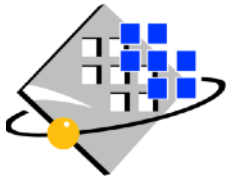
université
PARIS-SACLAY



L3 Mention Informatique
Parcours Informatique et MIAGE

Génie Logiciel Avancé - Advanced Software Engineering

A Brief Revision of UML

Burkhart Wolff
burkhart.wolff@universite-paris-saclay.fr

Why UML in a Software Engineering Course ?

- It is clearly not perfect
  - very syntactic, very diagrammatic flavour
  - diagrams do not necessarily scale up
  - not everyone in industry uses it (large companies typically have there own development process, reflecting their own «company culture »)
- BUT: Many in industry use it,
  - or use similar things (SysML), and most practitioners in industry would understand UML
  - we use it in requirements analysis, design, and for test generation techniques.

Plan of the Chapter

- The UML notation is used as document - core in SE processes (such as RUP or the V model)
- Syntax and semantics of class model elements and their visualisation in diagrams
  - Class Invariants
  - Constraints
  - Operations
  - Pre- and Post-Conditions
- Syntax and semantics of state machines
  Specify system components for test and verification

The UML ...

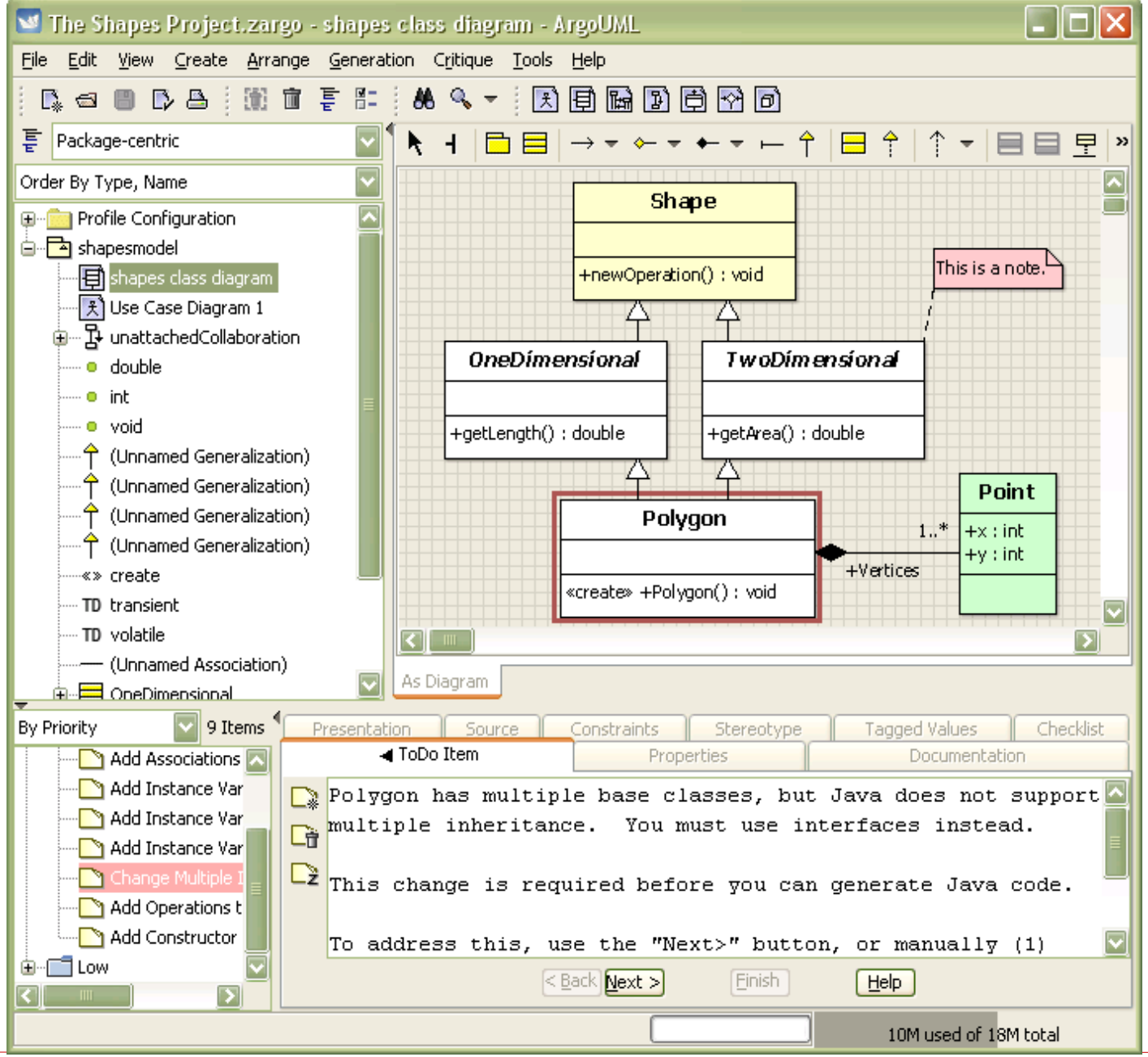
- ... is the Unified Modeling Language
- ... is a normed data-structure, a „technical format" of model-elements (that may contain model-elements) with consistent naming for
  - various system descriptions
  - various code formats
- ... has various external representations
  - as XMI exchange format (XML-formal)
  - as ECore Model
  - as UML diagrams

The UML offers the advantage ...

- ... of being a basis for

  Integrated Development Environments (IDEs)

  (like ArgoUML, Poseidon,
  Eclipse + Papyrus, IBM Rational Rhapsody, MS Visio,
  GenMyModels, MagicDraw, LucidChart,...)


The Shapes Project.zargo - shapes class diagram - ArgoUML
File Edit View Create Arrange Generation Critique Tools Help
Package-centric
Order By Type, Name
Profile Configuration
shapesmodel
shapes class diagram
Use Case Diagram 1
unattachedCollaboration
double
int
void
(Unnamed Generalization)
(Unnamed Generalization)
(Unnamed Generalization)
(Unnamed Generalization)
create
transient
volatile
(Unnamed Association)
OneDimensional
Shape
+newOperation() : void
This is a note.
OneDimensional
+getLength() : double
TwoDimensional
+getArea() : double
Polygon
«create» +Polygon() : void
1..*
+Vertices
Point
+x : int
+y : int
As Diagram
By Priority
9 Items
Add Associations
Add Instance Var
Add Instance Var
Add Instance Var
Change Multiple I
Add Operations t
Add Constructor
Low
Presentation
Source
Constraints
Stereotype
Tagged Values
Checklist
ToDo Item
Properties
Documentation
Polygon has multiple base classes, but Java does not support multiple inheritance. You must use interfaces instead.
This change is required before you can generate Java code.
To address this, use the "Next>" button, or manually (1)
< Back
Next >
Finish
Help
10M used of 18M total

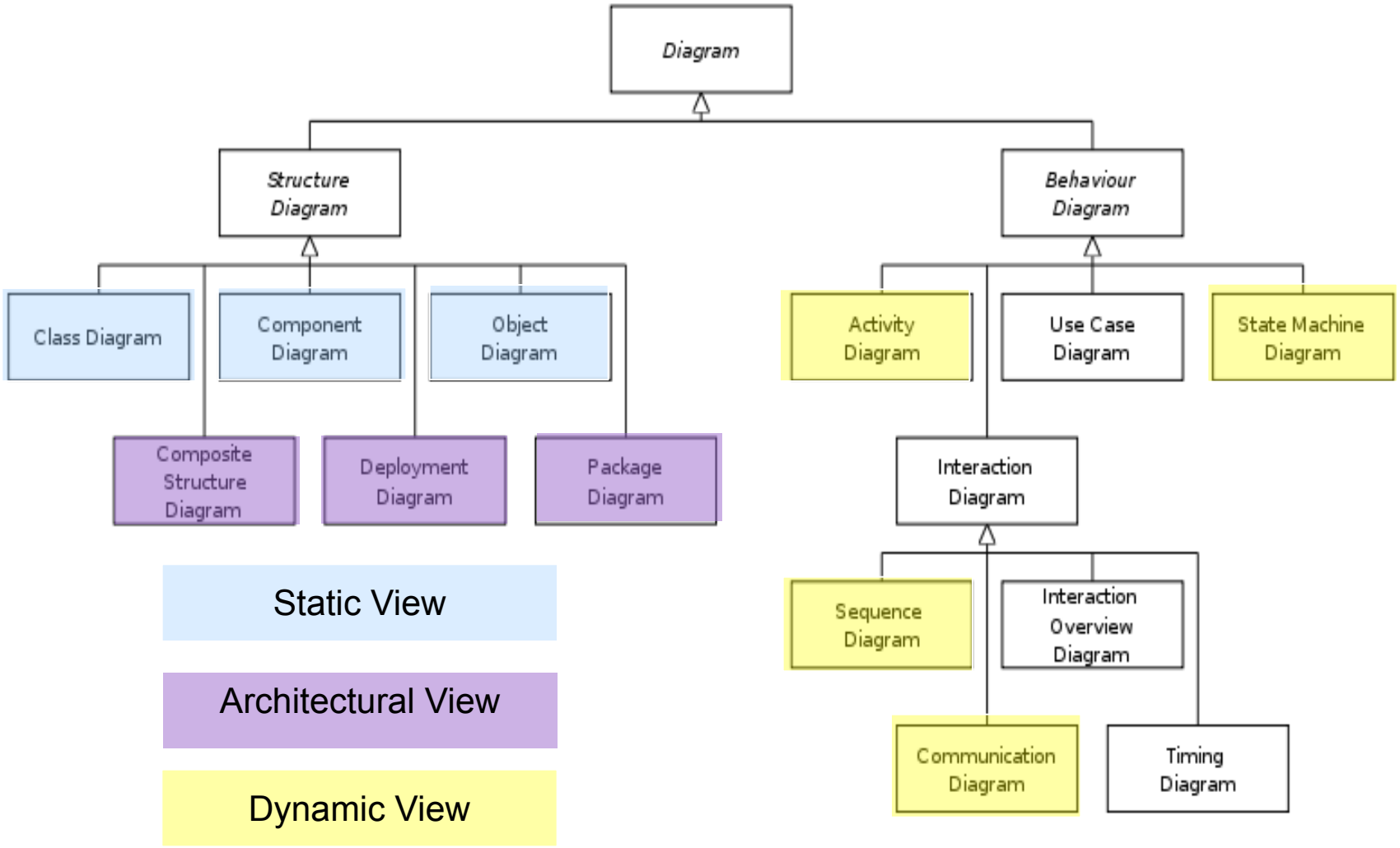
The UML offers the advantage ...

- ... to offer „object-oriented" specifications
- ... to offer a formal, mathematical semantics
  (well, at least to some parts of the UML)
- ... to be fairly widely used in industry, even
  if not always supported entirely or used
  in similar variants like SysML
- ... is the basis for a whole software-engineering
  paradigm called Model-Driven Engineering (MDE).

The UML 2.4 Diagrams (for corresp. models)

- UML, Version 1.1 : 9 types of diagrams
- UML, Version 2.4 adds 5 more types of diagrams
  - structure composition
  - communication
  - packaging
  - temporal constraints (timing)

The UML 2.4 Diagrams (corresponding to models)



Principal UML diagram types (1)

- **Structure and Visualization**
  - **Use Case Models and Use Case Diagrams**
  - **Sequence Models and Sequence Diagrams**
  - **State Machines and State Charts**
  - **Class Models and Class Diagrams**
  - **Object Graphs and Object Diagrams**

In Eclipse, all these Model Elements are described in a UML-document itself, the „Meta-Object-Framework" (MOF)

Bibligraphy

- **UML @ Classroom: An Introduction to Object-Oriented Modeling, Springer, 2015**
- **UML 2.0, Martin Fowler, Campus Press, 2004**
- **UML 2 et les Design Patterns, G. Larman, Campus Press, 2005**
- **http://www.omg.org/gettingstarted/what_is_uml.htm**
  http://www.eecs.ucf.edu/~leavens/JML/
  http://www.junit.org/

Using the UML

- A General Remark:
- The UML can be used in the

  Analysis Phase ("D1 - Documents")

  as well as the

  Design Phase ("D2 - Documents")
- This changes the character of the Models and the resp. Diagrams substantially.

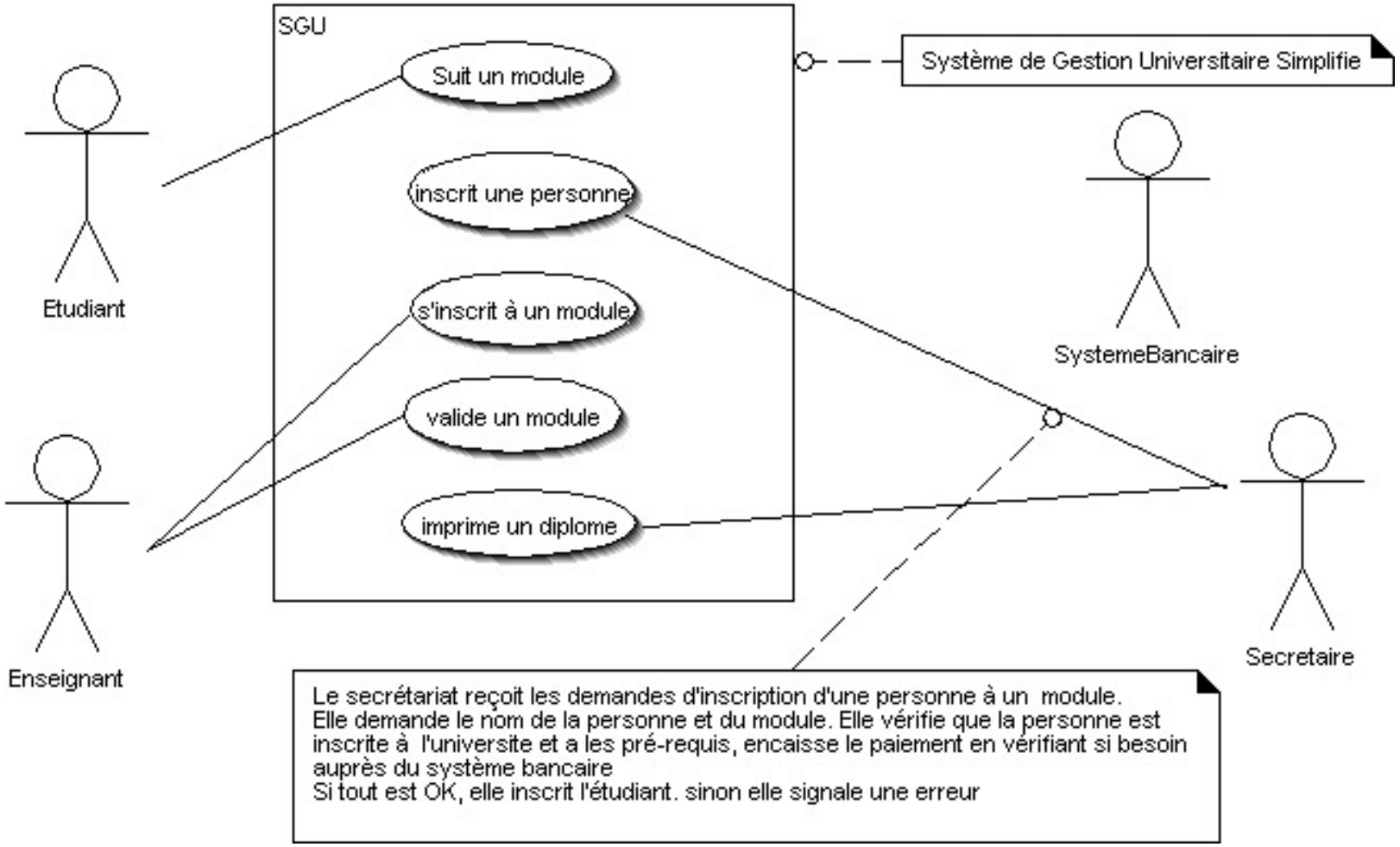
And now, lets get our hands dirty ...

Principal UML diagram types (1)

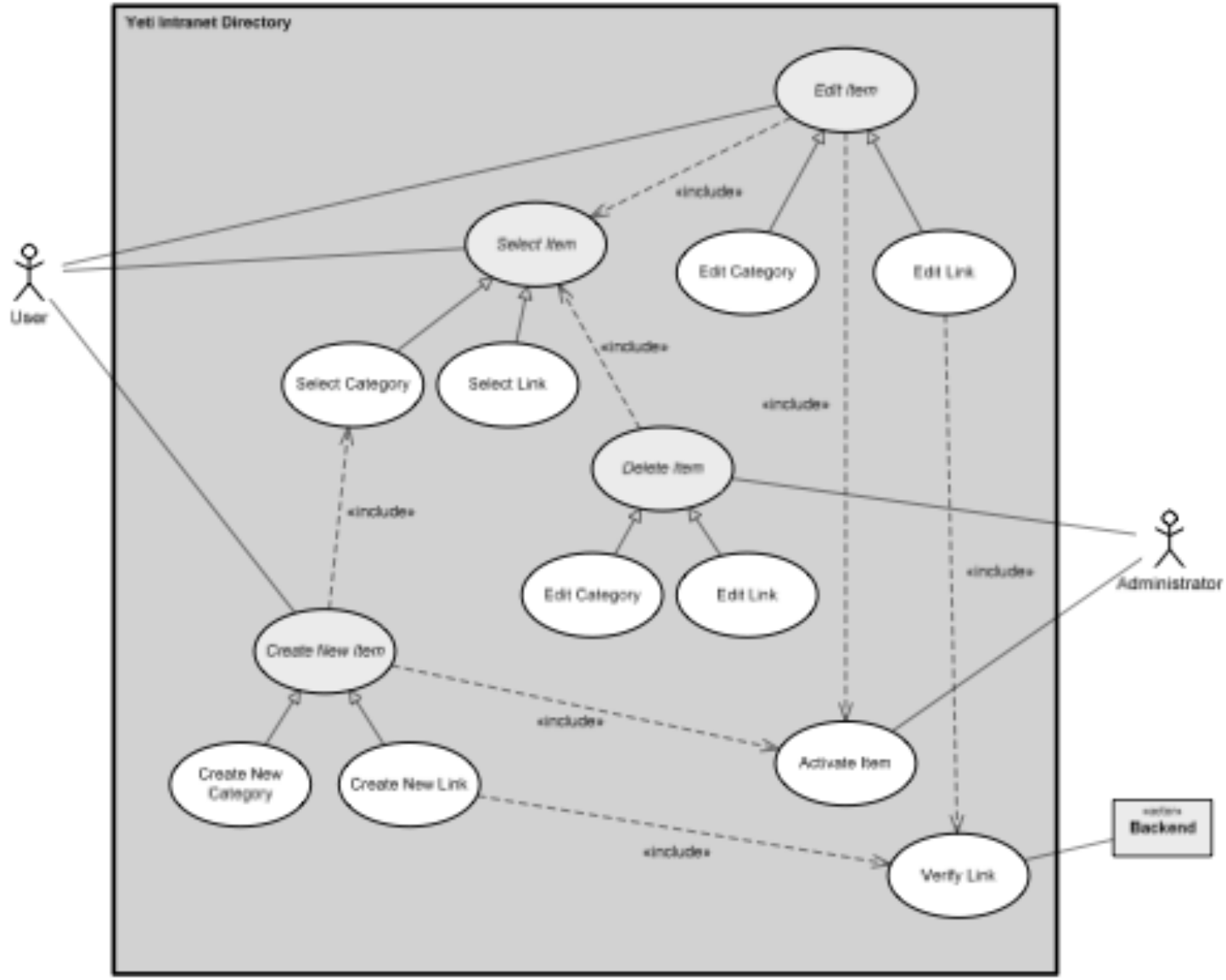
- **Use Case Diagrams** („Diagrammes des cas d'utilisation") : models the system **operations by**
  - the **interactions** of the system with the external world (external agents communicating with the system seen as a black box.)
  - Just the principle cases, the alternatives, the extensions

Emphasis on (top-level) functionality !

Example: Use Case Diagram (Analysis)



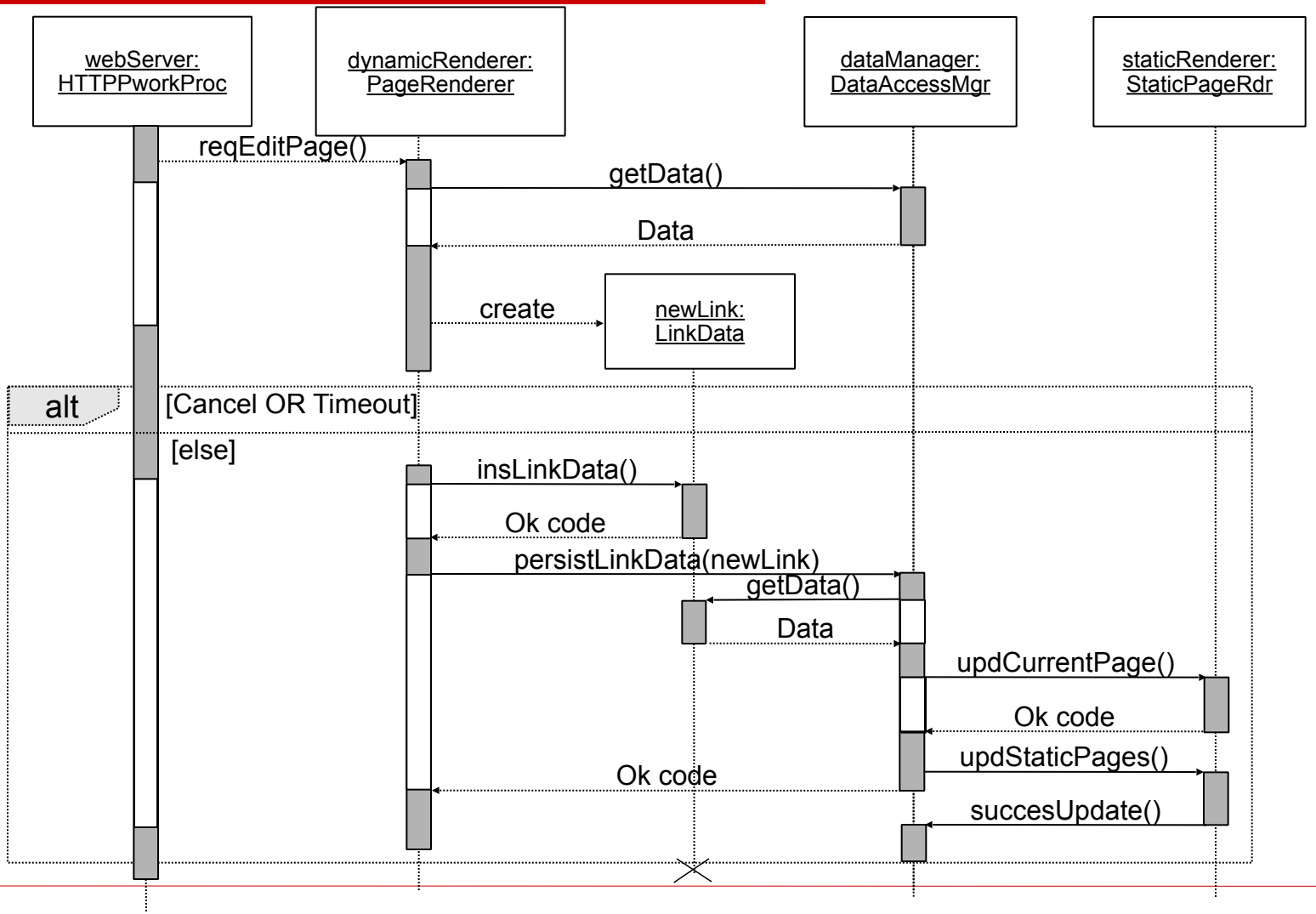
Example: Use Case Diagram (Design)



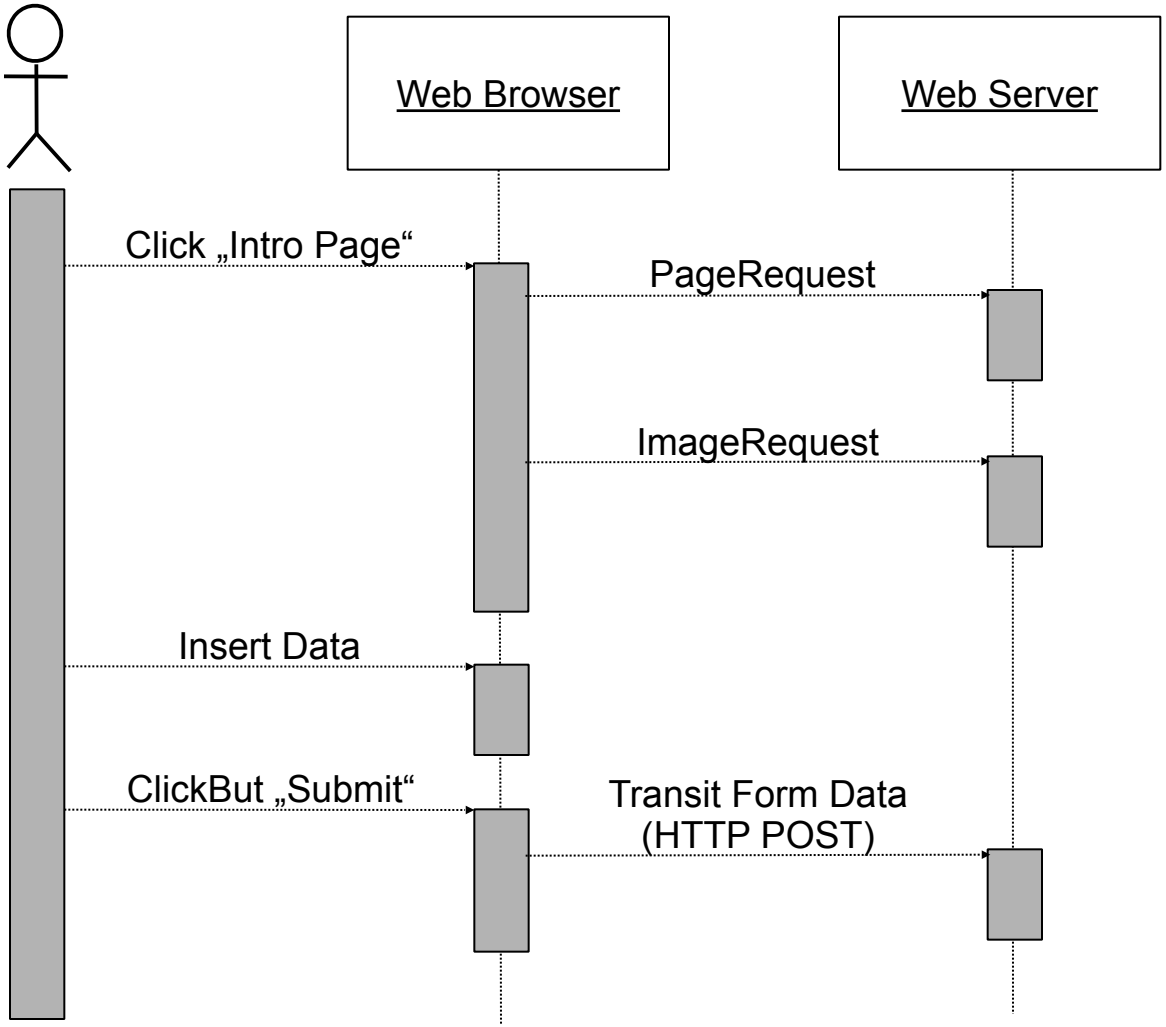
Principal UML diagram types (2)

- **Interaction Diagram** („Diagrammes d'interaction"):
  the interaction between objects for realizing a functionality
  - **SequenceDiagram**: privileged temporal description of exchanges of events. Notions of utilization **scenarios**.
  - **Collaboration Diagram**: centered around objects and top-level collaborations of them.

Example: Sequence Diagram (design-level)



Example: Sequence Diagram (analysis-level)



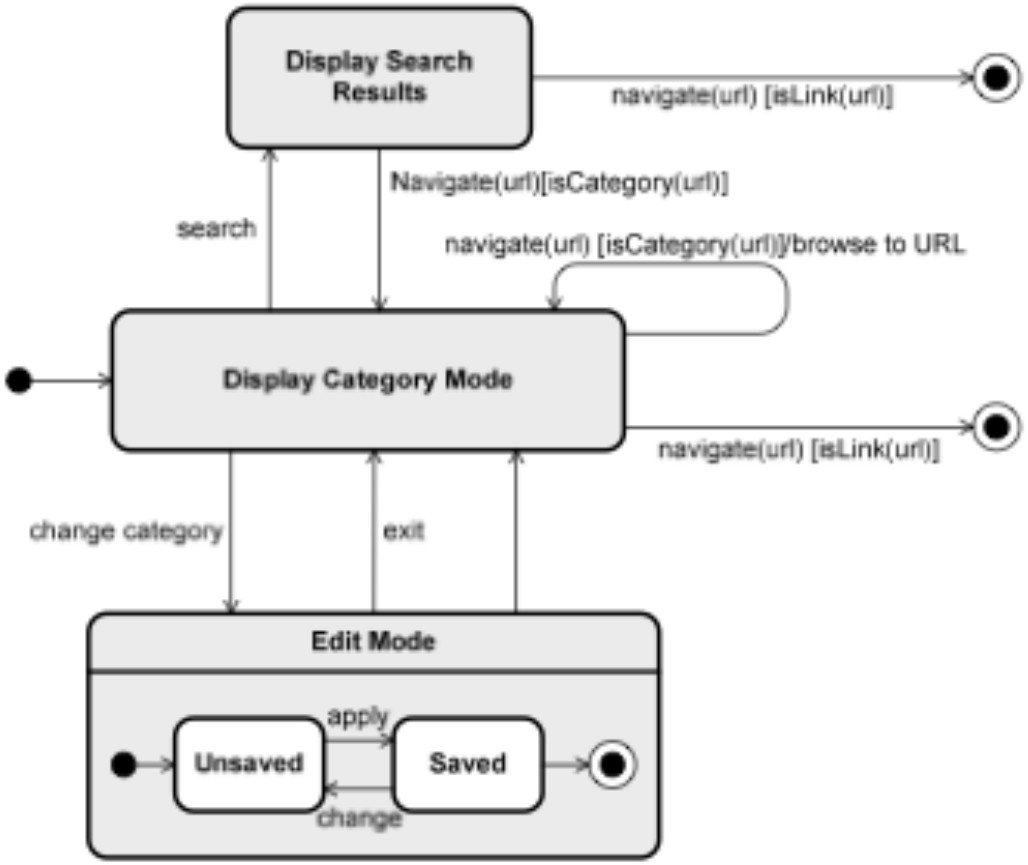
Principal UML diagram types (3)

- **State Charts** (ou « machine à états ») :

  a description of **behaviour** by (hierarchical) automata

  - interesting if an object reacts on
    events (asynchronous as well as synchronous)
    by the external environment
  - or if the internal state of an object leads to
    a somewhat interesting life-cycle of an object
    (transitions between well-characterized states of the object)

Example: **State Chart (design level)**



Summary: State Charts

- Two types can be distinguished:
  - **Semantics of Diagrams for requirements analysis:** many.
  - **Semantics of Diagrams for system design:** **many.**

    Can be interpreted in by automata, process calculi, Labelled Transition Systems (LTL) in several, reasonable ways (depends on context and application).

Main UML diagram type>:

- **Class Diagrams** („Diagrammes de classes") :

  the static **structure** of the DATA of the system

  - the classes of interest to be represented in the system
  - the relations between classes
  - the attributes and the methods
  - the types, required/defined interfaces …

  can be used for top-level views as specific interfaces for local code …

Example: A Class Diagram