A study of Object Evolution

Mourad Oussalah, Dalila Tamzalit

IRIN, 2, rue de la Houssinière, BP 92208 44322 Nantes cedex 03, FRANCE
Mourad.Oussalah@irin.univ-nantes.fr Dalila.Tamzalit@irin.univ-nantes.fr

Abstract. we propose a repository to characterize OO evolution problematic. The two main objectives are to characterize object evolution according to its own features, and to uniformly analyze and compare evolution strategies according to this proposed repository. For that, OO evolution is defined according to its three fundamental features that we call *facets*: the subject of evolution (the structure or the behavior of objects), the type of evolution (whether it is foreseeable or not) and the mechanism used to deal with object evolution (from class or instance toward classes or instances). We restrict and develop our study to the structure of an object. We propose a taxonomy on object structure (the *node* and the *arc*) and a taxonomy on evolution operations. We distinguish unary and binary operations that apply on the above defined concepts. We then analyze and position some evolution strategies according within this repository and according to those two taxonomies.

1 The three facets of Object Evolution

The three facets that we consider as the fundamental features of OO evolution are the *subject*, the *type* and the *mechanism of evolution*:

1. Subject of Evolution: the *structure* and the *behavior* of an object can evolve. They are the subject of evolution:
– The *structure* is all of the object attributes (simples attributes and semantic links like inheritance, association and composition).
– The *behavior* is all of its methods used during every execution of the system.

2. Mechanism of Evolution: evolution of a subject can be activated on a class or on an instance levels. We distinguish the *development* and the *emergence* mechanisms:
– The *development*: represents a class evolution with impacts on other classes and instances, like versioning [5,10], classification [3]…
– The *emergence*: represents an instance evolution with impacts on classes. It is each instance evolution provoking the evolution of class specifications. Few evolution strategies, at our known, propose an emergence mechanism [12,18].

3. Type of Evolution: a subject evolves according to two types of evolution:
– The *Preventive evolution*: when evolution is predictable, it can be achieved since changes are identified and integrated during the analysis and the design phases. So, when such changes appear, the model knows how to achieve its own evolution. The preventive evolution is said *with break* when the relation between the model statuses before and after evolution is not established (as for reorganization [4],

categorization [12] or conversion [13]). The preventive evolution is said *seamless* when the relation between the model statuses before and after evolution is known (as for versioning [10], role [14], or instance versioning [5]).
– The *Curative evolution*: when evolution can be achieved if unpredictable changes occur, like for categorization [12] and emergence [18].

To evolve, any evolution subject S_E needs a *mechanism of evolution* M_E according to a *type of evolution* T_E: $S_E \mapsto (M_E, T_E)$.

1.1 The structure of evolution

We restrict the subject facet to the structure. We propose a taxonomy of structures and a taxonomy of operations applied on this taxonomy of structures.

1. Taxonomy of structures: a structural hierarchy of objects is a graph composed of *nodes* and *arcs*. We consider that nodes and arcs have the same first-prevalence:

- **Node:** is an entity representing a structural information within a class hierarchy and which can be called to evolve. We consider the class and the instance nodes.
- **Arc:** is a link between two nodes. Its semantic depends on the two linked nodes. An arc can be from a class to another class [17], from a class to an instance [2,13], from an instance to another [10] or from an instance to a class [18].

2. Taxonomies of evolution operations: we consider unary and binary operations:

- **Unary operation:** *addition*, *modification* and *deletion* are operations applied on a single structure (node or arc). Each operation has a semantic according to the structure on which it is applied. For example, the addition of a class implies the addition of its attributes and methods, while the addition of an instance implies its creation according to its class. An unary operation has a given range depending on the concerned node. For example, the deletion of a class as an internal node or as a leaf node within a hierarchy has different ranges. The first one concerns the subclasses. The second one concerns the deletion of a class and its instances.
- **Binary operation:** is any evolution operation applied on two structures: from the *source* structure toward the *target* one. We identify the following binary operations: *Transfer*, *Split*, *Fusion* and *Emergence*:
  - *Transfer* moves a structure, like a class, within an hierarchy or to another one. It copy the source structure toward the target one, before deleting the source.
  - *Split* divides a structure in two new structures, like a class in two classes.
  - *Fusion* groups together two structures in one structure.
  - *Emergence* outlines a structure from the dynamic instance evolution.

1.2 The mechanism of evolution

The studied structure-evolution strategies are presented according to the evolution mechanism:

1. **Development:** strategies acting on the class-nodes with possible impacts on other class-nodes and on instance-nodes lean on a development mechanism. So it can be:
   - a *class-to-class:* from a class-node with impacts on other class-nodes. Most important strategies are: Class extension [17], Reorganization [4], Correction [2,13], Class versioning [10], Class classification [3], Characteristics migration [8,9].
   - a *class-to-instances*: from a class-node with impacts on instance-nodes, like Instance conversion [2,13], Instance emulation and Instance versioning [5].
   - an *instance-to-instance*: from an instance-node to other ones: Integrity constraints [11], Instance versions, Instance classification [16].
2. **Emergence:** strategies acting on the instance-nodes with impacts on class-nodes: *instance-to-class* (emergence [18]) and *class-to-class* (categorization [12]).

2 How to place an evolution strategy within the repository?

We propose a method to place any evolution strategy according to the triplet $S_E \mapsto (M_E, T_E)$. For each strategy, the following questions must be answered:

1. On what *subject of evolution* does it act: the *structure* or the *behavior*?
2. What *mechanism of evolution* does it ensure: *Development* or *emergence*?
3. What *type of evolution* does it propose: *Curative* evolution or *preventive* one?

Strategies	**Subject of evolution**			**Mechanism of evolution**		**Type of evolution**	
	structure taxonomy	operation taxonomy		development	emergence	preventive	curative
		Unary	binary				
Extension	Inheritance arc	addition	-	class-to-class	-	continue	-
Reorganization	Node Inheritance arc	addition, modification, deletion	-	class-to-class	class-to-class	break	-
Correction	Node Inheritance arc	addition, modification, deletion	-	class-to-class	-	break	-
Versioning	Node Class-class arc	addition, modification, deletion	fusion	class-to-class	-	break	-
Classification	Node Class- class arc	addition, modification, deletion	-	class-to-class	class-to-class	break	-
Characteristic migration	Node Class -class arc	-	fusion, split transfer	class-to-class	class-to-class	break	-
Conversion	Node	modification	-	class-to-instance	-	continue	-
Emulation	Node	modification	-	class-to-instance	-	continue	-
Versioning	Node Class-instance arc	addition, modification, deletion	fusion	class-to-instance	-	break	-
Integrity constraints	Node	modification	-	instance -to-instance	-	continue	-
Versions	Node	modification	fusion	instance -to-instance	-	break	-
Classification	Node Instance-inst arc	addition, modification, deletion	transfer	instance -to-instance	-	break	-
Emergence	Node Instance-class arc	addition, modification, deletion	emergence	-	instance -to-class	-	Curative
Categorization	Node Class-class arc	addition, modification, deletion	emergence	class-to-class	class-to-class	break	-

Table 1. evolution strategies according to their subject, mechanism and type of evolution

This method is applied on some strategies. They are placed within the proposed repository as summarized in Table1. We note that the OO structure-evolution problem is mainly treated under a development mechanism within a preventive evolution. The evolution problem was less treated *curatively* within the emergence mechanism.

3 Conclusions

To answer complex set of needs of object-evolution problems [1,2,6,7…], we have been convinced by a necessary analysis work that must be done upstream of any punctual evolution need. We have thus defined a classification of object evolution based on its own *facets*: the *subject*, the *type* and the *mechanism of evolution*. Classifying object evolution according to its facets offers a common referential, independent from evolution strategies. We focused on structure evolution. We proposed a taxonomy of structure and a taxonomy of evolution operations. We have placed some of the main existing evolution strategies within the repository. As far as we know, there is no work analyzing OO evolution according to its entire and fundamental features. Some works, like [15], propose an interesting evolution analysis, but concerning just a specific domain. We believe that our approach may widely contribute in the analysis and the design of evolution and to control its maintenance. We still persuaded that this analysis of OO evolution will not only allow to find out new research areas but also to guide them.

References

[1]Alhajj R., Polat F. "Rule-Based schema evolution in OODB", Knowledge Based Systems, Pp. 47-57, Vol. 16, Elsevier Science, 2003.

[2] Banerjee J., Kim W., Kim H., Korth H.F. "Semantics Implementation of Schema Evolution in Object-Oriented Databases" ACM 1987.

[3] Capponi C."Interactive class classification using types" E.Diday& eds., New approaches in classification and data analysis, Springer-Verlag, Berlin, pp204-211, 94

[4] Casais E. "Managing Evolution in Object Oriented Environments : An Algorithmic Approach" phd Thesis – Geneva university. 1991.

[5] Clamen S.M. "Schema Evolution and Integration " in Proceedings of the Distributed and Parallel Databases conference, vol 2, p 101-126. Kluwer Academic Publishers, Boston, 94.

[6] Claypool K.T., Rundenstein E.A., Heineman G.T. "ROVER: flexible yet consistent evolution of relationships" Data&Knowledge Engineering, Pp27-50, Vol. 39, Elsevier 2001.

[7] Coulondre S., Tibourel T. "An integrated object-role oriented database model", Data & Knowledge Engineering, Pp 113-141, Vol. 42, Elsevier 2002.

[8] Jacobson I., Lindtröm F. "Re-engineering of Old Systems to an Object-Oriented Architecture" Conference proceedings OOPSLA 1991.

[9] Jiang H., Castellani X. "Migration of Characteristics of Classes, a New Concept to Specify an Object System Evolution" 7th International Conf. IRMA, May 1996, Washington, USA.

[10] Kim W., Chou H.T. "Versions of schema for OODB" 14VLDB, California, 1988.

[11] Meyer B. "OO Software Construction" I. Series in Computer Science. Prenctice Hall, 88.

[12] Napoli A. "Représentation à objets raisonnement par classification en IA" Phd Nancy 92.

[13] Penney D.J., Stein J. "Class modification in the GemStone OODBMS" OOPSLA'87 Vl22,

[14] Pernici B. "Objects with Roles" IEEE Conf. on Office IS, vol16, n3, pp 417-438, sept91.

[15] Rashid A., Sawyer P. "Towards Database Evolution- a taxonomy for OODB" Technical report CSEG/5/2000, Cooperative Systems Engineering Group

[16]Rechenman F., Uvietta P. "SHIRKA : an object-centered KBMS " In ED. Artificial Intelligence in numerical and symbolic simulation, ALEAS, France, pp 9-23.

[17] Sun Microsystems, Programmation Java LJ20 Solaris 2.x, Windows NT, january 1998.

[18] Tamzalit D,Oussalah C"How to Manage Evolution in OODB?" ADBIS-DASFAA00, Prague