Introduction to UML and class modeling (part 1)

Fabrizio Maria Maggi
Institute of Computer Science
UML and OO approaches

Unified Modeling Language (UML) is a standardized, general-purpose modeling language. UML includes a set of graphic notation techniques to create visual models of object-oriented (OO) software-intensive systems.

An OO approach includes 4 aspects:

- Identity
  - Data organized into discrete distinguishable entities (*objects*)

- Classification/Abstraction
  - Objects with the same attributes and operations are grouped into a *class*
  - Each object is said to be an *instance* of its class

- Inheritance
  - Sharing of attributes and operations (*features*) among classes based on a hierarchical relationship
  - A superclass has general information that subclasses refine and elaborate

- Polymorphism
  - The same operation may behave differently for different classes
UML models

- **Class models**
  - Static structure of objects and their relationships
  - *Class diagrams*
    - Nodes are classes and arcs are relationships among classes

- **Interaction models**
  - Interactions can be modeled at different levels of abstraction
    - At a high level use cases describe how a system interacts with outside actors
      - Each use case represents a functionality that a system provides to the user
      - Use cases are helpful for capturing informal requirements
    - Sequence diagrams provide more details about which operations need to be invoked in a specific scenario
Software Development Methodology

Domain (Class) Model → Interaction Modelling → Application (Class) Model

Code Generation
Software Development Methodology

Domain Classes; Attributes; Relations

Domain (Class) Model

Interaction Modelling

Application (Class) Model

Code Generation
Domain (Class) Model
## Domain (Class) Model

| Domain Entities |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| ![Image of Domain Entities](image) | ![Image of Domain Entities](image) | ![Image of Domain Entities](image) | ![Image of Domain Entities](image) | ![Image of Domain Entities](image) | ![Image of Domain Entities](image) | ![Image of Domain Entities](image) | ![Image of Domain Entities](image) | ![Image of Domain Entities](image) | ![Image of Domain Entities](image) | ![Image of Domain Entities](image) | ![Image of Domain Entities](image) |
Domain (Class) Model

WHAT?
Domain (Class) Model

- To answer this question, the domain model provides classes with attributes and relations among them
- Operations are not specified
Software Development Methodology

Domain (Class) Model → Interaction Modelling → Application (Class) Model → Code Generation
Software Development Methodology

- Domain (Class) Model
- Interaction Modelling
- Application (Class) Model
- Code Generation

Instrument for identifying the right interfaces/operations
Interaction Modelling
How do objects interact?
Interaction Modelling
Software Development Methodology

- Domain (Class) Model
- Application (Class) Model
- Interaction Modelling
- Code Generation
Software Development

- Domain (Class) Model
  - Domain Classes
  - Application Classes (e.g., Patterns)
  - Attributes
  - Relations
  - Operations

- Interaction Modelling

- Application (Class) Model

- Code
Application (Class) Model
## Application (Class) Model

<table>
<thead>
<tr>
<th>Operations/Interfaces</th>
<th>carry</th>
<th>come</th>
<th>drink</th>
<th>go</th>
<th>lead</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ask</td>
<td>catch</td>
<td>cook</td>
<td>drive</td>
<td>hit</td>
</tr>
<tr>
<td></td>
<td>bake</td>
<td>clap</td>
<td>cry</td>
<td>eat</td>
<td>hop</td>
</tr>
<tr>
<td></td>
<td>bite</td>
<td>clean</td>
<td>cut</td>
<td>float</td>
<td>juggle</td>
</tr>
<tr>
<td></td>
<td>bounce</td>
<td>climb</td>
<td>dance</td>
<td>fly</td>
<td>jump</td>
</tr>
<tr>
<td></td>
<td>brush</td>
<td>close</td>
<td>dig</td>
<td>fold</td>
<td>kick</td>
</tr>
<tr>
<td></td>
<td>build</td>
<td>color</td>
<td>draw</td>
<td>follow</td>
<td>knock</td>
</tr>
<tr>
<td></td>
<td>call</td>
<td>comb</td>
<td>dream</td>
<td>give</td>
<td>laugh</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>open</td>
</tr>
</tbody>
</table>
Application (Class) Model
Application (Class) Model
Application (Class) Model
Software Development Methodology

Domain (Class) Model → Interaction Modelling → Application (Class) Model

Code Generation
Software Development Methodology

Domain Classes; Attributes; Relations

Domain (Class) Model

Interaction Modelling

Application (Class) Model

Code Generation

Domain Classes; Attributes; Relations
Class Modelling

- **Classes**
  - A class describes a group of objects with the same properties (attributes), behavior (operations), kinds of relationships and semantics
  - Classes often appears as nouns in problem descriptions with users

- **Objects**
  - An object is a concept, abstraction or thing with identity that has a meaning for an application
  - An object is an instance of a class
How Many Classes? And Instances?
Class Diagrams

- **Class**
  - UML notation: box with a class name

- **Object**
  - UML notation: box with an object name followed by a colon and a class name. The object name and the class name are both underlined
Attributes and Values

- **Attribute**
  - An attribute is a named property of a class that describes a value held by each object of the class.
  - UML notation: attributes are listed in the second compartment of the class box. Optional details, such as type and default value, may follow each attribute.

![Person](image)

- **Value**
  - A value is a piece of data.
  - UML notation: values are listed in the second compartment of the object box.

![JoeSmith: Person](image)  ![MarySharp: Person](image)
Class? Attributes? Values?
Class? Attributes? Values?
Operations and Methods

- **Operation**
  - An operation is a function or procedure that may be applied by or to objects in a class
  - UML notation: operations are listed in the third compartment of the class box

- **Method**
  - A method is the implementation of an operation for a class
Operations?
Visibility for Attributes and Operations

- + public
- # protected
- - private
- ~ package
## Visibility for Attributes and Operations

<table>
<thead>
<tr>
<th>Keyword</th>
<th>C#</th>
<th>C++</th>
<th>Java</th>
</tr>
</thead>
<tbody>
<tr>
<td>private</td>
<td>class</td>
<td>class</td>
<td>class</td>
</tr>
<tr>
<td>protected</td>
<td>derived classes</td>
<td>derived classes</td>
<td>derived classes and/or within same package</td>
</tr>
<tr>
<td>package</td>
<td>-</td>
<td>-</td>
<td>within its package</td>
</tr>
<tr>
<td>public</td>
<td>everybody</td>
<td>everybody</td>
<td>everybody</td>
</tr>
</tbody>
</table>
(Binary) Links and Associations

- **Link**
  - A link is a physical or conceptual connection among objects
  - UML notation: line between objects. A link can have a name (underlined)

- **Association**
  - An association is a description of a group of links with common structure and common semantics
  - UML notation: line between classes. An association can have a name (not underlined)
How Many Associations? Links?
Multiplicity

- Specifies the number of instances of one class that may relate to a single instance of an associated class
- UML notation: specified at the end of the association lines
  - Examples: “1” (exactly one); “3..5” (three to five, inclusive); “*” (many, zero or more)
Multiplicities?
Multiplicity many-to-many

[Diagram showing relationships between Person and Company, with specific instances for John, Mary, Sue, Alice, and Jeff.]
Multiplicity one-to-one

- **Country**
  - name
  - HasCapital
  - 1

- **CapitalCity**
  - name
  - HasCapital
  - 1

- **Canada : Country**
  - = Ottawa
  - name = “Canada”
  - : HasCapital

- **Ottawa : CapitalCity**
  - = Canada
  - name = “Ottawa”

- **France : Country**
  - = Paris
  - name = “France”
  - : HasCapital

- **Paris : CapitalCity**
  - = France
  - name = “Paris”

- **Senegal : Country**
  - = Dakar
  - name = “Senegal”
  - : HasCapital

- **Dakar : CapitalCity**
  - = Senegal
  - name = “Dakar”

---

Systems modelling – Fabrizio Maria Maggi
Association end names

- Association ends can be provided with a name as well as with a multiplicity
Association end names

- Association end names are necessary for associations between two objects of the same class. They can also distinguish multiple associations between a pair of classes.
- Association end names as pseudo attributes
Associations as References

Association end name
MagicDraw might not produce the Collection type declaration. For example, it might generate the code "Course takes;" instead of "Set<Course> takes;". If you are having this issue, you can go back to the class diagram, double click on the association end "takes", then select "Language Properties" and then "Container". There you can select the type java.util.Set<type> (or any other container type that you prefer).
Constraints

- A constraint is a condition involving model elements, such as objects, classes, attributes, links, associations.
  - A constraint specifies limitations that implementers need to satisfy.
## Constraints

<table>
<thead>
<tr>
<th>Collection Type</th>
<th>isOrdered</th>
<th>isUnique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiset, bag</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>Sequence, array</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>Set</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>Ordered set</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>
Packages

- You can fit a class model on a single page for many small and medium-sized problems
  - However it is often difficult to grasp the entirety of a large model.
- A package is a group of elements (classes, associations, and nested packages) with a common theme.
  - A package partitions a model making it easier to understand and manage,
- The UML notation for a package is a box with a tab:
  - The tab suggests the enclosed content, like a tabbed folder.
Supplementary Material about Java

Suggested:  http://www.learnjavaonline.org/

A more structured course:
https://www.udacity.com/course/java-programming-basics--ud282
https://www.udacity.com/course/Object-Oriented-Programming-in-Java

Other material:
https://www.sololearn.com/Course/Java