Journal of Theoretical and Applied Information Technology
15th April 2012. Vol. 38 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

# MODELING BIO INSPIRED SYSTEMS: TOWARD A NEW VIEW BASED ON MDA

[1]SEIF EDDINE MILI , [2]DJAMEL MESLATI
[1,2]Department of Computer Science, LRI Laboratory, University Of Annaba Algeria
E-mail: [1]seifmili@gmail.com, [2]meslati_djamel@yahoo.com

## ABSTRACT

Biologically inspired complex systems are increasing. As the abstractions presented by biologically inspired systems, systems architects will be required to include the abstractions in their architecture in order to communicate the design to system implementers. The paper argues that in order to correctly present the architectures of bio inspired system a need of bio inspired views will be required. The paper describes a new formalism based on biology and Model Driven Architecture (MDA) in order to find a new and easy way to design and understand (reverse engineering) a complex bio inspired system. The paper also describes then a set of bio inspired views which are used when describing bio inspired system. Finally we use the proposed approach to model Xor artificial neural network.

**Keywords:** *Bio Inspired System, Model Driven Architecture, Architectural Units, Design Pattern, Neural Network*.

## 1. INTRODUCTION

In a perfect world, a good engineer builds a perfect system, the customer is satisfied, and the maintainer of the system has little to do to keep the system up and running [1].

Artificial neural networks, multi-agent systems, artificial immune system, swarm intelligence, genetic algorithms, cellular system, artificial ontogeny, and cognitive intelligent have one thing in common. Each of them presents a biologically inspired computing paradigm. These abstractions of biological systems have provided much inspiration in the development of complex systems [2].

As Bio inspired system move from research laboratories into industry the need for methodologies to describe their design arises [3]. The architects of industrial software systems will be required to translate abstract concepts of biology into concrete models that can be used by system implementers. Many modelling and design techniques have been proposed [4,5].

The purpose of the paper is to present a new additional manner for describing bio inspired system based on MDA and biology.

The paper is organised as follows. Section 2 provides background to the topics discussed in the paper. In section 3 our framework description is presented. In section 4 an overview of bio inspired systems with our framework is presented in section 5 we present the transformation design in section 6 we present case of study upon xor representation in neural network. Finally section 7 looks at related and future work.

## 2. BACKGROUND

In the background section, the main concepts, terminology, and ideas presented in the paper are defined. The section first explores some concepts relating to the field of biology than looks at MDA.

### 2.1. Biology: ontogeny, phylogeny, epigeny

Living multi-cellular organisms are not created in the completely achieved form we usually know. The organism begins life as a single cell, endowed with a developmental program coded in its genome. The latter is continuously processed by the cell, which leads to its repeated division in a multitude of identical cells that have the same genome. Then, a form of communication appears between cells, allowing each one to execute the part of the genome corresponding to its position in the whole. In other words ontogeny investigates the "developmental model" of an individual organism from the earliest embryonic stage to maturity.
Phylogeny is the study of phylogenesis within given species; reproduction consists in transmitting the genome of one or two parents to offspring. The genome of the descendant first cell is obtained from that/those of the parents, through mutations and

Journal of Theoretical and Applied Information Technology
15th April 2012. Vol. 38 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

crossing over. Therefore, the living species evolves by the combination of genes within a population of individuals genetically compatible[6].
Epigenesis, relating to epigenetics, uses specific structures to store and handle a huge number of interactions with the environment. The epigenetic process is supported by three systems: the nervous system, the endocrine system and the immune system [7]. The structures used in these systems are easily alterable by the environment and allows the complex living organisms to learn and achieve symbolic processing of information.
The section has explored some of the biological processes involved in shaping and guiding the evolution and development of biological individuals.

## 2.2. Model Driven Architecture

One of the most important principles to cope with the complexity in software engineering is the separation of concerns principle. This principle states that a given problem involves different kinds of concerns, which should be identified and separated to cope with complexity, and to achieve the required engineering quality factors such as robustness, adaptability, maintainability, and reusability [8].
In this context, Model-Driven Architecture (MDA for more details see [9]) promotes the production of models with sufficient detail so that they can be used to generate or be transformed into executable software, running on target systems [10].
Key to MDA is the importance of models in the software development process. Within MDA the software development process is driven by the activity of modelling the business software system. The MDA development process does not look very different from a traditional lifecycle, containing the same phases (requirements, analysis, low level design, coding, testing, and deployment). One of the major differences to traditional development processes lies in the nature of the artefacts that are created during the development process. These artefacts are formal models, i.e. models that can be understood by computers and finally be transformed into a representation that lends itself to execution [11].

In summary MDA is a framework defined that separates the platform specific concerns from platform independent concerns, which is represented by different views of a system.

## 3. THE PROPOSED APPROACH

Our method is inspired from both: POE model (for more details see [7,12]) and MDA approach, the Architectural Units (AU) are the central points of the method. Our purpose is to define several Architectural Units to describe the ontogenetic, epigenetic and phylogenetic views.

The AU consists of a number n of input models and a transformation that produces the k output models. Transformations can have attributes and operators that are applied to produce the output models (see figure 1). Models as well as transformations can be of various types. The environment supplies diverse stimuli such as events that help in triggering or stopping the transformation [1].

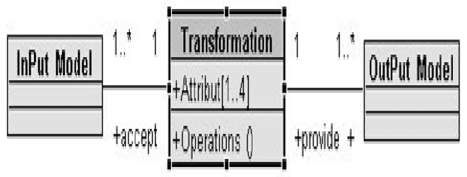


Figure 1. *This figure show the description of the architectural unit*

We can formulate the Architectural Unit like a function as show below:

Name_of_AU(IM1,IM2,...,IMn)→ OM1,OM2,...,OMk.

This form of description will be used below to describe the phelogenetic view of evolutionary systems.

There are three parts involved in all bio-inspired systems: the processes, the structures and the environment where the system is designed to operate [6]. Therefore, characterizing a system comes to characterize each part. The structure consists of all the models available in a system. We show in table 1, the derived criteria set.

TABLE I. *The model criteria set*

| |
|---|
| **Role**: A model can play two possible roles for each transformation where it is involved. The individual role or the species role. That is, a model can be involved simultaneously as a species in a process and as an individual in another. |
| **Description type**: A model can be a genome, a phenotype or any other description. Genome models are often coded using low level symbols such as a sequence of bits, while the phenotype is more abstract. Models can be implemented in hardware or stored in some memory. All models are interpretable. |

| |
|---|
| **Element/Set**: The model can be a single element or a set of elements. |
| **Granularity**: Characterizes the item available to transformations. Models range from fine grained to coarse grained. When we use phylogenesis to adjust a neural network, the grain is the weight attached to each connection. In other cases, the grain can be a symbol, a rule, an instruction or a function in a program. The finest grain is the bit. |
| **Alterability**: Defines how easy the model is alterable. Models can be highly alterable when they are stored in a soft memory. They are less alterable or reconfigurable when implemented in hardware. Furthermore alterability can be manual or fully/partially automated. |
| **Composition**: A model can be simple or composed. A composed model can be decomposed into sub-models and transformations. |

### 3.1. The Ontogenetic View

The Ontogenetic view is constructed using one AU: the development AU. Formally, the development unit can be written using the functional notation: Develop(D, M) → M' Which means that M' is obtained from M by a modification according to some description in D. M, D and M' are models.

*The Phelogenetic View*

The phylogenetic view process is constructed using two types of AU: the Reproduction AU and the Selection AU. The reproduction AU allows combination of input models using genetic operators (i.e. crossover and mutation) to produce output models. The transformation attributes include the mutation rates, the crossover type. Formally, the reproduction is written: Reproduce (RM,S) → S' Where RM is a model containing the description of the reproduction, S and S' are sets of models. Each element in S' is obtained (according to RM) from one or more elements of S using mutation and crossover operators. The abstraction levels of S and S' are the same.

The selection unit allows the selection of one or more models for the set of input models (i.e. output models are a subset of the input models). Models themselves are not altered. The transformation operators include the fitness functions and attributes, the selection threshold. Formally, the selection is written: Select(SM, S) → S' Where SM is a model containing the description of the selection, S' is a subset of S containing elements selected according to SM. The abstraction levels of S and S' are the same

### 3.2. The Epegenetic View

The epigenetic view is constructed using two AUs: the interpretation AU and the adjustment AU. The interpretation AU accepts executable models and data models as inputs and produces a data model as output. The adjustment unit adjusts one model according to another input model. The interpretation can be written: Interpret (P, I) → O.
O is obtained by transforming the I model according to some description in P. The abstraction levels of I and O are the same. However, compared to P, they may have greater or lesser abstraction level. The adjustment can be written: Adjust (M, P) →P'.

### 3.3. Characterization of Biological Process

In this section, we characterized the biological processes using the functional expressions:

**Ontogenesis**

Iterate (C, Assign(Ph, Develop(G,Ph)))

**Phylogenesis**

Iterate (C, Assign(S, Select(FM, Reproduce(RM,S))))

**Epigenesis**

( Assign(M, Null),
Iterate (C,( Iterate(SC, Assign(M, Develop(D, M))),
Assign(D, Adjust(Interpret(M,IDM), D ))
)))

From the previous, we remark that the three processes are similar since they all aim to deal with evolution, but in the same time there is some differences such as :

• The degree of alterability of the used models
• The abstraction levels of the used models
• The process cycle frequency
• The intervention of the environment on the processes

In figure 2 we summarize our vision of the relationship between the three biological processes.

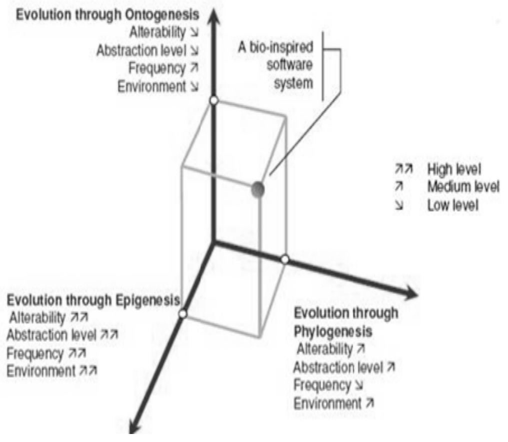


Figure 2. *Relationship between the biological processes*

Journal of Theoretical and Applied Information Technology
15th April 2012. Vol. 38 No.1
© 2005 - 2012 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

## 4. DESCRIPTION OF BIO INSPIRED SYSTEMS

Regarding to [13] there are different types of bio inspired system, in this paper we will discuss some types:

### 4.1. Artificial Neural Network

Artificial neural networks are computational models implemented that attempt to capture the behavioural and adaptive features of biological nervous systems. An artificial neural network is composed of several interconnected units, or neurons. Some of these units receive information directly from the environment (input units), some have a direct effect on the environment (output units), and others communicate only with units within the network (internal, or hidden, units)[13].
Each unit implements a simple operation that consists in becoming active if the total incoming signal is larger than its threshold. An active unit emits a signal that reaches all units to which it is connected. The connection, or synaptic point, operates like a filter that multiplies the signal by a signed weight, also known as synaptic strength. The behaviour aspect was given by the expression below:
Iterate(C,Assign(M, Adjust(AJ, Interpret(I,M,E), E)))
Where C: is the condition of end of learning.
E: the model that define the desirable solution that the network must deliver (supervised learning).
M: the model that define neural network.
AJ: is the model which gives to the link of network the adjustment for adapt the network according to the problem result, like retro propagation of error.
I: the model that define the convergence or divergence of the network
Below is the structural aspect of Neural Network
Role: Neural Network model play role of Individual.
Description type: the neural is described with array of one or more dimensions.
Element/Set: a Neural network system is a set of two classes' network and layer, which second one is composed of two other class which are neural and link.
Granularity: the granularity of phenotype is a set of slot in the array that represents the change of weight.
Alterability: the phenotype is alterable.
Composition: a neural network systems are a succession of two transformations the first is interpretation and second is adjustment.

We describe below the meta model of neural network (structural aspect) using UML 2.0.

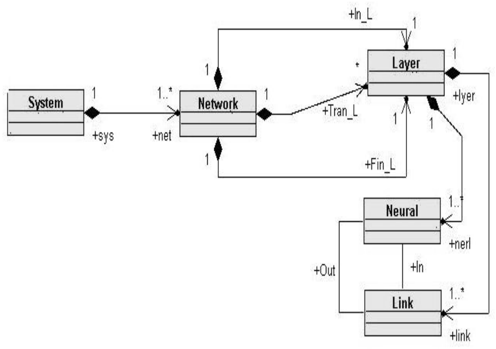


Figure 3. *A view of generic meta model of Artificial Neural Network*

System: class system define our system and it's the evolution of the network through the change incur in layer, neural and link class. We need to use singleton pattern to represent system because they are one instance of object system.
Network: define the features of the network like number of layer , the network evolve according neural and link changes (adjustment process).
Layer: define the characteristic of each layer of the network like number of neural, number of links, type of layer: input, output, hidden.
Neural: define the feature of neural like bias
Link: define the characteristic of link like weight
The Network class evolves from initial state to final state crossing transitional state according the adjust transformation and interpret transformation, the iteration transformation make refinement.

### 4.2. Negative Selection

The negative selection algorithm assumes that there is a collection P of fixed-length strings of symbols which must be protected from unauthorized change. For example, this collection could be the patterns of operation of a machine. In the absence of unauthorized changes P corresponds to a collection S which is called the self[13]. The goal of the algorithm is to generate a set of detectors that can signal the appearance in P of any string that does not belong to S, that is, the appearance in P of any nonself string. Nonself strings could be generated, for example, by the presence in the system of a virus or a network intrusion. The behaviour aspect was given by the expression below:
Iterate( C, assign (M, Select (SM, Interpret ( IM , M,E),E)))
C : the condition of end of transformation mechanism and it's until all detectors or antibody was compared.

M : define the model of detector that represent the population of antibody.
SM : define the selection model here the selection means the feature of good detectors.
IM : represent the interpretation model who compare the affinity between antibody and antigen.
E : the model that define the antigen.
Below is the structural aspect of Negative selection
Role: Negative selection model play role of Population.
Description type: the antibody can be described with several manner, in high level representation or in low level representation.
Element/Set: a negative selection system is a set of three classes' antibody, cell_population and antigen.
Granularity: the granularity of phenotype is variable due to the several manner of antibody description.
Alterability: the phenotype is alterable.
Composition: a negative selection systems are a succession of two transformations the first is interpretation and second is selection.

We describe below the meta model of negative selection (structural aspect) using UML 2.0.

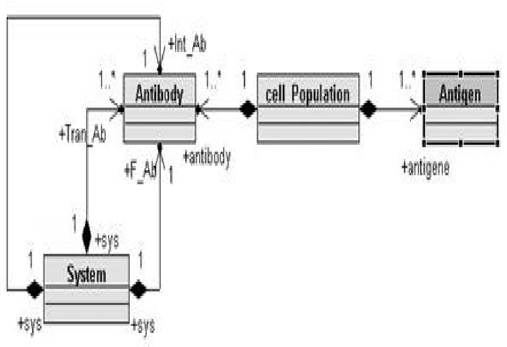


Figure 4. *A view of generic meta model of Negative Selection*

System: class system define our system and it's the filtering of population of antibody from initial state to final state. We need to use singleton pattern to represent system because they are one instance of object system.
Cell_Population: define the features of population like seize and individual, the population changes according antibody (selected one).
Antibody: define the characteristic of individual like representation, seize of individual…
Antigen: define the feature of the individual that represent the non self.

The Antibody class evolves from initial state to final state crossing transitional state according the selection transformation and interpret transformation, the iteration transformation make refinement.

### 4.3. Genetic Algorithm

Operate on binary representations of the individuals and emphasize the role of building blocks and crossover [13]. Genetic programming operates on tree-based representations of computer programs and circuits. Evolutionary programming often relies on tournament-based selection with gradual population replacement and does not use crossover [13]. The behaviour aspect was given by the expression below:
Iterate( C, assign (S, Select (SM, Reproduce ( OM , S))))
Where C : is the convergence condition as an example in optimisation the satisfaction of objective function
S: is the solution model that satisfies our problem
SM: is the selection model in this case we can have the proportionate selection model, generational replacement selection model, Truncated rank-based selection model or Tournament selection model
OM: is the reproduction model, and can be of two types, either by mutation or cross-over and both of them have their own mode, in this case it is more the cross over model that is used.
The expression function means that the system is reproduced according to the model OM this process with the selection one tune up the system with new population
Below is the structural aspect of genetic algorithm
Role: genetic algorithm model play role of Population.
Description type: the genome is describe in several manner as chromosome form in genetic algorithm, as tree in genetic programming..
Element/Set: a genetic algorithm system is a set of two classes' individual and population.
Granularity: the granularity of phenotype is a set of bit that represents words
Alterability: the phenotype is alterable and it's the main principal of genetic algorithm.
Composition: a genetic algorithm systems are a succession of two transformations the first is reproduction and second is selection.
We describe below the meta model of genetic algorithm using UML 2.0.

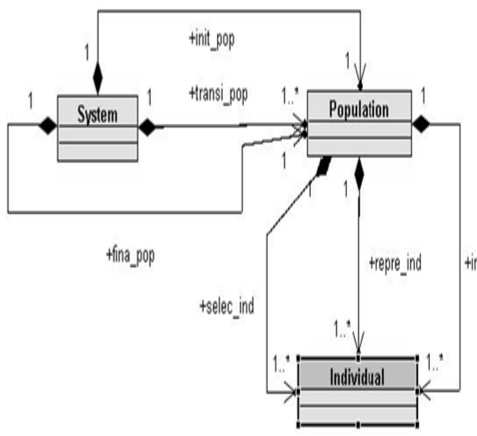


Figure 5. *A view of generic meta model of Genetic Algorithm*

System: class system define our system and it's the evolution of a population from initial state to final state. We need to use singleton pattern to represent system because they are one instance of object system.
Population: define the features of population like seize and individual, the population evolve according individual (selected one, reproduced one).
Individual: define the characteristic of individual like representation, seize of individual…[14].

### 4.4. Simulated Annealing

Is a function optimization procedure based on random perturbations of a candidate solution and a probabilistic decision to retain the mutated solution. Simulated annealing takes inspiration from the process of shaping hot metals into stable forms through a gradual cooling process whereby the material transits from a disordered, unstable, high-energy state to an ordered, stable, low-energy state. In simulated annealing, the material is a candidate solution (equivalent to the individual phenotype of an evolutionary algorithm) whose parameters are randomly initialized. The solution undergoes a mutation and, if its energy (equivalent to the inverse of the fitness) is lower than that at the previous stage, the mutated solution replaces the old one. The procedure stops when the annealing temperature approaches the zero value [13]. The behaviour aspect was given by the expression below:
Select (SM, Iterate( C, assign (S, Adjust ( OM , S,))))
C: is the condition of convergence, it is the temperature
S: is the solution model that satisfies our problem
SM: is the selection model, in this case we choose a random initial solution.
OM: is the adjustment model supporting only the mutation using mathematical methods like metropolis criterion.
Below is the structural aspect of simulated annealing
Role: simulated annealing model play role of individual.
Description type: the genome can be described in several manners as chromosome form, as tree form...
Element/Set: a simulated annealing system is an element that's representing by class named individual.
Granularity: the granularity of phenotype is a set of bit that represents words
Alterability: the phenotype is alterable and it's the main principal of simulated annealing
Composition: a simulated annealing systems are a succession of two transformations the first is selection and second is adjustment.
Figure 6 show the meta model of simulated annealing using UML 2.0 [14].

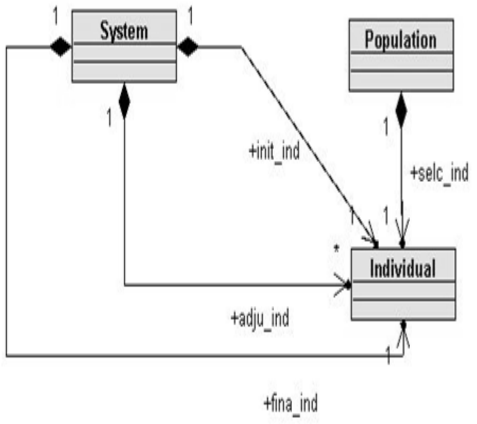


Figure 6. *A view of generic meta model of Simulated Annealing*

System: class system define our system and it's the evolution of a individual from initial state to final state. We need to use singleton pattern to represent system because they are one instance of object system.
Population: define the features of population like seize and individual, the population is used to select one individual for the evolution.
Individual: define the characteristic of individual like representation, seize of individual. The individual evolve according adjustment.

## 5. TRANSFORMATION DESIGN

We describe in this section some of the transformations implies in bio inspired system listed above.

### 5.1. Selection Transformation

Figure 7 shows the representation in UML 2.0 of the selected transformation, witch take population model like input and provide new population model like output. We use a strategy pattern to design selected transformation model

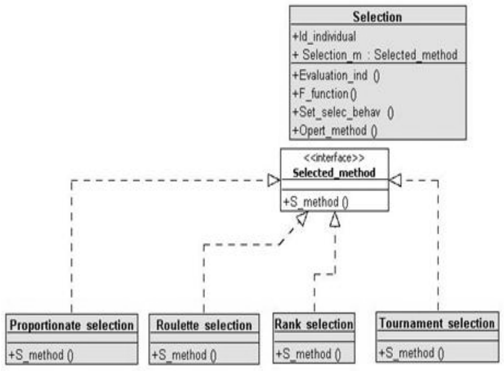


Figure 7. *A view of generic meta model of selection transformation*

A selection class has two attribute: Id_individual that identify the individual and Selection_m. the last one is used in Set_selec_behav method to describe the selection behaviour. Operat_method is used to encapsulate the behaviour of selection and use S_method for each behaviour type (roulette, rank…).S_method define an algorithm for each selection behaviour.

### 5.2. Adjustment Transformation

We present below the Adjust transformation

The figure 8 shows the representation in UML 2.0 of the adjustment transformation, witch take individual model like input and provide new individual model like output. We use a strategy pattern to design adjustment transformation model.

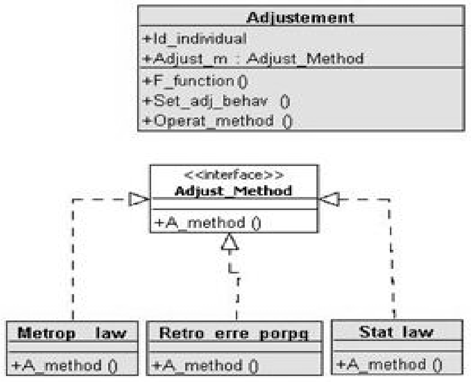


Figure 8. *A view of generic meta model of adjustment transformation*

An Adjustment class has two attribute: Id_individual that identify the individual and Adjust_m. the last one is used in Set_adj_behav method to describe the adjustment behaviour. Operat_method is used to encapsulate the behaviour of adjustment and use S_method for each behaviour type (metropolis, retropropagation of error…).S_method define an algorithm for each adjustment behaviour.

### 5.3. Reproduction Transformation

The figure 9 shows the representation in UML 2.0 of the reproduction transformation, which take individual model like input and provide new individual model like output. We use a strategy pattern to design reproduction transformation model.

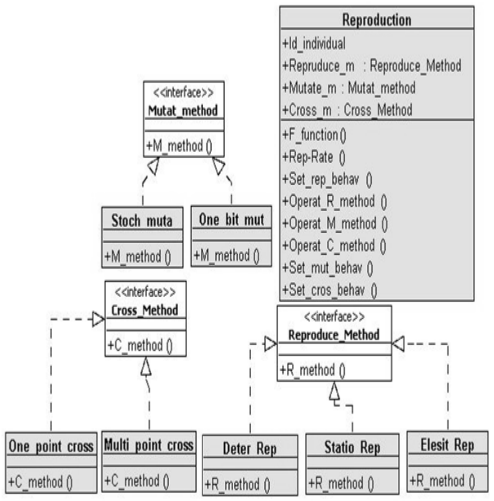


Figure 9. *A view of generic meta model of reproduction transformation*

A reproduction class has four attribute: Id_individual that identify the individual repruduce_m which is used in Set_rep_behav method to describe the reproduction behaviour, Mutate_m and Cross_m. are used respectively in Set_mut_behav and Set_cros_behav methods to describe the mutation and crossover behaviour. Operat_R_method, Operat_M_method, Operat_C_method are used respectively to encapsulate the behaviours of reproduction, mutation and crossover.

The population class evolves from initial state to final state crossing transitional state according the selection transformation and reproduced transformation, the iteration transformation make refinement.

## 6. CASE OF STUDY AND IMPLEMENTATION

To illustrate our formalism we choose to solve Xor problem using neural network, below we describe the evolution of Xor Artificial Neural Network using MDA. The Xor function is non-linearly separable, we can't separate out in class 0 of those in Class 1 by using a simple perceptron consists of one in layer and one out layer. For this, the system we describe consists of an input layer, a hidden layer and an output layer.

In this context, the learning phase is not taken into account, and the system is considered valid. We focus on the functioning of the system, the propagation of data from one layer to another. The figure below describe the simplified meta model of neural network and here relation among initial state of Xor system.

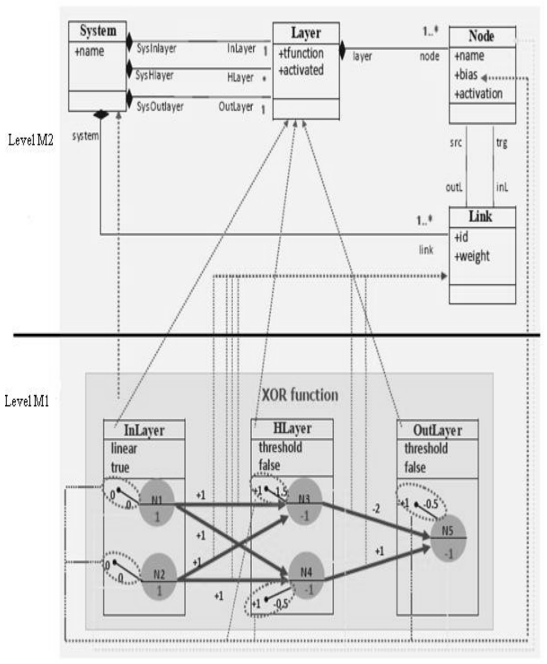


Figure 10. *view of relationship between initial state Xor system model and here meta model*

The model described above is a graphical representation of the Xor function system in its initial state (the system is found in its initial state when all layers are turned off except the input layer, and is found in the final state when all layers are turned on). We use the standard XMI [9] to expressed the system. Below some listing of XMI representation.

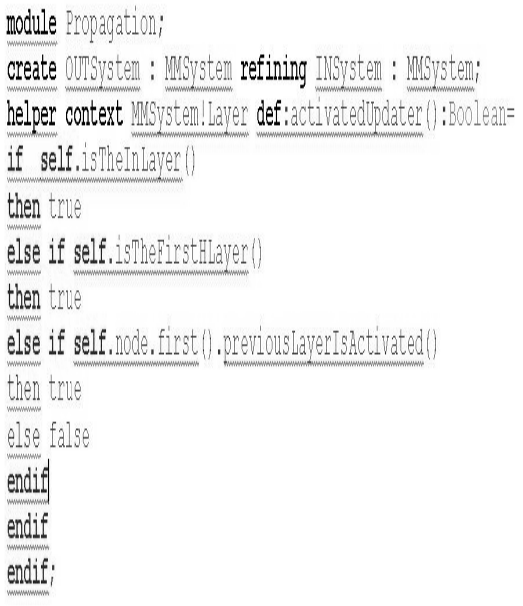
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0"
  xmlns:xmi="http://www.omg.org/XMI"
xmlns="MMSystem">
<System name="XOR function">
  <InLayer tfunction="linear" activated="true" >
    <node name="N1" bias="1" activation="1"
outL="/0/@link.0 /0/@link.1"/>
    <node name="N2" bias="1" activation="1"
outL="/0/@link.2 /0/@link.3"/>
  </InLayer>
  <HLayer tfunction="threshold"
activated="false">
    <node name="N3" bias="-1.5" activation="-1"
inL="/0/@link.0 /0/@link.2" outL="/0/@link.4" />
    <node name="N4" bias="-0.5" activation="-1"
inL="/0/@link.1 /0/@link.3" outL="/0/@link.5" />
  </HLayer>
  <OutLayer tfunction="threshold"
activated="false">
    <node name="N5" bias="-0.5" activation="0.0"
```

```
inL="/0/@link.4 /0/@link.5" />
  </OutLayer>
<link id="0" weight="1.0" src="/0/@InLayer/@node.0" trg="/0/@HLayer.0/@node.0"/>
  <link id="1" weight="1.0" src="/0/@InLayer/@node.0" trg="/0/@HLayer.0/@node.1"/>
  <link id="2" weight="1.0" src="/0/@InLayer/@node.1" trg="/0/@HLayer.0/@node.0"/>
  <link id="3" weight="1.0" src="/0/@InLayer/@node.1" trg="/0/@HLayer.0/@node.1"/>
  <link id="4" weight="-2" src="/0/@HLayer.0/@node.0" trg="/0/@OutLayer/@node.0"/>
  <link id="5" weight="1.0" src="/0/@HLayer.0/@node.1" trg="/0/@OutLayer/@node.0"/>
</System>
</xmi:XMI>
```

The transformations that we propose consist of a set of deals that will be applied to source model ie (level M1 in figure above) in purpose to achieve target model (in our state source model and target model blong to the same metamodel ie level M2 in figure above). The transformations are listed below:

- Determination of the state of the system ( if it is in final state we stop refining)
- Propagation of data from previous layer to next layer
- For each propagation, the nods that compose the layer situated after the last layer activated in the system do the sum pondered of their input and apply the function of transfer to this layer. After the propagation the non activated layer becomes activated.

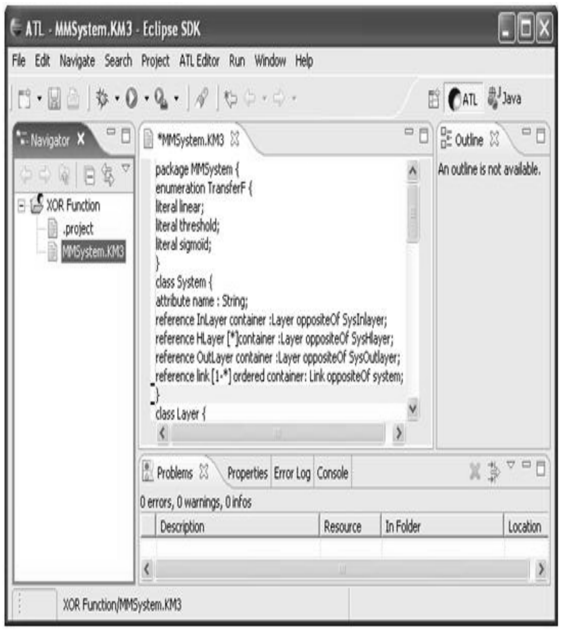
The figure below shows one of the transformation representation listed above in ATL language[15].



```
module Propagation;
create OUTSystem : MMSystem refining INSystem : MMSystem;
helper context MMSystem!Layer def:activatedUpdater():Boolean=
if  self.isTheInLayer()
then true
else if self.isTheFirstHLayer()
then true
else if self.node.first().previousLayerIsActivated()
then true
else false
endif
endif
endif;
```

Figure 11. *This figure show transformation model in ATL language*

we uses the environment eclipse from Java to make this system realizable the figure below show the system represented in Eclipse Platform with plug in Kernel Metametamodel (KM3)[9].


ATL - MMSystem.KM3 - Eclipse SDK
File Edit Navigate Search Project ATL Editor Run Window Help
ATL
Java
Navigator
XOR Function
.project
MMSystem.KM3
*MMSystem.KM3
Outline
An outline is not available.
Problems
Properties
Error Log
Console
0 errors, 0 warnings, 0 infos
Description
Resource
In Folder
Location
XOR Function/MMSystem.KM3

```
package MMSystem {
enumeration TransferF {
literal linear;
literal threshold;
literal sigmoid;
}
class System {
attribute name : String;
reference InLayer container :Layer oppositeOf SysInlayer;
reference HLayer [*]container :Layer oppositeOf SysHlayer;
reference OutLayer container :Layer oppositeOf SysOutlayer;
reference link [1-*] ordered container: Link oppositeOf system;
}
class Layer {
```

Figure 12. *This figure show Xor system in KM3*

## 7. RELATED WORK

The majority of research in the domain of the software engineering aims the process of software development. This current of research is explained

by needs of enterprises. These needs summarize to the productivity, the everlastingness of the knowledge and the consideration of the platform. We saw in literature that the transformations languages currently used are at a time some declarative and imperative languages. They don't provide easiness for transformations that only aim the development of software but, in addition, they can be used, as we showed, to describe the running of systems. The imperative expressions permit the extremely complex dealing description while the declarative expressions are ignoring some details of transformations. These languages can be evolved in this context while introducing flexible techniques allowed the integration of useful concept for the description of all dealing type that can be done by systems.
The field of software engineering has until now focused more on actual architectural solutions, analyses and designs and less on architectural description as put forward in the paper. We hope by developing this formalism to:

- Facilitating the study of bio-inspired systems
- Finding new promising inspiration directions
- Unifying bio-inspired systems terminology

Elicitation of system requirements.

## 8. CONCLUSION

The paper highlights the need for a new formalism and mechanisms to describe bio inspired systems. The paper argues that the need of biological inspired point view is the best opportunity to describe bio inspired systems. The benefit of using biologically inspired view is, for example, to cover the lack of naturalness. In other words, while some biological mechanisms are being intensively used, it seems difficult to maintain a correspondence between the designed systems and their counterparts in the nature. Even if this has no effect on the system effectiveness, it can be a helpful quality in its comprehension. For example, when using an evolutionary process within a robot, it is not obvious to identify what is the individual and what is the species. The robot is what corresponds, at first glance, to an individual, but, within one individual, phylogeny is meaningless.

The paper shows the contribution of Model Driven Architecture to the bio inspired systems modelling. For example the models provide an advantage for artificial neural to have a flexible (adaptable) behaviour. We hope with the combination of MDA and bio inspired views to make the development of complex bio inspired system an easier task

## REFRENCES:

[1]. S. Demeyer, S. Ducasse and O. Nierstrasz. Object-Oriented Reengineering: Patterns and Techniques. In Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM'05), pages 723–724, Budapest (Hungary), 25-30 September 2005.

[2]. T.L. van Zyl and E.M. Ehlers. A Need for Biologically Inspired Architectural Description: The Agent Ontogenesis Case. 10th Pacific Rim International Conference on Multi-Agents (PRIMA 2007), Bangkok (Thailan), 21-23 November, 2007. Lecture Notes in Computer Science, Springer, Vol. 5044, Agent Computing and Multi-Agent Systems, pages 146-157, April 2009.

[3]. M. Luck, P. McBurney, O. Shehory and S. Willmott. Agent technology: Computing as interaction (A roadmap for agent based computing), University of Southampton, January 2005.

[4]. J. Li, X. Mao and Y. Shu. An OO-based Design Model of Software Agent. In Proceedings of the 6th International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT'05), pages 434-440, Dalian (China), 5-8 December2005.

[5]. F. Pagliarecci, L. Spalazzi and G. Capuzzi. Formal Definition of an Agent-Object Programming Language. In Proceedings of International Symposium on Collaborative Technologies and Systems (CTS'06), pages 298-305, Las Vegas, NV (USA), 14-17 May 2006.

[6]. D. Meslati, L. Souici-Meslati and S.E Mili. Software Evolution and Natural Processes: A Taxonomy of Approaches. 3rd International Symposium on Innovation and Information and Communication Technology (ISIICT 2009), pages 48-57, Philadelphia University, Amman (Jordan), 15-17 December 2009.

[7]. M. Sipper, E. Sanche, D. Mange, M. Tomassini, A. Pérez-Uribe and A. Stauffer. A Phylogenetic, Ontogenetic, and Epigenetic View of Bio-Inspired Hardware Systems. IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, pages 83-97, April 1997.

[8]. B. Tekinerdogan, M. Aksit and F. Henninger. Impact of Evolution of Concerns in the Model-Driven Architecture Design Approach. 2sd

International Workshop on Aspect-Based and Model-Based Separation of Concerns in Software Systems, Bilbao (Spain), 10 July 2006. Electronic Notes in Theoretical Computer Science, Elsevier, Vol. 163, No. 2, pages 45-64, April 2007.

[9]. X. Blanc. MDA en action: Ingénierie logicielle guidée par les modèles. Ed, Eyrolles, April 2005 (in french).

[10]. L. Xiao and D. Greer. Adaptive Agent Model: Software Adaptivity using an Agent-oriented Model-Driven Architecture. Journal of Information and Software Technology, Elsevier, Vol. 51, No. 1, pages 109-137, January 2009.

[11]. B. Bauer and J. Odell. UML 2.0 and agents: how to build agent-based systems with the new UML standard. Journal of Engineering Applications of Artificial Intelligence, Elsevier, Vol. 18, No. 2, Special issue on Agent-oriented Software Development, pages 141-157, March 2005.

[12]. E. Sanchez, Phylogeny, Ontogeny, and Epigenesis. From Biology to Hardware, Volume 1259 de LCNS, pages 33-54 Springer, Berlin 1997.

[13]. Dario Floreano,Claudio Mattiussi, Bio-Inspired Artificial Intelligence Theories, Methods, and Technologies, The MIT Press Cambridge, Massachusetts London, England, 2008.

[14]. Seif Eddine Mili, Djamel Meslati , Modeling the phylogenetic dimension of bio-inspired systems: Toward a New Taxonomy Of Bio-inspired Systems, 3rd International conferece On Computer Science And Its Aplications, CIIA 2011 Univesity Of Saida Algeria proceedings p115-121, December 13-15 2011

[15]. F. Allilaire, T. Idrissi, ADT: Eclipse development tools for ATL , In: Proceedings of the second European workshop on Model Driven Architecture (MDA) with an emphasis on methodologies and transformations (EWMDA-2). Computing laboratory, University of Kent, Canterbury, UK. England 2004, p. 171-178.