A Study on Using Biological Metaphors to Develop Evolutionary Software Systems

Somkiat Kitjongthawonkul
School of Business
Australian Catholic University, St Patrick's Campus
Melbourne, Australia
somkiat.kitjongthawonkul@acu.edu.au

Thuy-Linh Nguyen
School of Business
Australian Catholic University, St Patrick's Campus
Melbourne, Australia
thuy-linh.nguyen@acu.edu.au

***Abstract*—In this paper, we examine the process of establishing a design methodology for software that displays evolutionary characteristics that are similar to living organisms. We use biological metaphors to develop this design. To do this, we first identify what the basic evolutionary entities of the software are, then we investigate what features are required of them, and what evolutionary mechanism and evolutionary path can be designed for them so that the software may evolve in a biological like manner. Finally, we show that object-orientation, a web system, and meta-modelling have great potential for design a system with biologically life-like characteristics.**

Keywords-component; biological evolution; biological metaphors; software evoltuion; object-orientation; meta-modelling techology; statefulness; web system

I. INTRODUCTION

Software maintenance and evolution has always been a thorny issue in the software industry. The costs for the maintenance and management of software [1], which are continually rising, as well as continually changing user needs, have posed throbbing questions that push researchers and practitioners to search for more solutions. Amongst others, the use of a biological metaphors approach to software evolution looks appealing as biological evolution and software evolution have many similarities from basic principles to underlying mechanisms [2]. However, the evolutionary process involved here does not have much documented study, despite a recent increase in awareness of and interest in the problem.[3, 4].

In this paper, we use biological metaphors to establish a design methodology for software which exhibits evolutionary characteristics that are similar to living organisms. The metaphors are derived from the well-understood and well-established Darwinian theory on evolution [5] and many related fields, such as the discoveries of the human genome and DNA (deoxyribonucleic acids). More specifically we suggest how biological metaphors may be used in software development to write software which is capable of evolving in a biological like form.

The first section provides some works in biological discipline that have inspired and given many useful insights into the research for a solution to our problem as well as introducing the relevant biological elements. The subsequent sections attempt to show how by using biological metaphors, a web system, object-orientation, and meta-modelling to work together to produce evolvable software.

II. BACKGROUND AND RELATED WORK

The use of biological metaphors in software engineering is not new. In fact, it was John von Neumann [6] who made one of the first computers by emulating biological systems. The components of his computer were designed after organs, and logic gates after the activity of neurons. The Genetic Algorithm (GA), invented in 1975 by Holland [7], benchmarked the first successful application of Darwinian evolutionary theory in computer science with a simple yet powerful mathematic model. Richard Dawkins [8], who is considered an "astonishingly influential… revolutionary evolutionist" of this time, viewed life as "a process of digital-information transfer" [9]. Peter Small [10] presented his ideas about A-Life Avatar, an artificial host cell on the Internet that can take various forms and behaviour, depending on the artificial life form that resides in it at one particular time. In another book, Small speculated on an analogy between molecules and objects (in object-orientation), with similar message passing mechanisms[11]. Rohit Khare et al. [12] used the term document species to denote document type, which discussed the advent of XML as an "evolution" of documents on the World Wide Web. "Evolution" mentioned there, however, was only at the level of observed facts about changes in document types (in terms of syntax, style, structure and semantics) over time.

Works in these areas has inspired and given many useful insights into the search for a solution to our problem. None of them however has dealt with the problem of evolvability or software evolvability. To our knowledge, we did not find any works dealing with software maintenance and software evolution that use biological analogy or a web system in their approach. Some use object-orientation, or attempt to improve object-orientation, to address the issue of software evolution. These include adaptive programming [13], aspect-oriented programming [14], XAspects [15], subject-oriented programming [16], and reuse contracts [17]. These and other similar works use conventional Computer Science methods to solve their problems. Our approach, which uses biological metaphor, is radically different from them. However as we do share a common base in object-orientation, there would be similarities and lessons to learn from each other, especially when the flexible modular structure requirement in our approach is developed.

2166-0662/13 $26.00 © 2013 IEEE
DOI 10.1109/ISMS.2013.65

Another major area of work in software evolution studies the software evolution phenomenon in the industry and tries to establish theories and practices of software evolution in terms of system dynamics and software maintenance processes [18]. Pioneering in this area are the studies undertaken by Lehman [19]. These works have a different domain to ours.

III. BIOLOGICAL EVOLUTION

It is believed that living organisms evolve over long period of time to withstand changes in the environment, resulting in numerous different species. All life activities require support at the molecular level.

A. Life at the Molecular Level

Life forms require the support at the molecular level. For individuals are made up of cells, which in turn, are made up of what are called organic compounds. All of these organic substances have one common constituent in their composition, which is carbon (C), an element that exists in all forms that have life. Carbon possesses two special features that cannot be found in any other elements [20]. The first is the ability of carbon to combine with other elements to form numerous complex and branching chain structures, which are connected together either by skeletal links or as side-chained functional groups. The presence of a functional group alters characteristic properties of an organic compound. The second is that the connections between elements may only be weakly bound. They can unwind to allow for connections to other elements to be made under certain conditions. In a molecule, a single bond is stronger than a double bond, which in turn, is stronger than a triple bond. It is the composition of these compounds and the complex arrangement of their constituent atoms that create an almost endless variety of cells differing in hereditary components. The weak bindings in their structure play a crucial role in all metabolic reactions to form new substances, as well as in other life processes such as reproduction.

B. Life at the Bilogical Level

Life manifests itself in the capacity to perform two major functional activities, namely, *reproduction* and *metabolism* [21]. Reproduction is the process whereby a living organism produces its offspring, and is controlled by genetic information processing made possible by deoxyribonucleic acids (DNA) [21, 22]. Gregor Mendel found the basic building block involved in this process in 1866, known as a gene [23]. The genes contain coded information capable of directing the synthesis of specific proteins, and substances that make up the cell itself [24]. The discovery of DNA reveals its role as both a model from which specifications of proteins are defined, and as a manufacturer producing these proteins, which subsequently construct and define the characteristics of the living organism. The DNA is called the code of life since it defines the type (species) of the living organism and controls its creation and reproduction.

Metabolism refers to the sum of the chemical processes by which the cell transforms energy to perform life activities such as reproduction, growth, response to stimuli, and elimination of waste materials.

C. Evolution

Over time living organisms evolve into diversified and modified forms to withstand changes in the living conditions. Evolution was not understood until Charles Darwin, in his famous book, On the Origin of Species By Means of Natural Selection [5], introduced the theory. Darwinian theory of evolution has been a topic of investigation from that time, and there is still much debate on the evolutionary process. It has been generally agreed, however, that evolution is a process of repetitive selection of the fittest (called "survival of the fittest") made possible by selective use of the variety of genes, through reproduction. This variety comes from the introduction of new genes into the gene pool of a population of a species. These new genes may be the result of a random genetic combination from each parent in sexual reproduction, or a series of small gene mutations. If these genetic changes increase the survivability of the species, they will be retained through the natural selection process, and multiplied throughout the population; otherwise they will be disadvantaged and disappear [25, 26] .

D. Species and Speciation

Species is the basic category in the biological classification system, and is an attempt to classify organisms into groups that reflect the evolutionary processes underlying the similarities and differences between them. Members in a species can mate and produce offspring with one another, but do not breed with members of other species. Closely related species are grouped into a genus (plural genera). Genera are then grouped into families, families into orders, orders into classes, classes into phyla, and phyla into kingdoms [26, 27].

Speciation is the complex process of forming a new species, and is believed to happen along the evolutionary process. Speciation probably starts first with extrinsic isolation, where a species becomes subdivided because of some extrinsic event such as a climatic change or a geographical separation. Secondly, the isolated population becomes genetically differentiated, possibly because its individuals multiply more quickly than there are chances to mate with individuals in other populations. Genetic divergence can happen as a result of natural selection or randomly. Thirdly, intrinsic isolation, some form of isolation within individuals among the differentiated populations, develops. This could be the result of some preferences in courtship or some genetic incompatibilities, making offspring of mating between individuals of differentiated populations no longer viable or fertile. Finally, the independence of a species is confirmed when newly separated populations continue to evolve independently, and may subsequently invade each other's habitats without hybridization [5, 26].

IV. BIOLOGICAL EVOLUTION VS SOFTWARE EVOLUTION

This section discusses the features which can be sought for in the design of a biological life-like software system. By this means we will look at how a web system, object-

orientation, and meta-modelling can support this design and where development should be concentrated.

A. The Biological Metaphors

One condition for evolution to happen is that there exists a population in which individuals can connect, communicate and interact with each other to spread the change. A web system, either the World Wide Web, or a private web system that is built upon an intranet, appears to be a good candidate to satisfy this requirement. Here the system is analogous to the planet, a web site to a geographical region, a web document to a living organism, a web object that belongs to a web document to an organ, a document type to a species, and the communities of users and developers to the environment (see Table I). Looked at in this way a web document and a web object are both evolutionary entities, similar to biological life forms. As far as evolution is concerned they have similar roles and the term "web form" is used here to denote either a web document or a web object when it is not necessary to distinguish them. The basic evolutionary entity however, would be web object, which can be made to represent an object-oriented software object. A web document can thus represent an object-oriented software system, and a web community a software domain.

TABLE I. A WEB SYSTEM AS AN ECOLOGICAL SYSTEM OF THE BILOGOGICAL WORLD

Biological world	Web system
Geographical region	Web site
Living organism	Web document
Organ	Web object belonging to a web document
Species	Document type
Environment	Communities of users and developers

Many resemblances to living world can be seen in both the World Wide Web and a private web system. For their spaces are populated by web forms uniquely identified by their URLs in much the same way our planet is populated with living creatures, each of which is a unique entity in the universe. Forms are created and destroyed, just as living creatures are born and die. The interconnectivity inherent in the Internet or intranet allows for a network of web forms communicating and interacting with each other through various protocols; likewise, living creatures make connections, communicate and interact with each other through their own standards. And as living creatures respond to the environment in which they live, web forms can also respond to user inputs.

Although a web system as a whole appears to be very lively, it is not "alive". In contrast to the Earth's inhabitants, who live and thus breathe life into the planet, current web forms do not live. Although they do exhibit some living qualities, but being stateless they cannot grow on their own, nor can they reproduce themselves. Consequently, web forms cannot evolve themselves and direct the evaluation of the whole web in a systematic way, as biological living organisms do to the biological world. Table II and Table III summarise the similarities and differences between the biological world and a web, respectively.

TABLE II. SIMILARITIES BETWEEN THE BIOLOGICAL WORLD AND A WEB SYSTEM

Biological world	Web system
Each individual is uniquely identifiable	Each web object can be uniquely identified
Individuals populate the Earth	Web objects populate the web space
Individuals are born and die	Web objects are created and destroyed
Individuals can interact with each other and respond to stimuli	Web objects can interact with each other and respond to user inputs
Individuals live in groups or communities where they can communicate with each other through protocols	Web objects belong to sites or communities, are interlinked on a communication network and can communicate with each other through protocols
Individuals are stateful	Web objects are stateless
Individuals can reproduce, grow, and evolve themselves	Web objects cannot reproduce, grow, and evolve themselves
Growth, change, and evolution happen at individual level	Growth, change, and evolution happen at system level

TABLE III. DIFFERENCES BETWEEN THE BIOLOGICAL WORLD AND A WEB SYSTEM

Biological world	Web system
Individual are stateful	Web objects are stateless
Individuals can reproduce, grow, and evolve themselves	Web objects cannot reproduce, grow, and evolve themselves
Growth, change, and evolution happen at individual level	Growth, change and evolution happen at system level

If a web system can be regarded as an entity like Earth, then it is possible to populate it with species. The class hierarchy in the object-oriented paradigm can be compared to the species hierarchy in biology. To bring life to web forms, it is necessary for them to change, grow and reproduce themselves. Observations of biological life provide very helpful input for projecting life into the software design.

B. Flexible Modular Structure

In nature, link flexibility (or in terms of molecular chemistry, chemical bond stability), is the basis of all life activities. It was also observed that there are several levels of chemical bond stability, which are dependent on the types of the bonds. For instance, single bonds are stronger than double bonds, which in turn, are stronger than triple bonds.

In object-oriented technology, link flexibility is a concept that has been formalised and implemented with such notions as final class in Java [28], or read-only attribute in Sather [29, 30]. Link type, or relationship, however, is still a debatable issue, and does not relate to link flexibility [31]. The most commonly used relationships are association, aggregation (or composition), and generalisation (specialisation, or inheritance) [32, 33]. Even these relationships may be defined differently in differing object systems, and their semantics often does not give enough information to assign a degree of flexibility for the links of its type. Association, for example, may itself be considered a

generic type of many subtypes, such as use, or define. Use, as in the case of "Calculator uses Memory", and define, as in the case of "Genome defines Organism", however, would both be categorised as association, but cannot have the same degree of flexibility because of their semantic differences. While it is possible to remove or replace the Memory instance from a Calculator instance (to make a non-memory calculator or to upgrade memory), it is not possible to remove or replace the Genome instance from an Organism instance (because the removal would make the Organism instance undefined and the replacement would lose its identity).

C. Genes

In order for an object to reproduce, it must "know" its own specifications. In order to grow and evolve, it must be able to alter its own specifications. This suggests a scheme in which the object carries its own specifications, similar to a protein carrying its own genetic code of life (DNA). This scheme may also be relaxed so that the object may have access to its specifications without actually carrying them. In either way, objects can reproduce, change, grow, and evolve by themselves when certain conditions are satisfied or some events occur. In a similar way, living organisms reproduce, grow, and evolve in response to stimuli. Besides, each individual living organism has its own specific genes, but all belonging to the same species share the same genome (set of genes). This suggests the notions of individual gene that is specific to an individual object and group gene that describe a class of objects.

The genetic requirement entails that a "living" object must be defined by, and has access to, its own individual "gene" and group "gene". Intuitively, an individual "gene" is a special element that is embedded in, or linked to, an object. The individual "gene" contains the (compact) core information specific to the object, which is necessary and sufficient for the creation and reproduction of that object. It controls these (creation and reproduction) and other "life" processes (growth and evolution) that take place in the object. A group "gene" defines the common features of similar objects, including the essential functionalities for the objects to perform "life" activities, and plays the role of a framework, model, or schema, for the objects it describes.

Under this view we can see the strong parallels between the object-oriented and the biological worlds. An object is an instance of a class in the same way as a creature is an instance of a genome. What needs further work in object-oriented technology is to design individual "genes", to make the object carry or have access to its "genes", and to ensure that these "genes" are modifiable and transferable across objects of the same class (during reproduction).

D. Meta-Genes

From the genetic point of view, biological speciation happens when individuals from two populations are no longer genetically compatible, that is, the differences between their genes have exceeded a certain limit. We do not know of any scientific study of to what extent such differences will trigger the speciation process, or how to measure these differences. However, it is not impossible to imagine the existence of a meta-gene that controls the process. This assumption stands up well when applied to the meta-modelling concepts described in [32, 34-36] and accepted as the formal basis for information system evolution [32, 35]. Meta-models are models about models, or in data modelling terms, a model is an instance of its meta-model. As meta-models are models themselves, they provide the abstraction and constraints needed to interlink the elements of different data models in a heterogeneous environment and along their evolution. Thus, in the same way that a genome defines an individual, a meta-genome defines a species, a meta-meta-genome (M2-genome) a genus, and so on. Speciation would start when a genome has so diverged that it no longer conforms to the specifications defined by the meta-genome. Similar processes would also happen at higher meta-levels.

The recursive abstraction to higher meta-levels of genes forms a pyramid structure in which the higher the level, the more abstract it is, and the fewer distinct groups it contains. The lowest level at the bottom of the pyramid, corresponding to the concrete level, is non-abstract (specific) and contains not groups but (concrete) individuals. The highest level at the top of the pyramid, corresponding to the union level, is the most abstract, and unifies all groups into one single model. This design, which we call recursive gene, is reflected fairly well in the biological classification system, where individuals are grouped into species, species into genera, genera into families, and so on, until there are only a few kingdoms, and finally one single large group of living organisms.

With regard to the concept of "living" object, assuming a design for "gene" is in place, it would be possible to use meta-modelling technique for the construction of "object meta¬genes" and "object genes" at higher meta-levels. Recursive gene design also generally agrees with studies of meta-modelling. Researchers and practitioners in this field generally accept that four levels of instantiation are needed to control the evolution of an information system [37, 38]. Meta-modelling technology, however, does not have the concept of the union level, which unifies "the whole world". In this sense, our recursive gene design extends the architectures proposed and used in meta-modelling. In our opinion, in practice, the number of necessary meta-levels may not be restricted to four, but depends on the need to link, unify and cooperate elements across multiple heterogeneous and distributed systems.

E. Functionalities

Biological life is characterised by activities such as reproduction, growth, metabolism, and response to stimuli. In addition, an organism must also be born (created), evolve, and die (be destroyed). These activities establish the functional requirements in our design. An "object gene" must define functions for the "living" object to perform the aforementioned life activities. Here, object-orientation natively provides for this requirement by means of (appropriate) methods [33]. A set of genetic functionalities to support the biological life activities aforementioned is shown

in Table IV. These form the basic functionalities that an object gene must define, and a "living" object must perform.

TABLE IV. GENETIC FUNCTIONALITIES AND LIFE ACTIVITIES

Genetic functionalities	Life activities
N/A	Metabolism
N/A	Response to stimuli
Self-construction	Creation
Self-duplication	Reproduction
Self-evolution	Evolution
Self-modification	Growth
Self-destruction	Destruction

Regarding the life activities listed, we make following notes: a) Metabolism and response to stimuli are not supported by life functionalities. This is because these activities are already natural behaviour of computer software. Metabolism, in the software engineering context, is analogous to the total of data processing processes in an object by which inputs (from users or other objects) are transformed into outputs and for use in other software functionalities (including "life" activities). Response to stimuli is the production of output in response to a given input; b) Reproduction and evolution are meaningful only in an interconnecting space in which "living" objects are created, function, interact with each other, and are destroyed; and c) Growth requires another condition, statefulness, as explained below.

F. Statefulness

Growth is the process of changing from one state to another that is considered an increase or development. For an object to grow therefore, it is necessary that it be stateful. Object-oriented technology provides a native way to define object state by a set of attributes and values, and state information can be held inside an object [33]. The current Hypertext Transfer Protocol (HTTP) [39] on which a web system is built, however, is a stateless protocol. The question here is of producing persistent objects, that is, how to maintain object state across web sessions. Assume that we have been able to design an "object gene" and it is compact, as it should, it would be possible for these "object genes" to be wholly distributed and permanently kept at client sites. This would make the web stateful.

G. Evolution

Three essential elements have been identified previously for the evolutionary process to happen: a) A gene pool with a variety of genes and a supply of new genes; b) A driving force to select the fittest gene or set of genes for a given requirement; and c) A population of interconnected and self-reproducing individuals in which the favourable genes can multiply and unfavourable ones become extinct.

Applying these conditions to a web system, provided that suitable designs for "living" objects, "object genes" and "object genes" at higher meta-levels are in place, it can be seen that such system is almost ready as an environment for the evolutionary process to happen. Here, using the analogy between the ecological system and a web system, the driving forces would be users' and developers' requirements. The communication network on which the web is built provides for an interconnected space in which "living" objects, that is, web objects, interact, reproduce, grow and evolve. The population(s) of "living" objects (web objects), each with its own gene, changeable under the aforementioned driving forces, constitutes the gene pool and is the supply for new genes.

H. Species and Speciation

The formation of a new "species" on a web system would happen as a consequence of the evolutionary process. Patterned after biological speciation, and using the same analogy between the ecological system and the web as in the previous section, species and speciation on the web may be conceptually described as follows:

Species on the web, or document species, is a distinct type of web document, with characteristic features pertaining to the community or situation from which it is formed, and which might remain stable over a long period of time. Instances of a species are interoperable and compatible at the genetic level. In terms of our recursive gene design, their genes are defined by meta-genes that conform to each other. Instances of differing species, on the contrary, are genetically non-interoperable and incompatible, and their genes are defined by non-conforming meta-genes.

Speciation on the web is the process of forming a new type of web document that no longer conforms with the type it derives from. This process would start when some extrinsic factor, such as the formation of a new web community with new user requirements, causes the formation of a new population of web objects with slightly different genetic traits. Users' preferences and developers' needs then continue to diverge, leading to a genetic differentiation among the newly formed populations. A new species would form when "living" objects belonging to differentiated populations are no longer interoperable or compatible at the genetic level. In terms of recursive gene design, this means the genome of the new species no longer meta-genetically conforms to the original one from which it was derived.

V. CONCLUSION

In this paper, we show that object-orientation has great potential for designing a system with biological life-like characteristics. The network inherent in a web system establishes an interconnected space, necessary for "living" objects to interact, reproduce, change, grow and evolve. Meta-models permit recursive gene design, an evolutionary architecture for software to evolve. Some new concepts still need to be introduced, and adjustments made to object-oriented design. The many parallels between biological life, object-orientation and the web, however, make it possible for the design of software that can evolve itself in a similar way to biological life forms. The similarities and differences between object-orientation and biological life are summarised in Table V and Table VI.

We are currently working on formalising a software design methodology based on this study then a prototype will be developed to prove our concepts.

TABLE V. SIMIARITIES BETWEEN OBJECT-ORIENTATION AND BIOLOGICAL LIFE

Biological life	Object-orientation
Species hierarchy	Class hierarchy
Species genes	Class specifications
Meta-genes	Meta-model
Molecule	Object
Molecular links	Relationships
Flexible molecular structures	Flexible modular structures
Weak/strong links	Derivable/final classes, or read-write/read-only attributes
Chemical messages	Messages
Cascaded chemical reactions	Cascaded reactions

TABLE VI. DIFFERENCES BETWEEN OBJECT-ORIENTATION AND BIOLOGICAL LIFE

Biological life	Object-orientation
Individual and species genes	Only class specifications
An organism contains its own genes	An object does not contain its own specifications
Genes in organisms are functional	Class specifications are non-functional

REFERENCES

[1] J. Koskinen. (2003, 20 July 2006). Software maintenance costs. Available: http://users.jyu.fi/~koskinen/smcosts.htm

[2] D. Svetinovic and M. Godfrey, "Software and Biological Evolution: Some Common Principles, Mechanisms, and a Definition," 2008.

[3] M. M. Lehman and J. F. Ramil, "An approach to a theory of software evolution," in Proceedings of 4th International Worshop on Principles of Software Evolution, 2003.

[4] M. W. Godfrey and D. M. German, "The Past, Present, and Future of Software Evolution," in 2008 Frontiers of Software Maintenance, Bejing, China, 2008, pp. 129-138.

[5] C. Darwin, The Origin Of Species By Means Of Natural Selection: Penguin Books, 1859.

[6] J. von Neumann, The computer and the brain. New Haven: Yale University Press, 1958.

[7] J. H. Holland, Adaptation in natural and artificial systems. Ann Arbor, MI.: The University of Michigan Press, 1975.

[8] R. Dawkins, The selfish gene: Oxford University Press, 1976.

[9] M. Schrage. (1995, 10 August). Revolutionary Evolutionist. Available: http://www.wired.com/wired/archive/3.07/dawkins.html?pg=1

[10] P. Small, Magical A-Life Avatars - a new paradigm for the Internet. Connecticut: Manning Publications, 1998.

[11] P. Small, Lingo Socery - The Magic of Lists, Objects and Intelligent Agents. West Sussex: John Wiley & Sons, 1996.

[12] R. Khare and A. Rifkin, "The origin of (document) species," Computer Networks and ISDN Systems, vol. 30, pp. 389-397, 1998.

[13] K. Lieberherr. (1996, 20 July 2006). Adaptive Object-Oriented Software - the Demeter method. Available: http://www.ccs.neu.edu/research/demeter/biblio/dem-book.html

[14] S. Clarke and R. J. Walker, "Composition patterns: an approach to designing reusable aspects," in Proceedings of the 23th International conference on Software engineering, 2001, pp. 5-14.

[15] M. Shonle, K. Lieberherr, and A. Shah, "XAspects: an extensible system for domain-specific Aspect languages," in Companion of the 18th annual ACM SIGPLAN conference on Object-Oriented Programming Systems Languages and Applications, 2003, pp. 28-37.

[16] SOP. (2004, 10 August 2004). Subject-oriented programming. Available: http://www.research.ibm.com/sop/

[17] C. Lucas, P. Steyaert, and K. Mens, "Managing software evolution through reuse contracts," in Proceedings of First Euromicro Conference on Software Maintenance and Reengineering, 1997, pp. 165-168.

[18] K. H. Bennett and V. T. Rajlich, "Software maintenance and evolution: a roadmap," in Proceedings of the Conference on The Future of Software Engineering, Limerick Ireland, 2000, pp. 73-87.

[19] M. M. Lehman, "On understanding laws, evolution and conversation in the large program lifecycle," Journal of Software & Systems, vol. 1, pp. 213-221, 1980.

[20] D. H. Andrews and R. J. Kokes, Fundamental Chemistry: John Wiley & Sons Inc., 1962.

[21] B. Alberts, D. Bary, J. Lewis, M. Raff, K. Roberts, and J. D. Watson, Molecular biology of the cell. New York & London: Garland Publishing, Inc., 1983.

[22] D. T. Denhardt, Replication of DNA. Burlington, NC: Scientific Publications Department, Carolina Biological Supply Co., 1983.

[23] A. F. Corcos and F. V. Monaghan, Gregor Mendel's Experiments on plant hybrids: a guided study. NJ: Rutgers University Press, 1993.

[24] J. D. Watson and J. Tooze, The DNA story: Freeman, 1983.

[25] R. J. Ayala, Genetic variation and evolution. Burlington, NC.: Scientific Publications Department, Carolina Biological Supply Co., 1984.

[26] J. M. Smith, The theory of evolution, 3rd ed. Great Britain: University Cambridge Press, 1993.

[27] C. Jeffrey, Biological Nomenclature, 2nd ed. Crane: Russak, 1978.

[28] P. J. Deitel and H. M. Deitel, Java How to Program, Seventh ed. New Jersy: Pearson Education, Inc., 2007.

[29] H. Schmidt and S. Omohundro, "Clos, Eiffel and Sather: A Comparison," ICSI, Berkeley, Technical Report TR-91-047, 1991.

[30] D. Stoutamire and S. Omohundro. (1996, 10 August 2004). Sather 1.1 Specifcations. Available: http://www.icsi.berkeley.edu/~sather/

[31] B. Henderson-Sellers, "Towards the formalization of relationships for object modelling," in Technology of Object-Oriented Languages and Systems: TOOLS 25, Melbourne, 1997, pp. 267-284.

[32] OMG. (2005, 20 October 2012). Unified Modeling Language: Superstructure: version 2.0. Available: http://www.omg.org/cgi-bin/doc?formal/05-07-04

[33] M. Blaha and J. Rumbaugh, Object Oriented Modeling and Design: Pearson Prentice Hall, 2005.

[34] R. Conradi, C. Fernstrom, and A. Fuggetta, A conceptual framework for evolving software processes vol. 18, 1993.

[35] M. Jarke, K. Pohl, K. Weidenhaupt, K. Lyytinen, P. Marttiin, J.-P. Tolvanen, and M. Papasoglou, "Meta Modelling: A Formal Basis for Interoperability and Adaptability," in Information System Interoperability, B. Kramer, M. Papazoglou, and H.-W. Schmidt, Eds., ed: Research Studies Press Ltd, John Wiley & Sons Co., 1998, pp. 229-263.

[36] T. Kühne, "Matters of (Meta-) Modeling," Software and Systems Modeling, vol. 5, pp. 369-385, 2006.

[37] J. Kotteman and B. Konsynsky, "Information Systems Planning and Development: Strategic Postures and Methodologies," Journal of Management Information Systems, vol. 1, pp. 45-63, 1984.

[38] I. I. I. Standards, "Information Technology - Information Resource Dictionary Systems (IRDS) Framework," in ISO/IEC 10027:1990, ed, 1990.

[39] R. Fielding, U. Irvine, J. Gettys, M. J., H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. (1999, 2 February 2009). Hypertext Transfer Protocol -- HTTP/1.1.