



UML essential

Stéphane Ducasse
Stephane.Ducasse@univ-savoie.fr
<http://www.iam.unibe.ch/~ducasse/>

S.Ducasse

1

Goals

- What/Why UML?
- Static
- Dynamic
- 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.



S.Ducasse

2



UML

- What is UML?
- uniform notation: Booch + OMT + Use Cases (+ state charts)
 - UML is not a method or process
 - ... The Unified Development Process is
- Why a Graphical Modeling Language?
 - 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

S.Ducasse

3



Why UML?

- Why UML?
- Represents industry standard
 - more tool support, more people understand your diagrams, less education
- Is reasonably well-defined
 - ... although there are interpretations and dialects
- Is open
 - stereotypes, tags and constraints to extend basic constructs
 - has a meta-meta-model for advanced extensions

S.Ducasse

4



Goals

- What/Why UML
- **Static**
- Dynamic



S.Ducasse

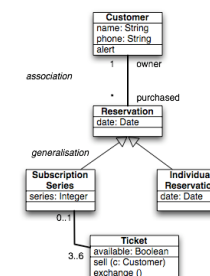
5



Class Diagrams

“Class diagrams show generic descriptions of possible systems, and object diagrams show particular instantiations of systems and their behaviour.”

Attributes and operations are also collectively called **features**.

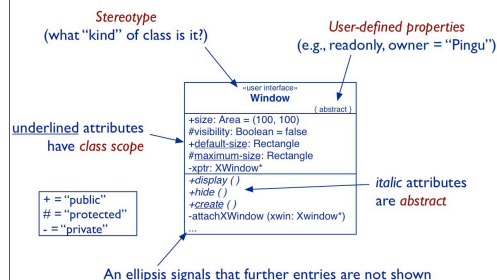


S.Ducasse

6



Features

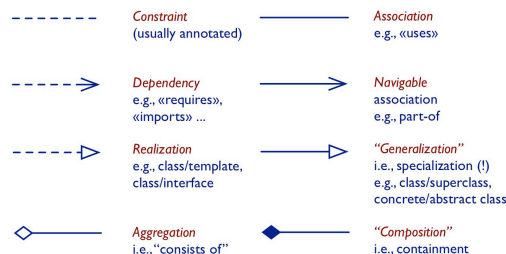


S.Ducasse

7



UML Lines and Arrows



S.Ducasse

8



Objects

Objects are shown as rectangles with their name and type underlined in one compartment, and attribute values, optionally, in a second compartment.

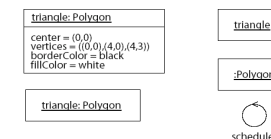


Figure 13-134. Object notation

At least one of the name or the type must be present.

S.Ducasse

9



Generalization

A **subclass** specializes its superclass:

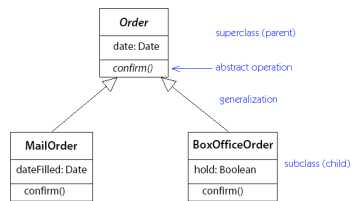


Figure 4-7. Generalization notation



Goals

- What/Why UML
- Static
- **Dynamic**
 - Use Case Diagrams
 - Sequence Diagrams
 - Collaboration (Communication) Diagrams
 - Statechart Diagrams



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.

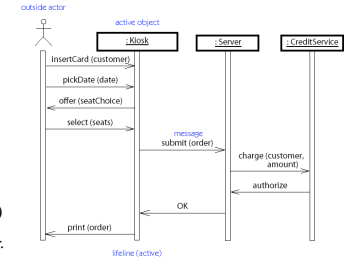


Figure 8-1. Sequence diagram



Collaboration Diagrams

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

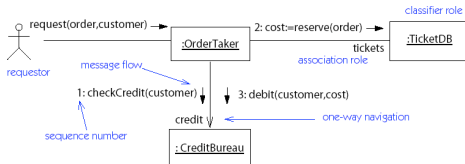


Figure 8-3. Collaboration diagram



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**.

1. **Prior messages** from other threads (e.g. "[A1.3, B6.7.1]") only needed with concurrent flow of control
2. 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" ["1" = 1..n])
conditional indicator (e.g., "2.3" ["#items = 0"])
- **Return value binding** (e.g., "status :=")
- **Message name**
1. event or operation name
5. **Argument list**



Nested Message Flows

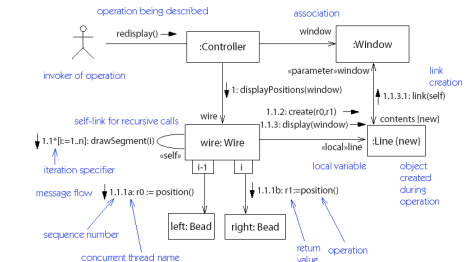


Figure 13-51. Collaboration diagram with message flows



Statechart Diagrams

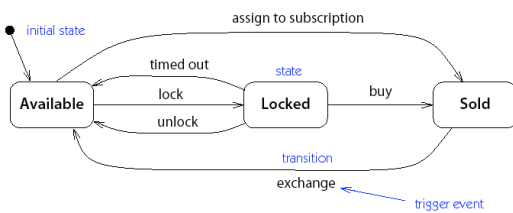


Figure 3-5. Statechart diagram



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



UML

You will get a complete lecture on that...

A notation not a process

Static

Classes

Relationships

Generalization

Dynamic

Messages

Flows

