From Object Evolution to Object Emergence

Dalila Tamzalit
LGI2P / Site EERIE - EMA
Parc Scientifique Georges Besse
30 035 Nîmes Cedex1 - France
(33) 4 66 38 70 39
Dalila.Tamzalit@site-eerie.ema.fr

Chabane Oussalah
Université de Nantes
IRIN, 2, rue de la Houssinière BP 92208
44 322 Nantes Cedex 03 - France
(33) 2 51 12 58 47
Mourad.Oussalah@irin.univ-nantes.fr

ABSTRACT

Database applications which model aspects of the real world should be able to express as accurately as possible the different nuances of reality; that includes the need to evolve internally in response to signals of updates coming from the environment. These updates are not always supplied in an ideal and complete manner and are not always predefined or precisely defined. In practice, requirements for evolution generally occur during the manipulation of objects while running the database. It is frequently necessary to change individual objects, less frequently the database schema. Database systems need to have mechanisms capable, whenever and as well as possible, of assimilating this new information correctly and diagnosing and implementing the changes necessary.

This paper concerns the evolution of objects inside databases. Our two main objectives are:

- to allow objects to evolve their structures dynamically during database maintenance and use, with all necessary impacts on the database schema;
- to allow, similarly, the creation and display of different plans for evolving the design, like ways of schema evolution, giving in this way a simulation tool for database design and maintenance.

So, we propose a Genetic Evolution Object Model developed to have inherent capabilities for auto-adaptation between classes and instances.

Keywords

design evolution — object-PTYPE — class-GTYPE — development and emergence processes — crossing-over.

1. INTRODUCTION

Nowadays, database systems and applications are increasing in robustness but also in complexity, bringing them up against crucial problems of evolution, adaptation and maintenance.

With its concepts of inheritance, composition, abstraction, polymorphism..., Object Orientation is often seen as a solution allowing the extension and adaptation of database systems by reuse of as much as possible of existing specifications. OODB methods contribute to the development of applications that facilitate modification by design and implementation but these still encounter evolutionary and auto-adaptation problems - even for existing specifications. Class based-languages force objects to conform to predefined class specifications, but, in the context of database specification and evolution, conformity often becomes constraint, exhibiting shortcomings in flexibility and power. In addition, most evolutionary strategies incorporate only the concept of evolution, not analysis and design of the evolutionary process.

The aim that we set ourselves is an evolutionary model able to handle both unforeseen and inaccurately anticipated needs. This will allow an object to evolve autonomously using internal and external information and to permit the creation of new abstractions. With this in mind, we propose to extend instance evolution from simple value modifications to structural modifications (addition and deletion of attributes). We will study for this the evolutionary processes, which allow objects to adapt by themselves when change takes place.

1.1 Object Evolution Vs Artificial Evolution

Operations of addition, deletion or modification of data or functionalities in an OODB or application lead automatically to evolution. Thus, changes in a class hierarchy or class definition must be propagated to instances and subclasses involved. Many strategies have been developed to manage impacts. We have studied the most important ones in three categories[17]: *class evolution* ([3], [11], [1], [13], [10], [7], [18], [2]...); *impacts of class evolution on instances* ([1], [9], [15], [4]...) and *instance evolution* ([10], [14]...).

The main conclusion of this comparative study is that the principal gap in existing evolutionary models is their incapacity to cope with unidentified or poorly defined needs and incomplete data. Moreover, instance evolution is always limited by class hierarchy - a rigid and unnatural aspect of their evolution.

1.2 Artificial Evolution

We have brought together under the title of Artificial Evolution all research work concerned with the definition and implementation of evolutionary and adaptive artificial systems. Artificial Life and Genetic Algorithms fall under this head [8] and constitute a basis for this study:

1. **Artificial Life:** its principal objective is simulation and synthesis of biological phenomena [5]. It attempt to generalize the principles underlying biological phenomena and to recreate them. It borrowed the concepts of GTYPE (genetic information of a system) and PTYPE (representative individuals of a system) respectively from the genotype and the phenotype of biology. GTYPE and PTYPE are interacting together unceasingly, enriching themselves through the processes of *development* (of GTYPE to create new individuals) and *emergence* (of new individuals properties to be inserted into GTYPE).
2. **Genetic Algorithms:** are particularly adapted to searching for better solutions to a given problem, iteratively, evolving "blindly" by reproducing and then perpetuating best genes through new individuals [6]. Genetic mechanisms are used: random *selection* of adapted individuals (implies a quantitative measurement of this adaptation); *crossing-over* of their genetic code in order to recover best genes; *mutation* to mutate a gene favorably.

1.3 Object Evolution and Artificial Evolution

Taking into account the role of classes and instances in the makeup of a real or artificial system, we liken class to genotype and instance to phenotype. We propose to present the general evolution of an object model as a retroactive and iterative loop (Figure 1). For our part, we consider that object evolution presents an insufficiency in the evolutionary process - namely in the emergence of new properties from instance evolution.

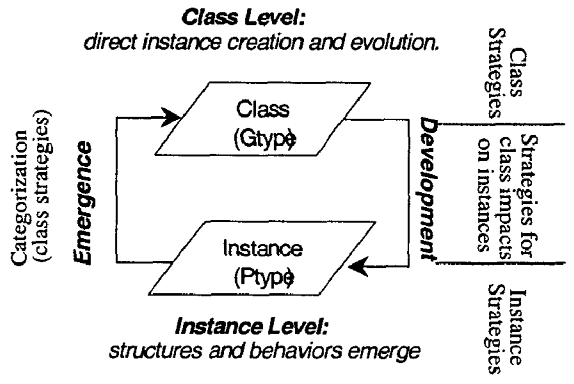
Class Level:
direct instance creation and evolution.
Class (Gtype)
Instance (Ptype)
Instance Level:
structures and behaviors emerge
Categorization (class strategies)
Emergence
Development
Class Strategies
Strategies for class impacts on instances
Instance Strategies

Figure 1: Object evolution processes under Artificial Evolution viewpoint.

2. A GENETIC EVOLUTION OBJECT MODEL

We propose to adapt artificial evolution concepts to those of object evolution through the GENOME model[17]. In order to illustrate the concepts and evolutionary processes, we use the example of Figure 2:

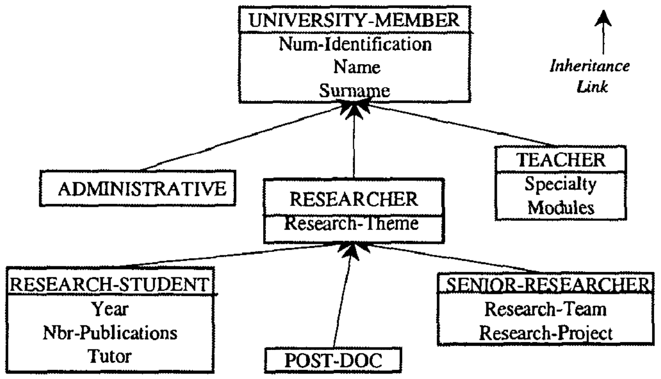

Figure 2: Members of a university described at the class level.

2.1 Concepts

2.1.1 Basic concepts: population, Instance-PTYPE and Class-GTYPE

- *Population and Genetic patrimony:* a group of classes representing various abstractions of one and the same entity forms a *population* (like the population of members of a university). All the attributes constitute its *Genetic Patrimony.*
- *Instance-PTYPE:* instances are the phenotype and represent entities called upon to evolve.
- *Class-GTYPE:* classes define instances features, their genetic code.

2.1.2 Advanced concepts: Fundamental, Inherited and Specific Genotypes

In a class, not every gene plays the same role or has the same prevalence. We consider that any class is entirely specified through three types of genotypes:

- *Fundamental Genotype or FG:* any object presents fundamental features, represented by particular genes representing the minimal semantics inherent to all classes of a same population.
- *Inherited Genotype or IG:* properties inherited by a class from its super-class constitute the *Inherited Genotype.*
- *Specific Genotype or SG:* it consists of properties locally defined within a class, specific to it.

Inherited genotype and specific genotype are issued from the environment and the context specificities where the "species" lives whereas fundamental genotype corresponds to the transmission of characteristics specific to the whole species.

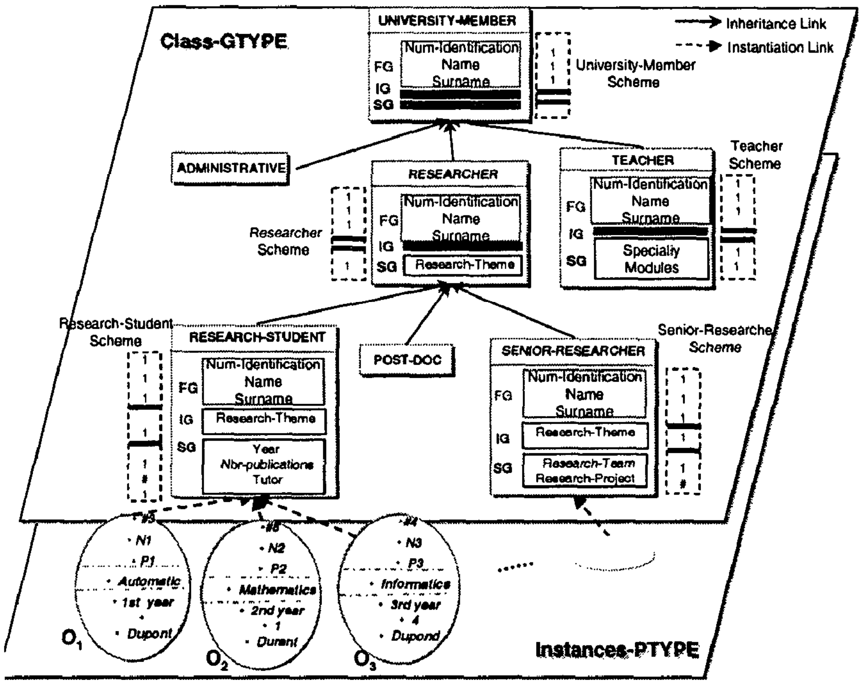

Figure 3: Classes and instances model in GENOME.

2.1.3 *Concept of scheme*

The concept of scheme is borrowed from genetic algorithms. It is an entity having the same genetic structure as the represented population. Each feature is represented by 0, 1 or #: 0 for absence of the gene, 1 for its presence and # for its indifference. The scheme is a simple and powerful means to model groups of individuals. We consider two kinds of schemes:

- *Permanent scheme:* with each specified class is associated a permanent scheme. It also contains three parts: fundamental, inherited and specific genotypes parts.
- *Temporary scheme:* it is a selection unit of one or a group of entities (instances or classes). It is especially useful during an instance evolutionary process and is used like a filter allowing selection of adapted entities.

2.2 Evolutionary processes

An evolutionary process is triggered when a change, envisaged or not, appears in the model. The process must be able to detect this change, find the entity implicated in the evolution and reflect this change adequately. Let us recall here that we place ourselves within the framework of the instance evolutionary process:

2.2.1 *Phases*

We consider that an instance's evolutionary process is carried out in three phases: an *extraction* phase, an *exploration* phase, and finally an *exploitation* phase:

- *Extraction Phase:* it consists of the detection of an instance evolution and the extraction of the object's genetic code within a temporary scheme.
- *Exploration phase :* it explores all the model's classes to locate adapted, even partially, classes. In order to avoid fruitless searches, this exploration follows precise steps: first it selects the set of populations concerned, then it carries out the search in that set.
- *Exploitation phase:* it manage the various impacts by way of development or emergence:
  - *Development process:* represents the impact of class evolution on instances.
  - *Emergence processes:* concern any emergence of new conceptual information, by way of impacts on classes. There are two possible outcomes:
    - *Local emergence:* relating to the emergence of new information within existing class(es). The genetic code of the object has mutated and this can force mutation[1] in its class or a semantically close relation.
    - *Global emergence:* related to the emergence of a new conceptual (class) entity either by direct projection from the object doing the evolving or by crossing-over existing entities.

2.2.2 *Genetic object operators*

It is necessary to define basic operators to handle instances and classes. The two most important are those of selection and crossing-over:

- *Selection:* is defined to determine, after structural evolution of an instance, which class holds part or all of its specification.
- *Crossing-over:* works on two entities via their genetic code (scheme) to allow them to interchange their genes in order to define a new group of genes. Crossing-over works indifferently

[1] We consider a mutation as every appearance of a new information inside a model, and that will provoke changes in the genetic patrimony.

on classes and instances. It constitutes the core of the emergence process.

- *Adaptation value Av:* allows calculation of the semantic distance between the evolved object and semantically close classes, thanks to a function having as input parameters two schemes of the same length. The first represents the entity having evolved and the second represents a closely related class. Denoting the evolved object's scheme by Sch_{obj} and the close class's scheme by Sch_{param}, the adaptive function is defined, using the operator $\wedge$ (and_logic), as:

$$Av\,(Sch_{param}) = \sum_{(i\,=\,1 \rightarrow\, n)} \{\, Sch_{obj}[i] \wedge Sch_{param}[i] \,\} \,/\, n$$

Where n is number of genes specified in the evolved object; i is the variable index from 1 to n, defining, at each stage, the position of two respective genes of the analyzed schemes. Two schemes are initially compared in their FG. The other genes are then compared.

In the following section, we present the architecture of GENOME as well as the operation of its evolution processes:

2.3 Operation of the model

Extraction, exploration and exploitation phases follow one another. Each phase is unrolled on an example.

2.3.1 Extraction Phase

It extracts the scheme of the evolved instances. New introduced attributes, which are reified, are created with a transitory status (their value is preceded in by a ~). Then all the attribute references are joined together to create the temporary scheme.

- Example: the three instances of `Research-Student` (example in Figure 3):

	Genetic Patrimony	Instances of `Research-Student`		
		O1	O2	O3
FG	*Identification-Num*	#3	#8	#4
FG	*Name*	N1	N2	N3
FG	*Surname*	P1	P2	P3
	Research-theme	Automatic	Mathematics	Computing
	Year	1st Year	2nd Year	3rd Year
	Nbr-Publications		1	4
	Tutor	Dupont	Durant	Dupond
	Research-Team			
	Research-Project			
	Specialty			
	Modules			

These instances will evolve to become (Attribute is a deleted attribute):

	Genetic Patrimony	O1	O2	O3
FG	*Identification-Num*	#3	#8	#4
FG	*Name*	N1	N2	N3
FG	*Surname*	P1	P2	P3
	Research-theme	Automatics	Mathematics	Objet
	Year	1st Year	2nd Year	3rd Year
	Nbr-Publications		1	4
	Tutor	Dupont	Durant	Dupond
	Research-Team	**Vision**		**Object**
	Research-Project			
	Specialty			
	Modules	**Segmentation**		

New Attributes

Rank	**Professor**		
Position		**Engineer**	
Responsibility			**Supervisor**

The new attributes *Rank*, *Position* and *Responsibility* take transitory status and temporary schemes (TS) are:

1. TSO_1=[11110001001~100]
2. TSO_2=[111000000000~10]
3. TSO_3=[1111000100000~1].

2.3.2 Exploration phase

It is carried out within the selected population. Note that the new attributes (marked transitory) are not concerned in the selection since they are not in the possession of any class.

- **O_1's Temporary Scheme:** An Av. is calculated for each class:

Entity	Scheme (IG and SG)								Av.
Objet O_1	1	0	0	0	1	0	0	1	
`University-member`	0	0	0	0	0	0	0	0	0/3
`Researcher`	1	0	0	0	0	0	0	0	1/3
`Research-Student`	1	1	#	1	0	0	0	0	1/3
`Senior-Researcher`	1	0	0	0	1	#	0	0	2/3
`Teacher`	0	0	0	0	0	0	1	1	1/3

With : ■ IG, and : □ SG

`University-member` (Av=0), `Researcher` and `Research-Student` are ignored, because they are in the same branch as `Senior-Researcher` which has the best Av. In contrast, `Senior-Researcher` and `Teacher` are partially adapted, and are thus selected.

- **O_2's Temporary Scheme:** Calculation of each class's Av.:

Entity	Scheme (IG and SG)								Av.
Objet O_2	0	0	0	0	0	0	0	0	
`University-member`	0	0	0	0	0	0	0	0	0
`Researcher`	1	0	0	0	0	0	0	0	0
`Research-Student`	1	1	#	1	0	0	0	0	0
`Senior-Researcher`	1	0	0	0	1	#	0	0	0
`Teacher`	0	0	0	0	0	0	1	1	0

No class is adapted. They are all ignored. Only the `University-member` class remains because it has the same FG as the object.

- **O_3's Temporary Scheme:** Calculation of each class's Av.:

Entity	Scheme (IG and SG)								Av.
Objet O_3	1	0	0	0	1	0	0	0	
University-member	0	0	0	0	0	0	0	0	0
Researcher	1	0	0	0	0	0	0	0	1/2
Research-Student	1	1	#	1	0	0	0	0	1/2
Senior-Researcher	1	0	0	0	1	#	0	0	1
Teacher	0	0	0	0	0	0	1	1	0

The Senior-Researcher class is completely adapted: it can contain the object in its new state, if it takes into account the new attribute.

2.3.3 *Exploitation Phase*

2.3.3.1 *Local Emergence processes*

O_3's Temporary Scheme: an existing class, Senior-Researcher, holds the genes of the object O_3 after an evolution step, which introduces the new attribute of *Responsibility*. This will have: to be inserted into the Senior-Researcher class by local emergence; to enrich the genetic patrimony and to lose its transitory status.

2.3.3.2 *Global Emergence processes*

The emergent abstraction can be defined by crossing-over or directly:

1. Crossing-over:

The permanent schemes of the selected classes constitute the starting population for the crossing-over, and fixe its length. The new schemes will replace their parents.

Two schemes are randomly selected to be crossed. The choice of crossing-over point is significant. It is based on the *uniform crossing-over* of the genetic algorithm [16]. It amounts to granting a weight relating to each parent for the transmission of genes to one of the children. This weight is calculated according to the Av. of each scheme (see example). After this, a random number is generated, then compared with the weight. If it is lower, the scheme 1 gene is transmitted to child 1 and the scheme 2 gene is transmitted to child 2. This operation of random bonding is repeated until all the genes have been reviewed.

We add to that a significant constraint which enables us to ensure coherence for the crossing-over operation: a permanent scheme presents at most two significant blocks (without considering the FG): IG and SG, which must be respected and transmitted. We thus impose a constraint of *blocks of genes*.

- **O_1's Temporary Scheme:** Senior-Researcher and Teacher schemes are selected:
  - *The first iteration:*

O_1	1	0	0	0	1	0	0	1	
Random bonds	*0.1*	*0.5*	*0.63*	*0.24*	*0.58*	*0.95*	*0.15*	*0.48*	*Av.*
Senior-Researcher	1	0	0	0	1	#	0	0	0.65
Teacher	0	0	0	0	0	0	1	1	0.35
Child Scheme1	1	0	0	0	*1*	*#*	0	0	2/3
Child Scheme2	0	0	0	0	0	#	**1**	**1**	1/3

Since the Child Scheme 1 (or 2) has more chance to inherit genes of the parent 1 (or 2), each child scheme inherits the block constraints of its predominant parent.

- *Second iteration:*

O_1	1	0	0	0	1	0	0	1	
Random bonds	*0.65*	*0.58*	*0.23*	*0.47*	*0.95*	*0.28*	*0.64*	*0.75*	*Av.*
Child Scheme1	1	0	0	0	*1*	*#*	0	0	0.65
Child Scheme2	0	0	0	0	0	#	**1**	**1**	0.35
Child Scheme3	1	0	0	0	*1*	*#*	0	1	1
Child Scheme4	0	0	0	0	0	#	**1**	**1**	1/3

Child Scheme3 could be the parent scheme of O_1: it is the *emergent scheme.*

- **O_2's Temporary Scheme:** the scheme belongs to the population but cannot be deduced from any existing class. So it represents a new abstraction.

2. Global Emergence processes:

It can provoke the emergence of a simple class or of a collection of related classes. Necessary attributes (existing and new) are specified within the new abstraction. The methods, which call upon these attributes, are integrated and proposed to the user for validation. The user will validate and complete the specifications of the abstraction. O_1's Temporary Scheme is thus a *simple emergent object.*

Now, the question is where this new abstraction must be inserted? There are two possibilities:

1. Within the hierarchy branch of Researcher, as a sub-class of Senior-Researcher, with a mutation on the *Module* gene defined by Teacher.
2. Within the hierarchy branch of Teacher, as a sub-class, by integrating together, by means of a mutation, *Research-Theme*, *Research-Team* and *Research-Project*.

The most judicious choice can be easily deduced from the Av.

- **O_2's Temporary Scheme:** The object O_2 defines a new abstraction, that of Engineer, which is created and attached to the root University-member population.

2.3.3.3 *Integration of the emergent entity and notion of semantic distance*

The insertion of a new abstraction must reflect the influence of its parents but also the semantics carried by the emergent entity. It is logical that an abstraction, which takes the greater part of its genes from another abstraction, has to be in the same hierarchy as this last. In order to infer such information, we have recourse to the Avs already calculated during the exploration phase. The parent having the strongest Av. will influence the final scheme the most. All the more so if the emergent scheme contains the IG of the parent scheme. Likewise, in a global emergence, the emergent abstraction will be attached to its branch. But in the case of Avs are around 50% (±10%), multiple inheritance is the solution. When the IG of each of the parent schemes is found in the emergent scheme is also another criterion.

That's why the corresponding emergent abstraction of the object O_1, in the example, will be attached to `Senior-Researcher`, and this for two main reasons:

1. the emergent scheme presents the same block of IG as `Senior-Researcher`.
2. the emergent scheme is closer to `Senior-Researcher` than to other classes. This proximity is evaluated by comparison of the two schemes, using the adaptive function. The obtained value is the *semantic distance* between the two schemes:

Child Scheme3	1	0	0	0	1	#	0	1	***Semantic distance***
`Senior-Researcher`	1	0	0	0	1	#	0	0	3/4
`Teacher`	0	0	0	0	0	0	1	1	1/4

Once the emergent abstraction is inserted, the object O_1 is attached to it and an evolution link is created from its initial class to its new class: *the model has learned a new behavior and a possible direction of evolution.* Having concluded these evolution processes, the model becomes:

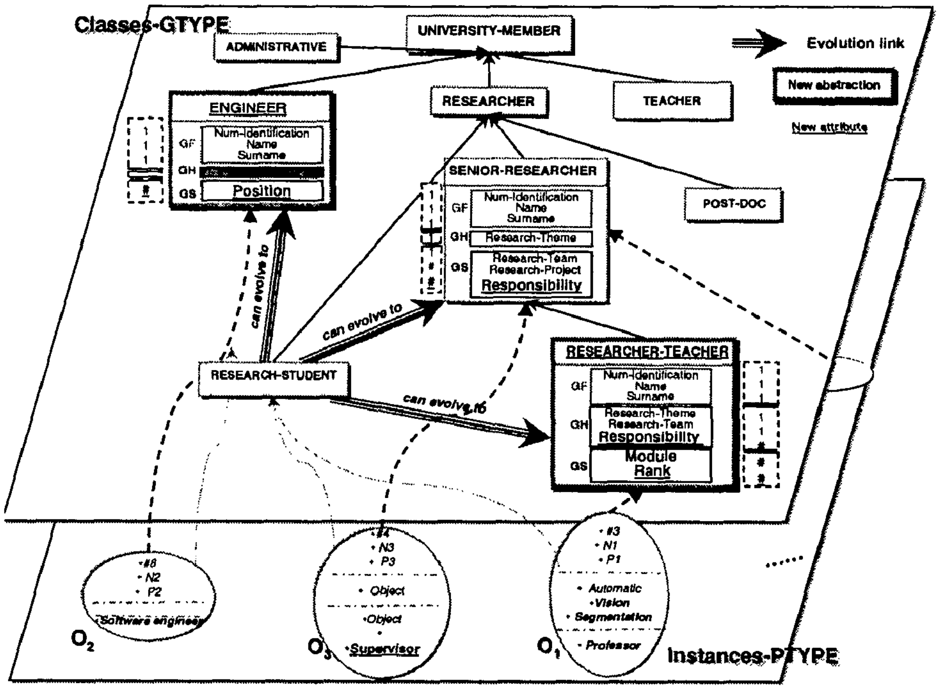

Figure 4: The example model after evolution in GENOME.

2.4 Synoptic chart of the model

To describe the principal concepts of GENOME, we borrow UML formalism. Concepts of class, inheritance, composition and associations will be used, as well as the concepts of roles and multiplicity.

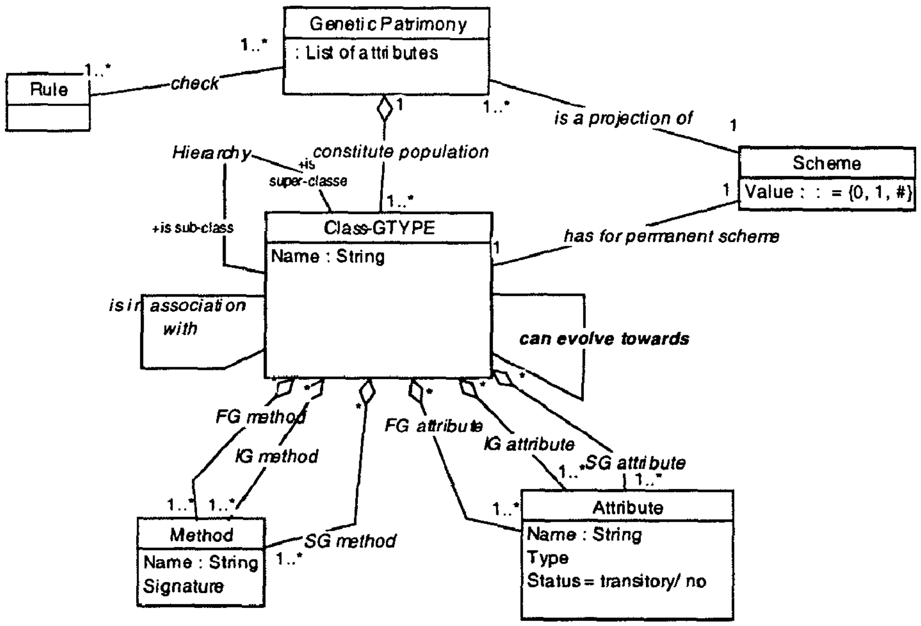

Figure 5 : Structure of GENOME's model using UML formalism.

3. CONCLUSION

The proposed and studied evolution processes are based on models organized around the inheritance link. We aim to extend GENOME to manage also complex objects. We have to study the emergence processes according to other structural links: composition and association links, implying like this, several populations.

Another future extension of the model is the evaluation of the quality of the emergence. In fact, many ways of evolution can be proposed and a choice must be done among them. We want to develop semantical metrics of emergence to make the best choice among several emergent abstractions.

It seems to us that this new way of considering object evolution is promising for analysis and design, and can provide new tools of design and simulation of complex systems.

4. REFERENCES

[1] BANERJEE J., KIM W., KIM H., KORTH H.F. "Semantics Implementation of Schema Evolution in Object-Oriented Databases" ACM 87.

[2] BRATSBERG E. "Evolution and Integration of Classes in Object-Oriented Databases" PhD thesis, Norwegian Institute of Technology, jun. 93.

[3] CASAIS E. "Managing Evolution in Object Oriented Environnements: An Algorithmic Approach" Thèse - Université de Genève. 91.

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

[5] HEUDIN J.C. "La Vie Artificielle ". Edition Hermes, 94.

[6] HOLLAND J. "Adaptation in Natural and Artificial Systems", University of Michigan Press, Ann Arbor, Mich., 75.

[7] KIM W., CHOU H.T. "Versions of schema for object oriented databases" In Proceedings of the 14th VLDB Conference, Los Angeles, Californie, 88.

[8] LANGTON C., TAYLOR C., FARMER J.D., RASMUSSEN S. "Artificial Life II" proceedings vol in the Santa Fe Institute studies in the sciences of complexity, New Mexico, February 90.

[9] LERNER B.S., HABERMANN A.N. "Beyond Schema Evolution to Database Reorganization" Proc. ACM Conf. OOPSLA and Proc. ECOOP, Ottawa, Canada. Published as ACM SIGPLAN Notices 25(10), pp. 67-76. October 90.

[10] MEYER B. "Object-Oriented Software Construction" International Series in Computer Science. Prenctice Hall, 88.

[11] NAPOLI A. "Représentation à objets et raisonnement par classification en I.A." Thèse de doctorat, Nancy 92.

[12] OUSSALAH C. & ALII "Ingénierie Objet : Concepts et techniques" InterEditions, 97.

[13] PENNEY D.J., STEIN J. "Class modification in the GemStone object-oriented DBMS" SIGPLAN Notices (Proc OOPSLA'87) Vol. 22, No. 12, pp. 111-117. 87.

[14] RECHENMAN F., UVIETTA P."SHIRKA : an object-centered knowledge bases management system" In A. Pavé and G. Vansteenkiste ED. AI in numerical and symbolic simulation, pp 9-23., ALEAS, Lyon, France 89.

[15] SKARRA A.H., ZDONIK S.B. "Type Evolution in an Object-Oriented Databases" in Research Directions in Object-Oriented Programming, MIT Press Series in Computer Systems, MIT Press, Cambridge, MA, 1987, pp. 393-415. 87.

[16] SYSWERDA G. "Uniform Crossover in Genetic Algorithms" dans les actes de Intl. Conf. On Genetic Algorithms, pages 2-9, 89.

[17] TAMZALIT D., OUSSALAH C., MAGNAN M. "How to Introduce Emergence in Object Evolution" OOIS'98, , pp: 293-310, 9-11 September 98, Paris.

[18] ZDONIK S.B. "Object-Oriented Type Evolution" in Advances in Database Programming Languages, François Bancilhon and Peter Buneman eds, pp.277-288, ACM Press, New York 90.