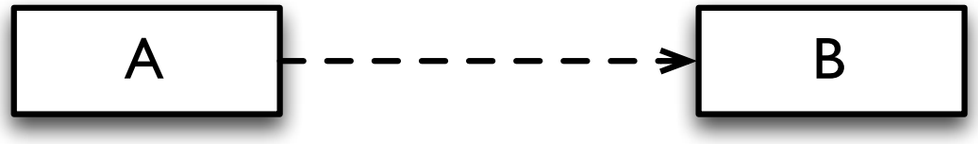
Relationships: inheritance with interfaces

- A implements I
- I2 extends I1

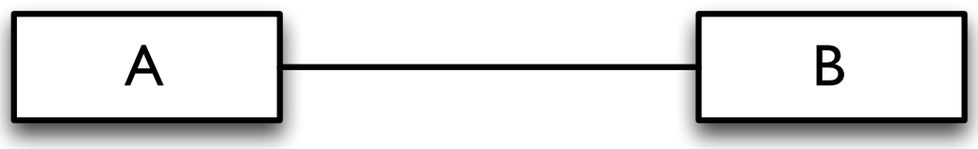




Relationships: (generic) dependency and association

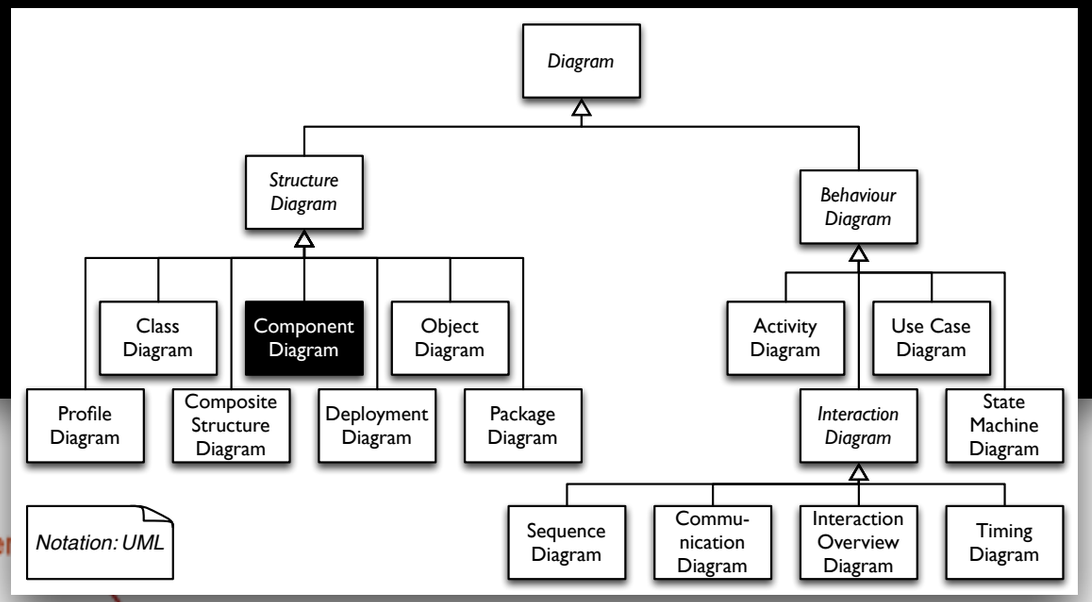


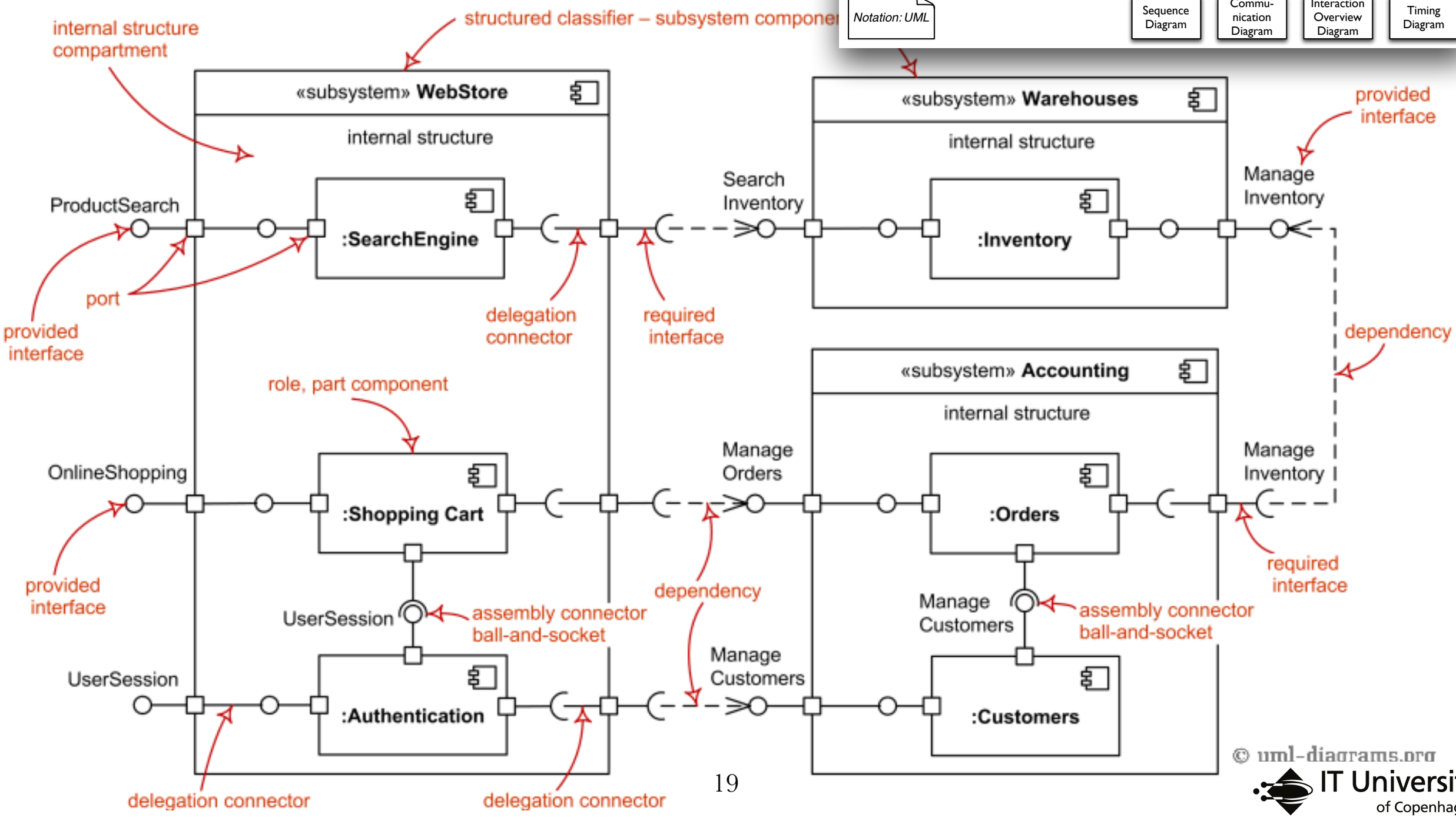
- A depends on B if A needs B to execute.
  - A calls B
  - A uses the type B



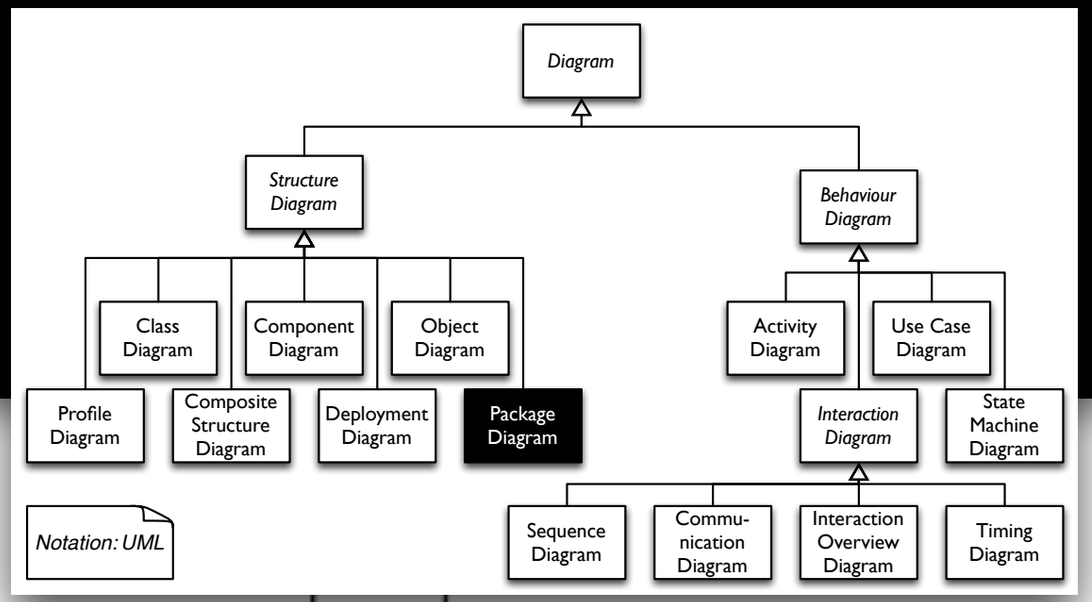
- A generic (but interesting) relationship between A and B.

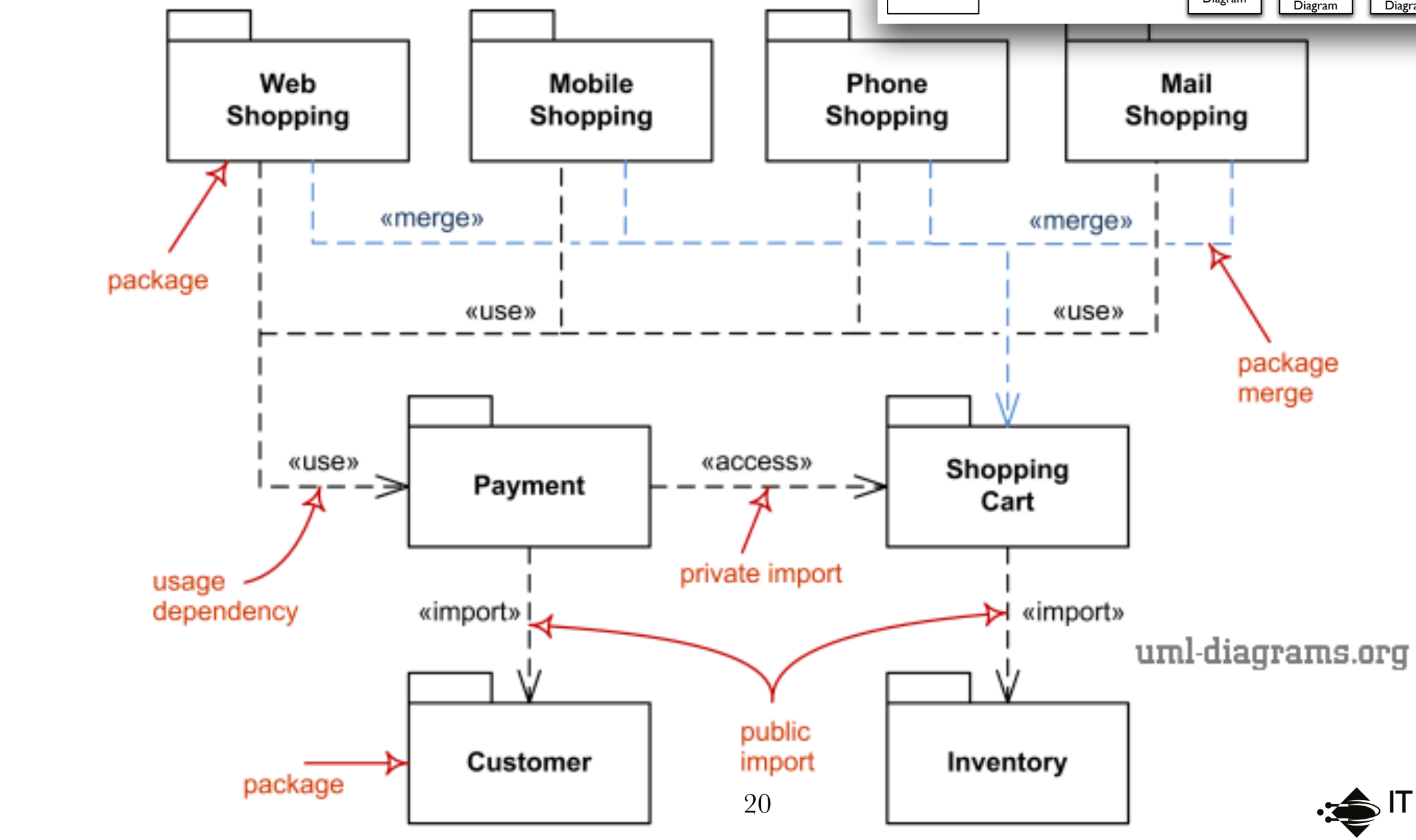
Component diagram



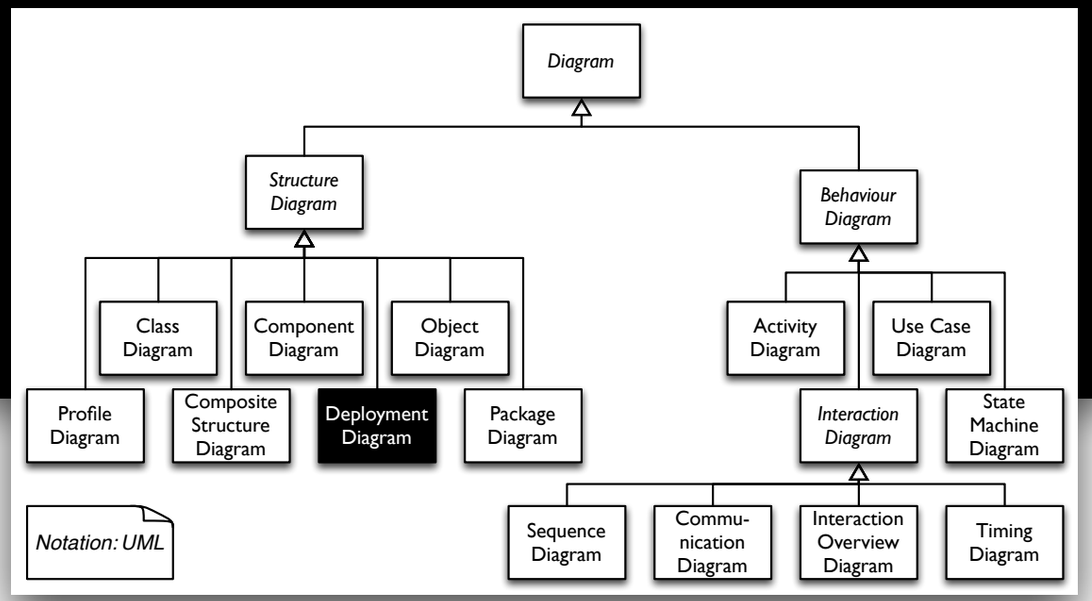


Package diagram



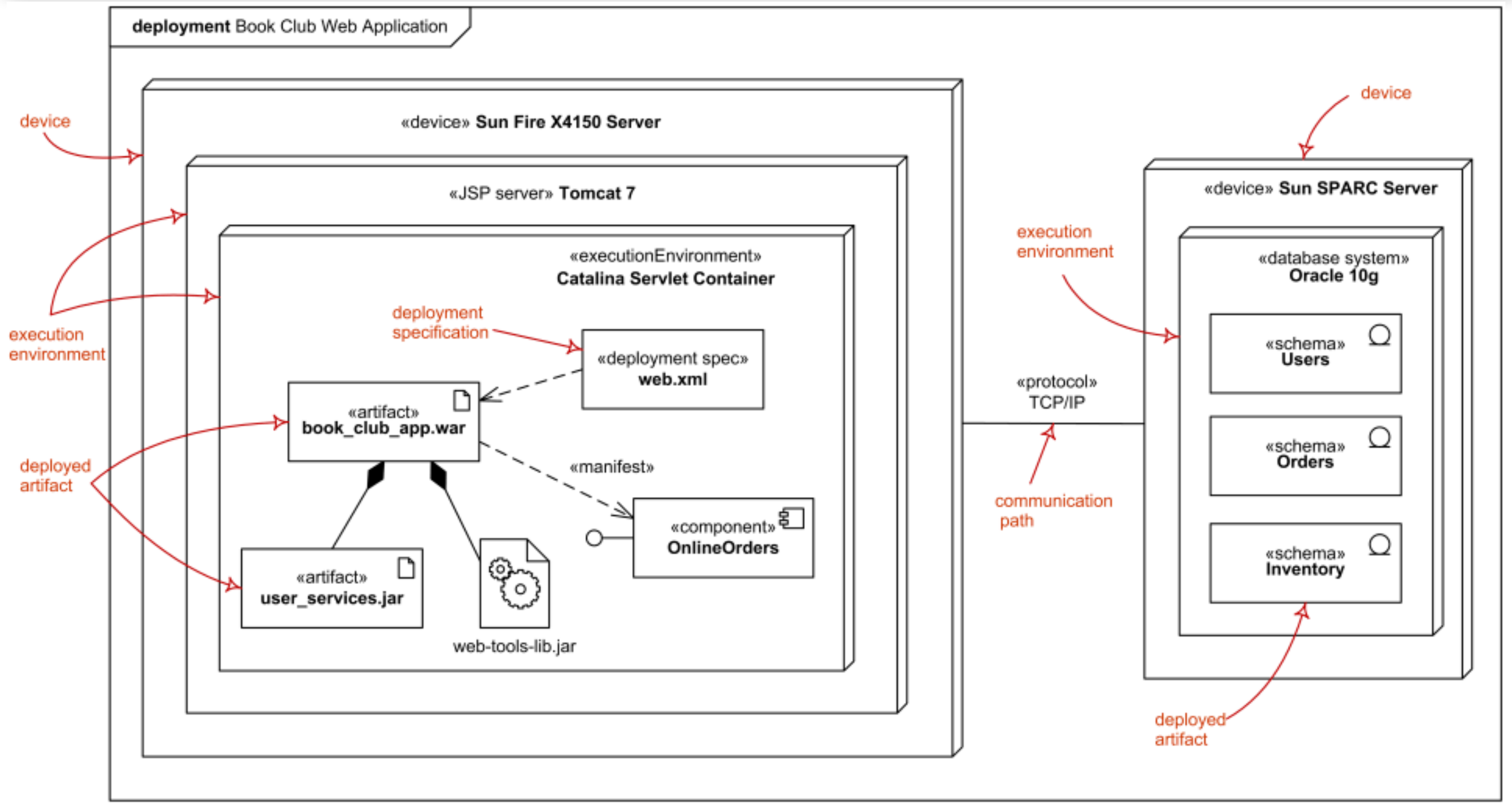


Deployment diagram

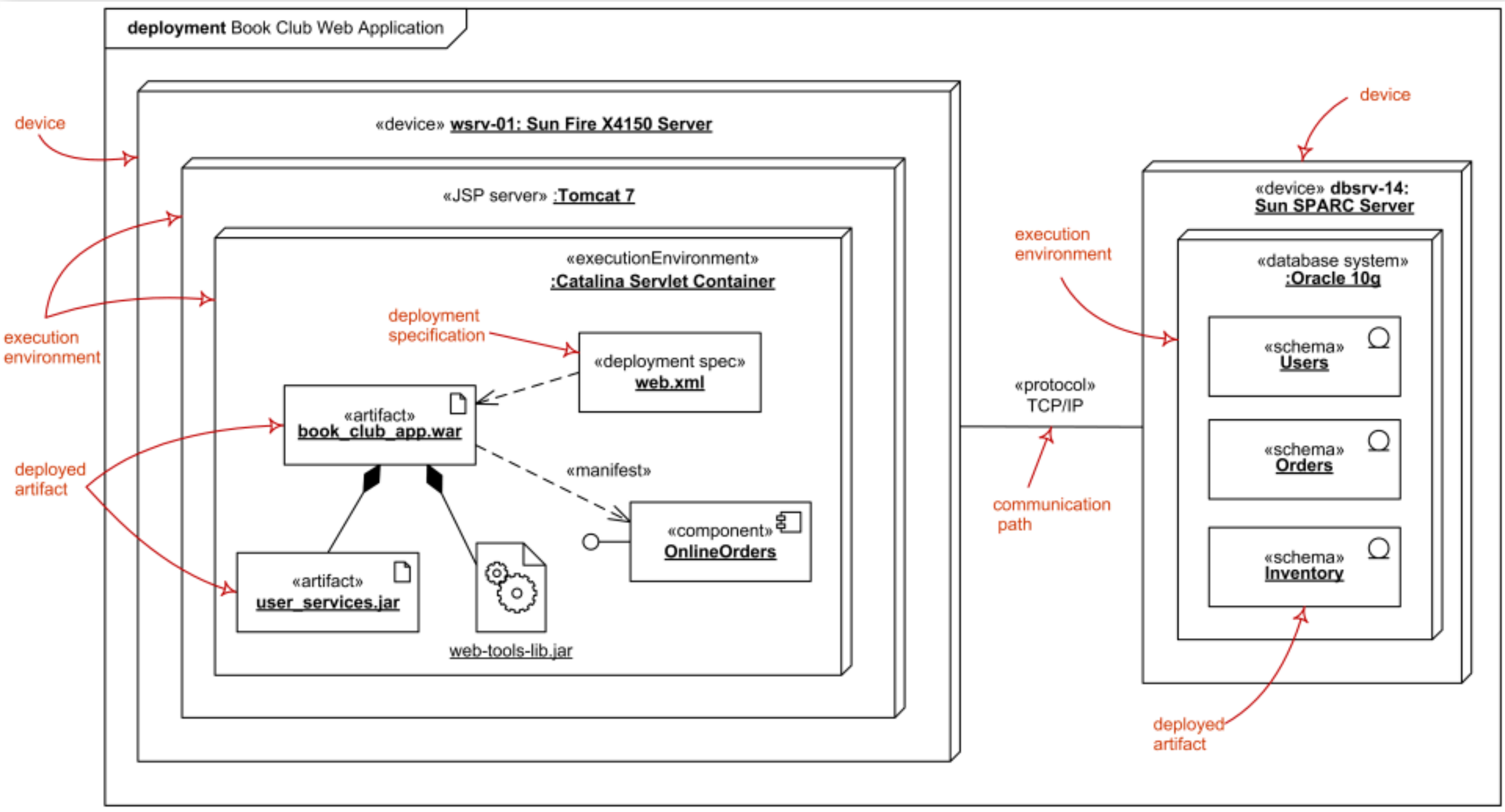


- Structure diagrams.
- They are used to depict where the various elements of a system are located.
- For instance, a distributed system based on a client/server architecture.
- Deployment diagrams contain
  - *nodes* and *communication paths*
  - devices
  - *execution environments*
  - components and dependancies

Specification level
Deployment diagram

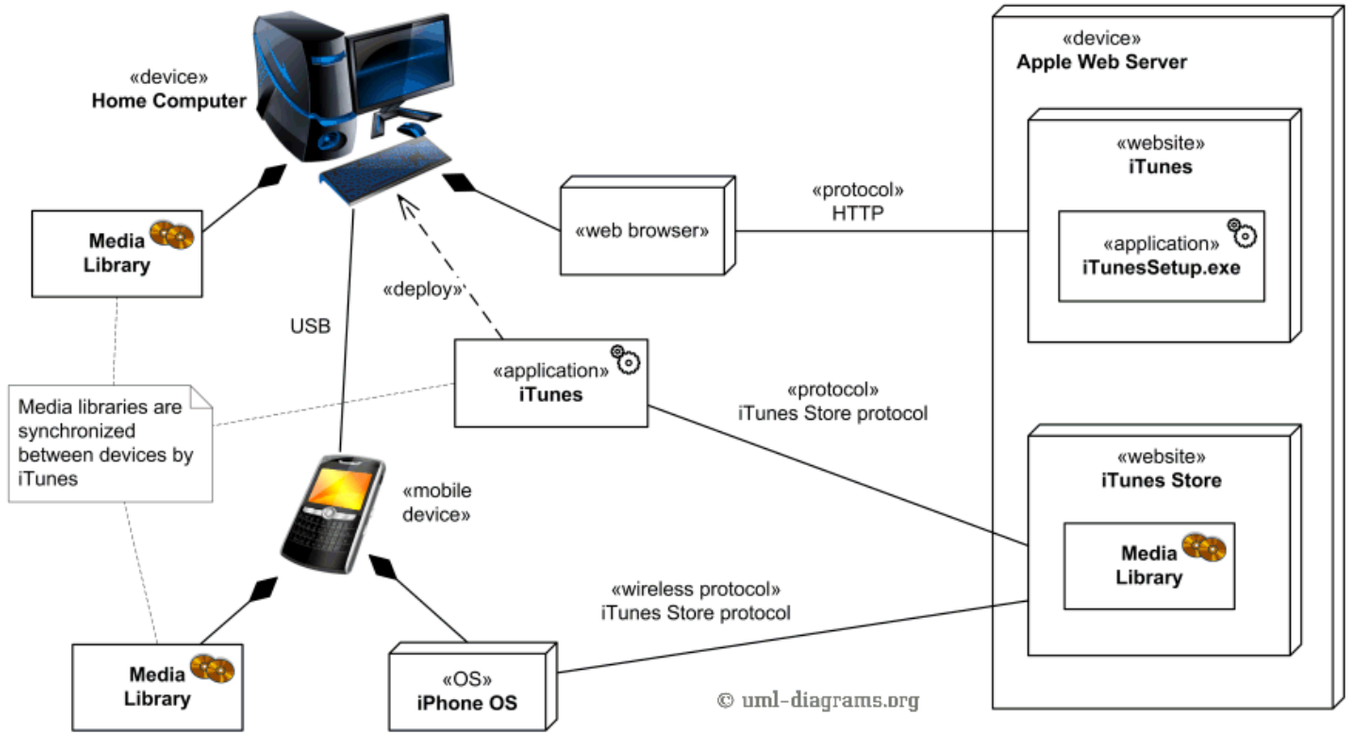


Instance level
Deployment diagram

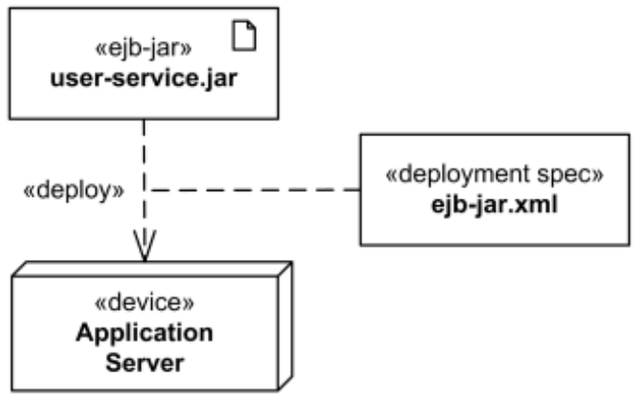


Some interesting bits

- Custom icons

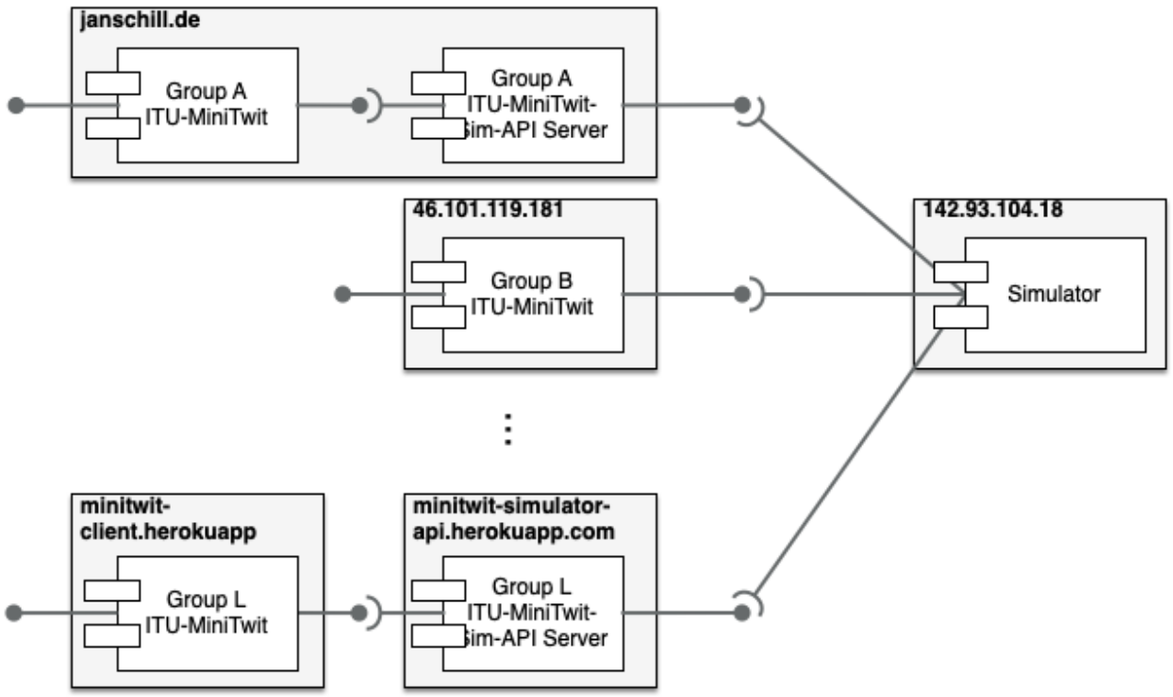


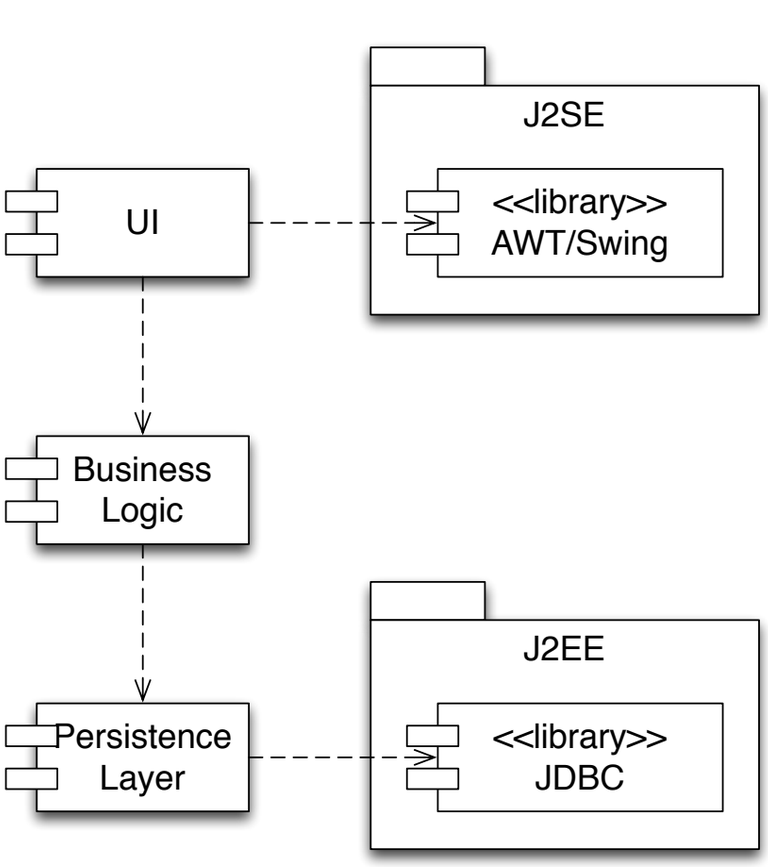
- Deployment specification association



Mixing diagrams

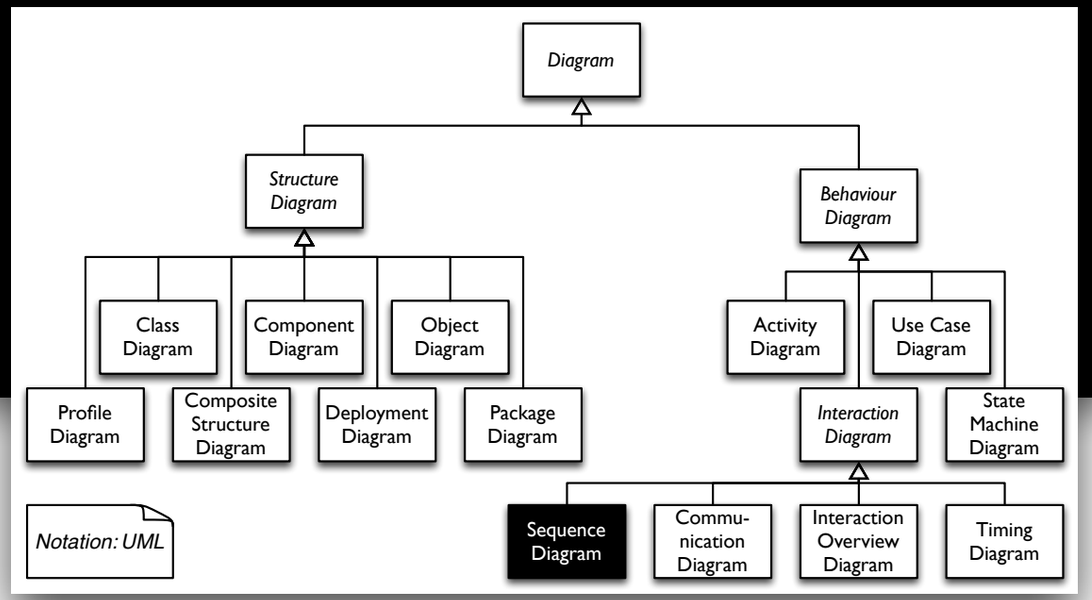
- Component and deployment

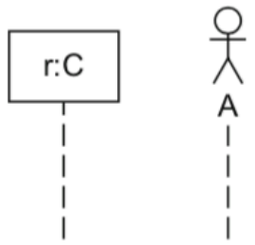
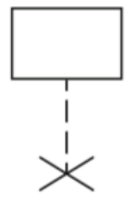
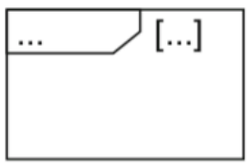
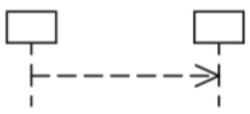


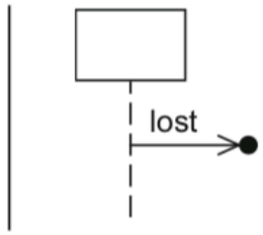
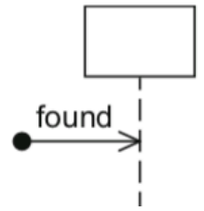


- Package and deployment

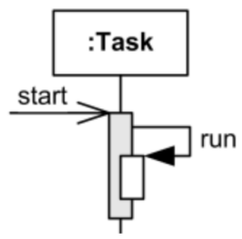
Sequence diagram



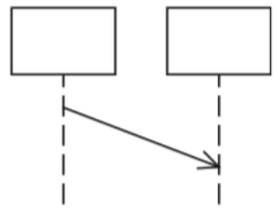
Name	*Notation*	*Description*
Lifeline		Interaction partners involved in the communication
Destruction event		Time at which an interaction partner ceases to exist
Combined fragment		Control constructs
Synchronous message		Sender waits for a response message
Response message		Response to a synchronous message
Asynchronous message		Sender continues its own work after sending the asynchronous message

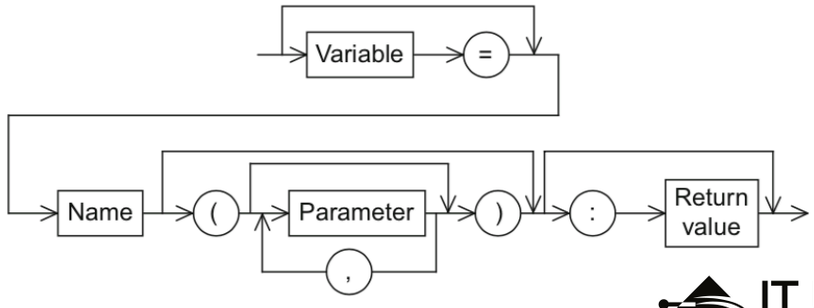
Lost message		Message to an unknown receiver
Found message		Message from an unknown sender

Execution specification



Time-consuming message

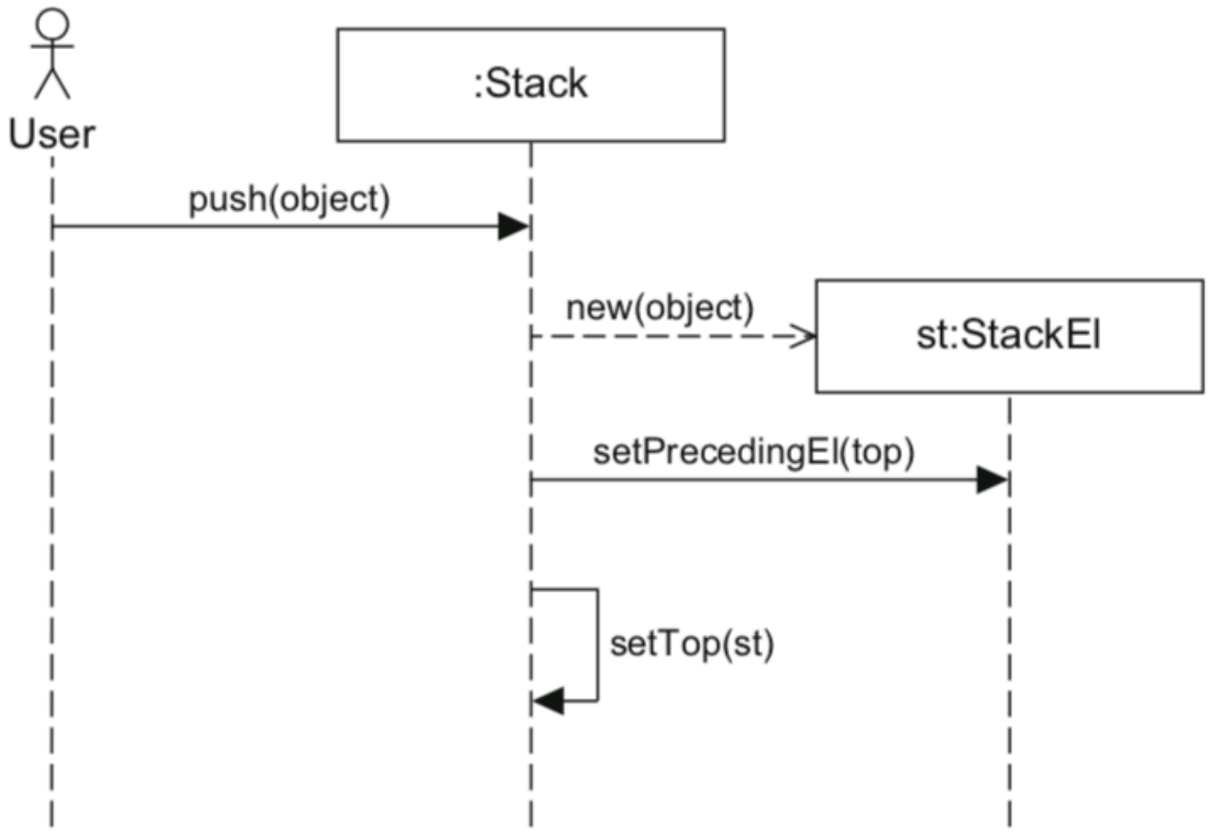


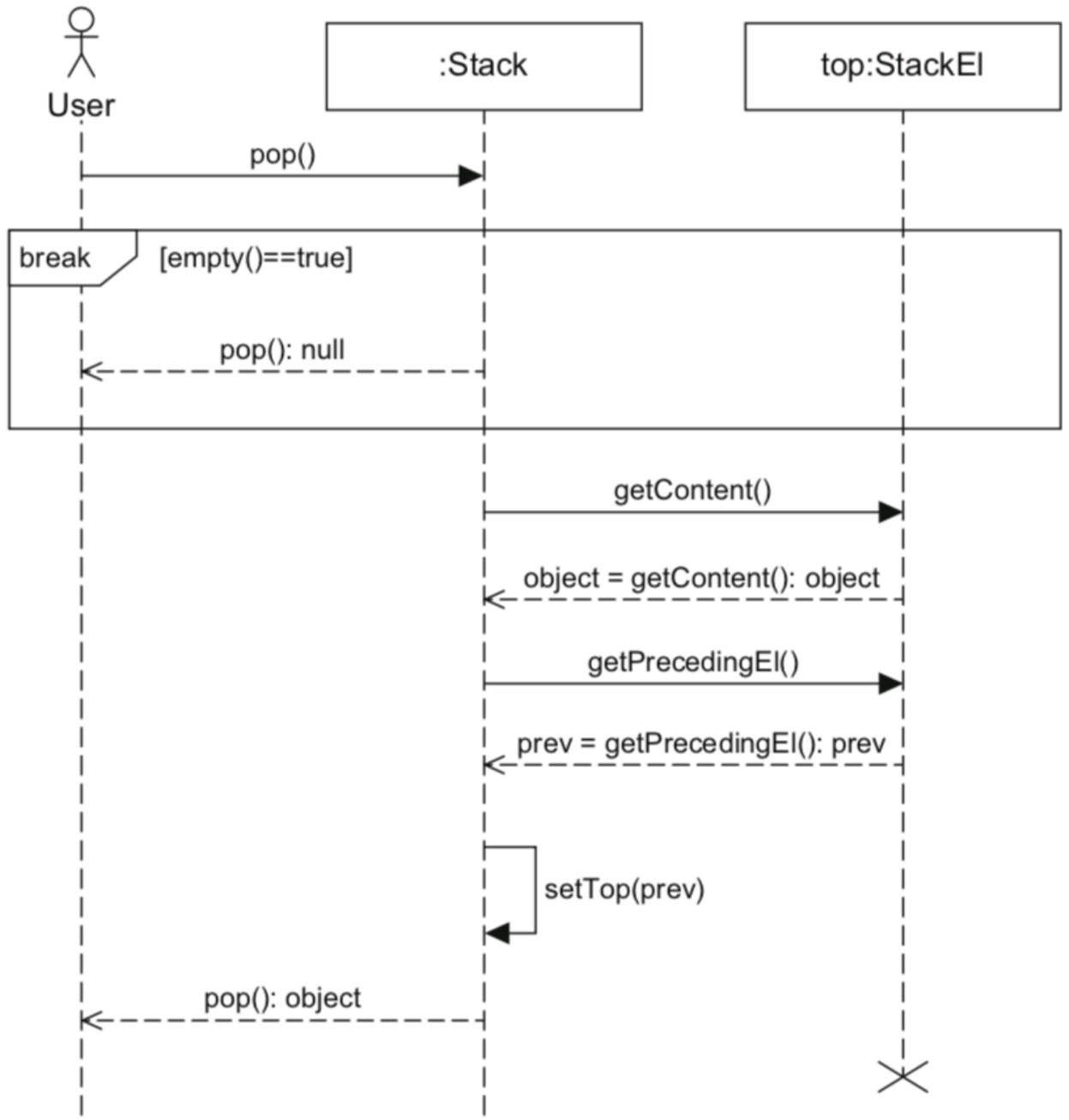


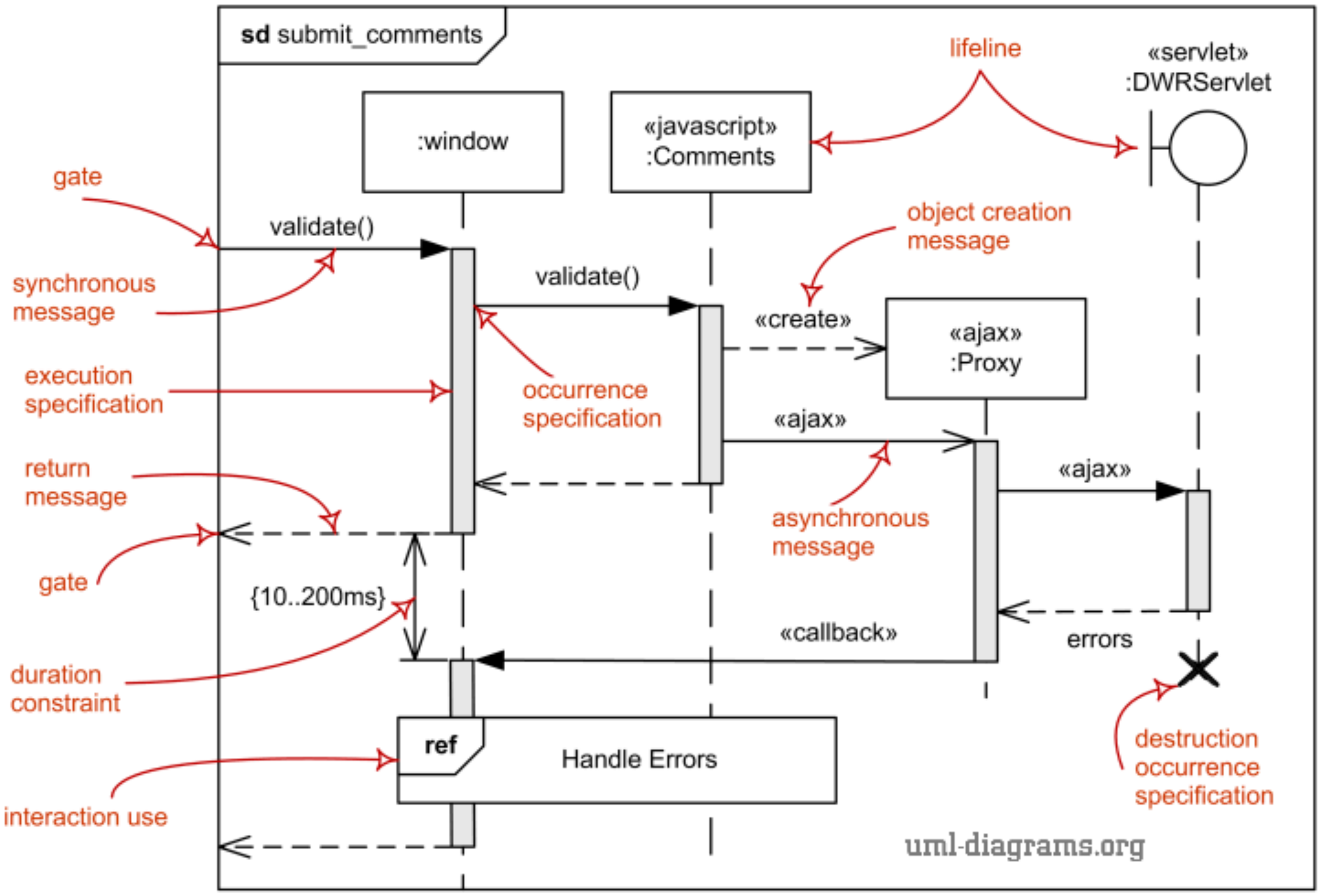
Source:
Seidl, Martina, Marion Brandsteidl, Christian Huemer, and Gerti Kappel. *UML@ Classroom*. Dpunkt, 2012.

An Example

Source:
Seidl, Martina, Marion Brandsteidl, Christian Huemer, and Gerti Kappel. *UML@ Classroom*. Dpunkt, 2012.

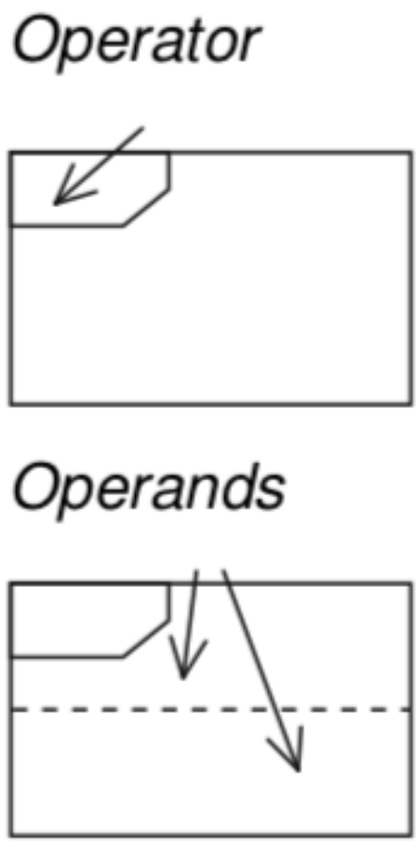




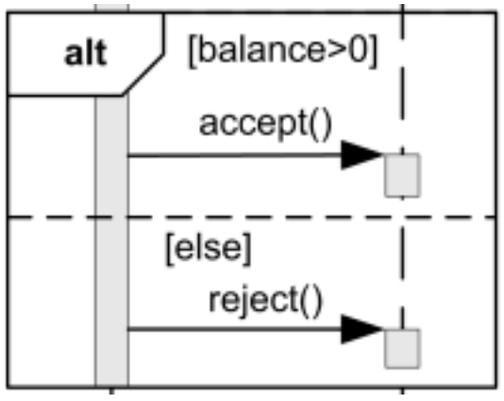

sd submit_comments
lifeline
«servlet»
:DWRServlet
:window
«javascript»
:Comments
gate
validate()
object creation
message
synchronous
message
validate()
«create»
«ajax»
:Proxy
execution
specification
occurrence
specification
«ajax»
return
message
«ajax»
asynchronous
message
gate
{10..200ms}
«callback»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

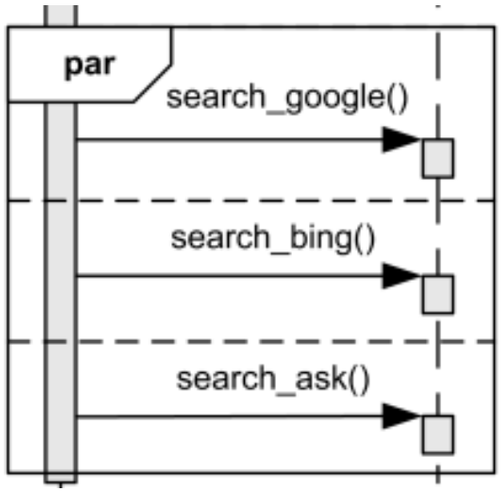
Combined fragments

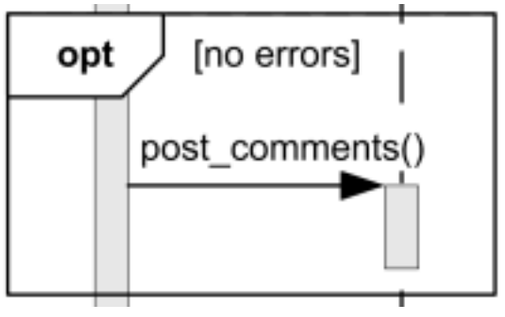
Source:
Seidl, Martina, Marion Brandsteidl, Christian Huemer, and Gerti Kappel. *UML@ Classroom*. Dpunkt, 2012.

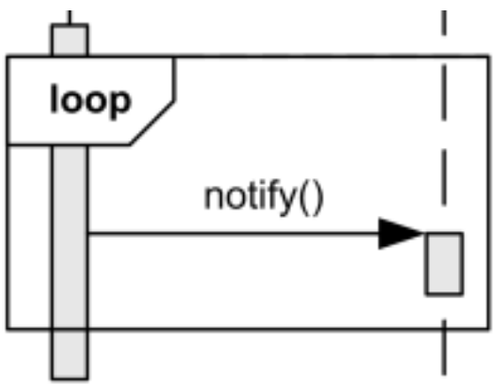


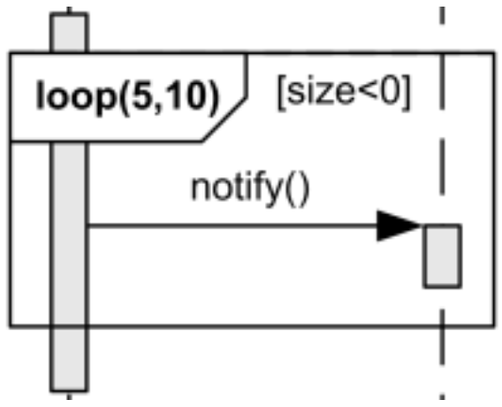
	Operator	Purpose
Branches and loops	alt	Alternative interaction
	opt	Optional interaction
	loop	Iterative interaction
	break	Exception interaction
Concurrency and order	seq	Weak order
	strict	Strict order
	par	Concurrent interaction
	critical	Atomic interaction
Filters and assertions	ignore	Irrelevant interaction parts
	consider	Relevant interaction parts
	assert	Asserted interaction
	neg	Invalid interaction

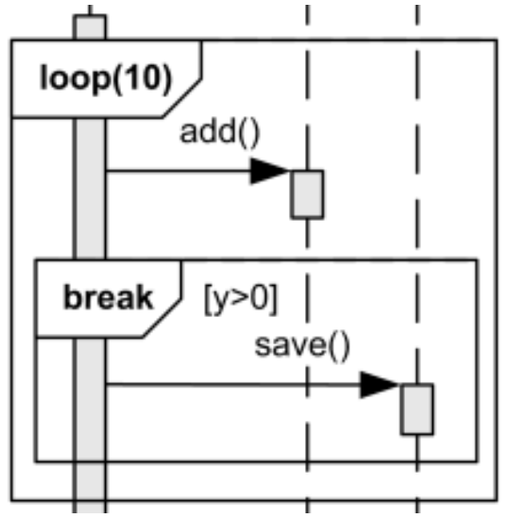

alt
[balance>0]
accept()
[else]
reject()


par
search_google()
search_bing()
search_ask()


opt
[no errors]
post_comments()


loop
notify()

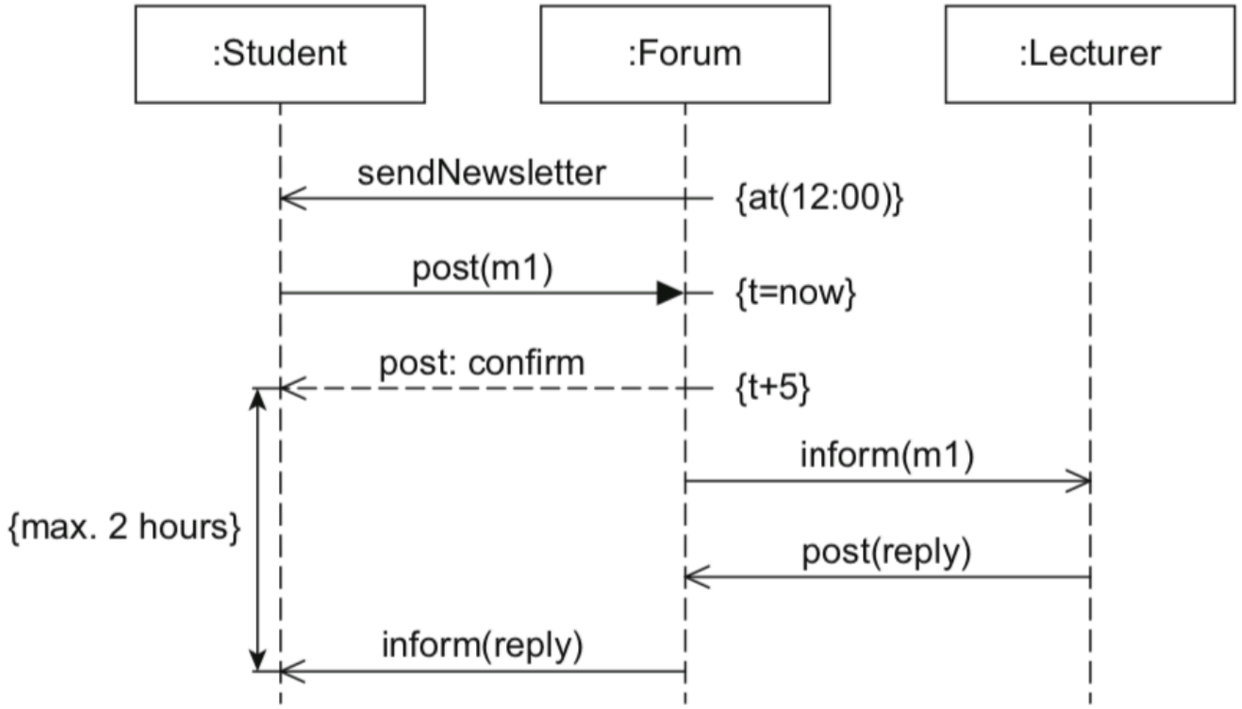

loop(5,10)
[size<0]
notify()


loop(10)
add()
break
[y>0]
save()

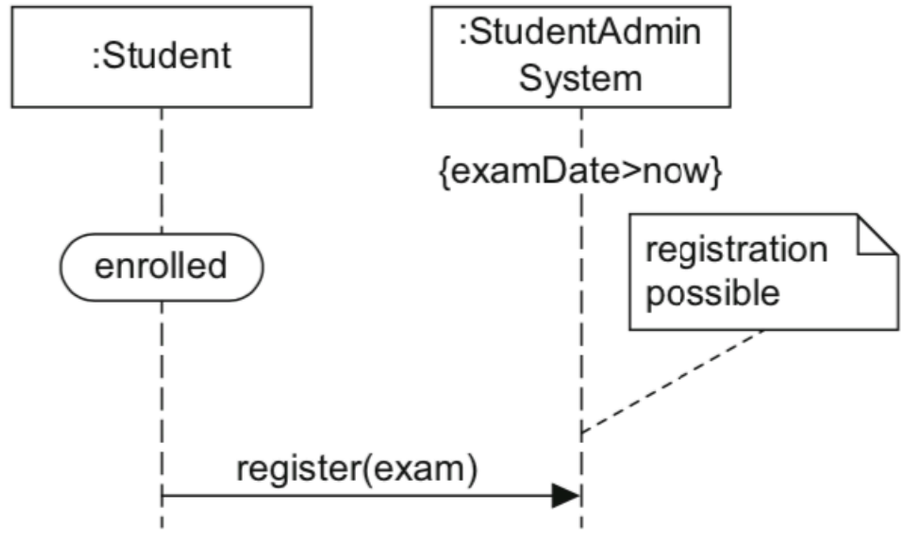
Selected examples

Source:
Seidl, Martina, Marion Brandsteidl, Christian Huemer, and Gerti Kappel. *UML@ Classroom*. Dpunkt, 2012.

- Time constraints

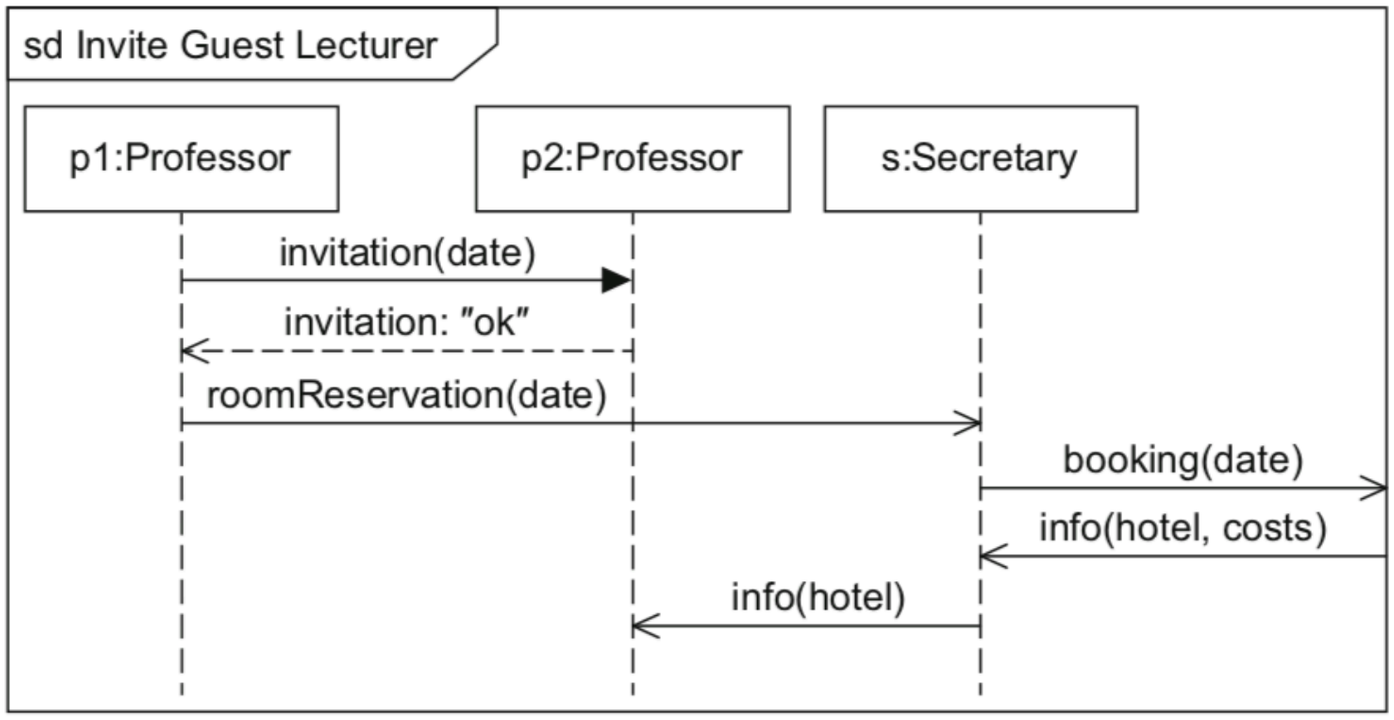


- State invariants

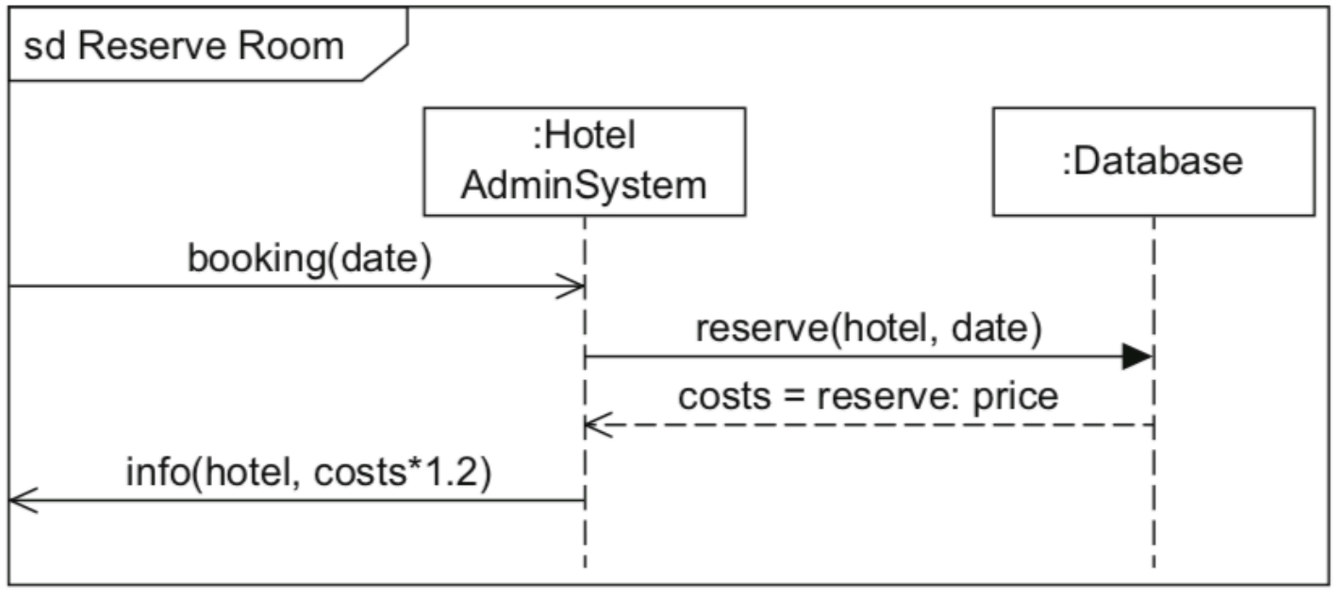


Selected examples

Source:
Seidl, Martina, Marion Brandsteidl, Christian Huemer, and Gerti Kappel. *UML@ Classroom*. Dpunkt, 2012.

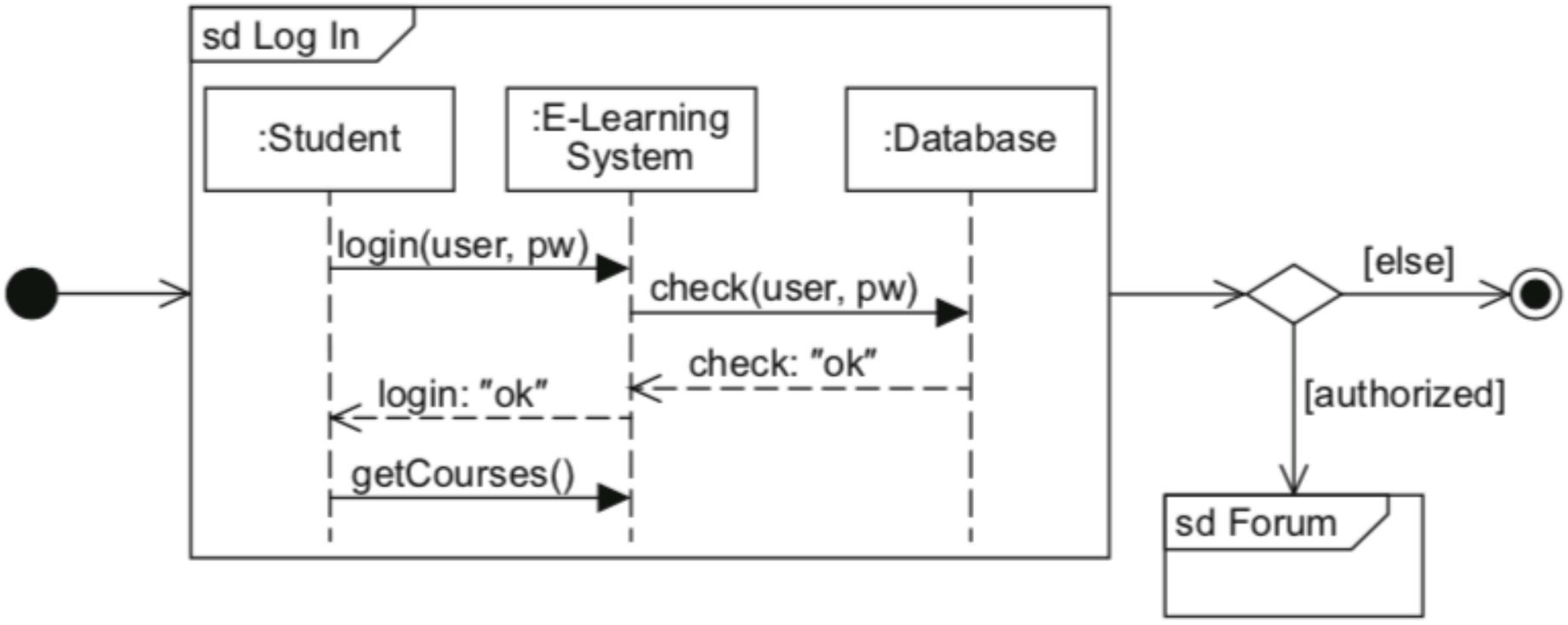


- Gates

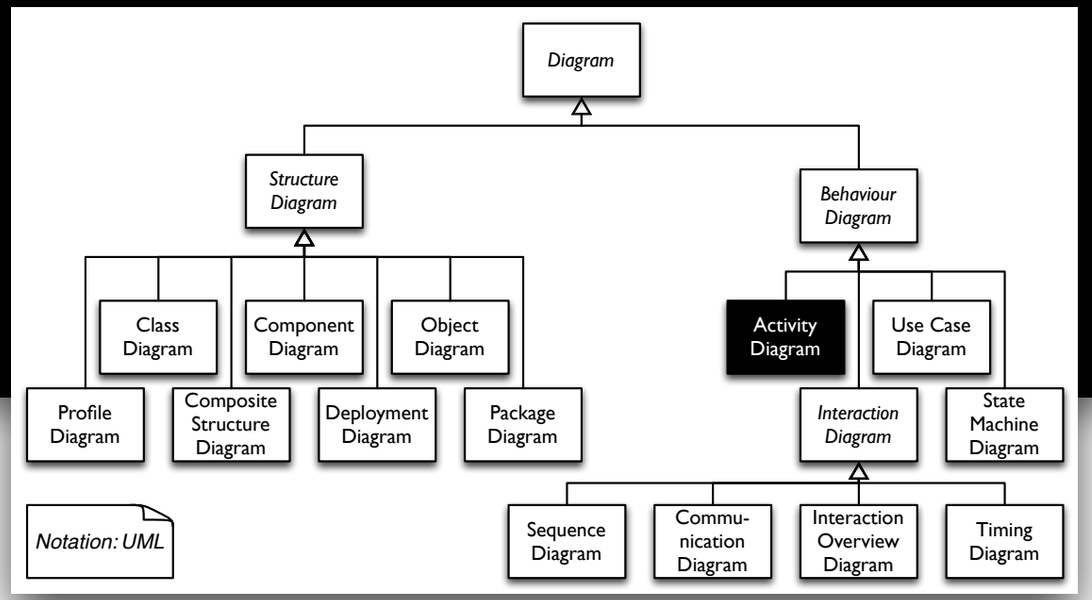


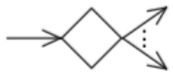
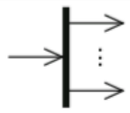
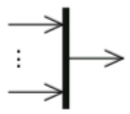
Interaction overview diagrams

Source:
Seidl, Martina, Marion Brandsteidl, Christian Huemer, and Gerti Kappel. *UML@ Classroom*. Dpunkt, 2012.



Activity diagram



Name	*Notation*	*Description*
Action node	Action	Actions are atomic, i.e., they cannot be broken down further
Activity node	Activity	Activities can be broken down further
Initial node	●	Start of the execution of an activity
Activity final node	◉	End of ALL execution paths of an activity
Flow final node	⊗	End of ONE execution path of an activity
Decision node		Splitting of one execution path into two or more alternative execution paths
Merge node		Merging of two or more alternative execution paths into one execution path
Parallelization node		Splitting of one execution path into two or more concurrent execution paths
Synchronization node		Merging of two or more concurrent execution paths into one execution path
Edge	A → B	Connection between the nodes of an activity
Call behavior action	A	Action A refers to an activity of the same name
Object node	Object	Contains data and objects that are created, changed, and read
Parameters for activities	Activity	Contain data and objects as input and output parameters
Parameters for actions (pins)	Action	Contain data and objects as input and output parameters

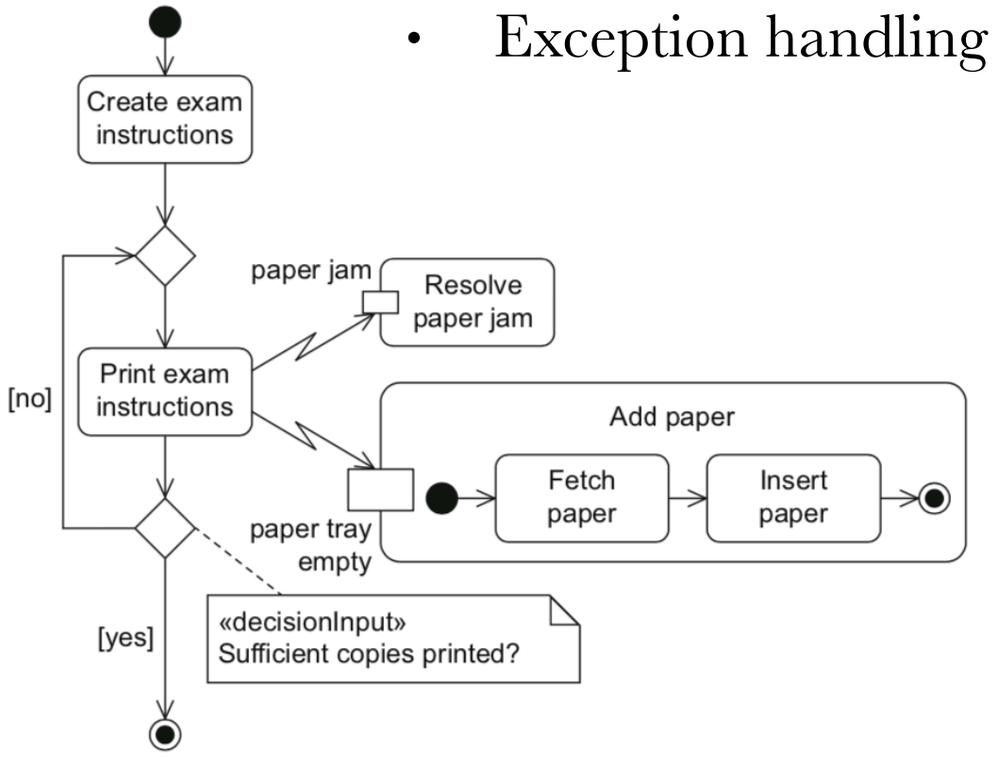
Name	*Notation*	*Description*
Partition	A B / A / B	Grouping of nodes and edges within an activity
Send signal action	S	Transmission of a signal to a receiver
Asynchronous accept (time) event action	E or T	Wait for an event E or a time event T
Exception handler	e Exception-Handler; Action	Exception handler is executed instead of the action in the event of an error e
Interruptible activity region	B; E; A	Flow continues on a different path if event E is detected

Source:
Seidl, Martina, Marion Brandsteidl, Christian Huemer, and Gerti Kappel. *UML@ Classroom*. Dpunkt, 2012.

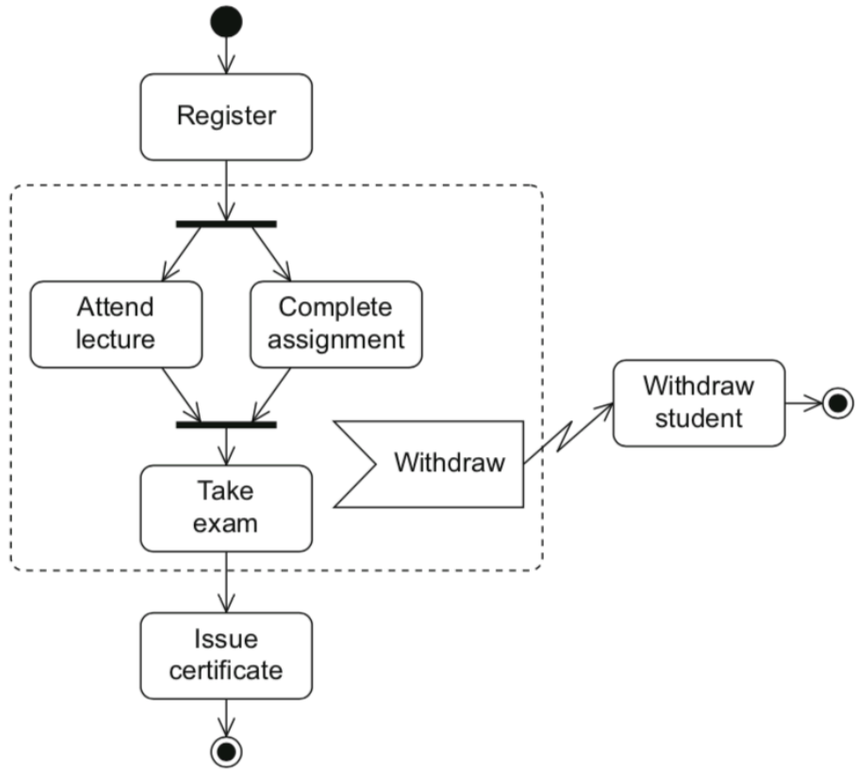
Some interesting bits

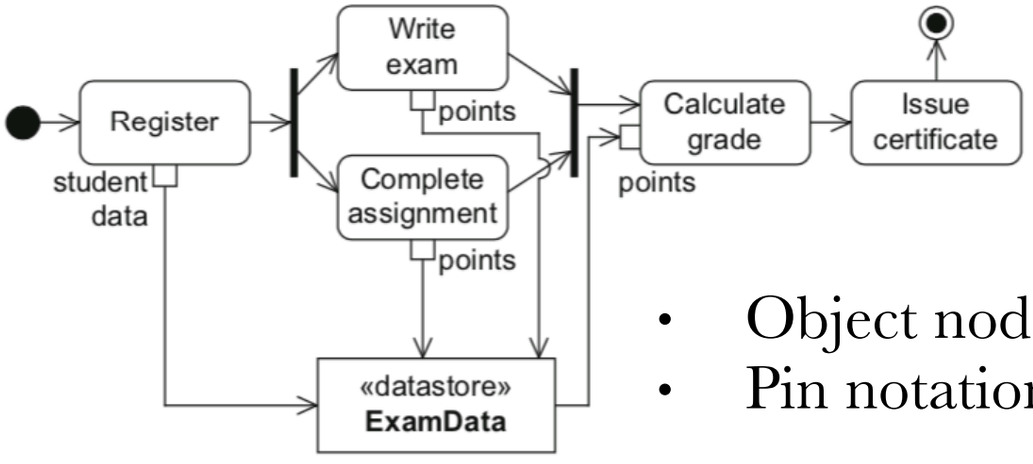
Source:
Seidl, Martina, Marion Brandsteidl, Christian Huemer, and Gerti Kappel. *UML@ Classroom*. Dpunkt, 2012.

- Exception handling



- Interruptible activity region





- Object nodes (datastore)
- Pin notation