The main concepts for object approaches and UML

Part 1 :
« The static models »

Content

- Part 1 : the static models
  - The foundations of object approaches
    - Model and representation
    - Artificial intelligence and computer science
    - The various languages
    - UML… history and diagrams
  - The basic concepts of object approaches
    - The objects and their description
    - The classes and the instantiation
    - The associations among classes
    - The specialization among classes
    - The constraints
Notion de Modèle et de Représentation

- A model: a simplification of reality
  - A is a model of B if we can answer to some questions on B by manipulating A (Minsky 74)

- A representation: the form of a model
  - Mathematical equations
  - Geometry (e.g. GIS)
  - Logics
  - The object-centered representations (Artificial Intelligence)
  - The object-oriented languages (Computer Science)

The origins of object-oriented representations

Artificial intelligence: Knowledge representation

Logics

Object-centered representations

Ontologies

Computer science: Program structure

Functional programming (equations)

Object-oriented programming
Artificial intelligence: knowledge representation

The logical approach
- To describe the world by a set of facts, rules about what is true or false
- Cognitivism

The object-centered approach
- To describe the world by concrete (the notion of table, of chair, etc.) or abstract (the form, the beauty,...) concepts and their relations:
  - Frame languages (Minsky) or semantic networks
- Organization in taxonomies: the ontologies, the description logics

Computer science: program structure

The functional approach
- Representation of variables, functions and equations.
- Scientific culture

The object-oriented approach
- Abstract representations of entities having a material (tree, person) or virtual (social security, bank account,...).
- Implicit and philosophical culture
- Confluence of various computer science disciplines
Le cycle du logiciel

- **Requirement analysis:**
  - Specification of the needs of the user
    - Ex.: to write letters

- **Analysis:**
  - Description of the notions and functionalities from the user point of view
    - Ex.: what is a letter, how it is composed, which manipulations?

- **Design:**
  - Translation of these notions in computer science terms
    - Ex.: to insert characters within a buffer

- **Implementation:**
  - Realization in a given language
    - Ex.: `addressBuffer.insert(typedChar, currentPosition);`

UML: why?

- **Before:**
  - Only trace of the process: the final program

- **Need:**
  - Traceability of the entire cycle:
    - Requirement analysis: formal representation of the needs
    - Analysis: descriptions of the user's notions and practices
    - Design: choice of software structure: data and algorithms at various levels of details
    - Implementation: writing of an actual program in a programming language
UML: why?

Need of standardized representations through the whole process:

- **Object-centered representations for the analysis**
  - We propose to use UML as a kind of knowledge representation language
- **Object-oriented representations for the design**
  - UML in the computer scientist way
- **Object-oriented languages (or others) for implementation**
  - Simula-67, Smalltalk-82, Objective-C, C++, CLOS, Ada95, Java,…

History (1)

- More than 50 object methods in 90-95: Booch, Class-Relation, OMT, OOA, OOD, OOM,…
- Only 3 methods really emerged:
    - OMT: Object Modeling Techniques
    - General Electric
  - The BOOCH’93 method by Grady Booch: OOD, BOOCH’93 (RATIONAL society)
    - 1987 for ADA,
    - 1990 general
  - The OOSE method by Ivar Jacobson: Objectory 1992,
    - OOSE: Object Oriented Software Engineering
    - Ericsson
- Grouping of BOOCH-OMT then Objectory
UML: a formal description language

- Accepted by OMG l’OMG – 1997
- A unique common language:
  - A meta-model
  - A « less » ambiguous language
  - A simple graphical notation,
    - Understandable by non computer scientist
    - Allows communication among designers/actors
- Became THE reference for object-oriented modelling

UML: Its diagrams

Some UML diagrams

- Use-case diagrams  ➔ users’ needs
- Class & object diagrams ➔ structural descriptions (static)
- Statechart diagrams ➔ object dynamics
- Activity diagrams ➔ operation dynamics
- Interaction diagrams ➔ scenarios and messages streams
- Implementation diagrams ➔ work units
- Composite diagrams ➔ diagrams articulation
Part 1: the static models

- The foundations of object approaches
  - Model and representation
  - Artificial intelligence and computer science
  - The various languages
  - UML… history and diagrams

- The basic concepts of object approaches
  - The objects and their description
  - The classes and instances
  - The associations
  - The specialization and generalization

In UML, the structural or static model is represented with two kinds of diagrams

- Class diagrams
  - Descriptions of categories with classes, taxonomies and associations.

- Object diagrams
  - Descriptions of a concrete system with objects and links.
An object has two aspects:

- **A static aspect**
  - The properties or attributes describing the external appearance/structure
  - weight, size, sex, age...

- **A dynamical aspect**
  - The spontaneous/proactive behaviours:
    - To walk, to swim, to reproduce, to grow,...
  - The reactive behaviours:
    - To feed, to push,...

The objects: example

- Any object has these two aspects:
  - A static aspect
  - A dynamic aspect

**A tortoise in a simulation**

**Properties**
- weight
- size
- sex
- age...

**Behaviours**
- walk
- swim
- reproduce
- grow

**A window on a UI**

**Properties**
- position
- size
- texture
- title
...

**Behaviours**
- open
- activate
- close
- suspend
- reduce (coeff)
- increase (coeff)
- move (place)
Content

Part 1 : the static models

- The foundations of object approaches
  - Model and representation
  - Artificial intelligence and computer science
  - The various languages
  - UML... history and diagrams
- The basic concepts of object approaches
  - The objects and their description
  - The classes and instances
  - The associations
  - The specialization and generalization

Classes and instances

First abstraction

- A class can be seen as

  - The description of a category of objects (intentional description)
    - Having the same structure (same set of properties),
    - Doing the same thing (spontaneous or reactive, possibly in various ways),
  - The set of all the objects belonging to this category (extensional description)
Classes and instances

- A class is a structural model of a set of similar objects called the instances of this class.

- An object is an instance of only one class:
  - It has the structure specified by the class,
  - It has a specific value for each property specified in the class,
  - It has the behaviours specified by the class.

- Each object has a unique identity.
The structural model

UML: the structural diagrams (static)

- **Class diagram**
  - **Class name**: Car
  - **Attributes**: List of properties
    - name
    - company
    - color
    - paint
  - **Operations**: List of behaviours
  - Objects - J.P. Müller, P. Bommel

Object diagram
- titine : Car
  - « 205 »
  - « Peugeot »
  - « red »

Classes and Instances

- **The class « Tortoise »**
  - **Properties**
    - weight : entier
    - size : entier
    - sex : chaine
    - age : entier
    - walk
    - swim
    - eat
    - reproduce
    - grow
  - **Behaviours**
  - Object diagram
    - momo : Tortoise
      - weight : 87
      - size : 50
      - sex : « masculin »
      - age : 127
    - mimie : Tortoise
      - weight : 89
      - size : 47
      - sex : « feminin »
      - age : 132

Objects - J.P. Müller, P. Bommel
Classes and instances

Herd
- population : integer
- nbMales : integer
- nbFemales : string

Farmer
- name : string
- size : float
- age : integer

Instance of

:Herd
population 12
nbMales 7
nbFemales 5

Instance of

:Farmer
name « Moussa »
size 50
age 52

Content

Part 1 : the static models
- The foundations of object approaches
  - Model and representation
  - Artificial intelligence and computer science
  - The various languages
  - UML… history and diagrams
- The basic concepts of object approaches
  - The objects and their description
  - The classes and instances
  - The associations
  - The specialization and generalization
The associations

- An association is a permanent semantic relationship between two classes (*Roques, 2005*)

  Ex: A farmer owns a herd

- A relation is non-directional

  Ex: there is a « ownership » relation between a farmer and a herd

The associations: representation

**Association / Link**  
(same as Class / Instance)

```
<table>
<thead>
<tr>
<th>Farmer</th>
<th>ownership</th>
<th>Farm</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Association</td>
<td>name</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>:f2: Farmer</td>
<td>ownership</td>
<td>:Farm</td>
</tr>
<tr>
<td>name='Léopold'</td>
<td>Link</td>
<td>Name='La prairie merveilleuse'</td>
</tr>
</tbody>
</table>
```
**The associations**

**Multiplicity and roles of an association**

- **Farmer**
  - name: string
  - size: integer
  - age: integer

- **Farm**
  - name: string
  - surface: integer

- **Ownership**
  - owner: 1
  - farm: 1

- **Employment**
  - employer: 1..*
  - employee: 0..1

- **Tortoise**
  - weight
  - size
  - age
  - sex
  - partner: 0..1

- **Company**
  - name
  - address

- **Person**
  - name
  - birthday
  - address
  - chief: 0..1
  - subordinate: 1..*
Example of an object diagram: parents-children

```
Paul : Person
<table>
<thead>
<tr>
<th>parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
</tr>
<tr>
<td>Marie</td>
</tr>
<tr>
<td>Boris</td>
</tr>
</tbody>
</table>

Lea : Person
<table>
<thead>
<tr>
<th>parent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
</tr>
<tr>
<td>Marie</td>
</tr>
</tbody>
</table>

Lulu : Person
<table>
<thead>
<tr>
<th>child</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marie</td>
</tr>
</tbody>
</table>
```

Multiplicity

1 - exactly 1

0..1 - at most 1

0..* - none, 1 or more (default)

1..* - at least 1

2..4* - from 2 to 4

2,4 - 2 or 4
Difference between an attribute and an association

Les attributs:

- An attribute is the role played in an association with a basic category (strings, integers, floats, etc.).

![Diagram showing a Farmer with attributes size, age, and name, associated with Integer and String types.]

- What is considered basic can be different in different domains.

Specific associations (composition / aggregation)

Aggregation

- A specific non-symmetric association

- Express the whole-part relationships with a notion of containment

- The parts do not disappear with the whole
**Aggregation**

Collection/element semantics

- Forest 1
- Farm 1
- Herd 1
- Tree 1..*
- Plots 1..*
- Animal 1..*
- Country 1
- Region 1
- Department 1

**Specific associations (composition / aggregation)**

- **Composition**
  - Express the strong whole-part relationship

  - Directory 1
    - File 1..*
  - Book 1
    - Page 1..*

  ➤ The parts disappear with the whole
Part 1: the static models
- The foundations of object approaches
  - Model and representation
  - Artificial intelligence and computer science
  - The various languages
  - UML... history and diagrams
- The basic concepts of object approaches
  - The objects and their description
  - The classes and instances
  - The associations
  - The specialization and generalization

Specialization and generalization
- Intellectual mechanism
  - Either one adds details (specialisation)
  - Or one abstracts from details (generalisation):
    => Second abstraction
- Semantics
  - Set-therorectic point of view (The ostriches are included in the birds)
  - Logical point of view (The birds have feathers, then the ostriches have)
**Specialization and generalization**

- **Person**
  - name
  - size
  - age
  - grow

- **Farmer**
  - cultivate

- **Merchant**
  - stock
  - buy
  - sell

**Inheritance**

- A subclass is a specialization of its superclass
- A subclass « inherits » its descriptions from its superclass:
  - Same properties,
  - Same behaviours (even if performed differently),
  - Same associations,
  - Same constraints
- A subclass can refine the properties and add new ones, define behaviours in more specific ways
Aggregation et generalization!

- **Aggregation**
  - Links between objects
  - An aggregation tree describes the structure of an object

- **Generalization**
  - Links between classes
  - A generalization tree describes a taxonomy of categories

!! Do not confuse!!

**Aggregation and generalization!**

```
Object
   /     \
LivingBeing  ExploitationUnit  Surface
   /         \                  /
   Plant     TreePlot          Tree
         /     \              /
        Forest  Landscape  
```
Aggregation and generalization

A classical example

QuickTime™ et un décompresseur TIFF (LZW) sont requis pour visionner cette image.