



**SOFTWARE BIO-INSPIRED SYSTEMS
AN ARTIFICIAL GENOME METHODOLOGY**

BY

IMAD ALDIN MAHMOUD ALSHIEKH

SUPERVISOR

PROF. SAEED GHOU

**This Thesis was Submitted in Partial Fulfillment of the
Requirements for the Master`s Degree in Computer Science**

Deanship of Academic Research and Graduate Studies

Philadelphia University

January 2012

**SOFTWARE BIO-INSPIRED SYSTEMS
AN ARTIFICIAL GENOME METHODOLOGY**

**BY
IMAD ALDIN MAHMOUD ALSHIEKH**

**SUPERVISOR
PROF. SAEED GHOU**

**This Thesis was Submitted in Partial Fulfillment of the
Requirements for the Master`s Degree in Computer Science**

**Deanship of Academic Research and Graduate Studies
Philadelphia University**

January 2012

Dedication

I fully dedicate this thesis to my whole family; father, mother, sister, and brothers.

Also to my professors and all my friends.

Imad Aldin Alshiekh

Acknowledgment

I would like to express my regards and appreciation to Prof. Saeed Ghoul, who has evaluated my work from the beginning and supported me, and to all of those who helped and encouraged me.

Imad Aldin Alshiekh

Table of Contents

Subject	Page
Authorization Form	i
Title	ii
Examination Committee	iii
Dedication	iv
Acknowledgment	v
Table of Contents	vi
List of Figures	viii
List of Abbreviations	x
Abstract	xi
Chapter one: Introduction	1
Chapter two : Literature Review	6
2.1 Overview	7
2.1 Object-Oriented Modeling (OOM)	8
2.2 Aspect-Oriented Modeling (ASP)	10
2.3 Bio -Inspired Modeling (BIM)	11
Chapter three: An Artificial Genome Model	18
3.1 An artificial Genome Elements	19
3.2 UML Notations	21
3.3 An artificial Genome Class Diagram	24
3.4 Genome Genes and Genome Control Genes	25

3.5 Species Architecture Genes and Architecture Control Genes	26
3.6 Organ Genes and Organ Control Genes	27
3.7 Phylogenesis Genes and Phylogenesis Control Genes	28
3.8 Epigenesis Genes and Epigenesis Control Genes	30
3.9 Ontogenesis Genes and Ontogenesis Control Genes	32
3.10 (POE) Control Genes	34
3.11 Artificial Genome Elements Behavior	34
3.12 Case Study	38
3.13 An Artificial Genome Design Methodology	43
Chapter four: Evaluations, Discussions, Conclusions and Perspectives	44
4.1 Evaluations	45
4.2 Discussions	47
4.3 Conclusions	48
References	49

List of Figures

Figures Number	Figure Title	Page
Figure (2-1)	Relations Between Genotype and Phenotype	12
Figure (2-2)	Relations between Cells, Chromosomes, DNA, and Genes	13
Figure (2-3)	POE Model	14
Figure (3-1)	Elements of Genome (S.Ghoul, 2010)	19
Figure (3-2)	Abstract Genome model by (S.Ghoul, 2010)	20
Figure (3-3)	General Artificial Genome Architecture	20
Figure(3-4)	UML Used Notations	22
Figure (3-5)	Artificial Genome Class Diagram	24
Figure (3-6)	Genome Genes Class	25
Figure (3-7)	Genome Control Class	25
Figure(3-8)	Species Architecture class	26
Figure (3-9)	Species Control Genes Class	26
Figure(3-10)	Organ Genes Class	27
Figure(3-11)	Organ Control Genes Class	27
Figure(3-12)	Phylogenesis Genes Class	28
Figure(3-13)	Phylogenesis Control Genes class	28
Figure (3-14)	Epigenesis Class	31
Figure(3-15)	Epigenesis Control Genes Class	31
Figure (3-16)	Ontogenesis Genes Class	33
Figure (3-17)	Ontogenesis Control Genes Class	33
Figure(3-18)	(POE) Control Interface	35
Figure(3-19)	Artificial Genome State diagram	35

Figure (3-20)	Phenotype Growth State Diagram	37
Figure (3-21)	OS with different types of Genomes	38
Figure (3-22)	Genome List in inactive state	39
Figure (3-23)	Queue type selection and activation	40
Figure (3-24)	Genotype selection and genotype network creation	41
Figure (3-25)	Genotype Definition	41
Figure (3-26)	phenotype growth and its communication with the environment (OS)	42

List of Abbreviations

OOM	Object-Oriented Modeling
AOM	Aspect-Oriented Modeling
BIM	Bio-Inspired Modeling
UML	Unified Modeling Language
POE	Phylogeny, Ontogeny, Epigeny
AL	Artificial Life
GRN	Genes Regulatory Networks

Abstract

It is commonly held in current computer software systems that inspiration from biological entities was entirely used as inspiration of biological entities structures, emulation of their motions and reflection of their behaviors. In fact, the literatures have investigated many biological entities as a base of biological inspiration such as ants, swarms, genetics, bees, endocrine systems, and many others.

However the genetics were and still the first biological inspiration, while the extensive methodologies appeared in role of mapping the real genetics into an artificial ones, but these mapping methods are still limited to reflect what happened in real genetics on the real life. This limitation is due to lacking of general framework and proper methodology for such mapping.

In this thesis, we develop a kernel model that integrates the three axes of the space of the software bio-inspired systems, where these axes are Phylogeny, Ontogeny, and Epigeny, i.e. the (POE) model.

We present a rebuttal of the main arguments against POE axes integration and show that they can be herein within a genome kernel to present a novel and new artificial genome model which is presented and illustrated visually by using UML diagrams along with new software modeling methodology.

We also present an example of applying the proposed model to resolving common problem that happens in Operating Systems. The results show new general framework of bio-inspired software systems and new software modeling methodology. The results also show some discussions about this model, evaluations, implementations issues, related frameworks, conclusions and future perspectives.

CHAPTER ONE
INTRODUCTION

Bio-inspired systems is about using natural and biological entities structures, functions, and behaviors not only for designing and developing software systems that benefit from biological characteristics, but combining these characteristics and integrating them together into one general model to allow this model to be used as a general template for designing and producing software systems that have complex, and multiple bio-inspired characteristics.

This is especially important when the functions to be performed by the software system are too complicated to be described by one biological aspect, especially when the software must evolve, grow during its life cycle achieving some capabilities as learning new subjects by interaction with the environment and developing new immunology techniques to defend itself again a threat may attack this software during its functioning inside a specific environment.

Software bio-inspired systems have been inspired and developed during many years ago to be used in many domains as medical applications, real time applications, network applications, security applications, simulation applications, decision making applications, and many others.

Software bio-inspired systems are usable in a vast area of applications as mentioned above. This biological inspiration allows the software systems to improve themselves in dynamic environments where the software system may adapt itself, may change its state, and may force to perform new actions and drop others.

All the software systems information of self modifications and adaptations can be stored on genes, where these genes perform the basic biological entities of inspiration where in all functions, structures, and behaviors of the software bio-inspired system are stored.

There are many software bio-inspired systems that are based on genetics, in the field of artificial evolution (I. Harvey, 2001) involving evolutionary algorithms, (M. Oltean, 2005) or evolutionary computations (P. J. Bentley et al, 2001) which include genetic algorithms (D. Whitley, 2001), evolutionary programming (C. Lee, and X. Fellow, 2004), and genetic programming(E. K. Burke et al, 2004), where in concepts like genome, genetic code, generation, crossover, and mutation are used.

In the field of artificial inheritance the concepts of embryonic class and genetic code are used (D. Dori and E. tatcher, 1994), and the concepts of species, variation, and genetic program are used (D. Meslati and S. Ghoul, 1997). While in the field of bio-inspired hardware systems,

(M. Hinchey et al, 2008), (L. Kang et al, 2007), (A. J. Ijspeert et al, 2006), the components of the model Phylogeny, Ontogeny, and Epigenesis (POE) (M. Sipper et al, 2003) are used.

Study Problem

The problem is that while there are many models based on genetics to design and to produce software systems that are inspired from genetics, there is no general model or framework to gather all these models together as well as lacking of proper methodology for that generalization.

The actual modeling techniques can not reflect the real or part of the real mapping of genetics into an artificial one, however, many efforts are endeavored in this field but this mapping still limited, dividing the space of bio-inspired software systems into three axes, the POE axes may provide us with a new opportunity to integrate these three axes into one general model where the artificial genome is the core or the kernel of that general model, while till now this integration is not achieved yet.

In work (S. Ghoul, 2010), the author presented a general outlines of the artificial genome declaring some genetics concepts as chromosomes, genes, alleles, genotype, phenotype and peristase, etc., however the author introduced his point of view about the artificial genome model, but he did not introduce it in more detailed and sophisticated manner.

Research Objectives

The following research objectives will guide the study and literature review:

1. Investigating the actual modeling techniques and showing their weaknesses in field of software evolution, development, and learning.
2. Proposing general framework for integrating POE axes within an artificial genome model.
3. Introducing the proposed artificial genome visually by using UML diagrams.
4. Proposing a new methodology for modeling bio-inspired software systems.

5. Discussing the proposed artificial genome model by presenting a case study to show how this model can be applied, introducing an evaluation, implementations, and future enhancement for this model.

Rational of the Study

In this study, the biological terms as genes, organ, species, etc..., are used, so as to make this study more appropriate for computer software systems, however, we will use object oriented terminology to illustrate the new model.

The scope of this thesis is software modeling as an important part of software production, for this reason, we can consider our work as an enhancement, generalization, and integration of modeling techniques that are used to model and design the software systems that are inspired from nature and biology especially those that used the genes, and genetic science as a base for their biological inspiration.

In order to present the structure of the proposed artificial genome model we will present each element of the model as a UML class and the whole artificial genome model will be presented as a class diagram with the relations among its classes.

To reflect the behavior of the artificial genome model, we will use the UML state diagram to present the states in which the model changes itself on them.

Significance of the Study

The study will be infinitely important to present more general view of the of bio-inspired software systems, by suggesting new general frame work that integrates the POE axes within it.

Proposing new methodology for modeling bio-inspired software systems along with actual modeling methodologies that are used nowadays such as object oriented, and aspect oriented methodologies. This new methodology can be approved and enhanced as a future work.

In the chapter two, we will give a short survey over the actual modeling methodologies and illustrating their strengths and weaknesses.

After that, in chapter three; we will introduce the proposed Artificial Genome model with an artificial genome modeling methodology, in chapter four; we will present its implementation issues, evaluation, conclusion and future works.

Related Works

Integrating the (POE) axes into one model has not been used yet, some researchers used only the (PO) axes like (Lidia et al, 2007), in their research they have divided all the nature-related computing disciplines (e.g. genetic algorithms, genetic programming, cell computing, neural networks,...etc) into these two axes. While the (Tyrrell, A., et al,2003) integrates the three model of (POE) in computer hard project concerning hardware evolution, so their projects called POEtic project, as the aim of this project is giving the computer hardware the ability to generate more digital circuits to be used for the additional computing task, In the POEtic project the genetic material specifying the functionality of the circuit that stored in the genotype layer (all possible functions), the rules and mechanisms of gene expression that map the genotype into a phenotype are stored and executed in the configuration layer, and the resulting function performed by each cell is implemented and executed in the phenotype layer. The POEtic project depends on direct mapping between genotype-to-phenotype depending on what is the kind of the circuits required and its function.

Direct mapping from genotype to phenotype, does not reflect the real operations inside natural genome, so in our proposed artificial genome, we tried to map the interactions, behaviors, and functions of natural genome into our proposed artificial one. Sure, we can not make mapping for the real genome into an artificial one because the nature is very complicated for any person to understand it.!

However the works (S .Ghoul, 2010) presents a general philosophy for our genome model, which we have detailed by presenting its composed structure elements, functions, and its design methodology with UML.

CHAPTER TWO
LITERATURE REVIEW

2.1 Overview

The concept of software modeling has become important in computer science. There are many usages of the term of software modeling in many disciplines of computer science, especially those concerned in software life cycle and software engineering techniques. Software modeling is the art of abstraction of software-development progress to increase communication, to enhance planning, to minimize risk and cost (B.Gruschko et al, 2007), (A.Cicchetti et al, 2008). While other researchers considered that the code of software must speak itself, which means that the code itself is a model, (J.Bezivin and Jouault, F., 2006), (L.Yamamoto, 2007). The key characteristics of software modeling are widely understood to improve the understandability of the system, to ease system evaluation and maintenance, to increase decomposition and modularization, while most important characteristic is giving more abstract view for the system to be understood easily.

In this context, software modeling in fact has begun to appear since 1960`s and has developed since 60`s up today improving the software life cycles (Hosier, 1961), (Royce, 1970), (Boehm, 1976), (Scacchi, 1984), (Somerville, 1999), (Meyers, B, and Vangheluwe H., 2009), (Meyers. B, and Vangheluwe. H, 2011). And these modeling techniques differentiated from waterfall, spiral, to iterative techniques. Due to the huge change of complexity of software production and change of the nature of software where many major changes that happened to software like (evolution, self-learning, self-immunology, and self-adaptation) and many other behaviors inspired from nature.

For those reasons, it was necessary for rising methodologies of software modeling, which begins with object-oriented modeling (OOM) to pass over an aspect oriented modeling (AOM), and to the most-recent methodologies based on inspiration from biology. All or most used modeling techniques are created to solve the increasing of software complexity, to changing in the nature of software products, and to present more general modeling techniques that may model and handle all types of software product.

In order to model an artificial genome, we will present some significant actual modeling techniques: Object-Oriented technique, Aspect-Oriented technique, and bio-inspired technique.

2.1 Object-Oriented Modeling (OOM)

This approach is used to model the system as a group of interacting objects. Each object represents some entity of interest in the system being modeled, and being characterized by its class, its state (data elements), and its actions. Various models created to show the static structure, dynamic actions, and run-time deployment of these collaborating objects (Grady Booch, 2007).

An object-oriented system is composed of objects. The action of the system results from those objects` Collaborations(Fritz son et al, 2011), (Gianni Ferretti and Francesco Casella, 2010). With this technique, a computer system has been developed on a component basis which enables the reuse of existing components and facilitates the sharing of its components by other systems (Gianni Ferretti and Francesco Casella, 2011). By the following of the OOM, higher productivity, lower maintenance cost and better quality can be achieved.

In fact, there are many object-oriented methodologies' schools as (Coad and Yourdon Method, 1991), (J.Rumbaugh et al,1991), (G.Booch,1994), (Sally Shlaer and Stephen J. Mellor, 1988), (James Martin and james J.Odell, 1992), (I.Jacobson,1993), (D. Coleman, 1994), (Grady Booch,2007), (Pefkaros, Kenneth,2008), (Shahar Maoz et al, 2011).

These schools are the most famous and the founders of OOM but with some differences among them. However, to apply notations that can be used as a standard for all OOM's, a graphical notation language that calls (Unified Modeling Language), (UML) has been approved by (Object Management Group), (OMG). Last version of UML has appeared and (UML 2.4, 2011), while other modeling languages were introduced by (OMG) as (Action Language for Foundational UML (ALF), 2010), (Object Constraints Language (OCL), 2010).

The benefits of OOM such as improving productivity, delivering high-quality system, reducing the maintenance cost, allowing and facilitating the software reuse, etc., are largely recognized. OOM depends on some aspects used in life. These aspects are polymorphism, encapsulation, inheritance and abstraction. So these characteristics of OOM to emulate the life behaviors in classification, representation the entities, and solving the problems were used, but this inspiration from life in OOM has its drawbacks where as these aspects still inefficient, weak, and not capable to reach the perfection of life's behaviors. In fact, the information distinguishing an object from another is actually limited to some elementary characteristics (an identifier, an elementary descriptor, etc.) revealing fewer semantics. Effectively, this information does not show what really the object is. Person's identity card, for example, gives a little description of the person's external aspects but does not give his reality: from where did he come (precise origin setting up his expected physical and psychic characteristics)!, or what is he actually (actual physical and psychic characteristics)!, and what he will probably be (probable evolution of his physical and psychic characteristics, hereditary predisposition, etc.)!.

The OOM approach sounds great in order to model software systems that are based on classes and objects, even OOM is inspired from actual life as a modeling technique and uses some characteristics of our daily life as polymorphism whose meaning appears in multiple faces for the same entity depending on the situation, as well as inheritance which depends on heredity of characteristics from ascendants to descendents, but OOM does not reflect the real inheritance operation as it happens in real life, where a subclass in OOM must inherit all operations of the superclass where this operation in real life is not true, so one child may inherit some characteristics from its parents and the other child may inherit other characteristics, while there are selectivity in inheritance procedure in real life which can not be achieved in OOM.

The object in OOM belongs to a specific class, so during the life time of that object inside an environment, the object still belongs to that class where the object is created, but in real life this operation is not true, as the student belongs to the students class in some period of time but after graduating the same student may belong to employee class which means the same person losing the students' class characteristics and acquiring the employees' class characteristics. This real life behavior can be achieved by OOM. While other important real life aspects which can not be

achieved by OOM make him growing, learning from environment, and defending itself against the threats, so in OOM all operations which will be added or acquired by the object are defined by its class and any changes on these operation must be addressed in the class level. In other words, the object in OOM can not make any changes to itself without applying these changes to its class, where as this aspect is not real in our life.

From the above, OOM has weaknesses in reflecting the real life behaviors to the software modeling, especially, in term of software evolution, growing, learning, and immunology.

2.3 Aspect-Oriented Modeling (AOM)

Aspect-orientation is introduced to overcome some problems with object-oriented techniques (A.Hovssepian et al, 2010). At beginning, let us define the aspect-oriented methodology, AOM is Aspect-Oriented software development technique with an emphasis on the separation of concern's principle (R. Chitchyan and I. Sommerville, 2004). AOM takes the next step, after OOM, in developing well modularized software, by separating the crosscutting concerns. A significant part of work in (AOM) focuses on software evolution support as well as dynamic change due to run-time weaving of aspects (Johan Brichau et al, 2008). AOM provides new mechanisms, such as join points, point cuts and advice, which can be used to facilitate dynamic changes (A.Hovssepian et al, 2010).

In addition, from a reuse perspective, modules that contain code relating to many concerns are likely to be less generally useful in different situations. The phenomenon where multiple concerns intermixed in the code is known as tangling. Of course, good use of design patterns will help in many situations, but the repetition and concern-mixing phenomena will appear even in cases that the design patterns are used to simplify the software design complexity (Yacoub S. and Ammar H, 2004). AOM can overcome the problems caused by code tangling and code scattering, by adding newer functionality to the software model and by creating new aspects. AOM appeared to overcome some shortcoming of OOM especially those related to code scattering, code tangling, and cross-cutting concerns, AOM still suffering from maintenance and debugging problems.

AOM has the same important insufficiencies of OOM in modeling the real life aspects to create software systems that evolve, grow, learn, and immune themselves.

2.4 Bio-Inspired Modeling (BIM)

In fact, using different biological concepts to be applied in multiple computer science disciplines is very old, where as the goal was and still is to reach and emulate the biological behaviors. The beginning was by applying the genetic concept as a new trend in programming depending on evolutionary-based algorithm methodology (Holland, 1975). Concepts as chromosomes, mutation, crossover, population, and natural selection are the main core of genetic programming (GP), (Fogel and David, 2006), (A.Moraglio et al, 2009), (A.Moraglio et al, 2010).

To illustrate the bio-inspired systems that are based on genetics, we will give a small introduction about the genetics science, after that we will determine the actual bio-inspired modeling techniques through dividing the space of the bio-inspired systems into three axes where each axe has its common bio-inspired system modeling technique as will we see later.

Genetics concepts

The genetics is the science of the heredity and variation. The heredity is a process which allows the transmission of physical and psychic characteristics from ascendants to descendants. Genes are the essence of heredity. Every human being has thousands of genes, which are made up of DNA and chromosomes building blocks of life. We distinguish the grain heredity and the environment heredity. The first corresponds to the biological transmission of characteristics, called genotype, that are specific to species.

Genotype is the internally coded, inheritable information carried by all living organisms. This stored information is used as a blueprint or set of instructions for building and maintaining a living creature. These instructions are found within almost all cells (the internal part), they are written in a coded language (the genetic code), they are copied at the time of cell division or reproduction and are passed from one generation to the next (inheritable). These instructions are intimately involved with all aspects of the life of a cell or an organism. The second corresponds to non-biological transmission of characteristics, called peristase, that are more

imprecise and result from the environment, in which lives the entities, The genotype and peristase of an entity define its phenotype, where the phenotype is the physical manifestation of the organism. These are the physical parts, the sum of the atoms, molecules, macromolecules, cells, structures, metabolism, energy utilization, tissues, organs, reflexes and behaviors; anything that is part of the observable structure, function or behavior of a living organism. The variation is the existence of hereditary or non-hereditary differences that are among individuals or groups of individuals. These variations are due to genetic differences.

Figure (2-1), shows the relationship between the genotype and phenotype, where this relation can be defined as:

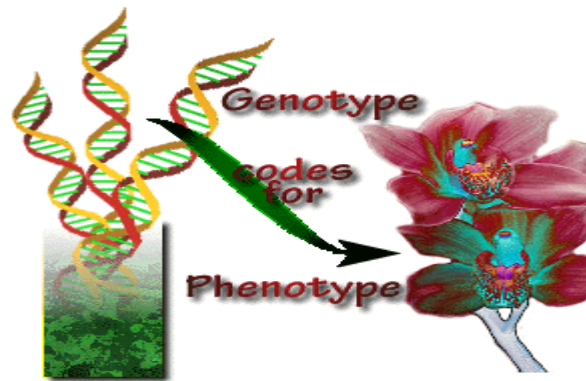


Figure (2-1): Relations between Genotype and Phenotype

Genotype, carried by all living organisms, holds the critical instructions that are used and interpreted by the cellular machinery of the cells to produce the physical manifestation, or Phenotype of the organism.

The genetics as a science studies the transmission of biological characteristics and discusses its rules and mechanisms. Through heredity, living things inherit traits from their parents. Traits are physical characteristics. We resemble our parents because we inherited our hair and skin color, nose shape, height, and other traits from them.

Cells (figure (2-2)) are the basic unit of structure and function of all living things. Tiny biochemical structures inside each cell called genes carry traits from one generation to the next. Genes are made of a chemical called DNA (deoxyribonucleic acid). Genes are bond together to form long chains of DNA in structures known as chromosomes. Genes are like blueprints for building a house, except that they carry the plans for building cells, tissues, organs, and bodies. They have the instructions for making the thousands of chemical building blocks in the body. These building blocks are called proteins. Proteins are made of smaller units called amino acids. Differences in genes called (Alleles) cause the building of different amino acids and proteins. These differences cause individuals to have different traits such as hair color or blood types.

A gene gives only the potential for the development of a trait. How this potential is achieved depends partly on the interaction of the gene with other genes. But it also depends partly on the environment. For example, a person may have a genetic tendency toward being overweight. But the person's actual weight will depend on such environmental factors as how and what kinds of food the person eats and how much exercise that person does.

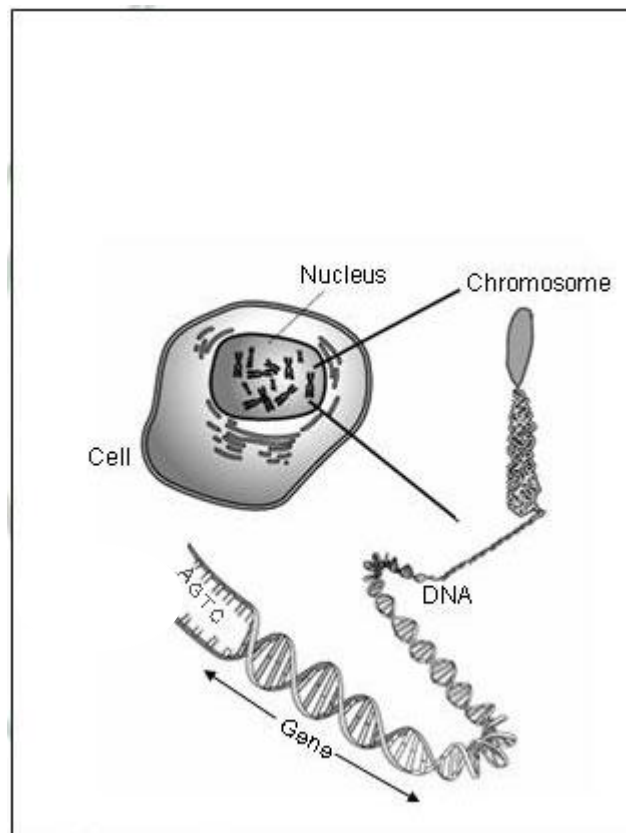


Figure (2-2): Relations between Cells, Chromosomes, DNA, and Genes

In bio-inspired software systems the development of a phenotype (Object, multiple objects, or complete software systems) from its Genotype (the basic information about that object, its functions, its behaviors, etc..) is still at its beginning. Because it must attempt to understand how the genes or any other genetic information element (Genotype) coordinate and harmonize the development and the differentiation of (Phenotype). It is obvious that a phenotype depends on its specific genotype and on its development environment, due to the literature (Cuntz et al, 2008),(Larson.P, 2008),(Shoghuchi et al, 2009), this goal is not achieved properly till now, because there are a huge variety of models and techniques trying to developing software systems (phenotypes) from their related genetic data (genotypes). Here in this thesis the main goal for us is unifying these different models and techniques into one general model to simplify the modeling and designing the bio-inspired software systems.

POE axes

In order to determine the actual Bio-Inspired modeling Systems, the space of bio-inspired systems can be partitioned along these three axes: Phylogeny, Ontogeny, and Epigenesis . It is referred to this as the POE model (figure (2-3)), from which the following definitions are adopted.

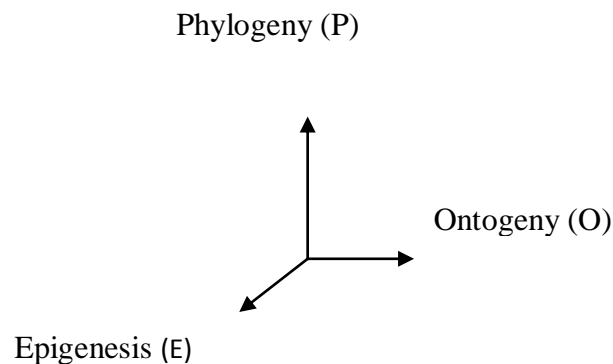


Figure (2-3): The POE Model. Partitioning the space of bio-inspired systems along three axes: Phylogeny, Ontogeny, and epigenesis

- **Phylogeny.** This axis concerns evolution of species (the software). In this category there are many models that are based on evolution as, evolutionary programming (Eiben and Smith, 2003), evolutionary algorithms (Koza et al, 2003), evolutionary strategies (Jens Jagerskupper, 2006), artificial evolution (I. Harvey, 2001), and many others.
- **Ontogeny.** This axis involves the growth or construction of a phenotype (object, multiple objects, or the complete system) from its own genotype (its genetic material). In this category there are many modeling systems as cellular automata (Emmanuel Sapin et al, 2007), membrane computing (Gheorge Paun, 2007), amorphous computing (H. Abelson et al, (2008), and many others.
- **Epigenesis.** This axis involves the interaction of the phenotype with the environment. In biology this interaction is ensured via the nervous system, the immune system, and the endocrine system. This axis concerns learning new subjects and developing new immunology techniques. In this category many models have appeared as artificial neural networks (G. Daniel, 2007), artificial immunology systems (John Timmis, 2008), and many others.

As we noticed the POE axes include the bio-inspired modeling and methodologies techniques that used nowadays to develop the bio-inspired software system.

From the genetics point of view, we observe that the evolutionary computation technique that automatically solves problems without requiring the user to know or specify the form or structure of the solution in advance. Genetic programming is a systematic, domain-independent method to solve problems automatically starting from a high-level statement of what is required (Koza et al, 2003). Genes are the first biological inspiration used in extensive mode in both software system programming and modeling, evolutionary computation used genes or genetic information in an iterative process to apply growth and development of optimized solution (species) for complex problem while there are many other techniques such as, genetic algorithm (GA) (Koza et al, 2003), evolutionary programming (EP) (Eiben and Smith, 2003), evolutionary strategy (ES) (Jens Jagerskupper, 2006), and learnable evolution model (LEM) (Wjtusiak, 2006).

On one hand, as we noticed above, the genes were and still the main inspiration, so more genes-related concepts appeared in bio-inspired methodologies as embryonic, artificial inheritance, variations, phylogeny, ontogeny, epigenesis, genome, genotype, and phenotype. All these concepts used and applied on new generation of artificial life (AL) (A. Eldridge, 2009), which are defined as applying and mapping the processing, evolution, and behavior of life entities into computer system (Takaya Arita, 2009). In the other hand, AL studies the logic of living systems in an artificial environment, where there are two types of AL, soft-life for software, and hard-life for hardware (Takaya Arita, 2009).

Several works on building bio-inspired systems based on the genome have been introduced by (J. Meslati and Ghoul, 2005), (J. Meslati et al 2004), (S. Ghoul, 2010), but still incomplete and do not reflect all aspects of (Genome) as a basis of bio-inspired system modeling, and they did not introduce general frame-work of bio-inspired system modeling POE Model used in modeling bio-inspired hardware (hard-life) (M. Sipper et al, 2003), so the POE model was not used as a complete model on the bio-inspired software model. Phylogeny and Ontogeny (PO) were used as a software models (Lidia et al, 2007).

However, using genes as a basis for building new emergent bio-inspired systems sounds promising but the main challenging problems are:

- There are many types of bio-inspired genes basic systems using genetic information to provide suitable solutions for specific problems.
- Using one or two characteristics of genes to solve a problem discarding other genes` characteristics (e.g. mutation, crossover) used in evolutionary computation.
- Lacking of universal platform or model of bio-inspired systems based on genes.
- Lacking of general bio-inspired methodology to model all real or greater part of aspects of genes from collecting genetic data until producing the solution required.

In conclusion, genes were and still the basic biological inspiration used in many models starting from genetic algorithms (Holland, 1975) ending to DNA computing (Lewin .D, 2002), (Kahn .M et al, 2008). So the main pitfall is that no existing general methodology appeared. Until now,

none existing, approved and agreed modeling language standard for bio-inspired methodology as a future trend like UML as a standard general modeling language for OOM been introduced and approved by Object Management Group. This thesis will introduce a first general model for BIM detailing and formalizing the model presented as outlined by (S.Ghoul, 2010).

Work (S.Ghoul,2010), can be the philosophy background of our new detailed artificial genome model that will be presented and evaluated in next chapters.

CHAPTER THREE
AN ARTIFICIAL GENOME MODEL

In the nature, each biological entity is defined by its temporal trajectory: it comes up by some initial events, completes its growth through some steps defining its life cycle, and goes away at the end. This trajectory is produced and controlled by genetic information according to the varying environment influence. So, the genetic information is supported by meta processes having the environment influence as parameters. This influence is mainly based on learning including culture, experiences, etc, all these processes are part of the genome. In the paper “Bio-inspired systems: An Integrated Model” by (S.Ghoul,2010), the author referred to the main points of the integrated model by suggesting a model that integrates the POE three axes and the author suggested that the artificial genome can be the core of that model, however, the author did not define the bio-inspired modeling techniques that are partitioned in POE space, the author as well did not present a clear view of the internal components of the artificial genome as well as its structures, relations, and behaviors, so we can conclude that the proposed artificial genome of (S.Ghoul,2010) is incomplete, and abstract. As a complement and improvement to the work (S. Ghoul, 2010) we will present the artificial genome in more detailed, and sophisticated manner. In the following we propose an artificial genome model, kernel of bio-inspired systems, along with its functions and behaviors.

3.1 Artificial Genome Elements

The genetic patrimony, genome, of a species includes the definition of all of its possible characteristics (organic, functional, and behavioral) along with the information controlling their coherence.

Each characteristic might be developed in alternative ways (figure 3-1). Each way constitutes a version of this characteristic (figure 3-2), (S.Ghoul,2010). So, a characteristic is defined by an allele of genes; each one is responsible for the development of a version. The physical development and phenotype of organisms can be thought of as a product of genes interacting with each other and with the environment.

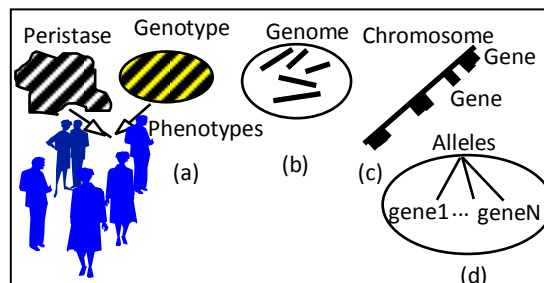


Figure (3-1): Elements of Genome (S.Ghoul,2010)

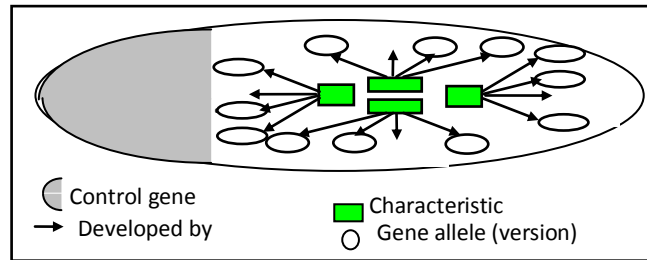


Figure (3-2): Abstract Genome model by (S.Ghoul,2010)

The elements of the genome are all components which develop, manage, and control all operations inside the genome beginning from evolution of the species, during species growth and its communication with environment till death. So these elements are represented inside the genome which can interact, and collaborate together to produce final entities which are the phenotypes (physical entities), figure (3-3) shows the general architecture of the Artificial Genome integrated with POE three axes:

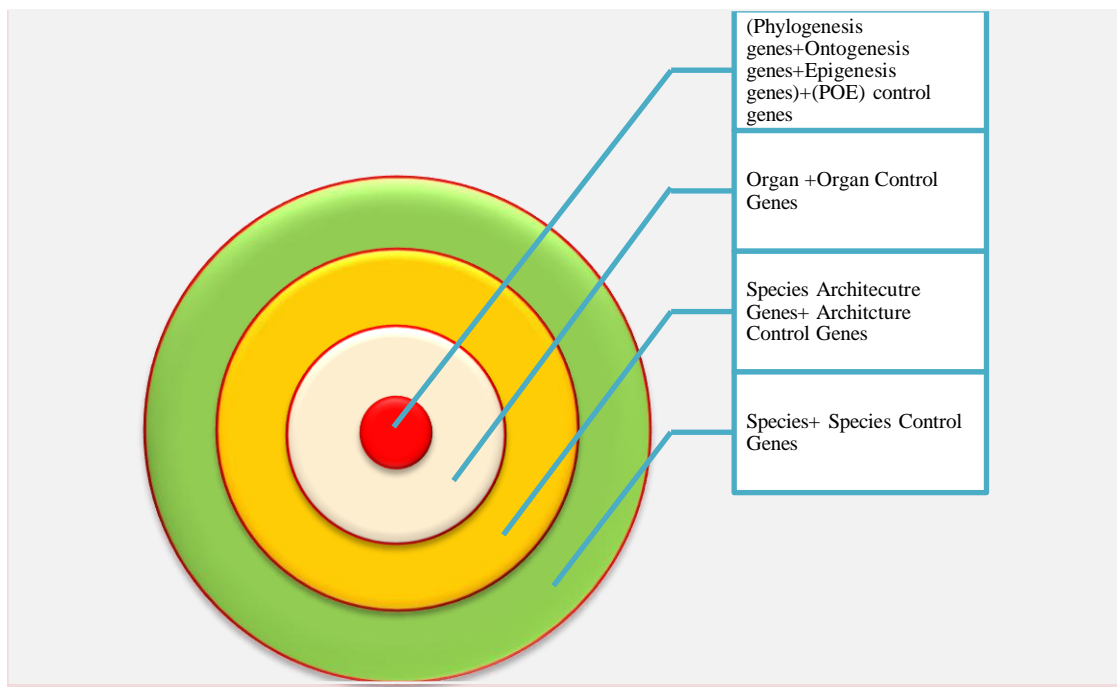


Figure (3-3): General Artificial Genome Architecture

To model the genome, first, we will present the definition of genome elements using textual notations, and then we will convert this definition into visual notation using UML notations.

Second, we will introduce the behavior of the genome, and finally, we will support our model with a case study, and a genome design methodology.

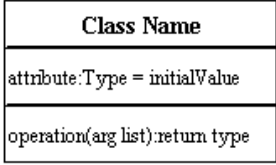
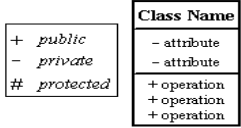
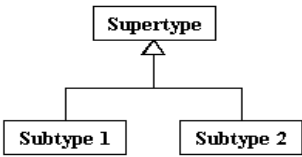
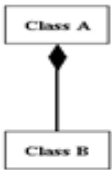
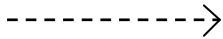

3.2 UML Notations

Currently, there are no existing, complete, and approved modeling language to model the dynamic, self-organized systems where each component of these systems evolves and changes over the time.

Our proposed artificial genome is dynamic, and self-organized system which has the following main properties:

- ***Emergence.*** This property concerns arising global behavior of phenotype behavior resulting from the interaction among the components of the system (genes and their controls).
- ***Adaptive.*** This property concerns genome self-modifying which means changing the state of the elements into other states within different environments.
- ***Evolution.*** The proposed artificial genome is an evolutionary system which is a result of emergence and adaptation which construct the components (genetic materials) and the behavior (functions and operations) that may appear or disappear when needed.

Because of the nature of our proposed artificial genome model, we will use the textual notations. To visualize these notations we will use the approved and formalized modeling language used to model the Object-Oriented systems this language is Unified Modeling Language (UML 2.0). Using as well UML class diagram to illustrate the static view of the components of our system, and using UML state diagram to illustrate the behavior of our proposed artificial genome see (figure (3-4)).

UML 2.0 Notations	Notations shape	Notations description
Class		<p>-Class: It is a blueprint or a definition of objects that share given structural or behavioral characteristics.</p> <p>- Attribute: A typed value attached to each instance of a class.</p> <p>-Operation: A method or function that can be performed by instances of a class.</p>
visibility		<p>-Positive sign(+): Denotes that the access can be performed from out of the class or package (public).</p> <p>-Negative sign(-): Denotes that the access can be performed only inside of the class (private).</p> <p>-protected (#): Denotes that the access can be performed by the class and by all its children (subclasses).</p>
Generalization		<p>-Generalization: Or inheritance is an "is a" relationship. It refers to a relationship between two classes where one class is a specialized version of another.</p>
Composition		<p>-Composition: Is a special type of aggregation that denotes strong ownership between Class A, the whole, and Class B, as its part.</p>
Dependency		<p>-Dependency: The definition or implementation of the dependent classifier might change if the classifier at the arrowhead end is changed</p>
Association		<p>-Association: Association represents static relationships between classes, and the name of the association is put above the line.</p>

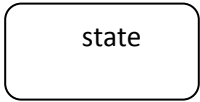
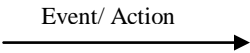
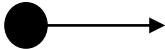
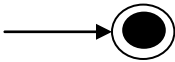
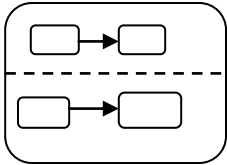
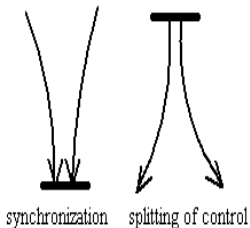
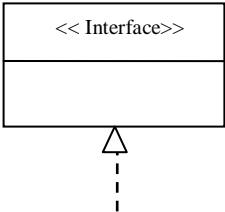
State diagram:		
1. State		-State: Represents the situations during the life time of an object.
2. Transition		-Transition: A solid arrow represents the path between different states of an object.
3. Initial state		-Initial state: A filled circle followed by an arrow represents the Object's Initial state.
4. Final state		-Final state: An arrow pointing to filled circle nested with another circle represents the final state of the object.
5. Composite state		-Composite state: It is a state that has regions where these regions have other substates which can operate in concurrent fashion.
6. Synchronization and splitting	 synchronization splitting of control	-Synchronization and Splitting: Used to perform two or more activity concurrently and must be completed before reaching next state.
Interface		-interface: It is a class that have empty methods which can Implemented from other classes, which means one class can implement multiple interfaces .

Figure (3-4): UML Used Notations

3.3 An Artificial Genome Class Diagram

We start by presenting the general artificial genome model by its class diagram (figure (3-5)) and step by step we define this general model.

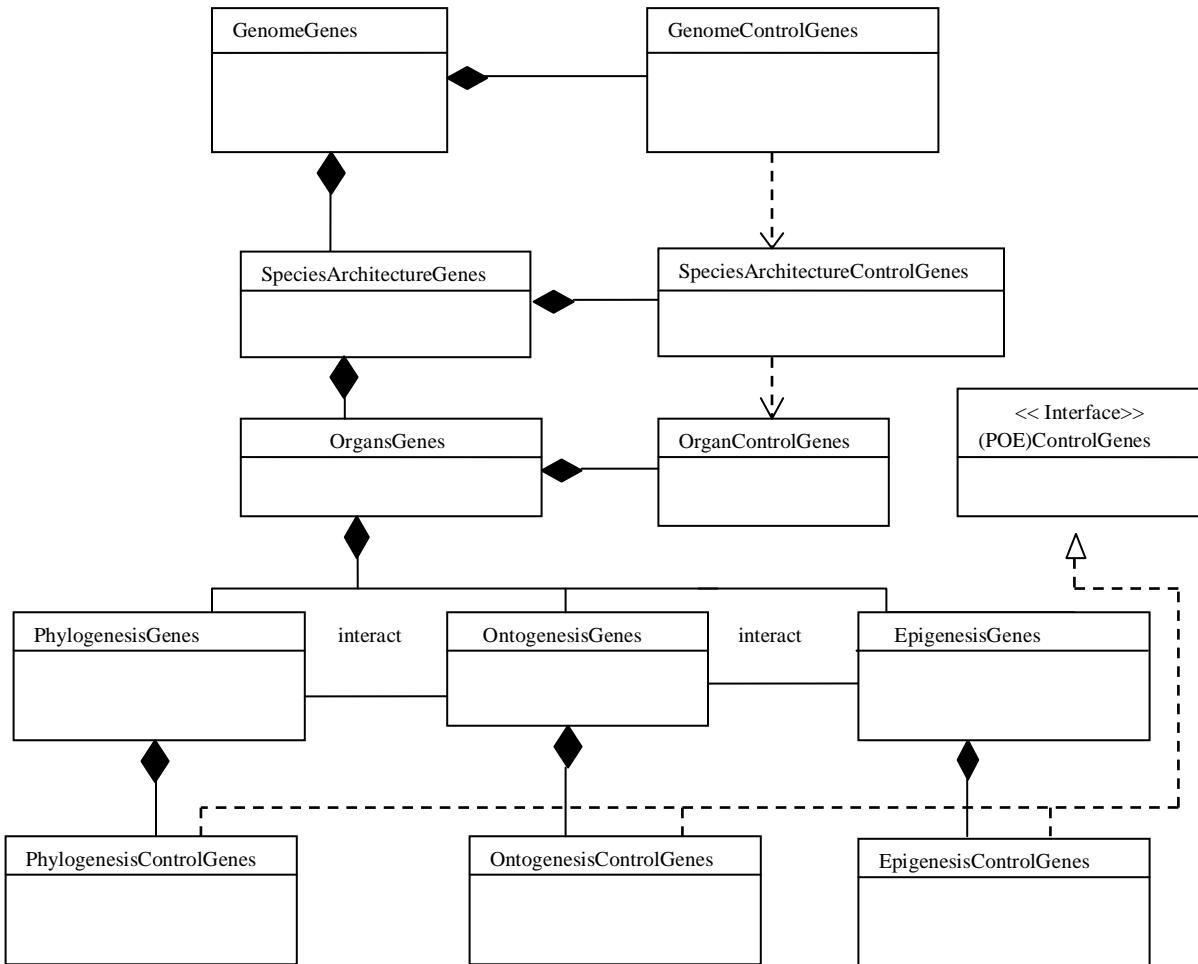


Figure (3-5): Artificial Genome Class Diagram

3.4 Genome Genes and Genome Control Genes

A genome of a species is composed by a set of different architectures (configurations). Each architecture defines the characteristics of a kind of species entities. The genome of human species is composed from two different architectures: male and female. This may be defined as follows:

$$Genome = \{genomegenes_{I, I=1 \rightarrow N}\} \cup \{genomecontro genes\}$$

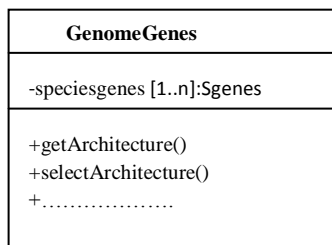


Figure (3-6): Genome Genes Class

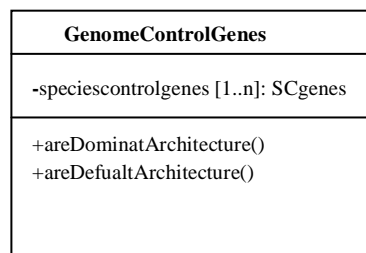


Figure (3-7): Genome Control Class

In figure (3-6), genome architectural genes are genes grouping together with the genes defining an architecture of the species and defining the role or the functions of the species by:

- GetArchitecture(): to return the final and complete shape of the species.
- SelectArchitecture () : to select the required species architecture to be composed and generated.

While figure (3-7) shows the genome control class allowing the control of the selection of an architecture.

$$GenomeControlGenes = \{AreDefaultArchitectures genes\} \cup \{AreDominantArchitectures genes\}$$

- AreDefaultArchitecture : to determine the default architecture of the species.
- AreDominantArchitecture : when the selection of two architectures can produce conflicting choice.

3.5 Species Architecture Genes and Species Architecture Control Genes

Species architecture genes have a list of organs which may be created and used to build the whole architecture of a phenotype of the species.

$$\text{Species Architecture Genes} = \{ \text{Organ_List genes} \} \cup \{ \text{Architecture Control Genes} \}$$

SpeciesArchitecture
- organlist [1...n]: Organs
+getOrgan(): organs +selectGenotype()

Figure (3-8): Species Architecture class

Species ArchitectureControlGenes
-architecturecontrolgenes [1..n]: AGenes
+areImpliedInArchitecture(organlist):organ +areDefaultInArchitecture():organ +areExcludedFromArchitecture(organlist):organ

Figure (3-9): Species Control Genes Class

In figure (3-8), the species architecture class has a list of all organs that are implied in species architecture:

- GetOrgan(): is responsible in displaying all organs that composing the whole species architecture.
- SelectGenotype(): to select the related genetic material to create the genes regulatory networks.

Figure (3-9), shows the species architecture control genes which are the genes allowing the control of the appearing organs into a species by:

- AreImpliedInArchitecure (): defines the organs that are parts of the species architecture.
- AreDefaultInArchitecture (): defines the organs appearing in the species architecture, which are selected by default.
- AreExcludedFromArchitecture (): defines the organs which are excluded from species architecture.

3.6 Organ Genes and Organ Control Genes

The organ genes have references to Phylogenesis, Ontogenesis, and Epigenesis genes the POE genes which have all necessary information about organs, such as description of an organ, its functions, behaviors, mutation, learning, and growth limit, etc....

Organ genes = {Phylogenesis genes, Ontogenesis gene, Epigenesis genes} U {Organ Control Genes}

OrganGenes
- phylogenesisgenes[1..n]:PHgenes - ontogenesisgenes[1..n]:ONgenes - epigenesisgenes [1..n]:EPgenes
+getOrganBasicfunctions() +getOrganSize() +getOrganAge() +developePhenotype() +genomeEvolve()

Figure (3-10): Organ Genes Class

OrganControlGenes
-organcontrolgenes [1..n]:OCgenes
+performSameFunctionOrgan (Organs):organs[1..n] +controlOrganSize() +controlOrganAge() +controlOrganBasicFunctions() +controlOrgansRelations() +controlOrgansBehavior() +.....

Figure (3-11): Organ Control Genes Class

The organ genes class shown in figure (3-10) shows some operations used by organ objects when created, while other operations can be created during the life time of the organ objects, so the basic operations are:

- GetOrganBasicFunctions(): defines all basic functions for an organ (e.g. Eye → See, Open, Close, Flush)
- GetOrganSize (): manages the size of an organ.
- GetOrganAge(): manages the age of an organ.
- DevelopePhenotype(): initiates the construction of phenotype.
- GenomeEvolve(): initiates the process of the genome evolution.

The organ control genes class as shown in figure (3-11) provides all controls over the structure and functions of an organ by:

- PerformSameFuntionOrgan () : defines if other organ performs the same function for specific organ.

- ControlOrganSize(): controls the organ size.
- ControlOrganAge(): controls the organ age.
- SpecifyOrganBasicFunction(): controls the basic function of each organ.
- GetOrgansRelations(): controls the relation among organs.
- GetOrganBehavior(): controls a specific organ behavior.

While other control genes may be appeared and be activated depending on appearing new additions and modifications in organ`s structure and its functions.

3.7 Phylogenesis Genes and Phylogenesis Control Genes

Phylogenesis genes concerns genetic evolution of the species, by searching through the genetic data space to choose the related data used to create the integrated aspects.

Phylogenesis genes = { coding_population genes, coding_function genes, selection_criteria genes, } U { phylogenesis control genes }

PhylogenesisGenes
-codingpopulationgenes [1..n]:CPgenes -codingfunctiongenes [1..n]:CFgenes -selectioncreteriagenes [1..n]:SCgenes +.....
+ Select(CP_genes,CF_genes,SC_genes):CP_genes +Recombine(): Cp_genes +Mutate():Boolean +Replace (Select()): CP_genes +Test(SP_genes,SC_genes):Boolean +Iterate(SP_genes,Replace()):CP_genes +.....

Figure (3-12): Phylogenesis Genes Class

PhylogenesisControlGenes
-Phylogenesiscontrolgenes [1..n]:PCgenes
+AreRelatedtoSolution():Boolean +SolutionSelected():Boolean +SolutionState():Boolean

Figure (3-13): Phylogenesis Control Genes class

The phylogenesis class as shown in figure (3-12), has the genes involved in evolution process, here the evolution process depends on genetic algorithm as basic evolution technique that uses the set of genes (chromosomes) to represent the possible solution population, then the genetic algorithm creates a population of solutions and applies genetic operators such as mutation and crossover, etc., to evolve the solutions in order to find the best one(s).

The phylogenesis genes currently using the genetic evolution (e.g. genetic algorithms) so phylogenesis genes may have and add other evolution techniques if they exist.

The Phylogenesis genes components are described as following:

1. Coding population genes: They have all coded and non coded genes, they are 2 types:
 - Genetic coding information :to determine the [genetic coding location, genetic size] for the data which will be used in evolution
 - Non genetic coding information :(junk data)
2. Coding function genes: they have all the functions that operate over coding population genes, so the functions are shown as following:
 - Select(): selects all possible genetic information available .
 - Recombine(): gathers genetic information together.
 - Mutate(): determines if there are changes in genetic information (good or bad).
 - Replace(): generates new genetic information .
 - Test(): checks if the selected genetic information is the one required depending on Selection criteria.
 - Iterate(): iterates over the genes that have coding genetic information.
3. Selection criteria genes: They include the genes that have all possible criteria to select the genetic information.

Figure (3-13) shows Phylogenesis control genes that have all the control functions over the evolution process:

- AreRelatedtoSolution(): determines if the collected genetic materials have the required solutions.
- SolutionSelected(): controls the searching process for required genetic data, if the genetic data have the solution then the solution selected is set to True.
- SolutionState (on()—off()): to determine if solution state is activated or not.

3.8 Epigenesis Genes and Epigenesis Control Genes

Epigenesis genes concern learning and immunology, so the epigenesis genes achieve learning by activating and deactivating the connections among these genes. Learning is based on genes local information, also learning is the main process for epigenesis genes.

Epigenesis genes also achieve learning by applying the classification, so these genes have the ability to classify information into classes depending on the observed attributes of that information. Epigenesis genes may be considered as data self-adaptive genes which may adapt them-self without out-side interfering.

In immunology side, the epigenesis genes immunity behavior is an innate construction which means they have the capability of initiation and regulation of the immune response.

Epigenesis genes immunity behavior is standard behavior, during which the growth of the organism dynamic immunity behavior is appearing, so this type of immunology is called adaptive immunity which allows the organism to lunch an attack against any threat that the innate system may not remove.

The adaptive immunity is evolved by the growth of the organism and during the learning of that organism through its life cycle. Epigenesis genes can have any additional learning techniques as neural networks and their types as Hopfield neural networks, Associative neural networks,...etc. also epigenesis genes may be able to add any new learning techniques and decision making techniques as fuzzy logic or any others as well as the capability of epigenesis genes to add any new immunology techniques.

Epigenesis genes={ learning genes, immunity genes}U {epigenesis control genes}

Learning genes={subject_activation genes, subject_classification genes}

//Subject_activation genes: They have the pre-genetic learned subjects that are installed in species within thier creation from its genetic evolution, the standard learning activities (e.g eat,see,drink,...etc)

//Subject_classification genes:to classify the learning materials

Immunity genes={innate_immunity_genes,immunity_initiation_genes,immunity_regulation_genes}

Innate_immunity_genes={ defense_mechanism_genes}

Defense_mechanism_genes={negative selection //to protect the body from self-reactions and attack),clonal selection //to differentiate the normal cell from threatened cell, Adaptive defense}

Immunity_initiation_genes={initiate_standard_defense(negative selection and clonal defense),initiate_dynamic_defense(adaptive defense)}

Immunity_regulation_genes={creation_immunological_memory}

EpigenesisGenes
-learninggenes [1...n]: LGenes -immunity genes [1...n]: IGenes
+activateLearning():Boolean +initiateNegativeDefense() +initiateClonalDefense() +initiateAdoptiveDefense() +createImmunolgicalMemory(): +classifyLearningSubject(LGenes) +storingLearnedData(Attacks Data) +.....

Figure (3-14): Epigenesis Class

EpigenesisControlGenes
-Epigenesiscontrolgenes [1..n]: ECgenes
+lunchDefenseMechanism(Initiate_Negative_Defense() ,Initiate_Clonal_Defense(),Initiate_Adoptive_Defense()) +lunchImmunityResponse() +isGood():boolean +isBad(): Boolean +learningLimit()

Figure (3-15): Epigenesis Control Genes Class

In figure (3-14), Epigenesis class contains the following components:

1. Learning genes: They have two types of genes:

- Subject activation genes: which have the pre_installed subjects of learning (e.g. eat, drink,....etc) .
- Subject classifying genes: to classify the learning materials or to recognize what is bad and what is good to be learned.

2. Immunity genes: They have all genes that play the role of the immunology and these genes are:

- **Innate immunity genes:** These genes have references to defense_mechanism genes where defense_mechanism genes have three types of defense:
 - **Negative_selection:** protects the body from self attack.
 - **Clonal_selection:** differentiates the normal cells From the threats cells .
 - **Adaptive defense:** initiates new learning defense style that may be learned from Interaction with environment .
- **Immunity_initiation genes:** They are the genes that initiate the Standard defense mechanisms (negative selection, clonal selection) or Initiate (adaptive defense).
- **Immunity_regulation genes:** They are the genes that are storing all Information about the threats (threats Type, location, duration,...etc) ,to Create the immunological memory to keep all attack history to be used and Retrieved for future defense action.

The epigenesis control genes class shown in figure (3-15), are genes that are controlling the learning and immunity processes through life time of the species by controlling the limit of learning, and differentiating among bad and good learning behaviors and also controlling immunity defense mechanisms by activating the basic and developing new ones. These genes controlling the immunology mechanisms by:

- **LunchDefenseMechanism():** initiates the suitable defense mechanisms for out side attacks.
- **LunchImmunityResponse() :** responds immediately to any attack.
- **IsGood():** returns True if the new subject is good to be learned.
- **IsBad():** returns True if the new subject is bad to be learned.
- **LearnLimit():** determines the limit of learning capability of the species.

3.9 Ontogenesis Genes and Ontogenesis Control Genes

Ontogenesis genes are responsible for the growth and development of the phenotypes (physical entities).The growth and development of the species is a self-reproduction automaton process.

Ontogenesis genes are identical but they behave differently depending on deferent sittings for the same trait to produce different features from same genetic data. Ontogenesis genes are growing by copying themselves, the growing process begins by what is called (Embryo_Seed), which is

the basic genetic material. The organ structure and behavior differentiation comes from different appears of the same genetic data. Ontogenesis process can be divided into four main processes:

- Transition rules (from genotype to phenotype).
- The geometry of the organ (its size and shape).
- The instructions or functions for that organ.
- The behavior of that organ in a specific environment

Transition rules, determine how a state may succeed another state on phase of translation of genetic data into more complex structure to create the phenotype. The geometry will correspond to the structure of the organ, and to which bias may grow. The instruction determines the behaviors or the functions of a specified organ, and the behavior reflects how a specific organ can react in specific situations. Also the ontogenesis genes has the ability to add any new software growth techniques.

Ontogenesis genes={organ definition genes, organ functions genes, organ behavior genes} U { ontogenesis control genes}

Organ definition genes = {organ non coding information, organ coding information}

Organ non coding information = (C) compulsory/ (O) optional

Organ coding information = organ name: Als {gene alleles} endAls

// alleles are the possible settings for the same trait(the eye can be, blue, green.....etc).

OntogenesisGenes
-organdefuntiongenes [1..n]:ODgenes -organdefintiongenes [1..n]:ODgenes -organbehaviorgenes [1...n]:OBgenes
+areSelectedState():boolean +arePriority():Boolean +areActive():Boolean +defineOrganLocation() +defineOrganAge() +areBehaveNormally() +areFunctioning() +.....

Figure (3-16): Ontogenesis Genes Class

OntogenesisControlGenes
-ontogenesiscontrolgenes [1...n]: OCgenes
+areRelatedToAspect() +areExclusive() +areDominant()

Figure (3-17): Ontogenesis Control Genes Class

The Ontogenesis genes elements description as shown in figure (3-16) will be as following:

1. Organ definition genes: These genes are:

- Organ coding information: which have the organ name and organ alleles where alleles are Possible settings for the same trait (e.g. EYE can be BLUE, GREEN....etc).
- Organ non -coding information: these information may be Compulsory (C) or optional (O)To support the organ coding information.

2. Organ function genes: these genes are all the basic functions for an organ:

- AreSelectedState(): to determine organ`s function is selected or not.
- ArePriority():to determine if the selected function has higher priority or not.
- AreActive(): to determine if a specific function Is active or not.

3. Organ behavior genes: these genes are responsible for addressing the behavior of an organ under specific situation and inside specific environment, to know how this organ can behave and if it can Initiate its normal functions or not.

- AreBehaveNormally(): shows if an organ behaves in normal way.
- AreFunctioning(): shows if the organ is functioning or not.

Figure (3-17) shows the ontogenesis control genes, which are genes for controlling the growth and development of an organ by managing the following relations:

- AreRelatedToAspect(): to group the organs functions, relations, and behaviors.
- AreExclusive(): to define the exclusive definitions, functions, and behaviors.
- AreDominant(): to define the dominant definitions, functions, and behaviors.

3.10 (POE) Control Genes

These genes are the main (POE) control genes that can add additional control over Specialized controls of (Phylogenesis, Epigenesis ,and Ontogenesis control genes).

(POE) control genes can be represented by UML diagram as <<interface>> that can provide empty functions which can be implemented by (Phylogenesis, Ontogenesis ,and Epigenesis control genes) as shown in figure (3-18):

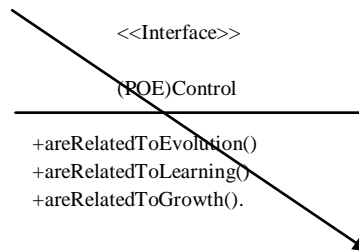


Figure (3-18): (POE) Control Interface

The (POE) control interface has the following empty methods:

- AreRelatedToEvolution(): to be realized by Phylogenesis control genes
- AreRelatedToLearning():to be realized by Epigenesis control genes
- AreRelatedToGrowth():to be realized by Ontogenesis control genes.

3.11 Artificial Genome Elements Behavior

To show the behavior of the model elements, we will use UML statechart diagram. Statechart diagram captures the behavior of the software system, which can be used to model the behavior of class, sub system, or entire software system, also captures the transitions of the system from one state to another.

Our proposed artificial genome behavior state diagram can be shown in figure (3-19) as following:

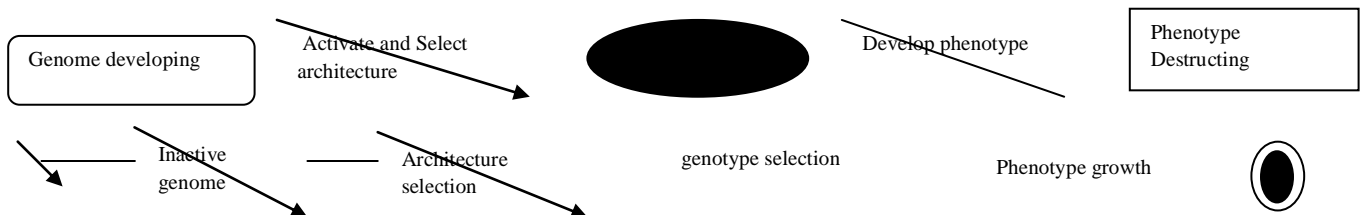


Figure (3-19): Artificial Genome State diagram

In figure (3-19), the state chart diagram shows the states in which the elements of our proposed genome move from one state to another, also we will show every state in more details.

Inactive Genome. Here in this state the entire genome is inactive where each element of the genome is in its initial state and the values of elements attributes are default and elements` objects are not created yet.

This state reflects the situation on which the Genome is in idle state where all genes are (silent) and no making any transitions which means they do not receive stimuli signal from out side environment and do not generate any new activity to transit to new state.

Architecture Selection. After receiving stimuli signal to select architecture from the environment where the genome exists, while the genome being activated which means each element initiates activation process by assigning values to attributes, and beginning to create required information used to determine the selected architecture.

Genotype Selection. In this state, after selection of the architecture type, then genotype selection stimuli begins by activating process (Genotype_Def()). The genotype selection means preparing all genes that have the genetic data materials for specific architecture variants with all data and all operations required for the selected architecture

Phenotype Growth step1. After creating a genotype for a selected architecture, the next process will be the development and the growth of the phenotype or physical entity from its related genotype by activating the process (Phenotype_Init()) to activate organic, functional and control genes, then the phenotype begin to grow up .

Phenotype Growth step2. phylogenesis, epigenesis, and ontogenesis genes begin to interact together and working in concurrent manner to allow the phenotype to grow up, to lean, to defend itself, and to interact with the environment. The details of this state (step1 and step2) are shown as following figure (3-20):

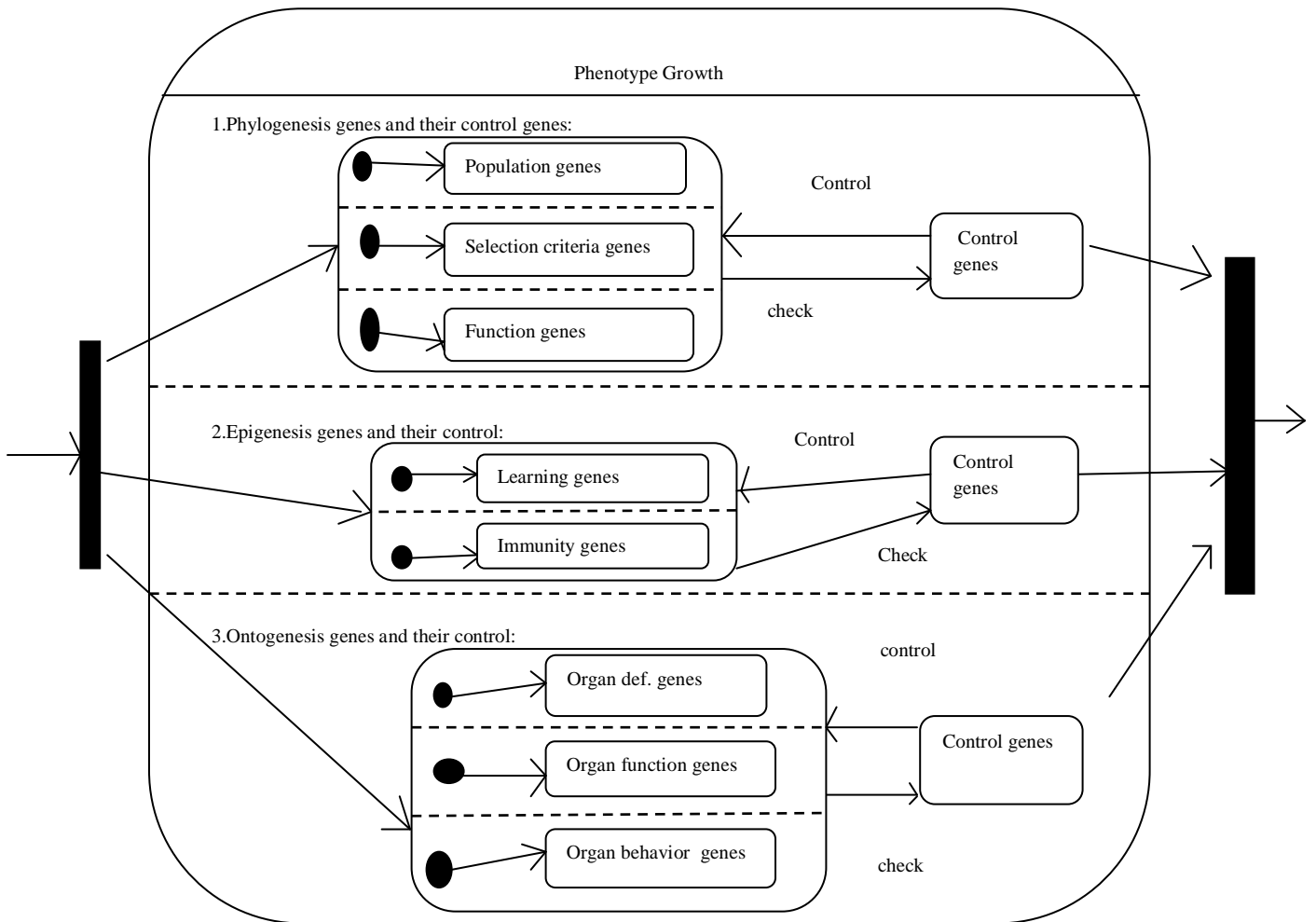


Figure (3-20): Phenotype Growth State Diagram

Figure (3-20) shows the sub states of phenotype growth state, here the phylogenesis, ontogenesis, and Epigenesis genes run concurrently as shown in state diagram in figure (3-20_ and controlled by their control genes, so these genes provide all required information that support the phenotype from its creation through its growth till its death, this information may be the basic functions of the phenotype, the basic behavior, the basic learning activities, basic defense mechanism, the maximum age of the phenotype..etc.

In next section we give a small case study to represent how can we apply the proposed Artificial Genome in software development.

3.12 Case Study

In fact, here we will present an example of how applying our proposed artificial genome to model software systems. Here we will present List Genome as a case study. Here we will present a common case that happens in Operating Systems (OS).

The OS manages all computer resources hardware and software from this point of view the OS handles all the tasks to be executed by the processor, OS stores these tasks in Lists, however the OS may require to create more Lists to store the tasks, here we assume that the OS acquires to create Lists of type queue to be used in storing and managing the tasks to be executed by the processor.

Here we assume that the OS has many artificial Genomes for every operations of the OS, which means that OS may have artificial genome for file management, for memory management, etc...

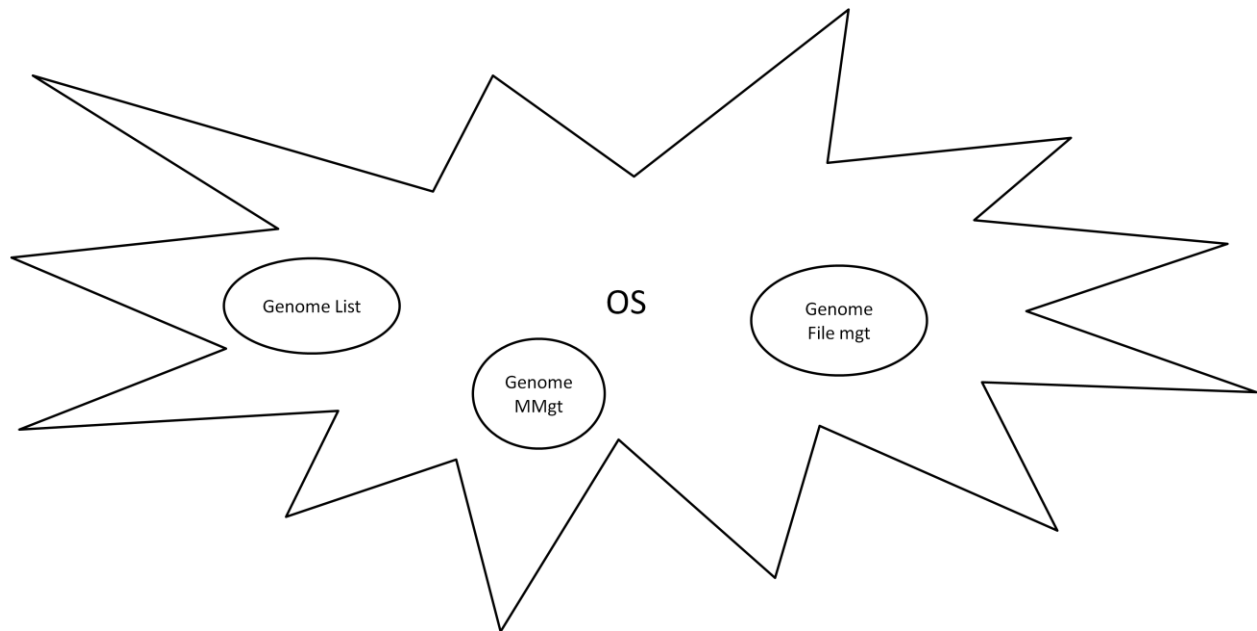


Figure (3-21): OS with different types of Genomes

To apply the artificial genome, we will assume that the OS requires additional queues to be created to handle the tasks by applying the following steps:

- **State:** The Genome list is in silent state (Inactive).
- **Environment influence and objectives:** here the OS will send a stimuli signal to the Genome List with OS requirements, these requirements are creating queues with these specifications:
 - Queue1= {data_type={int}, Q_Size={200}, Q_function={Enqueue(), Dequeue()},Q_structure={Array} }.
 - Queue2={data_type={OBJ},Q_Size={600},Q_function={Enqueue(), Dequeue(),IsFull(),QSearch()},Q_structure={Linked_List} }.
- **Activation and beginning the operations:** here the Genome list will be activated and begins to handle the OS requests by operating in the following steps:
 - Architecture selection.
 - Genotype selection.
 - Phenotype growth.

Genome List in inactive state can be shown as following:

```

Genome List (Type A, Size X){
//Architecture

Queue_List_genes [1...n];// silent.
Stack_List_genes [1....n]; // silent;
Array_List_genes [1...n];// silent;
.....
//Genotype selection
there are no genotypes networks.
//phenotype growth

There are no phenotypes to be created.

```

Figure (3-22): Genome List in inactive state

- **Architecture selection:** the queue architecture will be selected and activated while the rest architecture remain silent to avoid the architecture conflicts, here in this stage the Genome List is activated and beginning to select the required architectures, and the Genome List will initiate the Process (Architect_Select()) where this Genome List class method can select the required architecture by calling another Genome class method which is (Activate()) to activate the selected architecture (the code used here is java like) :

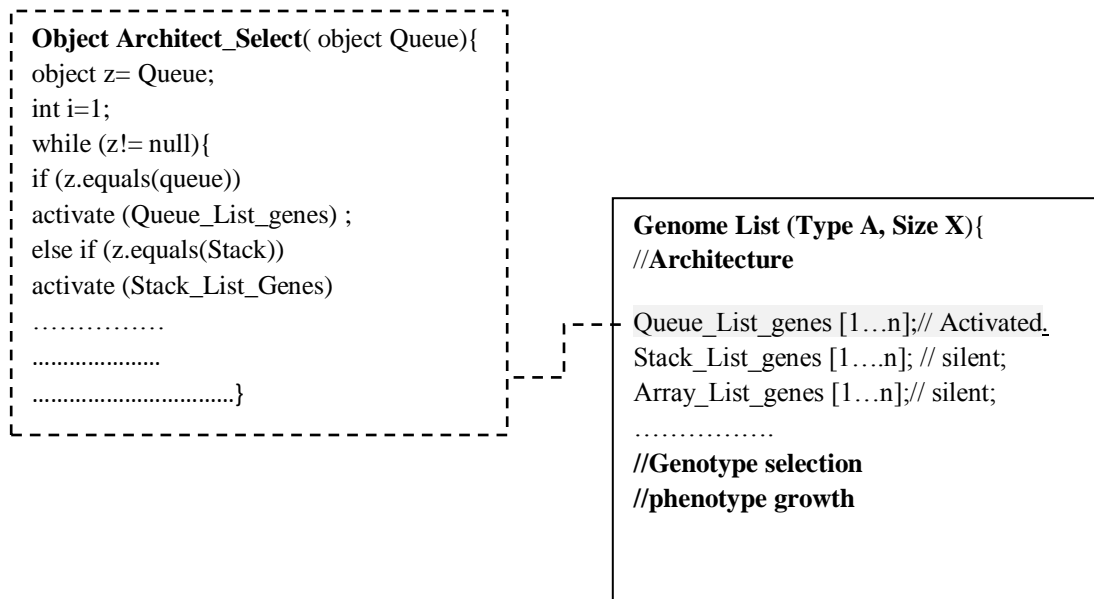


Figure (3-23) : Queue type selection and activation

- **Genotype selection:**

In this phase, the specifications of the selected architecture (e.g. Queue) will be selected and achieved by activation (**Genotype_Def()**) to define the required specification, here the specifications of the required architecture are stored in genes, considering the gene as an array which stored the related genetic data for specific architecture, so every stored data has an address inside that gene.

Firstly, we assign weight value for each index inside the genes, we assume that all data weight values are (0), after that the Genome Class method (**Iterate()**) will move over each

genes element, if the required specification is found then its weight value will be turned into (1). Some data element has references to other genes that stored that data.

Secondly, after activation the required specification, the Genome List will activate the (Genotype_Net()) method to gather all activated specifications and binding them to create a network of related genotype data as shown in figure (3-24):

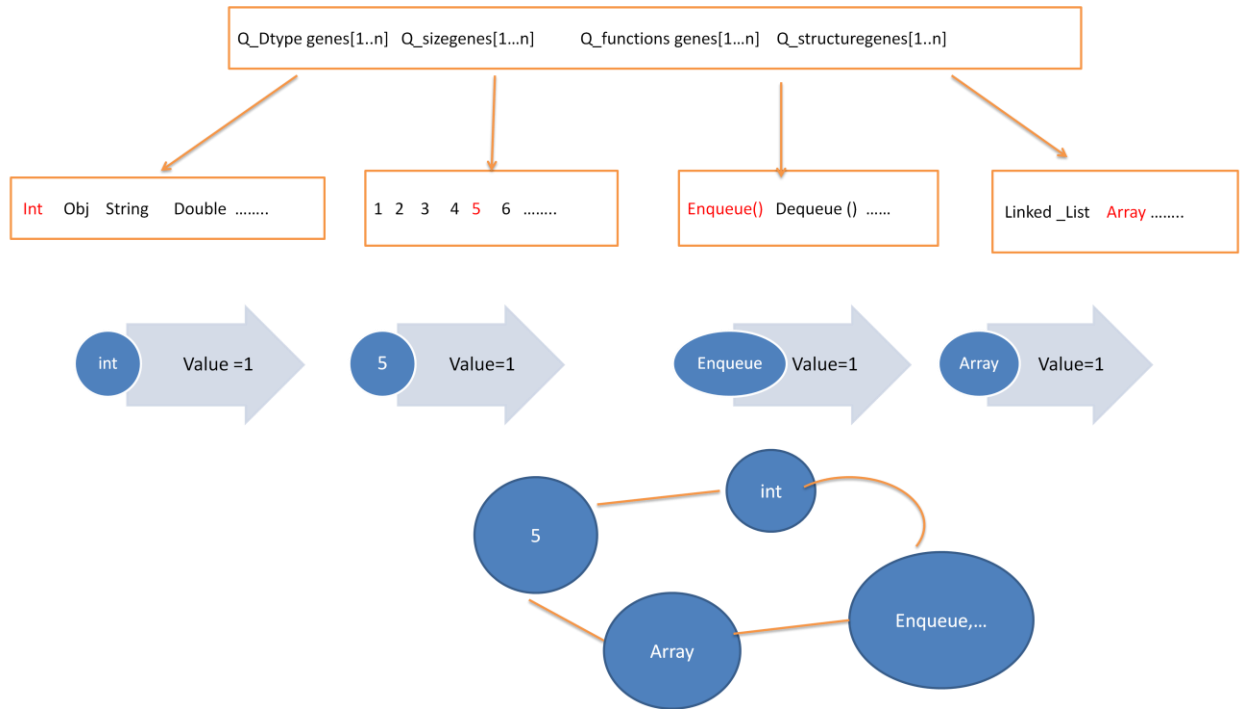


Figure (3-24): Genotype selection and genotype network creation

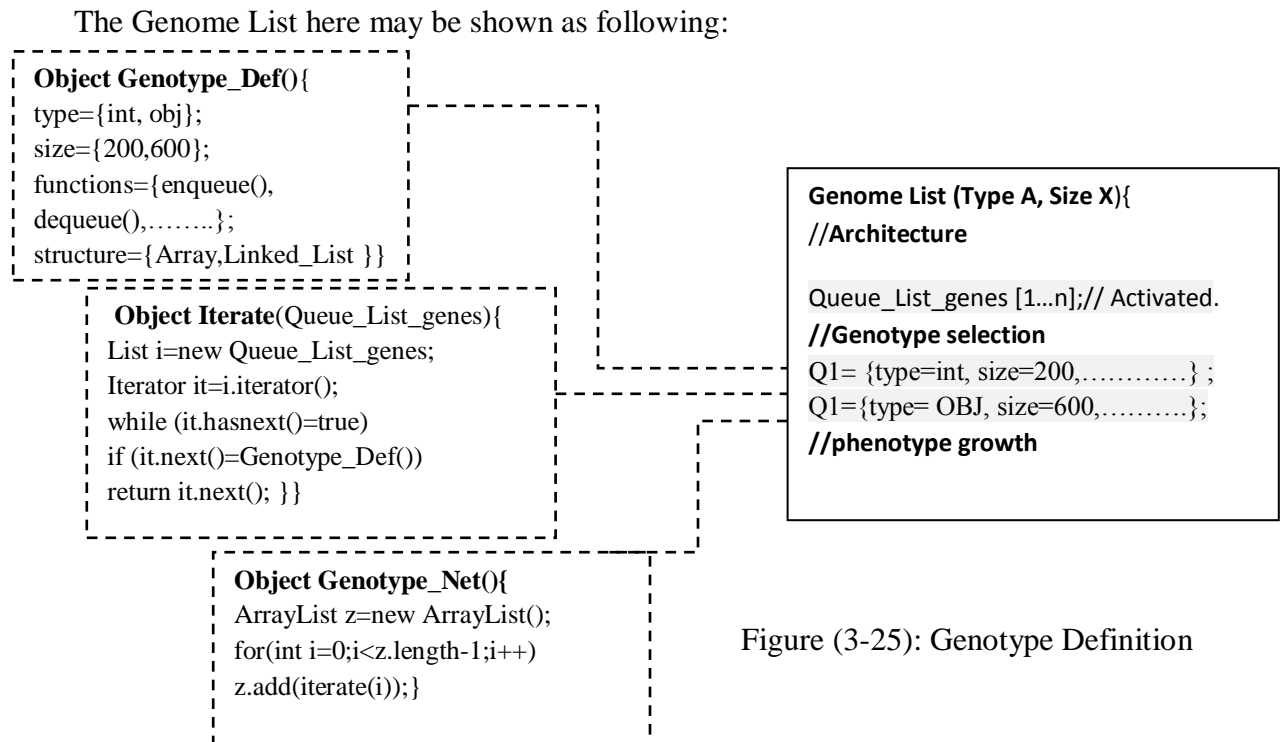


Figure (3-25): Genotype Definition

- **Phenotype Growth:**

In this stage the Genome List begins to create the phenotype (object) from its related genotype (genetic material) by activating process (Phenotype_Init()), also along with this function the phylogeny, Ontogeny, Epigeny (POE) genes begins to operate and communicate together to allow the phenotype (object) to evolve, to grow, to learn and to defend itself. Figure (3-26) shows these steps:

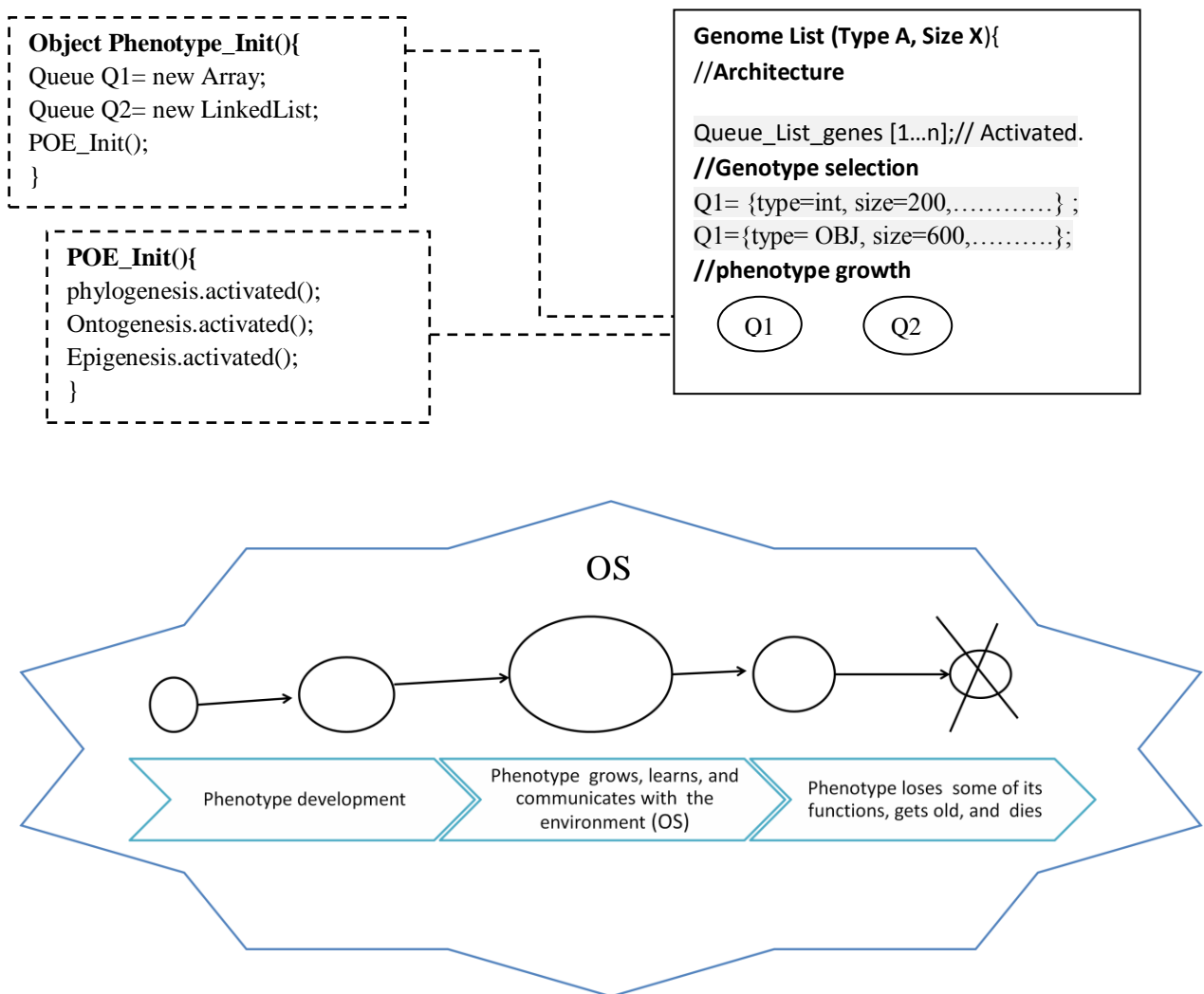


Figure (3-26): phenotype growth and its communication with the environment (OS)

In this small example, we have illustrated how the artificial genome may be applied to solve software problem complexity by gathering all the software data, functions, behaviors, and

structure inside one artificial genome which is able to create the required software in an efficient, and easy way by allowing this software to evolve, grow, and learn by itself.

3.13 An Artificial Genome Design Methodology

The integrated model suggests the following detailed design methodology specification. It is mainly based on the integration of phylogeny methods (supporting the genome evolution), ontogeny methods (supporting the genotype definition, phenotype instantiation, organs development control, and functions control), and epigenesis methods (supporting the behavior control including learning and immunity). Naturally, some steps of the methodology might be omitted according to the targeted goal.

- Step 1. Initial artificial genome design. By designing the genes of all Organs versions, Functions version, and Control. All genes are silent (not active) except those controlling genotype generation and genome evolution.
- Step 2. Evolved genome design. A genome might evolve by activating the genes supporting the process (`Genome_Evol()`) leading to another genome holding needed characteristics. This step may be repeated as well as new genomes are required (this is the case of artificial evolution).
- Step 3. Genotype design. It might be obtained by activating the genes supporting the process (`Genotype_Def()`) leading to locate, in a given genome, the versions of characteristics defining the genotype profile.
- Step 4. Phenotype growth. It might be obtained by activating the genes supporting the process (`Phenotype_Init()`) leading to activating organic genes, functional genes, and control genes; allowing the phenotype growth as well its learning and immunity.

CHAPTER FOUR

EVALUATIONS, DISCUSSIONS, CONCLUSIONS AND PERESPECTIVES

In this chapter we will present the evaluation of the proposed artificial genome, conclusions, and future works.

4.1 Evaluations

The evaluation of this work covers its main contribution, its academic and industrial applications, its implementation issues, and its comparison with similar works.

Contribution

This work has three main contributions:

- A UML artificial Genome model. We have used UML notations to identify the static view of the proposed artificial genome by modeling the genome using class diagram, where the elements of the artificial genome are represented as classes and relations among these classes (e.g. composition, dependency,...etc). Also in dynamic view of the artificial genome we used the state chart diagram to illustrate how the genome evolve and moves from one state to next one, the dynamic view of the genome reflects the behavior of the genome.
- An integration of the POE components in a single uniform artificial Genome model. This integration of the (POE) model into our proposed artificial genome makes our model closer to the nature, so integration evolution along with growth ,learning and immunity into the genome model give the capability for our proposed artificial genome to model more dynamic, self-organized, self-healed, and self-adapted software systems, all these efforts have one major scope which is producing what may be called “ living software systems”.
- An artificial Genome Design methodology. We may consider our proposed artificial genome model as a new modeling methodology which covers some pitfalls of existing modeling methodologies as Object-Oriented Methodology, and Bio-Inspired Methodology. But our new methodology is still in infancy stage. So there are a big amount of work enhancing our new Artificial Genome Modeling, like formalization, adding more biologically characteristics, and developing new modeling language .

Application Area

The Bio-inspired Genome design methodology may be used in all applications dealing with multi form objects of a same class. One example of this may be the Operating System which manages different forms of a same kind of devices (drivers, printers, etc), uses memory management policies which may be based on several forms of blocks: Linear Linked, Circularly Linear Linked, Double Linked, etc. In general, this methodology may be used in all configurations management systems software as well as hardware.

Implementation Issues

The closest implementing support is the Object-Oriented Model (OOM). But it is weakly bio-inspired. So, we have enhanced it by improving the class, variant, and instance concepts. This constitutes a minimal experimental environment for our model. A new classification imposed itself for modeling genomes. This is due to the existence of characteristics in several versions in the same genome (modeling the alleles): in organs, methods, and in behaviors.

One of the direct and practical consequences was the elimination of the omnipresent class versioning problem. Effectively, all the classes defining each one of specific variation (in conventional classification) are reduced to a single class (defining species of those variations).

We have defined a meta gene supporting (Genotype_Def) process. An activation of this meta gene allows the creation of another gene defining a genotype for the associated genome. A phenotype is a genome instance and always attached to one of its genotypes. A phenotype is considered as a process for which the growth is an inherent characteristic. It provides functions with respect to some constraints and policies.

The implementation environment of this methodology requires only some extensions to any object -oriented programming language.

- Characteristics alternatives definition allowing the definition of same concepts (structure, function, behavior, ...) in several ways.
- Characteristics attributes allowing its manipulation: selection, activation, deactivation, etc.

4.2 Discussions

Modeling the software systems that tend to behave as biological entities is very old, many modeling methodologies were investigated in this domain by applying many biological behaviors, structures, and functions to develop software systems that have most or at least one of natural and biological characteristics. For decades, software scientist have been trying to capture many characteristics of biological entities as Ants, Bees, Genetics, Neural Networks, Sea Wave, etc., these efforts produced many modeling techniques along with many methodologies.

Here in our work, we suppose a general frame work with new methodology that can unify all models of building bio-inspired system into one general model, by integrating the POE axes into one general model where the artificial genome plays the role of kernel or core of this integrated model, we illustrate this model graphically by using UML class diagram and by using state diagram to show the dynamic behavior of the proposed model and we applied this model into a case study to show how it works.

The proposed model presented as a set of classes that interact together and each class has its attributes (genes) and its operations that operate over these attributes. Here, we have suggested each class attributes and operations where more of them may be added if needed.

In conclusion, we proposed a general model for building bio-inspired software systems and presented this model by UML class, the semantic meaning of each element of the proposed model is out of scope of this thesis, our work here is modeling, the semantic and proving the model can be a future work in term of enhancement of the proposed artificial genome model.

4.3 Conclusions

By this research, we build a new modeling methodology based on inspiration from biology. Artificial Genome model is an enhancement of actual modeling techniques used nowadays, as Object-Oriented Methodology (OOM), Aspect-Oriented Methodology (AOM), and Bio-Inspired Methodologies (BIM's). So we think that our new artificial genome model has more natural characteristics than other methodologies, which may give our model the ability to model the

software systems that evolve in continuous manner. The aim of our research is suggesting the new model which is based on inspiration from nature to reflecting the evolution, changing over time, and the behavior of software systems along with its methodology, that may be considered or at least we hope, as a new trend in software modeling.

The bio-inspired modeling methodology can be summarized as following:

1. A model for nature definition of software system (define type, structure, size and age).
2. A model for defining software functions and behavior.
3. A model for software evolution and controlling this evolution.

4.4 Future Works

This work may be extended to:

1. Adding more biologically characteristics to our proposed model.
2. Formalizing the model to be more sophisticated and correct.
3. Simulate the genome model to capture and include more details about this model.
4. Applying this model to many different problems types as, real time systems, security, etc.
5. Using the Aspect Oriented technique to model our proposed artificial genome model.
6. Defining the semantic meaning of the proposed model.

REFERENCES

- A. Cicchetti, Ruscio, D.D., Eramo, R., Pierantonio, A.,(2008) “Automating co-evolution in model-driven engineering” In: EDOC '08: Proceedings of the 12th IEEE International EDOC Conference, Munchen, Germany.
- A.E.Eiben and J.E.Smith,(2003),”Introduction to Evolutionary Computing”,Springer, Natural Computing Series 1st edition, 2003, ISBN: 3-540-40184-9.
- A. Eldridge, (2009), “of The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music”. J. Romero and P. Machado (Eds). in Artificial Life 15:2 pp 255 – 258
- A.Hovsepian et al,(2010),”Model Composition vs. Code Weaving:the Maintenance Perspective”.
- A. J. Ijspeert, T. Masuzawa, and S. Kusumoto, (2006), “Biologically Inspired Approaches to Advanced Information Technology”, Proceedings of the Second International Workshop on Biologically Inspired Approaches to Advanced Information Technology, BioADIT 2006, Japan.
- A. Morgallio and J.Togilius,(2009),” Geometric Differential Evolution.” Genetic and Evolutionary Computation Conference (GECCO'09), 1705-1712.
- A. Morgallio and J.Togilius,(2010),” Geometric Differential Evolution for combinatorial and programs space.” Technical Report, School of Computing, University of Kent, Canterbury (UK).
- Boehm, B., (1976),Software Engineering, IEEE Trans. Computer, C-25,12,1226-1241.
- B. Gruschko, Kolovos, D., Paige., R., (2007), “Towards synchronizing models with evolving metamodels”. In: Workshop on Model-Driven Software Evolution,MODSE 2007, 11th European Conference on Software Maintenance and Reengineering, Amsterdam, the Netherland.

- C. Lee, and X. Fellow, (2004), "Evolutionary Programming Using Mutations Based on the Lévy Probability Distribution" IEEE Transactions on Evolutionary Computation, Vol. 8, No. 1.
- Cuntz et al, (2008), "the morphological identity of insects dendrites", PLOS Comput, Biol 4.
- D. Coleman, (1994), "Object Oriented Development: the Fusion Method",
ACCU Review The OO Development (Fusion Method) by Coleman.
- D. Dori and E. tatcher,(1994), "Embryonic classes: Enabling Selective Multiple Inheritance",
Journal of Object-Oriented Programming, 7 (3):36-40.
- D. Meslati and S. Ghoul, (1997), "Semantic classification: a genetic approach to classification in object-oriented models", Journal of Object-Oriented Programming, pp. 25-37.
- D. Whitley,(2001),"An overview of evolutionary algorithms: Practical issues and common pitfalls", Information and Software Technology.
- E. K. Burke, S. Gustafson, and G. Kendall, (2004), "Diversity in Genetic Programming: An Analysis of Measures and Correlation With Fitness", IEEE Transactions on Evolutionary Computations, Vol. 8, No. 1.
- Emmanuel Sapin et al, (2007),"Research of complexity in cellular automata through evolutionary algorithms", faculty of computing ,West England University.
- Fogel, David ,B, (2006), "Evolutionary Computation: Toward a new philosophy of machine intelligence", IEEE press.
- G. Daniel,(2007), " Principles of Neural Networks ", World Scientific Publishing; 2nd edition.
- Gheorge Paun, (2007),"Tracing some open problems in membrane computing", Romanian journal of information science and technology, vol 10, nr 4.
- Gianni Ferretti and Francesco Casella,(2010),"Object Oriented Modeling and Simulation",
Vienna Conference in Mathematical Modeling.

Gianni Ferretti and Francesco Casella,(2011),”Object Oriented Modeling :New Challenges”,
Vienna Conference in Mathematical Modeling.

Grady Booch,(2007),” Object-Oriented Analysis and Design with Applications.Addison-Wesley.

G. Booch, (1994),”Object Oriented Analysis and Design With Applications”.IBM publications.

G. Tempesti, (2003),“Ontogenetic Development and Fault Tolerance in the POEtic Tissue”,
Proceedings of the Fifth International Conference on Evolvable Systems: From Biology to
Hardware (ICES).

Guy Karlebach,Ron Shamir,(2008),”modeling and Analysis of Gene Regulatory Networks”,
Macmilan Publisher Limited,2008.

H. Abelson et al, (2008), “Amorphous Computing,” Comm. ACM, vol. 43, no. 5, pp. 74–82.

Holland, John H, (1975),”Adaptation in Natural and Artificial Systems”, University of Michigan
Press.

Hosair ,(1961), " Pitfalls and Safeguards in Real-Time Digital Systems with Emphasis on
Programming”, I R E Transactions on Engineering Management .

I. Harvey, (2001), “Artificial Evolution: A Continuing SAGA”, In T. Gomi (Ed), Evolutionary
Robotics: From Intelligent Robots to Artificial Life, Proc. of 8th Intl. Symposium on
Evolutionary Robotics, (ER2001), 2217 LNCS.

Jacobson, (1993),”Object Oriented Software Engineering: A Use Case Driven Approach”.IEEE
transactions.

James Martin and James J. Odell,(1992),”Principles of Object Oriented Analysis and Design”,
ACM press.

Jens Jägersküpper,(2006),”How the (1+1) ES using isotropic mutations minimizes positive
define quadratic forms”. Theor. Comput. Sci. 361(1): 38-56.

- J. Bezivin, J. Jouault, F.,(2006),”a DSL for Metamodel Specication”. In: Pro-ceedings of 8th IFIP International Conference on Formal Methods for OpenObject-Based Distributed Systems, LNCS 4037, Bologna, Italy.
- J. Le Boudec and S. Sarafjanovi'c,(2003), “An Artificial Immune System Approach to Misbehavior Detection in Mobile Ad-Hoc Networks”, Technical Report IC/2003/59, EPFL.
- John Timmis,(2008),” A Beginners Guide to Artificial Immune Systems. J. Timmis and P. Andrews. Chapter 3, pp:47-64 of In Silico Immunology Flower, Timmis (Eds)
- J. Meslati et al.,(2004), “The MAGE Ontogenic Model : Towards autonomously-developped Software”, 17th int. Conf. On Software & Systems Engineering and their Applications, Paris.
- J. Meslati and S. Ghoul, (2005), “Toward autonomously developed Software: A genetic approach in critical and embedded systems”, Journal of Computer Science 1 (4): 530: 537.
- J. Rumbaugh et al, (1991), “Object Oriented Modeling and Design”, Rumbaugh's Object Modeling Technology.
- Johan Brichau et al,(2008),” Aspect-Oriented Software Development: An Introduction” In Wiley Encyclopedia of Computer Science and Engineering.
- Kahn .M, et al,(2008),”Toward Molecular Computers that operation in biological environment” ,non-linear phenomena,237(9):1165-1172.
- Koza, John, Martin Keane, Matthew Streeter, William Mydlowec, Jessen Yu and Guido Lanza,(2003),” Genetic Programming IV: Routine Human-Competitive Machine Intelligence”. Kluwer Academic Publishers.
- Larson. P, (2008),”genetic blueprint and environment-which has the bigger impact”, Gerontologist, 48,324.
- Lewin,D,(2002),”DNA computing”, computing in science and engineering4,(3):5-8.
- Lidia Yamato, et al,(2007),”Self-Replication and Self-Modifying Programs in Fraglets”,

Proceedings of 10th European Conference on Genetic Programming, Valencia, Spain.

L. Kang, Y. Liu, S. Zeng, (2007), “Evolvable Systems: From Biology to Hardware”, Proceedings of the 7th International Conference, ICES 2007, Wuhan, China, Springer-Verlag, LNCS 4684.

L.Yamamoto et al,(2007),”Automatic Computer Systems: Self Healing Systems”, University of Basel.

Meyers, B, Vangheluwe H. (2009),”Evolution Of Modeling Languages”, 7th international Fujaba days.

Meyers,B, Vangheluwe,H,(2011),”A framework for evolution of modelling languages”. Science of Computer Programming.

M. Hinchey, A. Pagnoni, F. J. Rammig, and Schmeck ,(2008) “Biologically-Inspired Collaborative Computing”, IFIP 20th World Computer Congress, Second IFIP TC 10 International Conference on Biologically-Inspired Collaborative Computing, September, 2008, Milano, Italy, Springer

M. Oltean ,(2005) ,“Evolving Evolutionary Algorithms using Linear Genetic Programming”. Evolutionary Computation.

M. Sipper, et al.,(2003),“A Phylogenetic, Ontogenetic, and Epigenetic View of Bio-Inspired Hardware Systems" IEEE Transactions on Evolutionary Computations, Vol. 7.

Pefkaros, Kenneth, (2008), “Using object-oriented analysis and design over traditional structured analysis and design”.

Peter Coad and Edward Yourdon, (1991),”Object Oriented Analysis”, 2nd Ed. 1991.

Peter Fritzson, Adrian Pop, Martin Sjölund.,(2011), “Towards Modelica 4 Meta-Programming and Language Modeling with MetaModelica 2.0” Technical reports in Computer and Information Science, ISSN 1654-7233, No 10.

- P. J. Bentley, T. G. W. Gordon, J. Kim, and S. Kumar,(2001), “New Trends in Evolutionary Computation”, Proceedings of Congress on Evolutionary Computation (CEC-2001), Seoul Korea
- R. Chitchyan, I. Sommerville, (2004) AOP and Reflection for Dynamic Hyperslices. Workshop on Reflection, AOP and Meta-Data for Software Evolution (held with ECOOP 2004), Oslo, Norway.
- S. Ghoul, (1991), “Software Process Modeling: A Genetic Approach”. Proceedings of the 1st Maghrebien Symposium on Programming and Systems, USTHB, Algeria.
- S. Ghoul,(2010),”Bio-inspired Systems: An Integrated Model”, MISC2010, International Symposium on Modeling and Implementation of Complex systems, Constantine, Algeria
- Sally Shlaer and Stephen J. Mellor,(1988), “Object Oriented Systems Analysis: Modeling the World in Data”, 1/e.
- Scacchi, W., (1984), Managing Software Engineering Projects: A Social Analysis, IEEE Trans. Software Engineering, SE-10,1, 49-59.
- Shahar Maoz et al, (2011),” Summarizing Semantic Model Difference “, IEEE 14th International Conference on Model Driven Engineering Languages and System.
- Somerville, I. (1999),Software Engineering (7th. Edition), Addison-Wesley, Menlo Park, CA.
- Shoghuchi et al, (2009),”Genome-wide network of regulatory genes for construction of chordate embryo”,Div.Biol,316.
- Takaya Arita,(2009), "A Methodology for Ensuring Constructive Approaches based on Artificial Life Models", Journal of the Japanese Society for Artificial Intelligence, Vol. 24, No. 2, pp. 253-259.
- Tyrrell, A., et all. (2003).” POEtic Tissue: An Integrated Architecture for Bio-Inspired Hardware”. Proc. Of the 5thInt. Conf. on Evolvable System (ICES 03), LNCS 2606, Springer-Verlag

The Object Management Group,(2010, 2011),OMG Unified Modeling Language Specification,
<http://www.omg.org>.

Wojtusiak, J et al,(2006), "Intelligent Optimization via Learnable Evolution Model",
Proceedings of The 18th IEEE International Conference on Tools with Artificial Intelligence,
Washington D.C.

W.W. Roys,(1970), "Managing the Development of Large Software Systems: Concepts and
Techniques", Proceedings WESCON.

Yacoub S., Ammar H, (2004)," An Object-Oriented Framework for Feedback Control
Applications".

ملخص

من المعروف انه في الانظمة البرمجية الحاسوبية تم بناء بعض هذه الانظمة من خلال تطبيق الافكار المستوحاه من الطبيعه (biologically inspiration) حيث تم استخدام بعض العناصر الطبيعية كمصدر للالهام من خلال التركيب او من خلال محاكاة الحركة او من خلال عكس سلوكيات هذه العناصر على الانظمة البرمجية الحاسوبية.

دائما كانت الجينات من اول العناصر الطبيعية الملهمه للاستوحاء حيث تم استخدام العديد من المنهجيات (methodologies) لعمل المحاكاة الفعلية لهذه الجينات من الشكل الطبيعي الى الشكل الاصطناعي ولكن هذه المحاكاة كانت محدودة وذلك بسبب عدم وجود اطار عام ومنهجية واضحة وهكذا محاكاة. في هذه الرسالة سوف نقوم بدمج الاطارات العامة للانظمة البرمجية الحاسوبية وهذه الاطارات هي الخلق والتطور والتعلم سيتم دمجها في اطار واحد حيث سيكون لب او مركز هذا الاطار هو الجينوم الصناعي (Artificial Genome) حيث ستم عرض هذا النموذج المقترح باستخدام لغة النمذجة العامة (UML) وسنقدم دراسة حالة لشرح وتطبيق للنموذج المقترح داخل انظمة التشغيل الحاسوبية.

في النتائج سنقترح اطار عام ومنهجية عامة لبناء الانظمة البرمجية الحاسوبية المستوحاه من الطبيعة كذلك في النتائج سنناقش النموذج المقترح ونقيمه ونعرض الجانب التطبيقي له كما سنتطرق الى عرض الملخص لهذا النموذج مع عرض الاعمال المشابه وكذلك التطويرات المستقبلية.

الأنظمة البرمجية المستوحاة من الطبيعة
منهجية الجينوم الصناعي

من قبل

عماد الدين محمود احمد الشيخ

بإشراف

أ.د. سعيد الغول

قدمت هذه الرسالة استكمالاً لمتطلبات
الحصول على درجة الماجستير في علم الحاسوب

عمادة البحث العلمي والدراسات العليا

جامعة فيلادلفيا

كانون الثاني 2012