Multi Dimensional Taxonomy of Bio-inspired Systems Based on Model Driven Architecture

Seif Mili and Djamel Meslati
LISCO Laboratory, Badji Mokhtar-Annaba University, Algeria

Abstract: *The biological metaphor is an analogy between the biological world and the artificial world that enables us to benefit from artificial approaches by imitating some biological aspects while ignoring others. The biological metaphors, also called bio-inspired approaches, depend not only, on the biological field considered, but also on our understanding of that field and the paradigms and means we use to extract practical and useful elements to model some aspects of that field. Today, there is a huge number of metaphors which are different by their very nature and this number is expected to increase according to our inspiration capabilities. In front of this increasing numbers of metaphors it becomes necessary to define the main features of each one in order to evaluate their practical impact, to compare them, to ease their learning and use, to combine them, etc., finding the main or common features of bio-inspired approaches is not an easy task. Although, significant achievement can be noticed in some fields like artificial neural networks or genetic algorithms, a common agreement on definitions and concepts of a huge number of bio-inspired approaches is still lacking. In this article, we propose a multi-dimensional approach based on the Model Driven Architecture (MDA) to describe conceptually a wide range of bio-inspired approaches. Our starting point is to consider that each bio-inspired approach has two aspects: Structural aspect and Behavioural aspect. While the structural aspect is concerned with the involved elements and their relationships, the behavioural aspect deals with the process by which a computing is achieved in an artificial system based on the considered bio-inspired approach. Our choice of the MDA paradigm is justified by its ability to describe uniformly various intricate processes and artefacts involved in the development of software systems. As a preliminary result, our description approach proved to be effective in characterizing a wide range of bio-inspired systems.*

Keywords: *Bio-inspired system taxonomy, MDA, ontogeny, phylogeny, epigeny.*

Received September 20, 2012; accepted February 27, 2013; published online June 26, 2014

1. Introduction

The development of computing systems is now approaching levels of complexity such that their synthesis raises many problems whose solutions exceed the human skills. Since, biological systems have developed a set of well-tried mechanisms and desirable characteristics over millions of years, more and more engineers are beginning to look at nature to find inspiration for the design of software and hardware computing systems [8].

The recent proliferation of bio-inspired systems is due to:

- The confidence in the existence of biological metaphors that can appropriately resolve many problems.
- The success of certain approaches such as neural networks and genetic algorithms.
- The amount of improvement in hardware technology as well as development tools and methods.

In front of this proliferation, it becomes necessary to classify the bio-inspired systems for the following reasons:

- Characterizing and Relating Approaches.
- Searching for Common and Unifying Concepts.
- Facilitating the Study of Bio-Inspired Systems.
- Finding New Promising Inspiration Directions.
- Unifying Bio-Inspired Systems Terminology.
- Elicitating System Requirements.

Finding common characteristics is not an easy task this is due to the lack of common agreement on definitions and concepts. Two parts are involved in the bio-inspired systems: The processes (behavioural aspects) and the architectural structure (structural aspects). As a result to this, putting characteristics for systems means finding characteristics for both parts. We will use the MDA strengths to achieve this.

To cope with the behavioural aspect we need to understand some biological aspects like ontogeny, phylogeny, epigeny, which we describe hereafter.

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 executed 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. This "developmental process" of an individual organism from the earliest embryonic stage to maturity and over is called Ontogeny [8].

Phylogeny is the process of evolution not within an individual, like the ontogeny, but over whole specie.

The phylogeny is mainly achieved through the reproduction which consists in transmitting the genome of one or two parents to the offspring. The genome of the descendant first cell is obtained from that/those of the parents, through mutations and crossing over. Therefore, the living species evolve by the combination of genes within a population of individuals that are genetically compatible.

The epigeny or epigenetics, make use of specific structures to store and handle a huge number of the individual properties of interactions with the environment. The epigenetic process is supported by three systems: The nervous system, the endocrine system and the immune system [10]. The structures used in these systems are easily alterable by interactions with environment and allows the complex living organisms to learn and achieve symbolic processing of information [12].

To cope with the structural aspect of bio-inspired systems, we make use, in our approach, of various engineering concepts involved in the Model Driven Architecture (MDA) approach. We describe them hereafter.

A design pattern names, abstracts and identifies the key aspects of a common design structure that make it useful for creating a reusable object-oriented design [3]. In bio inspired systems design pattern can help developers to design a reusable model in order to facilitating the study of bio-inspired systems and unifying bio-inspired systems terminology.

UML allows modelling of object oriented applications, but it makes use of concepts that are sufficiently generic to be used in many contexts [2]. 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 [11].

In this context, MDA [2] 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 [14]. MDA is a framework defined to separate the platform specific concerns from platform independent concerns, which provide different views of a system [1].

In this paper, we propose a taxonomy of bio inspired systems. By taxonomy, we mean "a system for naming and organizing things into groups which share similar qualities" [6]. Such taxonomy can be used for a wide variety of purposes. To make this, we have to answer the questions:

- What are the important characteristics of bio inspired systems?
- Which MDA concepts can be used to cope with bio inspired systems?
- What are the quality requirements of bio-inspired systems?

This article is organized as follows. In section 2 we answer the questions mentioned above by proposing a new approach of viewing and designing bio-inspired systems. Section 3 shows how our approach is applied to some well known bio-inspired approaches. In section 4, we describe how our approach relates with biological processes. Finally, section 5, gives some related work.

2. Overview of the Proposed Approach

In this section, the main concepts and ideas of our approach are detailed. We first start with an overview of domain description then we will the consider system's computing process and finally system's architecture.

2.1. An Overview

Our method is inspired from both POE model (for more details see [10]) and MDA approach. The model uses three processes Phylogenetic, Ontogenetic and Epigenetic (POE) to characterize a system. In our approach, the Architectural Units (AU) are the central concept. We decomposed the POE processes using AU and we characterized how bio-inspired systems operate using these units or combination of these units. We also use criteria to discriminate the structure of the different bio-inspired systems. Figure 1 shows the general form of the architectural unit.

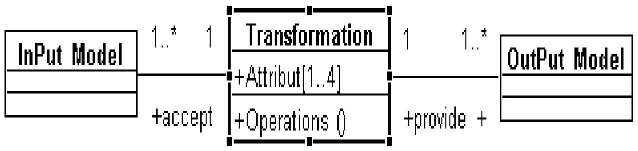

Figure 1. The general form of the architectural unit.

The AU consists of a number n 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. 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, the development AU is a specific AU that can be used in the ontogenetic process Figure 2.

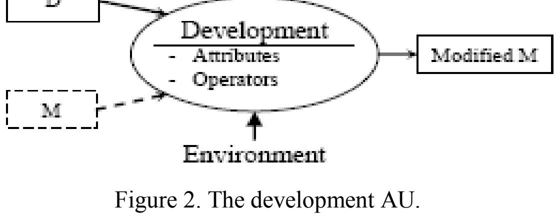

Figure 2. The development AU.

In Figure 2, D is a descriptive model which guides the transformation (plays the role of a genome). M is the model to transform (it is the innate part of a phenotype at the beginning). During the ontogenesis, the output model of one iteration of development is the input model for the next one (i.e., M and Modified M are two consecutives states of the same model).

Notice that, in the beginning, the innate part may be inexistent (this is indicated by dashed lines for model M). Both D and M (in their current state) are used to decide which basic operation is achieved each iteration [7].

We can write the architectural unit like a function as shown by Equation 1:

$$Name\_of\_AU(IM1,IM2,...,IMn) \rightarrow OM1,OM2,...,Omk \quad (1)$$

We propose several basic AU to describe various systems (Develop, Adjust, Interpret, Reproduce, Iterate, Assign). These units will be used below to describe different views of bio inspired systems.

2.2. Describing a System Computing Process and Structure

Each bio-inspired system operates using a particular computing process and has a structure that can be described using some characteristics. In this subsection we first present five description of computing processes then we present four structural characteristics which are involved in computing processes.

The usability of our approach as taxonomic mean for various bio-inspired systems depends on its ability to describe the three main biological processes that are behind any biologically inspired approach. Therefore, allowing the description of these processes using AU is important to show that our approach is able to describe any existing bio-inspired system but also any new one. In this subsection, we characterize biological processes using functional expressions corresponding to basic AU's.

- *The Ontogenetic View*: Is constructed using one AU called the development AU. Formally, the development unit can be written using the following 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 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 and 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.
- *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*, where *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'*.

  Hereafter, we give the UML description of the main transformations implies in bio inspired system's architecture.

- *Selection Transformation*: Figure 3 shows the representation in UML 2.0 of the selected transformation, which take population model as input and provide new population model as output. We use the strategy pattern to design selected transformation model.

A selection class has two attributes: *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* specifies an algorithm for each selection behaviour.

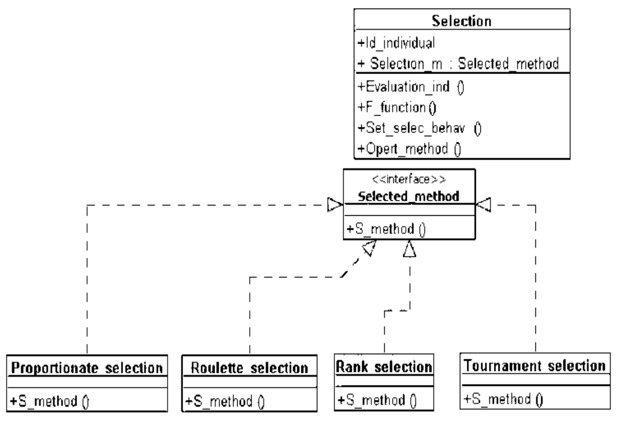

Figure 3. A view of generic metamodel of selection transformation.

- *Reproduction Transformation*: Figure 4 shows the representation in UML 2.0 of the reproduction transformation, which take individual model as input and provide new individual model as output. We use a strategy pattern to design reproduction transformation model.

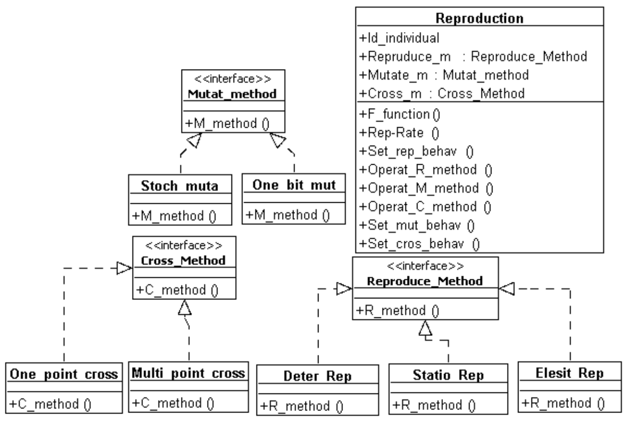

Figure 4. A view of the generic metamodel of reproduction transformation.

A reproduction class has four attributes: *Id_individual* that identifies the individual, *reproduce_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 to the selection and reproduced transformations, the iteration transformation make refinement.

- *Adjustment Transformation*: Figure 5 shows the representation in UML 2.0 of the adjustment transformation, which take individual model as input and provide new individual model as output. We use a strategy pattern to design adjustment transformation model.

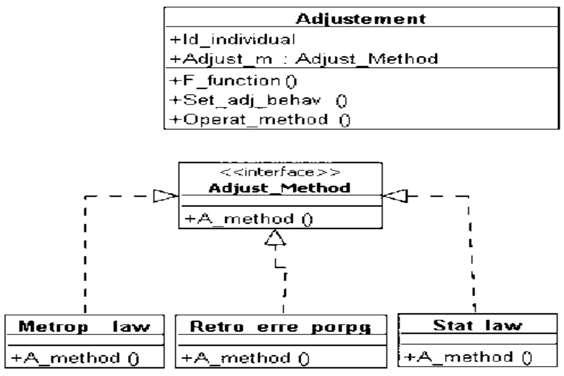

Figure 5. A view of the generic metamodel of adjustment transformation.

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, retro-propagation of error, …). *S_method* define an algorithm for each adjustment behavior.

In Table 1 we present concepts to describe the structure of a system at a given moment.

Table 1. The model criteria set.

Name of the Criterion	Definition
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 or 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.
Composition	A model can be simple or composed. A composed model can be decomposed into sub-models and transformations

2.3. Describing a System Qualities

Besides the previous characteristics that deal with intrinsic features of bio-inspired systems, there is a number of non-functional or quality requirements. In this subsection, we give, in Table 2, a brief description of those quality requirements.

Table 2. The qualities model criteria set.

Name of the Criterion	Definition
Performance and Scalability	A system should be able to cope with large number of entities and resources without sacrificing performance. An important issue that has to do with performance is whether the system computing process is interpreter-based or compiler-based [2]. Usually, compiler-based systems may have the benefit of improved performance over interpreter-based ones.
Extensibility	The flexibility of a system depends on the ease with which the system can be extended with new functionality. For example, many systems offer plug-in frameworks in order to facilitate the addition of third party code (so-called plug-ins) into the system in a standardized way [2].
Usability and Usefulness	The system should be useful, which means that it has to serve a practical purpose. On the other hand, it has to be usable too, which means that it should be intuitive and efficient to use [14].
Interoperability	A system should also be interoperable or easy to integrate with other systems [2].
Standardization	A system should be compliant with all relevant standards (such as XML, MOF, UML). For example, a system may need to support XMI for importing or exporting data models [2].

3. Bio inspired Systems Taxonomy

According to Cambridge dictionary, taxonomy is "A system for naming and organizing things into groups which share similar qualities" [6]. Some taxonomy such as the taxonomic organization of species in a biological context, are hierarchical, but this is not a prerequisite.

In our work, we propose a taxonomy for bio inspired systems that allows us to group tools, techniques or approaches for bio-inspired systems based on their common features. In order to identify these features, we proceeded as follows. Each of the previous subsections investigated an important question and suggested a number of objective criteria to be taken into consideration to provide a concrete answer to the question. Each criterion can be used to group together bio-inspired approaches satisfying this criterion. Our taxonomy provides a bio-inspired and multi-dimensional classification, allowing to group and compare bio-inspired approaches, based on their criteria. There is a huge number bio-inspired approaches, hereafter we will describe the main approaches [4, 5] using the computing process and architectural structure proposed in section 2.

3.1. Genetic Algorithms

Genetic algorithms may solve optimisation problems by operating on binary representations of the individuals and emphasize the role of building blocks and crossover. 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 [5]. The behaviour of systems based on such approaches can be described using functional Expression 2.

$$Iterate\ (C, assign\ (S, Select\ (SM, Reproduce\ (OM\ , S)))) \qquad (2)$$

Where, *Assign (M, M')*: assigns the value of *M'* to *M*. *Iterate (C, AU)*: executes repeatedly AU until the condition *C* is met.

- *C*: Is the convergence condition which is the satisfaction of objective function in optimization.
- *S*: Is the current solution model that resolve our problem (not yet the best one).
- *SM*: Is the selection model. We can use the proportional 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 crossover and both of them have their own model.

The functional expression means that the system reproduces individuals contained in the set *S* according to the model *OM* then selects a subset according to *SM*. The process is reiterated until some condition *C* is met. Concerning the structural aspect of a system based on a genetic algorithm, we can characterize it by the description in Table 3.

Figure 6 describes the met model of genetic algorithms using UML 2.0.

Table 3. The genetic algorithm criteria set

Name of the Criterion	**Description**
Role	The main entities in the system are individuals corresponding to solutions of a given problem. The whole system handles set of individuals that form populations.
Description Type	The individuals are coded in several ways. The most common is the binary representation that relates sequence of bits to individual features. Individuals are also coded as trees in genetic programming.
Element or Set	A system based on genetic algorithm uses both individual and population as main entities.
Granularity	The lowest granularity is a sequence of bits that represents individual features. However, in genetic programming, the granularity corresponds to constructs that can be created in the considered programming language.
Composition	When using AU, a system based on a genetic algorithm is a loop composed of a sequence of two transformations: reproduction and selection.

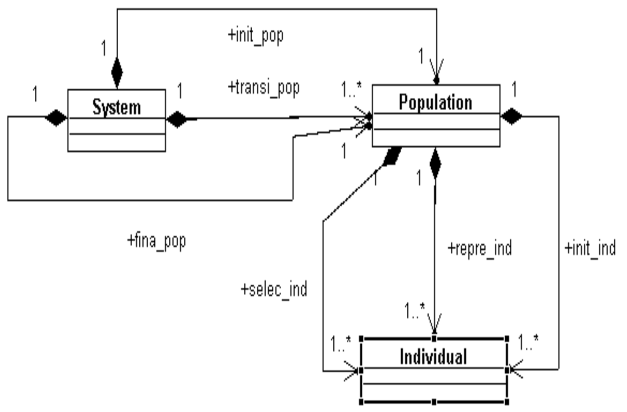

Figure 6. A view of the generic metamodel of genetic algorithms.

The (class *System*), in Figure 6, defines our system and the evolution of a population from an initial state to a final state. We need to use singleton pattern to represent system because there is only one instance.

The (class *Population*) defines the features of the population like seize and individuals. The population evolves according to individuals (selected one, reproduced one). The class Individual defines the characteristic of individuals like representation, seize of individual … [9].

3.2. Negative Selection

The negative selection algorithm assumes that there is a collection *P* of fixed-length strings of symbols that 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 [5]. 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 is given by the functional Expression 3:

$$Iterate\ (C, assign\ (M, Select\ (SM, Interpret\ (IM\ , M, E),E))) \qquad (3)$$

- *C*: Is the condition ending the transformation mechanism, until all detectors or antibodies are compared.
- *M*: Defines the model of detector that represent the population of antibodies.
- *SM*: Defines the selection model. Here, the selection means the feature of good detectors.
- *IM*: Represents the interpretation model which compares the affinity between antibody and antigen.
- *E*: The model that define the antigen.

We give in Table 4 the structural aspect of the negative selection system.

Table 4. The negative selection criteria set.

Name of the Criterion	Description
Role	A Negative selection model plays the role of Population.
Description Type	The antibody can be described in several manners, in high level representation or in low level representation.
Element or Set	There are three classes of elements : antibody, cell population and antigen
Granularity	The granularity of elements is variable due to the several manner of antibody description
Composition	A negative selection systems is a sequence of two transformations the first is an interpretation and second is a selection

We describe in Figure 7 the met model of negative selection (structural aspect) using UML 2.0.

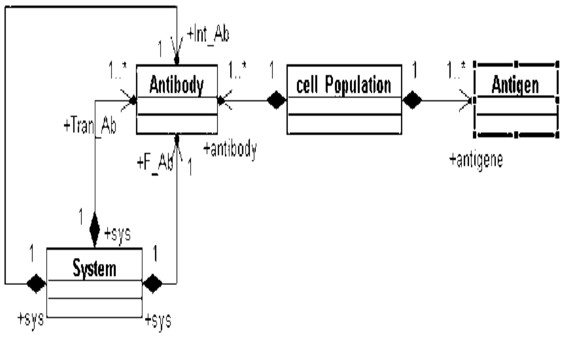

Figure 7. A view of generic metamodel of Negative Selection.

The class system defines our system and it's filtering of population of antibody from the initial state to the final state. We need to use singleton pattern to represent systems because they are one instance of object system. The class cell population defines the features of population like seize and individual, the population changes according antibody (selected one). The class Antibody defines the characteristics of individuals like representation, seize of individual.

Antigen defines the features of individuals that represent the nonself. The Antibody class evolves from initial state to final state crossing transitional states according to the selection transformation and interpret transformation, the iteration transformation make refinement.

3.3. Artificial Neural Networks

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) [5].

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 weight. The behaviour aspect is given by functional Expression 4:

$$Iterate\ (C, Assign\ (M, Adjust\ (AJ, Interpret\ (I, M, E), E)) \qquad (4)$$

Where:

- *C*: Is the condition of end of learning.
- *E*: The model that defines the desirable solution that the network must deliver (supervised learning).
- *M*: The model that define neural network.
- *AJ*: Is the model which gives the adjustment for adapting the network to solve the problem, like retro propagation of error.
- *I*: The model that defines the convergence or divergence of the network.

Table 5 shows the structural aspect of the Neural Network.

Table 5. The neural network criteria set.

Name of the Criterion	Description
Role	The neural network model play the role of one individual
Description Type	The neural network is described with an array of one or more dimensions
Element or Set	A neural network system is a set of layers that are composed of two types of elements : neuron and link
Granularity	The granularity of the phenotype is a set of slot in the array that represents the change of weight.
Composition	A neural network system is a sequence of two transformations the first is interpretation and second is adjustment.

We describe in Figure 8 the met model of the structural aspect of a neural network using UML 2.0.

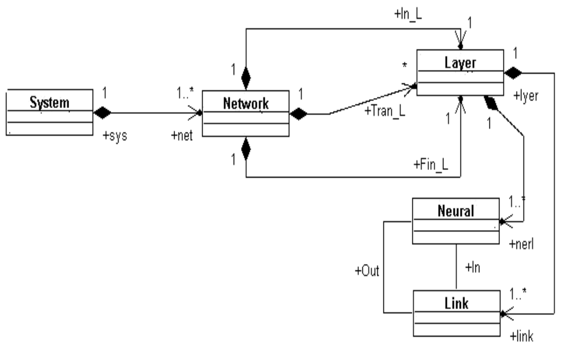

Figure 8. A view of the generic metamodel of artificial neural network.

The class system defines our system. We need to use singleton pattern to represent system because there is one only instance. The class network defines the features of the network like the number of layers. The network evolves according to neuron and link changes (i.e. an adjustment process). The class layer defines the characteristic of each layer of the network like number of neurons, number of links and the type of the layer (input, output or hidden). The class neuron defines the features of a neuron such as the bias. The class link defines the characteristics of the link such as the weight the network class evolves from an initial state to a final state crossing transitional states according to the adjustment transformations and interpret transformation. The iteration transformation makes refinement.

4. Relating Biological Process using our Approach

The following functional Expressions 5, 6 and 7 give the descriptions of the three main biological processes using the AU.

- *Ontogenesis*:

$$\mathit{Iterate}\ (C,\ \mathit{Assign}(Ph,\ \mathit{Develop}(G,Ph))) \tag{5}$$

- *Phylogenies*:

$$\mathit{Iterate}\ (C,\ \mathit{Assign}(S,\ \mathit{Select}(FM,\ \mathit{Reproduce}(RM,S)))) \tag{6}$$

- *Epigenesis*:

$$(\mathit{Assign}(M,\ \mathit{Null}),\mathit{Iterate}(C,\ (\mathit{Iterate}(SC,\ \mathit{Assign}(M,\ \mathit{Develop}\ (D,M))),\ \mathit{Assign}(D,\ \mathit{Adjust}(\mathit{Interpret}(M,\ IDM),D\ ))))) \tag{7}$$

When we analyze these expressions, we can deduce some similarities. While the ontogenetic process develops a phenotype Ph using a low level model (i.e., the genome), the phylogenetic process evolves a set of individuals *S* using a fitness function *FM* and non-deterministic operations.

If we consider abstractly *S*, without looking to the individuals, as a phenotype, we notice that the phylogenesis develops a phenotype using a particular genome *FM*. We can see this fact, when considering an ant colony, where each individual evolves by an ontogenetic process and the colony evolves by a phylogenetic process.

We can also consider the colony as one individual having a respiratory system (achieved by winged ants), a defense system (fighter ants), a reproduction system (the colony queen), etc., in this case, the previous phylogenetic process becomes an ontogenetic one.

At the opposite, each biological organism can be seen as a collection of cells that regenerates continuously and consequently the ontogenesis become a particular form of a phylogenesis.

On another hand, epigenesis provides an individual *M* with various epigenesis properties that are easily alterable. Thus, we can see the epigenesis as a particular ontogenesis that develops properties that are easily alterable, the genome being the society knowledge, where the individual lives and which adjusts the description *D* of *M*.

In the same way, the ontogenesis is an epigenesis that provides individuals with stable properties that are hard to change. We also notice a form of phylogenesis in the learning processes where individuals (i.e., possible solutions to a given problem) are enhanced iteratively to get a suitable solution.

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 9 we summarize our vision of the relationship between the three biological processes.

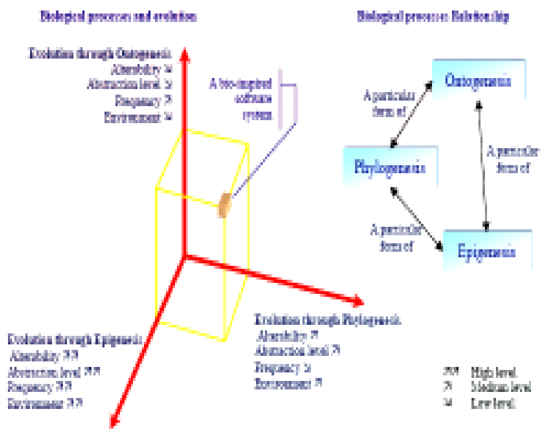

Figure 9. Relationship between the biological processes.

5. Related work

Most of the taxonomies are based on disciplines or sub-disciplines. To the best of our knowledge, only one work is directly related to ours. The authors of this work used the poetic model as the basis for the taxonomy of bio-inspired systems [9]. This poetic taxonomy is itself bio-inspired and deals with a wide range of systems (Figure 10 show the partitioned space of possible bio-inspired systems). However, some weaknesses can be raised:

- Some definitions used may be subject to discussions, such as considering that the environment has no effect during the ontogenesis, where it actually does.

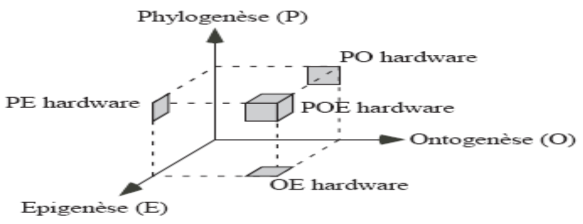

Figure 10. The POE model. Partitioning the space of bio-inspired hardware systems along three axes: Phylogeny, ontogeny and epigenesis.

- Processes can be combined, but the Poetic classification cannot discriminate the diverse forms of combinations (or hybridization). For example, within a combined phylogenetic-epigenetic approach, many combinations may exist.
- The dichotomy individual/species is not considered as an important criterion. We think that the dichotomy is important and allows a better understanding of approaches.

Our work is based on the poetic taxonomy and can be considered as a refinement that uses POE processes as the main discriminating criterion but adds a set of criteria to characterize a wide range of hybrid bio-inspired approaches.

6. Conclusions

We need to bear in mind that a good classification of bio-inspired systems is very useful for the design process of systems and improvement of approaches. Deriving a framework to position and relate software systems would be of great value for their development and maintenance. Unfortunately, little has been done in this direction. In this paper, we proposed an approach that characterizes software systems using POE processes and set of criteria on the three dimensions: Structure, process and environment. The originality of this approach lies in the fact that it can characterize a wide range of systems independently of their hybridization degree. Our resulting taxonomy of bio-inspired systems is mainly multi-dimensional.

As a future issue to this work, we are planning to apply our taxonomy to a large range of bio-inspired systems and try to find new promising directions for hybridizing bio-inspired systems.

References

[1] Bauer B. and Odell J., "UML 2.0 and Agents: How to Build Agent-based Systems with the New UML Standard," *Journal of Engineering Applications of Artificial Intelligence*, vol. 18, no. 2, pp. 141-157, 2005.

[2] Blanc X., *MDA en action: Ingénierie Logicielle Guidée Par Les Modèles*, Eyrolles, 2005.

[3] Cervenka R., Greenwood D., and Trencansky I., "The AML Approach to Modeling Autonomic Systems," *in Proceedings of International Conference on Autonomic and Autonomous Systems*, Silicon Valley, pp. 29, 2006.

[4] Floreano D. and Mattiussi C., *Bio-Inspired Artificial Intelligence Theories, Methods, and Technologies*, The MIT Press Cambridge, Massachusetts London, England, 2008.

[5] Kleppe A., Warmer J., and Bast W., *MDA Explained, The Model-Driven Architecture: Practice and Promise*, Addison Wesley, USA, 2003.

[6] Mens T. and Van-Gorp P., "A Taxonomy of Model Transformation, Electronic," available at: http://staffwww.dcs.shef.ac.uk/people/A.Simons/remodel/papers/MensVanGorpTaxonomy.pdf, last visited 2006.

[7] Meslati D., Souici L., and Ghoul S., "Classification of Software and Hardware Bio-inspired Systems," *in Proceedings of the 4th IEEE International Conference on Computer Systems and Applications*, Dubai, UAE, pp. 1023-1028, 2006.

[8] Meslati D., Souici-Meslati L., and Mili S., "Software Evolution and Natural Processes: A Taxonomy of Approaches," available at: http://www.bcs.org/upload/pdf/ewic_iict09_s2paper2.pdf, last visited 2009.

[9] Sendall S. and Kozaczynski W., "Model Transformation-The Heart and Soul of Model-Driven Software Development," vol. 20, no. 5, pp. 42-45, 2003.

[10] Sipper M., Sanche E., Mange D., Tomassini M., Pérez-Uribe A., and Stauffer A., "A Phylogenetic, Ontogenetic and Epigenetic View of Bio-Inspired Hardware Systems," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 83-97, 1997.

[11] Tekinerdogan B., Aksit M., and Henninger F., "Impact of Evolution of Concerns in the Model-Driven Architecture Design Approach," *in Proceedings of the 2nd International Workshop on Aspect-based and Model-based Separation of Concerns, Software Systems*, Elsevier, pp. 45-64, 2007.

[12] Van-Zyl L. and Ehlers M., "A Need for Biologically Inspired Architectural Description: The Agent Ontogenesis Case," *in Proceedings of 10th Pacific Rim International Conference on Multi-Agents*, Bangkok, pp.146-157, 2009.

[13] Weyns D., Omicini A., and Odell J., "Environment as a First Class Abstraction in Multiagent Systems," *Autonomous Agents and Multi-Agent Systems*, vol. 14, no. 1, pp. 5-30, 2007.

[14] Xiao L. and Greer D., "Adaptive Agent Model: Software Adaptivity using an Agent-Oriented Model-Driven Architecture," *Journal of Information and Software Technology*, vol. 51, no. 1, pp. 109-137, 2009.

Seif Mili received his Engineering Diploma in computer science from Mentouri University, Algeria in 2005. He received Ms degree in software engineering from Mentouri University, Algeria in 2008. Now he is pursuing his PhD in the Department of Computer Science at the University of Badji Mokhtar-Annaba, Algeria. His research interests include software engineering, bio-inspired system and model driven architecture.

Djamel Meslati: is a full professor in the Department Of Computer Science at the University of Badji Mokhtar-Annaba, Algeria. He has been the head of the research group on evolution and reuse of software systems at the Laboratory of Research on Computer Science (LRI) from 2001 to 2012. Currently, he is the head of the Laboratory of Complex Systems Engineering. His research interests include software development, evolution methodologies and separation of concerns.