The OFL Model to Customise Operational Semantics of Object Oriented Languages: Application to InterClasses Relationships

Pierre Crescenzo

The OFL Model to Customise Operational Semantics of Object-Oriented Languages: Application to Inter-Classes Relationships

Pierre Crescenzo

Pierre.Crescenzo@unice.fr
http://www.crescenzo.nom.fr/

Laboratoire I3S (UNSA/CNRS)
Projet OCL
2000, route des lucioles
Les Algorithmes, bâtiment Euclide B
BP 121
F-06903 Sophia Antipolis CEDEX
France

Abstract

Open Flexible Languages

OFL [CCCL 2000a, CCCL 2001, C 2001a, C 2001b] is the acronym for *Open Flexible Languages* and the name of a meta model for object-oriented programming languages based on classes. It relies on three essential concepts of

these languages: descriptions which are a generalisation of the notion of class, relationships such as inheritance or aggregation and languages themselves. OFL provides a customisation of these three concepts in order to adapt their operational semantics to the programmer's needs.

OFL allows to define and provide software components (for object-oriented programming language) such as:

a library of concepts-descriptions (equivalent to meta-descriptions)

- class
- generic class
- interface
- array
- basic type
- ...

a library of concepts-relationships (equivalent to meta-relationships)

- specialisation (and other derivatives of inheritance)
- generalisation (opposite of the previous, lacking in most of object-oriented programming language!)
- code-reuse (importation without polymorphism)
- aggregation (for attributes, parameters of method, ...)
- composition (for strengthened attributes, ...)
- view (like in DBMS)
- version (for evolution handling)
- ...

OFL allows to compose these concepts-descriptions and concepts-relationships into a concept-language (equivalent to a meta-language) to:

- modelise an existing object-oriented programming language (such as Java [AG 2000] or Eiffel [M 1992])
  - to use the language,
  - to adapt the language to a specific need (to make a more specialised language),
  - to avoid an inopportune feature (for a specific purpose) of the language (by disabling this feature),
  - to test a potential future evolution of the language (for instance, *multiple inheritance* or *genericity* in Java),

    - ...
- build a new object-oriented programming language (with all concepts-descriptions and concepts-relationships the programmer need)
    - to use this new language,
    - to make a very specific language,
    - to build a prototype of a future new language,
    - ...
- do meta-programming experiences...

OFL Concepts

Our goal is to avoid, as far as possible, long and fastidious meta-programming work. So, we have determined a set of *parameters* for each concept (concept-language, concept-description, and concept-relationship). Each parameter describes a *piece of operational semantics*. A system of *actions* (routines which represent operational semantics) is provided. Each action take in account the value of the parameters do realize its task.

For example, to define a relationship like *inheritance* (and thus, to influence execution of the action `lookup` which determines dynamic link), we have to set the value of some parameters such as:

- `Cardinality`: `1-1` (single inheritance),
- `Circularity`: `false` (no cycle in inheritance graph),
- `Polymorphism_direction`: `up` (inheritance is almost equivalent to specialisation),
- ...

And, to define a description like *class*, we must assign some others parameters such as:

- `Generator`: `true` (we can define constructors),
- `Overloading`: `true` (like in `Java`, unlike in `Eiffel`),
- `Attribute`: `allowed` (attributes are allowed),
- ...

OFL Tools

Several software products, in the process of being implemented, provide graphic

assistance to OFL meta-programmers and programmers.

The first tool called OFL-Meta is for the meta-programmer. It allows to graphically create, modify or delete the instances of the concepts. In other words, it allows to describe the operational semantics of a language which will be used when designing an application. These components which it handles can be stored by using various standard formalism such XML [W3C 2000, CCCL 2000b] or MOF [OMG 2000a].

The second tool is called OFL-ML in reference to UML [OMG 2000b]. It is intended for the application designer. It is also a graphic tool which allows to design the structure of the application (classes A and B, inheritance between classes A and B, ...), *i. e.* instances of the entities described in OFL-Meta. The programming task is included by a binding: the selected language is Java, that is body of all the methods is written in Java.

The third tool is called OFL-Parser. It is a translator, interpreter or compiler. Its task is to translate the structure of the application and the body of methods into a target language, Java in our case. The last step consist in executing the generated application and thus to use the methods and create the final data.

Conclusion

To sum up, we can say that OFL is a model which allows to describe object-oriented programming languages in order to adapt them to the programmer's need. The meta-programmer's work is mainly to define concepts-descriptions and concepts-relationships into concepts-languages by giving a value to their set of parameters.

References

All author's papers about OFL are available here.

[AG 2000] K. Arnold and J. Gosling. *"The Java Programming Language"* The Java Series... from the Source, Sun Microsystems, 3rd Edition, 2000. Web Site of this book

P. Crescenzo *"OFL : les relations et descriptions d'Eiffel et de*

[C 2001a] *Java"* Research Report I3S/RR--2001-06--FR, *Informatique, Signaux et Systèmes de Sophia Antipolis* (I3S) Laboratory, Sophia Antipolis, France, April 2001 Web Site of I3S and PostScript French version of the paper

[C 2001b] P. Crescenzo *"Le modèle OFL pour paramétrer la sémantique opérationnelle des langages à objets : Application aux relations inter-classes"* PhD. Thesis in Computer Science, University of Nice-Sophia Antipolis, Soon Published. Web site of the author

[CCCL 2000a] A. Capouillez, R. Chignoli, P. Crescenzo and P. Lahire *"Modeling Hypergeneric Relationships between Types in XML"* In Conference WOON'2000 (The White Object Oriented Nights), Saint-Petersburg, Russia, June 2000. Web site of WOON'2000 and PostScript version of the paper

[CCCL 2000b] A. Capouillez, R. Chignoli, P. Crescenzo and P. Lahire *"Modeling Hypergeneric Relationships between Types in XML"* In Symposium ETc 2000 (Electronics and Telecommunications), Timisoara, Romania, November 2000. Web site of ETc 2000 and PostScript version of the paper

[CCCL 2001] A. Capouillez, R. Chignoli, P. Crescenzo and P. Lahire *"Hyper-généricité pour les langages à objets : le modèle OFL"* In Conference LMO 2001 (Langages et Modèles à Objets), Le Croisic, France, January 2001. Web site of LMO 2001 and PostScript French version of the paper

[M 1992] B. Meyer *"Eiffel: The Language"* Object-Oriented Series, Prentice Hall, 1992. Web site of this book

[OMG 2000a] Object Management Group *"Meta Object Facility (MOF) Specification"* Version 1.3, March 2000. Web site of this document

[OMG 2000b] Object Management Group *"Unified Modeling Language Specification"* Version 1.3, March 2000. Web site of this document

[W3C 2000] World Wide Web Consortium *"Extensible Markup Language (XML)"* Version 1.0, W3C Recommendation, October 2000. Web site of this document