An Essay on the Origin of Software Evolution

Paolo Rocchi
IBM
via Shangai 53, 00144 Roma
LUISS UNIVERSITY.
via Salvini 2, 00197 Roma, Italy
procchi@luiss.it

Abstract

The biological domain holds interesting keys to the theorists who investigate the root causes of software maintenance. Several authors believe that software systems need to adapt to changing environment the way biological systems do. The objections raised against this generic comparison induced the author to attend additional lessons in biology.

Living beings exploit three main forms of adaptation: intelligent, specialist and genetic (or Darwinian). Of these, intelligent adaptation appears to be the most appropriate form to be examined in relation to computational phenomena; besides, it fits with the fundamental ideas of Artificial Intelligence.

This study shows how computers are adaptive devices, which aid general systems (companies, production lines, individuals etc.) to have successful behavior in the world. This assumption leads to the inference that the root-causes of software evolution and those of the software itself coincide.

Finally, all the factors that affect software maintenance have been surveyed and a measure to handle the software maintenance processes suggested.

Index Terms — **Nature of software, software changes, intelligent adaptation, information systems, programs classification, software maintenance, management of software projects.**

1. Introduction

Basically, a software program was the solution to a problem for the pioneers in computing. Most of the first programmers translated the immutable logic of mathematical functions into software programs with the notion that the algorithms would not be disproved later. Theorists and practitioners agreed with the static interpretation of software programming, which was influenced by the philosophical ideas of Turing [1].

As time passed, the interpretation of programs as objects that would not be subjected to change entered a crisis. Lehman and Belady [3] censured the partial understanding of software engineering and programming [2], devising a comprehensive map of three principal types of programs:

- The *S-program* addresses a completely defined problem; notably, there are one or more correct solutions to the problem as stated. The developer is concerned not with the correctness of the solution, but with the correctness of the solution's implementation.
- The *P-program* is based on a practical abstraction of the problem it addresses. The developer describes the problem in an abstract way, and writes the specifications of the system's requirements from that view.
- The *E-program* is embedded in the real world; it is an integral part of the world it models and changes as the world does.

Lehman recognized the predominant role of P-type and E-type programs in relation to that of S-type. The S-programs are derivable from logical or mathematical specifications, but in substance they form a minority group [4].

2. Biological Parallel

Lehman also highlights the importance of discovering *'what'* and *'why'* of software evolution. He regrets that this genre of investigations has been limited so far. Lehman is convinced that a deeper insight into the nature of software changes, and a better understanding of the evolving phenomena could lead to improved methods of planning, managing and implementing programs. Maintenance costs continue to be the major component of lifecycle costs despite decades of active research and considerable level of investments [5]. Studies focused on the root-causes of software evolvability could suggest useful solutions to maintenance and reengineering that continue to pose serious problems [6].

Biology offers a variety of suggestions to the issues posed by Lehman. Several theorists attempt to establish some links between biology and the dynamic life of software programs. Pfeffer and others offer, for consideration, a biologically inspired life-cycle that enables service evolution in addition to service adaptation [7]. Ray puts forward the idea of inoculating evolution by natural selection into the medium of the digital computer. Evolution finds forms and processes that naturally exploit the possibilities inherent in the medium [8]. A circle of scientists discusses the evolvability of software system with respect to a set of paradigms: self-organization, modularity, gene duplication, gene robustness, and symbiosis. Ramaswamy identifies some properties for architecting software systems that, in a way, conform to those paradigms [9]. The authors of [10] expand the concept of self-organization. Csete and Doyle discuss the control of biological systems and the computer nets [11].

However, some writers, who find discrepancies between artificial and natural systems [12] [13] [14], call those theoretical approaches into question. Skeptical commentators hold that the evolution of software programs is very much unlike that of living beings, whose existence, persistence, development, and integrity as single individuals are actively maintained by the biological functions of the individuals themselves over a long evolutionary history. In contrast, software systems do not appear intrinsically coherent. That is to say, programs – unlike biological individuals – do not engage actively in their own production, self-maintenance and adaptation to the environment.

For some time, the author has been under the belief that biology forms a convincing metaphor and can enlighten computing. However, the fast evolution of software systems and the aforementioned objections motivated the author toward gaining more insight into the role of biology in software evolvability. Pursuant to this, the author examined what biology teaches and what are the lessons that are most appropriate for programming.

3. Lessons from Biology

Biologists usually relate the concept of ‘evolution’ to some idea of ‘adaptation’ that is defined as ‘adjustment to environmental conditions’. It may be said that adaptation is the core and the archetype of any form of biological evolution. Scientists engaged in investigating living organisms discovered a series of adaptation mechanisms, which, over time, allow an individual to change form or behavior. One can classify those mechanisms into the following principal groups:

(i) Genetic adaptation,
(ii) Specialist adaptation,
(iii) Intelligent adaptation.

(i) The root idea of genetic adaptation comes from Darwin’s theory [15]. The various species of animals and plants have their origin in other pre-existing species and their distinguishable differences are due to *natural selection*.

Natural variations occur among the individuals of a population in a spontaneous and random manner. Many of these differences do not affect survival, but some new traits can improve the chances of survival of individuals. These traits have consistent positive effects upon the vitality of their bearers. Individuals with better traits are more likely to contribute offspring to the next generation, while individuals with worse traits are more likely to die early or fail to reproduce. Natural selection is driven by increased survivorship and/or increased reproductive success of individuals, which have favorable heritable traits. A new special trait is inherited because of genetic modification, and hence the term ‘*genetic adaptation’* is used here.

Natural selection is opposed to *artificial selection*, a process by which animals and plants with traits considered desirable by human breeders are favored for reproduction.

Genetic adaptation is ‘low’, because it crosses several generations, but turns out to be versatile as it can change the profile of individuals considerably.

(ii) Researchers have found an ample set of biological functions that bring the components of a living being into a more effective state. For example, human eyes are capable of adjusting themselves to various levels of darkness and light. Similarly, plants are able to obtain carbon dioxide through the stomata, but they also lose water by evaporation when the pores are open. In desert areas, cacti have the stomata open only at night when it is cool. A specialist adaptation can provide fast or even tardy responses. For example, the human body has both short-term and long-term adaptations to altitude that allow it to partially compensate for the lack of oxygen. In the short term, the lack of oxygen sensed by the carotid artery causes an increase in the breathing rate (hyperventilation). In the long term, the hematologic system increases the hematocrit and other hematic parameters.

Experts relate each mechanism to a single organ, say *visual adaptation*, or to a special event, such as *adaptation to hot environment* or *extreme altitude*, and thus all those mechanisms are qualified with the adjective ‘specialist’. Specialist adaptation mechanisms are also considered as physical reactions by some experts, because the actions are involuntary and do not happen under directions from the brain.

(iii) Intelligent adaptation works under the guidance of the brain, which is exclusive to humans and animals. Intelligent adaptation includes two principal stages: (a) The brain becomes aware of the context, that is to say, it adapts itself to the world. (b) The brain adapts body's behavior to the context. Let us examine them separately.

(iii-a) First, the psychologist Jean Piaget explains the emergence of intelligence in terms of adaptation [16]. Piaget holds that humans desire a state of cognitive equilibrium and reach that state through a mental adaptation process that includes two alternative stages: called *assimilation* and *accommodation*. Assimilation involves incorporating new information into previously existing mental structures; however, when new information does not fit into the existing structures, the individual creates new mental schemas. Accommodation involves the formation of new mental structures or schemas.

The outcome of assimilation and accommodation can be so remarkable as to create persons with far different psychological profiles. Intelligent adaptation shapes the personality of each individual with distinctive qualities and traits of characters.

(iii-b) The brain guides the body of an animal/human to operate in the most appropriate manner and to survive in the habitat.

In conclusion, intelligent adaptation includes the *learning* or *preparatory phase* (iii-a) and the *control, operating phase* (iii-b). The nervous system manages environmental solicitations – environmental factor could be any – during both the stages. Versatility proves to be the fundamental quality required for the intelligence in steps (iii-a) and (iii-b) [17].

Biological mechanism (i) revolves around gene duplication, genetic mutation and symbiosis, which are exclusive to living beings. Biological feedbacks (ii) consist of special chemical reactions and physical processes. Both (i) and (ii) appear rather apart from the software programs that manipulate information, while intelligent adaptation (iii) appears to be much more appropriate for computing.

In the pioneering age, von Neumann traced the first parallel between the human mind and the computer system [18]. During the following decades, several philosophers reverted to this subject [19]. Nowadays, various teams working in fields, such as Artificial Intelligence and Cognitive Informatics, are involved in studying the close relationship extant between intelligence and computing [20]. Lastly, one should mention technicians who optimize the brain-computer interfaces [21].

From these considerations, one can reasonably draw the following conclusion:

Conclusion #1 – *Intelligent adaptation turns out to be the biological mechanism that better fits computing.*

4. Velocity and Versatility

For a better understanding of how intelligent adaptation can be related to computing, it is worth examining more closely the behavior of animals and plants.

Animals – including humans – utilize complex substances (e.g. proteins and fats) that the digestive organs decompose to gain energy, whereas vegetables synthesize elementary substances such as carbon dioxide and water so as to subsist [22]. Animals must take organic molecules from biological beings, because only living entities contain carbohydrates, proteins, etc.; hence, they graze or hunt to feed themselves. In effect, all beasts and men act as serial killers, because they systematically slaughter other living beings. Vegetarians also share this lifestyle since they decimate grasses and plants of life. Once an individual eats a living being to subsist, that individual destroys its source of life and hence must search for another prey out of sheer necessity. Mobility is an intrinsic habit of animals and humans, which are serial killers, and have to cope with mutable circumstances to stay alive. Thus, intelligent adaptation proves to be a necessary function to keep an individual alive in his fast changing habitat. The brain of primitive species rapidly detects a prey; it avoids getting hurt against obstacles and circumvents barriers encountered during movement. The mind of evolved species devises sophisticated strategies.

Vegetables, on the other hand, acquire inorganic molecules from air and soil. They make use of the materials that are abundant in nature and that can be absorbed through leaves and roots. Plants synthesize elementary substances, such as carbon dioxide and water, and create complex molecules to subsist. A plant does not need to walk and therefore clings to a steady state. Vegetation has no concern with the variable world, in the sense that it acts in accordance with the prevailing weather and trends typical of each season.

Table 1 – Synopsis of biological adaptation mechanisms

Adaptation	*Versatility*	*Velocity*
Genetic	Broad	Low
Specialist	Narrow	High/Low
Intelligent	Very Broad	High

Animals, which need to rely on an available food supply, readily detect environmental signals and survive through feedback loops. Plants, on the other hand, do not need complex strategies to subsist. They have no brain and they do not have to be aware of the contextual physical reality.

In conclusion, from the philosophies of life – typical of animals/humans and of plants – it is evident how much the brain is required to provide prompt and flexible answers. The hallmarks of intelligence may all be linked to movement, to variable context and to rapid decision-making [23]. Universal experience shows how an intelligent person is capable of juggling many ideas instantly. A keen individual grasps a chance or skips a hazard in a short while; he/she is able to be involved in various topics of interest.

As a matter of fact, the intelligence quotient (IQ) measures rapid reactions to different challenging queries; one has to pass various IQ tests within a fixed time.

In the current literature, intelligence is variously defined: 'capacity for knowledge and the ability to acquire it'; 'capacity for reason and abstract thought'; 'ability to comprehend relationships'; and 'capacity for original and productive thought'. Philosophers argue upon different aspects of intelligence, but agree on the importance of its versatility and velocity. The faster and the more flexible is the response, the higher is the intelligence of an individual, regardless of the definition one prefers.

Table 1 lists the velocity and flexibility typical of mechanisms (i), (ii) and (iii), and based on these one can easily conclude thus:

Conclusion #2 – *Intelligent adaptation is fast and extremely versatile.*

At this point, intelligent adaptation (iii) should be expressed in more precise terms.

5. Intelligent Adaptation and Programs

The *open system S* (an animal, a man, a woman) is equipped with the *information system IS* (the brain, the nerves and nervous organs) and the *operating system OS* (muscles, glands and other operating organs). The environment *ENV* affects *S*, and, in turn, *IS* modifies the conduct of *OS*. The process (iii-b) can be expressed this way

$$ENV \rightarrow \left[S = (IS \rightarrow OS) \right] \quad (1)$$

The concept of open system, introduced for *natural systems* (animals and humans), is commonly extended to *artificial systems* (robots, industry production lines, etc.) and to *mixed systems* (companies, organizations, institutions, etc.) that include people and devices.

Most authors agree that artificial and mixed systems behave as self-adjusting entities [24]. The seminal work of Stanford Beer [25] presents a systemic interpretation of business. He illustrates some properties of his Viable Systems Model (VSM) in relation to software programming and conveys the idea – now familiar to the experts in the field – that information system is the '*brain of the company*'. John H. Holland, Murray Gell-Mann and others coined the term Complex Adaptive Systems (CAS) to denote a set of self-adaptive agents [26]. CAS is a complex, self-similar collectivity of interacting agents, characterized by a high degree of adaptive capacity, giving them resilience in the face of perturbation. Lastly, one should mention that the concept of adaptation is now widespread in Artificial Intelligence.

Most authors agree on model (1), and from Conclusion #1, one can frame the following definition:

Definition #1 - *The computer system is a mechanical IS or a mechanical component of IS responsible for intelligent adaptation of S to the context ENV.*

Just as the brain adapts human behavior to the context, the digital system adapts or helps to adapt *S* to *ENV*; the system *S* can be an organization, a device or even a single person. The computer has to follow the style of an intelligent agent summed up in Table 1 and this leads to the following conclusion:

Conclusion #3 – *A computer has to provide rapid responses and to become suitable to various situations and needs. Each duty, fulfilled by a digital system, could be required to change promptly as the human brain does.*

Universal experience confirms that computer specialists are frequently approached for modifying the functions of electronic devices in a hurry. Digital computers share the style of intelligent agents and rarely execute rigid, immutable tasks.

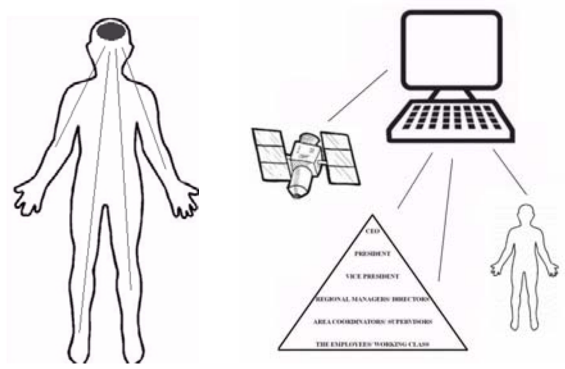
CEO
PRESIDENT
VICE PRESIDENT
REGIONAL MANAGERS/ DIRECTORS
AREA COORDINATORS/ SUPERVISORS
THE EMPLOYEES/ WORKING CLASS

Figure 1 – Natural and artificial intelligent adaptation

Stage (iii-a), that is the learning or acquisition phase, appears demanding for electronic engineers. Setting up hardware circuits requires considerable amount of time and material resources. Because of such considerable obstacles, technicians devised the appropriate solutions to install data processing in a fast manner, and definitely the computer device that executes variable tasks is guided by a programmable unit.

Programmable appliances have been invented long ago. The book [27] describes various machines – musical instruments, textile looms, calculators, etc. – that are controlled by written programs that carry out variable tasks. It is more rapid to write coded lines than to manufacture a wired circuit. Software programs, easier to create and to modify than the hardware components, comply with Conclusion #3; hence, one can formally close with the following one:

Conclusion #4 – *Computer systems are controlled by software programs, because prompt responses are required for an intelligent process.*

The mechanical information system is required to conform to the virtues of the human mind without interruption. Conclusions #3 and #4 are valid over an undefined range of time. This means that a computer system is tuned up not once, but several times; software programs need to be modified over and over again. Rapidly changing environmental requirements call for the adoption of software and its frequent maintenance. Definition #1 and Conclusions #3 and #4 yield that the root-causes of software evolution and the root-causes of the software itself coincide:

Conclusion #5 – *The unexpected and fast phenomena that determine the birth of software programming also give birth to software maintenance.*

Making subsequent changes in a generic program is a long-term process. Results of adaptation accumulate and lead to a sequel of editions of the intended program. This broad phenomenon is usually called '*software evolution*' [28]. The variants of a program are formally called *versions* or *releases* and mark the course of software evolution with precision. The stream of releases is not necessarily linear; sometimes it can form a complex tree; for example, see the distributions of Linux [29].

6. Two cases

Let us examine two real cases, which should make clearer that software programs and human being's survival paradigms are parallel.

A) Mister X is the sales manager of a company that produces beverages. X achieves the operations summed up in the timetable (Table 2).

Mister X (= *IS*) copes with the market (= *ENV*) and elaborates ever new ideas for about a month to make his company (= *OS*) more suitable to the current inclination of consumers. He executes the two steps typical of intelligent adaptation.
(iii-a) - He acquires information [operations #1,2,4,5,7]
(iii-b) - He guides the sellers [operations # 3,6].

The reader can note the two features typical of intelligent adaptation:
Versatility: Mister X examines different sources of information (regular reports, data warehouse and people)
Velocity: Mister X processes information in a few days.

Now let us analyze the software program Y, which achieves operations #2 and #5 using the data warehouse (DW) of the company. Y is written in SQL and the first version Y.1 makes a survey on the behavior of the shopkeepers, that is to say the Select command makes a search on the database tables describing the commercial distribution network. The second version Y.2 makes an inquiry on shopping of private consumers that is to say the program surveys the analytical data of DW. The two versions execute different algorithms, they adopt different programming languages, they pursue different

#	Day	Operations
1	1	X observes that the incomes are declining
2	2	X orders an analysis of the sales
3	10	X suggests up-to-date guidelines to the sellers
4	17	X sees that the sales do not match with the previsions
5	18	X orders a second analysis of the sales
6	22	X discusses with the sellers the best tactics to follow
7	29	X observes that the sales exhibit a positive trend

Table 2

informational scopes and demonstrate the evolution of a software product in an evident manner. The parallelism between the style of the human mind and the software program should emerge in clear terms as the sales manager X and the program Y share an identical goal: successful survival of the company. At last, one can observe that the different responses required by X and scheduled with strict deadlines cannot be obtained either by using a plugged computer or using a computer controlled by firmware. Instead, the software technology enables an expert the rapid updates of Y.

B) The previous case illustrates a software program, which offers support to a man; Y is a part of the overall information system. It may be that a software program is an autonomous *IS*.
Suppose the program M has the full control of a military missile flight. The first release M.a does not handle a certain type of noise and M.b replaces M.a. Later, military experts discover some electronic countermeasures in the wake of a war, thus they rapidly prepare the release M.c to neutralize those measures of the enemy.
The success of the missile flight relies exclusively on the guidance of M that evolves through the releases M.a, M.b and M.c. The program contends the unexpected emerging

factors with success that have been caused by Nature and people.
In conclusion, the program Y cooperates with the sales manages in adapting the company to the new orientation of consumers. The program M is fully responsible for the adaptation of the missile flight against the impediments which it encounters in the air.

7. Determining Factors

ENV never rests, and the *context factors* (CF) that influence the software algorithms cannot be easily catalogued. CFs form an unbounded set; one can randomly quote from this set: trends of the market, new laws of the state, commercial agreements, etc. In addition, there are *determining factors* (DF), which aggravate or mitigate the influence of a CF on software maintenance [30]. The most common DF can be briefly summarized as follows:

|1| Frequent/rare requests for changes,
|2| Predictability/unpredictability of requirements,
|3| Direct/indirect impact of the context on changing the contents,
|4| Short/long deadlines imposed,
|5| High/low economic pressure.

Stimuli coming from the context *ENV* can be more or less severe according to DF. For example, a program can be updated once a month or just once every 5 years (see DF 1). Moreover, the DFs can add to one another. When the above-listed factors overlap, they reinforce each other and make the management of changes to a program very challenging. As an example, suppose unexpected software changes to the program *Z* are suddenly required (see DFs 2 and 4); in addition, *ENV* has a direct effect on the algorithm *Z* (see DF 3); finally, a delay in the release of the updated program brings significant economical losses (see DF 5). All this makes the maintenance of Z somewhat dramatic.

At the other end, one finds software packages whose determining factors are feeble or inconsequential. Almost all mathematical programs belong to this second category. By definition, an '*abstract*' entity has virtual reference to reality or no connection to a specific instance. This implies that context factors do not influence a mathematical algorithm or that the DF's effects on it are anyway feeble owing to the systematic separation between abstract logic and the world. In short:

Conclusion #6 - *Determining factors 1-5 interact in various ways and create a somewhat continuous spectrum that includes all the programs in the world*.

At one extreme, software products are heavily dominated by environmental conditions. Going by Lehman's classification, the programs at this point can be classified as of E-type. In the middle are programs that receive moderate influence from *ENV*, which can be classified as P-type programs. At the other extreme are software packages uninfluenced by the environment, which can be classified as S-type programs. It may be said that *ENV* is very hot at one extreme (far left in Figure 1) and cool at the other (far right in Figure 2).

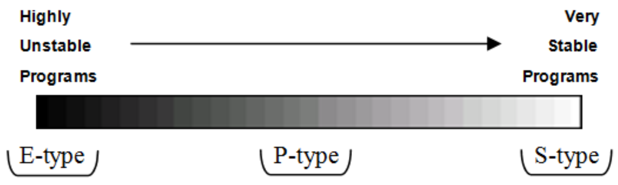

Figure 2 – Complete spectrum of software programs

As the temperature of a physical body depends on the movements of molecules, it is reasonable to define the temperature of *ENV* by means of the temporal frequency of the changes. If N_c is the number of changes requested during Δt, then *Tc* may be considered the *environmental temperature*

$$Tc = \frac{N_c}{\Delta t} \tag{2}$$

The most appropriate interval time (Δt) in (2) is the day, but one can increase it to a larger size. In principle, the physical temperature varies from zero to infinity, and likewise temperature *Tc* too.

The present logical framework suggests a somewhat different approach with respect to Lehman's theory. This paper discusses software maintenance on the basis of external powers, which affect the logic of programs and generates continuous distribution that includes all programs in the world. In contrast, Lehman defines three boxes, and, in a way, forces an expert to place each program in one box or the other. Second, the present approach supports the dynamic vision of programs, which matches with the universal experience:

Conclusion #7 – *Software maintenance depends on DF, thus a program can migrate from one place to another within the spectrum (Figure 2) as a consequence to variable determining factors.*

The event that provokes the movement of a software package along the spectrum can be any. For instance, let us consider the program H, which controls the mechanical process K and which is absolutely stable. After three decades, the technology of K evolves and generates a completely new problem that must be specified anew. Thus, the program H moves from extreme right to some intermediate position in the spectrum. The present theory is able to justify the mutable rhythms of software maintenance because it focuses on the context factors and determining factors, rather than on the program types. In contrast, Lehman finds it more difficult to justify the migration of a program from one type to another [31].

8. Conclusion

After surveying biological adaptation processes, the present paper demonstrates that intelligent adaptation is the mechanism most significant to software domain, not the Darwinian adaptation.

At different stages of presentation, specific conclusions – from #1 to #7 – were deduced through inferential reasoning. Each outcome was pin-pointed in formal terms to show how the concept of intelligent adaptation does not emerge from these pages as a suggestive metaphor for consideration by engineers and biologists, but can become a rigorous reference for the inquiries on program maintenance.

It was demonstrated why the computer has to conform to the dynamic style of human mind, and how, as a logical consequence, it is set up by means of software technology, of necessity. Second, it was proved that the behavior of a computer system needs lifelong adjustments, and this means that the root-causes of software maintenance and the root-causes of the software itself coincide. Finally, it was shown how a variety of contexts and determining factors affect software maintenance, and why expert services are necessitated to manage a variable set of situations. This conclusion matches with Lehman's general criterion, but the present paper basically focuses on the contextual and determining factors whereas Lehman examines the typology of programs.

The present different perspective is not trivial; it yields a precise practical rule for software maintenance leaders who are invited to pay attention not to the class of a program but to the specific CFs and DFs that affect the software package under maintenance.

The author has not developed a mathematical model of the software maintenance; anyway, he suggests a calculation to monitor the factors that affect software changes. The environmental temperature *Tc* provides a guideline to fix priorities, to assign resources, to schedule tasks, etc. in software management. The temperature *Tc* fits with some empirical criteria adopted by maintenance leaders [32].

9. References

[1] Morado R., Hernández-Quiroz F. - Some Assumptions about Problem Solving Representation in Turing's Model of Intelligence - *TripleC* 4(2) 2006: 136-142.

[2] Deek F.P., Turoff M., McHugh J.A. - A Common Model for Problem Solving and Program Development - *IEEE Trans. on Education*, 42(4) 1999: 331-336.

[3] Lehman M.M., Belady L.A. - *Program Evolution, Processes of Software Change* - Academic Press 1985.

[4] Cook S, Ji H., Harrison R. - Dynamic and Static Views of Software Evolution - *Proc. 17th IEEE Intl. Conf. on Software Maintenance* 2001: 592-601.

[5] Bennett K., Vaclav R - Software Maintenance and Evolution: A Roadmap - *Proc. ACM Conf. on the Future of Software Engineering*. 2000: 75-87.

[6] Mens T., Wermelinger M., Ducasse S., Demeyer S., Hirschfeld R., Jazayeri M. - Challenges in Software Evolution - *Proc. Intl. Workshop on Principles of Software Evolution* 2005 13-22.

[7] Pfeffer H., Linner D., Radusch I., Steglich S. - The Bio-inspired Service Life-Cycle: An Overview - *Proc. 3rd Intl. Conf. on Autonomic and Autonomous Systems* 2007: 27-36.

[8] Ray T.S. 1994 Evolving Autonomous Software Agents - *Proc. 3rd IEEE Intl. Workshop on Robot and Human Communication*: 7-11.

[9] Yu L., Ramaswamy S. - Software and Biological Evolvability: A Comparison Using Key Properties - *Proc. 2nd Intl. IEEE Workshop on Software Evolvability* 2006: 82-88.

[10] Cook S., Harrison R., Wernick P. - A Simulation Model of Self-organising Evolvability in Software Systems - *Proc. 1st IEEE Intl. Workshop on Software Evolvability* 2005: 17-22.

[11] Doyle J.C., Csete M. - Architecture, Constraints and Behavior - *Proc. of the National Academies of Science*. 108 (Suppl 3) 2011: 15624-15630.

[12] Nehaniv C.L., Hewitt J., Christiansen B., Wernick P. - What Software Evolution and Biological Evolution Don't Have in Common - *Proc 2nd Intl. IEEE Workshop on Software Evolvability* 2006: 58-65.

[13] Wagner G.P., Altenberg L. - Complex Adaptations and the Evolution of Evolvability - *Evolution,* 50(3) 1996: 967-976.

[14] Ackley D.H. - Real Artificial Life: Where We May Be - *Proc. 7th Intl. Conf. on Artificial Life* 2000: 487-496.

[15] Ward C.H. - *Charles Darwin and the Theory of Evolution* - New Home Library 1943.

[16] Inhelder B., Piaget J. - *The Growth of Logical Thinking from Childhood to Adolescence* - Basic Books 1958.

[17] Grossberg S. - *The Adaptive Brain* - Elsevier/North-Holland 1987.

[18] Neumann von J. - *The Computer and the Brain* - Yale University Press 2012.

[19] Searle J. - *Minds, Brains and Science* - Harvard University Press 1986.

[20] Wang Y. - On the Cognitive Informatics Foundations of Software Engineering - *Proc. 3rd IEEE Intl. Conf. on Cognitive Informatics* 2004: 22-31.

[21] Wolpaw J., Wolpaw E.W. - *Brain-Computer Interfaces: Principles and Practice* - Oxford University Press 2012.

[22] Jain V.K., Sharma J.P. - *Comprehensive Objective Biology* - Laxmi Publications 2005.

[23] Calvin W.H. - The Emergence of Intelligence - *Scientific American Presents,* 9(4) 1998: 44-51.

[24] Principe J.C., Euliano N.R., Lefebvre W.C - *Neural and Adaptive Systems: Fundamentals through Simulations* – Wiley 1999.

[25] Beer S. - *Cybernetics and Management* - English Universities Press 1967.

[26] Miller J.H., Page S.E. - *Complex Adaptive Systems: An Introduction to Computational Models of Social Life* - Princeton University Press 2007.

[27] Rocchi P. 2013 - *Logic of Analog and Digital Machines* – Second Revised Edition, Nova Science Publishers.

[28] Godfrey M.W. - The Past, Present, and Future of Software Evolution - Frontiers of Software Maintenance 2008: 129-138.

[29] Linux Distribution from Wikipedia available on July 2013 at:

http://en.wikipedia.org/wiki/Linux_distribution

[30] Rocchi P. - Intelligent Adaptation and the Nature of the Software Changes - Proc. 7th IEEE Intl. Conf. on Cognitive Informatics - see also in Advances in Cognitive Informatics, Wang Y., Zhang D., Kinsner W. (Eds), Springer Verlag 2008: 59-69.

[31] Cook S., Harrison R., Lehman M., Wernick P. - Evolution in Software Systems: Foundations of the SPE Classification Scheme - *J. of Software Maintenance and Evolution: Research and Practice*, 18 (1) 2005: 1-35.

[32] Xiaowei W. - The Metric System about Software Maintenance - *Proc. of Conf. on Information Technology, Computer Engineering and Management Sciences* 2011: 167-169.

Paolo Rocchi received the degree in physics from University 'la Sapienza' of Rome in 1969, and was associated to the same Institute of Physics as an assistant. In 1971, he joined IBM where he worked as a docent and researcher. He retired from this company in 2010 with the title of Emeritus Docent for his achievements in basic and applied research. Rocchi is also a Contract Professor at University 'Luiss' of Rome.

During his professional career, Rocchi took interest in various scientific domains, including: teaching methods in computer science, computer security, linguistic computing, principles of computing, foundations of the probability calculus, reliability theory, software engineering, and information theory. He received three prizes from IBM for his innovative proposals. Rocchi has written over 100 works including 12 volumes. He is a member of the council of AICA (Ass. Italiana per il Calcolo Automatico) in Roma and a member of various international academic associations.