A Road Map to Bio-inspired Software Engineering

Abstract
Software production research is rapidly evolving in two parallel approaches: conventional and bio-inspired. Bio-inspired approaches are generally developed and presented as enhancements of conventional ones. However, conventional approaches benefit from their integration with their global context, through software engineering methodologies, for being advantageous. The integration of bio-inspired approaches, with bio-inspired software engineering methodologies, will enrich bio-inspired approaches and let them be irrefutably the best. This paper identifies the motivation of the emergence of such bio-inspired software engineering, presents a first approach to it with a road map, and some of its challenges. The application of this first approach on different software systems categories is presented with its summary evaluation. However, the evaluation on industrial real software scale remains an open challenge.

Keywords
Conventional Software Engineering, Bio-inspired Software Engineering, Bio-inspired software Systems, Automated engineering

1 INTRODUCTION

Software engineering is a technology allowing the production of software from user requirements [15]. A deep gulf separates these requirements, which are at a very high abstraction level (natural), from software product, which is practically at the low abstraction level. The secure move, through this deep gulf, is guided by methodologies. Each *methodology* is defined by a set of steps organized and controlled by *coordination rules*. Each *step* carries out an *activity*. Thus, a methodology is a motherboard of software engineering, supporting coherent integration of techniques that collaborate to produce a software product. Software Methodology has a determinant impact on product quality and cost. The active recent works [3, 4, 7, 8, 10, 16], in software engineering, show clearly some important challenges related to: methodologies, cost and quality, abstractions levels, continuity, variability, autonomy, automation, etc.
Biology, being rich with a variety of high quality production systems, has inspired researchers with alternative techniques in order to face the above challenges. So, bio-inspired techniques emerged and have been in a rapid development in the last two decades. The obtained results were stimulant and lead to development of bio-inspired methods supporting more and more software engineering activities. Unfortunately, these methods are not normalized, their terminology is not sufficiently unified [11], and not integrated with a software engineering methodology. They are just limited to bio-inspired algorithms and applications of those algorithms [2, 17]. Even if these bio-inspired techniques were normalized and integrated in a conventional software engineering methodology, this integration might be inappropriate and incompatible. A

bio-inspired methodology, integrating bio-inspired methods, might be better and, consequently, bio-inspired software engineering might be required.

This paper abstracts and synthesizes such bio-inspired software engineering basics from recent and relevant research works. In the following, paragraph 2 presents evidences to this research motivation from current closest works perspectives. Paragraph 3 presents the proposed bio-inspired software engineering basics. Paragraph 4 presents an evaluation through an application of the above basics to different systems categories. At the end, paragraph 5 presents a conclusion and some bio-inspired software engineering challenges.

2 BIO-INSPIRED SOFTWARE ENGINEERING MOTIVATIONS

Despite the huge research work which is increasingly carried out in bio-inspired computing field, bio-inspired software engineering was not approached. All these works might be generally classified into two categories [17]: *algorithms* (design, improvements, and analysis), and *applications of algorithms*. These algorithms are inspired from bio processes which are produced by bio production engineering. Focusing the research interest only on a product and omitting how this product was produced is a restrictive research method. For this reason, one of the main problems these algorithms have is the lack of a universal platform, and of a proper methodology unifying, abstracting, and integrating them harmoniously.

Another kind of bio-inspired works deal with software modeling. In [11], the authors state the proliferation of bio-inspired systems and the lack of common agreement on definitions and concepts. They present three different views for biological systems: the phylogenetic (mutation), the ontogenetic (growth), and the epigenetic (learning) without any insinuation to the bio engineering behind these views. In [2, 9] the authors state the need of adaptive systems to bio-inspired approaches which may be adapted from existing ones. Others works, like [1, 5, 6, 12, 14, 18] present bio-inspired modeling approaches to specific activities in software engineering (design, implementation, and evolution). They do not deal with the whole software engineering lifecycle.

It is noteworthy that modern emergent systems like software product lines [13], self adaptive systems [9], and continuous software engineering [4] are mainly based on bio-inspired concepts: variability, automaticity, dynamicity, autonomy, and self control, although they do not implement them with bio-inspired approaches. This suggests that the technology of emergent software systems converges to bio-inspired software engineering.

In conclusion, it is easy to state, from the current relevant works, that bio-inspired technology is currently concerned with particular cases. A generalization of this technology from cases to patterns of theses cases (inspired from bio engineering) will increase the reuse of software artifacts, reduce the cost, and increase the quality of software products.

3 MATERIAL AND METHODS

In this section, are abstracted and synthesized some bio-inspired software engineering basic concepts, from recent and relevant research works perspectives. Generally these concepts are important requirements of emerging systems like software product lines [13], self adaptive systems [9], continuous software engineering [4], etc. They include bio-inspired features and methodology.

3.1 Features

A bio-inspired software engineering must support the following required features by new trends in software systems:

- *Variability.* The variability is the stone bed concept of bio-inspired software engineering. It means the availability of any software asset (function, process, methodology, etc) on multi versions. Figure 1 depicts such variability. Variability introduces relations (implication, exclusion, etc) between assets versions. A selection mechanism is needed for creating a coherent software product. This variability is well supported in software product lines and in software configuration management systems. While it is, in conventional software engineering, limited to some elementary assets (functions and data), it is comprehensive (for all assets) in bio-inspired software engineering (methodologies, processes, functions, etc).

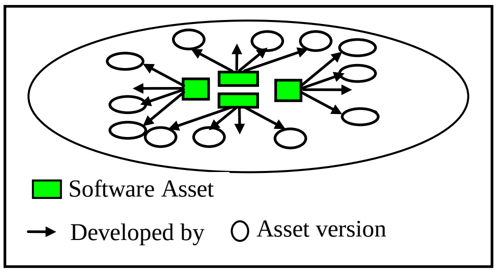

Figure 1. Software Asset variability

- *Individuality.* This feature allows software to automate (by self-decisions) adaptation to new environments and requirements, evolution and maintenance, mutation, and learning. This automation occurs statically and / or dynamically. Several implementation mechanisms are emerging in self adaptive systems.
- *Dynamicity.* This feature allows software, during its run time, to take decisions and carry them out. Several implementation mechanisms are emerging in self adaptive systems and software product lines. While this dynamicity is limited in conventional software engineering, it is comprehensive in bio-inspired software engineering (adaptation, evolution, mutation, and learning).
- *Continuity.* This feature allows self propagation of effects, of any operation on an asset, on all implied ones. This is like pipelines. Some implementations are suggested in continuous software engineering for specific needs.
- *Forward.* This feature allows the production methodology to go always forward and never returns back. In conventional software engineering, the control of a methodology allows going back from one activity to another, because time dimension is not captured. Whereas, in bio-inspired software engineering, *time dimension is very important* and nothing may return back to the past.
- *Meta Engineering.* In bio engineering, production systems are based on variability in engineering patterns. Humans, birds, etc are bio engineering production systems; having a common pattern which is instantiated with different versions of some pattern components. This leads to engineering patterns hierarchy, ending (at leaves) by specific engineering. This is a generalization of meta modeling in conventional software engineering. Figure 2, depicts meta-modeling patterns hierarchy.

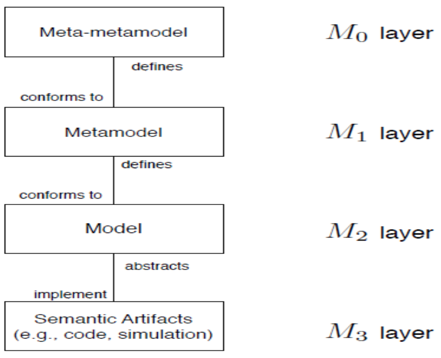

Figure 2. Meta-modeling patterns hierarchy

3.2 Methodology

A bio-inspired software engineering methodology must support at least all the previous software required features. The dominant one is its strong dependency on time dimension, letting it to be only forward. In fact, any bio engineering product springs in an elementary state, grows up through its temporal trajectory, and passes away at the end; never returns back to its past. This is a determinant difference with the conventional software engineering where all its methodologies allow reverse engineering and reengineering of software products. In bio-inspired software engineering, reengineering is part of forward engineering (Figure 3).

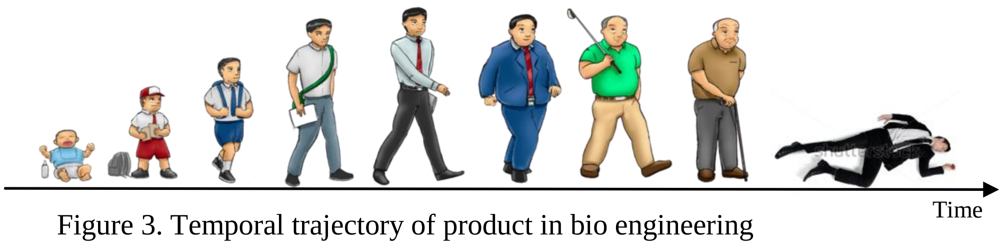

Figure 3. Temporal trajectory of product in bio engineering

While in conventional software engineering, a software product evolves on a single dimension, sequencing mutation, growth, and learning, in bio-inspired software engineering, a software product evolves in a parallel way on three distinct dimensions: phylogenetic (P: mutation), ontogenetic (O: growth), and epigenetic (E: learning), naturally through the time dimension (Figure 4)

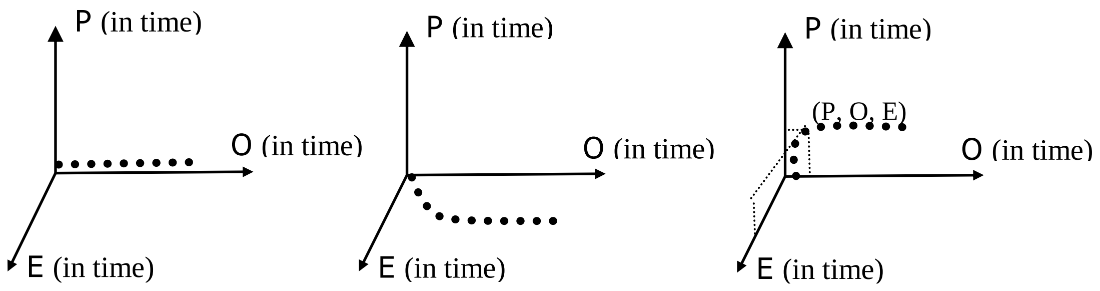

Figure 4. (a) Only growing (b) Growing and learning (c) Growing, learning, and mutation

A bio-inspired software engineering might be based on the following four principles:

Principle 1 (Software rise). Each software system is a release (instance) generated from a configuration defined on a Software Database (Figure 5).
This principle suggests the first three steps in bio-inspired software engineering:

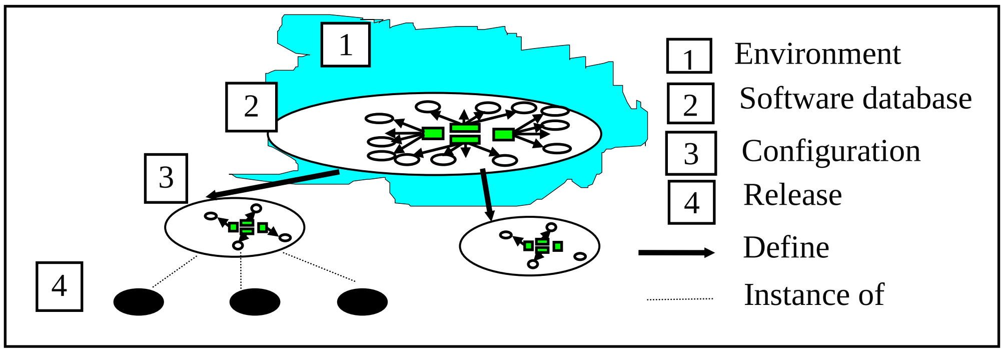

Figure 5. Software rise

1. Software database definition. This huge database might be specialized, in a specific business domain, or general according to patterns abstraction hierarchy supported power. It contains all software assets in multiple versions, their relations, and their coherence control processes. Thus, the methodology supporting product evolution along with its parallel sub methodologies supporting mutation, growth, and learning are also included.
2. Configurations definitions. A configuration is a program selecting versions of assets (one version for any one) composing a software product when executed. These configurations are parts of the software database with their variations, relations, and control processes.
3. Product instantiating (release). A product might be instantiated from a configuration. It holds all the assets selected versions by its configuration. The product is then at its first state (age = 0). According to its environment and to its owned growing, learning, and mutating methodologies the rising product evolves in the time through its evolution space defined by the three dimensions (P, O, E) tracing its trajectory. Like that, at any time, it is possible to answer the following three questions: Where this product is coming from (its past)? What is it actually (its present)? And what is its probable evolution (its probable future)? This provides a rich and precise semantics of software product allowing its automatic understandability and then its self evolution.

Naturally, each software database modification engenders a mutation, which will only affect new products. An aged product might mutate through its dynamic evolution according to the environment effect.
Principle 2 (Software components nature). The nature of software database, configuration, and release is completely operative (Figure 6).
This principle suggests the completely functional nature of any component in the software database, configuration, and instance. Structures are built by structures building functions, they operate and behave through operating and behavioural methods, and the evolution is supported and controlled by methodologies and control processes.

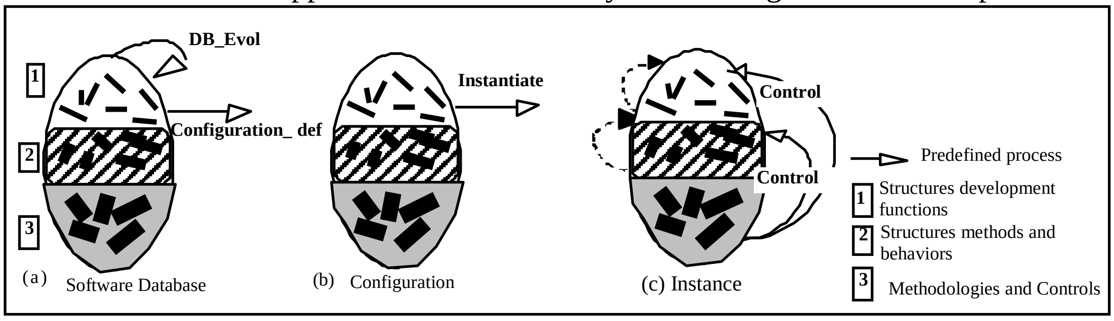

Figure 6. Operative nature of Database, configuration, and instance

Principle 3 (Instance incremental evolution). The structures, functions, and behaviors of an instance are developed incrementally in the time and through the (P, O, E) space.
The encoded evolution methodology along with its sub methodologies (Mutation, Growth, and Learning) ensures that structures, their methods, and their behaviors are developed incrementally by time.

Principle 4 (Control automation). Software database, configuration definition, product instantiation, growth, evolution, and Learning are controlled automatically.
The methodology and controls part of the software database ensure the correct evolution of any instance as it is predefined (statically) and according to the environment effect (dynamically). This allows self control of the evolution trajectory.

4 RESULTS AND DISCUSSION

There are no really close works to compare this work with. A real evaluation necessitates a large adoption of bio-inspired software engineering in the industry, which is far to happen. Then, some aspects were evaluated by specific applications.
Software database modeling for Aspect-oriented systems. In [5], the author presents a bio-inspired aspect-oriented software database modeling approach along with its supporting methodology. This approach has enriched the aspect-oriented paradigm and supporting methodologies with several useful concepts and processes.
Software database modeling for software product lines. In [18 and 19], the authors present a software database bio-inspired model based on features diagrams and supporting variability. A methodology supporting this modeling, configuration definition, and product instantiation is also provided. The bio-inspired approach has lead to enhancements on software product lines modeling methodology: features diagrams, configuration definition, and product instantiation.
Software database modeling for object-oriented systems. In [6], the authors present a bio-inspired object-oriented software database modeling approach along with its supporting methodology. This approach has enriched the object-oriented paradigm and supporting methodologies with several useful concepts and processes. Mainly the relation is-a implementing inheritance was devalued, whereas the value of the relation composed-by, between classes, has been increased.
Self adaptive software methodology. In [12], the authors present a bio-inspired methodology, included in a bio-inspired software database that supports self adaptive systems growth and mutation. This approach has enriched self adaptive systems engineering with a supporting methodology and several useful concepts and processes. Mainly this work is an example of a definition of growing and mutating methodologies inside a bio-inspired database and their effectiveness.
Meta engineering. In [14], the authors present a bio-inspired software meta-modeling approach along with its supporting methodology. The obtained result states clearly the suitability of Features Diagram formalism instead of UML for this kind of meta-modeling and identifies UML possible enhancement that may generalize it to support software variability meta-modeling.
These applications have not only proved the ease of the feasibility of some important aspects of bio-inspired software engineering (software database, configuration, instance, evolution methodology, mutation methodology, growing methodology, self control, meta engineering, etc) but have also induced valuable enhancements in conventional software engineering.

5 CONCLUSION AND CHALLENGES

This paper has outlined motivations to bio-inspired software engineering, proposed some fundamental features and methodological principles, and ended by an evaluation of such bio-inspired software engineering basics with applications in its software database modeling methodologies, its mutation and growing methodologies, and in its meta-engineering. The obtained results have demonstrated the relatively ease of bio-inspired software engineering basics implementation and have a valuable impact on conventional software engineering. However, several important challenges are to be faced.

The Software database is too large and complex. Its suitable model will require more efforts and experimentations. Our experimentations have proved the non relevancy of object-oriented and aspect-oriented modeling paradigms without important enhancements.

The self, dynamic, and continuous (adaptation, mutation, and learning) are so far to be understood and mastered in conventional software engineering. Because these features are inherent to nature, their study in the context of bio-inspired software engineering will lead to valuable end relatively simple solutions.

The meta-engineering patterns development by generalization of conventional meta-modeling techniques will lead to improve software quality and reduce considerably its complexity and understandability.

The actual computer architecture might be enhanced in order to efficiently support such powerful concepts, inspired from bio-engineering. Van Newman computer model was designed for computations below this natural level.

The evaluation at an industrial scale should be a dominant challenge, because it will be the decision maker of the acceptance or reject of such engineering.

6 SCIENTIFIC HONESTY

Funding Information
This work is sponsored, from 12/12/2013 to 12/12/2016, by Philadelphia University, through the research project “Bio-inspired Systems Variability Modeling” at the Bio-inspired System Modeling Research Laboratory

Author’s Contributions
All authors equally contributed in this work.

Ethics
This article is original and contains unpublished material. The corresponding author confirms that no ethical issues involved.

References

[01] Afaghani S, Ghoul S. A Genetic Based Approach for Reducing Null Values In Object-Oriented Database. MISC2010, International Symposium on Modeling and Implementation of Complex systems, 2010.

[02] Bakhouya M. and Gaber J. Bio-inspired Approaches for Engineering Adaptive Systems. Procedia Computer Science 32 (2014) 862 – 869

[03] Cruz S., Fabio Q.B. da Silva F., Capretz L. Forty years of research on personality in software engineering: A mapping study Computers in Human Behavior, Volume 46, May 2015, Pages 94-113

[04] Fitzgerald B, Stol K. Continuous software engineering: A roadmap and agenda Journal of Systems and Software, In Press, Corrected Proof, Available online 4 July 2015

[05] Ghoul S. Supporting AOP by Bio-inspired concepts. IEEE Xplore, 2011 .

[06] Hamoudah D., Ghoul S. A Bio-Inspired Approach to Selective Inheritance Modeling. International Journal Of Software Engineering and Its Applications, ITSEIA, Vol. 8, No. 1, 2014,

[07] Heaton D, Carver J. Claims about the use of software engineering practices in science: A systematic literature review Information and Software Technology, Volume 67, November 2015, Pages 207-219

[08] Jain R., Suman U. A Systematic Literature Review on Global Software Development Life Cycle. ACM SIGSOFT Software Engineering Notes (SIGSOFT) 40(2):1-14, 2015

[09] Krupitzer C. et al. A survey on engineering approaches for self-adaptive systems. Pervasive and Mobile Computing 17, pages184–206, 2015

[10] Lenberg P, Feldt. R, Wallgren L. Behavioral software engineering: A definition and systematic literature review Journal of Systems and Software, Volume 107, September 2015, Pages 15-37

[11] Mili S. and Mealati J. Multi Dimensional Taxonomy of Bio-inspired Systems Based on Model Driven Architecture. The International Arab Journal of Information Technology, Vol. 12, No. 3, May 2015

[12] Nafar E., Ghoul S. A Genetic Methodology for Object Evolution. International Journal Of Software Engineering and Its Applications, ITSEIA, Vol.8, No. 3, 2014.

[13] Sepúlveda S., Cravero A., Cacher C. Requirements Modeling Languages for Software Product Lines: a Systematic Literature Review. Information and Software Technology, In Press, Accepted Manuscript, Available online 7 September 2015

[14] Soltan A, Ghoul S. Modelling Variability in Algorithms Design Methods - Divide and Conquer Case, International Journal of Software Engineering and Its Applications Vol. 9, No. 2 (2015), pp. 47-58.

[15] Sommerville I. Software Engineering, 10/E, Pearson, 2016

[16] Stol K., Goedicke M., Jacobson I. Introduction to the special section — General Theories of Software Engineering: New advances and implications for research Information and Software Technology, In Press, Corrected Proof, Available online 17 August 2015

[17] Yang X. and Cui, Z. Bio-Inspired Computation: Success and Challenges of IJBIC. Int. J. Bio-Inspired Computation, Vol. 6, No. 1, pp.1-6 (2014).

[18] Younes O, Ghoul S. Systems Variability Modeling: A Textual Model Mixing Class and Feature Concepts. International Journal of Computer Science & Information Technology (IJCSIT), Vol. 5, No 5, October 2013.

[19] Younes O., Ghoul S. *A Textual Software Product Lines Design Model By Mixing Class and Feature Concepts.* LAB LAMBERT Academic Publishing, May 20, 2014