American Journal of Engineering Research (AJER)
e-ISSN : 2320-0847 p-ISSN : 2320-0936
Volume-03, Issue-05, pp-21-28
www.ajer.org

Research Paper Open Access

Software Evolution: Past, Present and Future

P.I. Okwu[1] and I.N. Onyeje[2]
[1]*Deputy Director, Electronics Development Institute (ELDI) Awka, Nigeria*
[2]*Department of Electrical/Electronic Engineering, Anambra State University, Uli Nigeria*

Abstract: **-** Change, being a fact of life, is inevitable even in software systems. On its own part, software has become both omnipresent and vital in our information-based society which is highly dependent on computers and software. Software need to be updated regularly to ensure preservation and maintenance of their values. There is therefore the need for software to evolve. In this paper, the concept and importance of evolution are explained while emphasis is laid on Lehman's laws and perspectives of software evolution. Also, the relationships and differences between software maintenance and software evolution are brought to the fore. The laws highlighted that a software system must be frequently modified; otherwise it gradually becomes less adequate in use. It is pointed out that software lifecycle undergoes a number of distinct stages. There is a review of various software development models. Despite the challenges facing software evolution, the emerging trends are open source software evolution and unanticipated software evolution.

Keywords: **-** *Lehman's laws of evolution, open source software evolution, software evolution, software maintenance, unanticipated software evolution*

I. INTRODUCTION

The famous Greek philosopher, Heraclitus, stated that change is central to the universe and that the only constant is change. [1][2] His statement implies that there is nothing permanent except change. Being a fact of life, change is therefore inevitable. Consequently, change is everywhere and the main challenge is learning how to handle it effectively. [3]

1.1 Evolution versus Revolution

There are two perspectives of change: evolutionary and revolutionary. [3] Usually evolutionary change occurs gradually and incrementally overtime; it takes place step by step and little by little. On the other the other hand, revolutionary change is fundamental, transformational; results in complete overhaul, renovation and reconstruction and at times irreversible. [3] As a matter of fact, in considering evolution, focus is always on whatever happens to a system over time. An organisation or a product can undergo an evolution.
Organisations may undergo an evolution due to external pressures and do so for effective address of the needs of the stakeholders while product evolution is to keep up with emerging technologies. With respect to change, software is no exception since it is impossible to produce systems of any size which do not need to be changed. [4] The inevitability of change makes it an essential characteristic of software development as software must respond to all environmental pressures and other evolving requirements and so on. [5]

1.2 Software Classification

The ubiquity of computers and hence of software in virtually all aspects of human activity has resulted in the society becoming ever -more software dependent. Software has become vital and universal. [6][7]Usually, software is classified into two: system software and application software. There is still another system of classification. It is known as SPE program classification scheme produced by Lehman and Belady. S represents specified; P stands for problem solving or practical –type and E represents evolutionary or embedded.
Here, S-type programs refer to those that can be specified formally in a set of specifications and whose solution is equally well understood. The stakeholders of these systems understand the problem and thus know what is needed to resolve it.

The P-types are those that cannot be specified and the solution is not immediately known thus employ iterative process in order to find working solution. Thus, the stakeholders of this type know the needed end result, but do not have the means of describing how to obtain it.

The E-type programs are for systems that are actively used and embedded in a real world domain. In fact, E-type software is a model-like reflection of a real world where the criterion for acceptability is satisfaction of stakeholders needs. E-type systems have an intrinsic property which entails that they must be continually changed if they must remain satisfactory in a constantly changing world. [8][9][10][11][12][27]

1.3 Software Evolution

It was in the late 60s that the phenomenon of software evolution was first identified. However, at that time the term evolution was not used until the 70s. [10] The process by which programs are modified and adapted in a changing environment is referred to as software evolution. [12] Software evolution aims at implementing the possible major changes in the system and ensuring its reliability and flexibility. [13] Despite its importance, software evolution has received little attention and the amount of information available to researchers is limited. Even Professor Meir Lehman of Imperial College London carried out an empirical study of software evolution within IBM with the idea of improving the company's effectiveness. Surprisingly, the study received little attention and thus had no expected impact on the company's development practice. [12]

In the course of his study, Lehman identified a number of observations of how software evolves over time and this led to the formulation of the laws of software evolution. Actually, these laws were originally based on observations concerning propriety software such as IBM's OS/360 and OS/370 evolutions. [7] [8][11][12][28]

1.4 Importance of Software Evolution

The successful evolution of software is becoming increasingly critical since the increasing dependence of computers and software at all levels of the society. Regular update of software along with its modernisation, therefore, is very important more so when several organisations invest heavily in their software systems as they consider them to be critical resources for their business. [7]These ensure preservation and maintenance of the values of these business assets. To all companies that are producing and consuming software there is a great need for software to evolve. [10] The relevance of software evolution is on the increase due to the fact that this evolution is inevitable. [14] Furthermore, in understanding and practice of software development, studies in software evolution plays a central role.

1.5 Software Evolution versus software maintenance

Software maintenance and software evolution are related and are at times used interchangeably. However, they are not the same. While software maintenance focuses on fixing of viruses and making minor improvements, software evolution lays emphasis on modification and resettlement. In software evolution process, programs transform their shapes and adjust to the situations in the market. Whereas maintenance is concerned with preservation and fixing of problems, evolution centres on what happens to a system over time and new designs evolving from the old ones. [5][7] Maintenance can be described in terms of constituent activities such as perfective, adaptive, corrective and preventive. [15]

Software evolution is seen as one of the difficult and challenging areas in the field of software engineering having about 60-80% of the cost of the life cycle dedicated to it. It plays an important role in system management. [7] This type of evolution is now widely recognised as a topic that deserves serious study. Software engineering education should include software evolution. Software evolution phenomenon is a topic that is necessary to investigate. [6] Since research in software evolution is very young, it entails continuous change in its focus and basic concepts.

The objectives of this study are to stress the inevitability of software evolution, to differentiate it from software maintenance, to note its centrality in software engineering and to encourage more research in this important field.

This paper is divided into five chapters. As usual, chapter one is the introduction. There is a review of software development, maintenance and evolution in chapter two. Chapter three deals extensively on the theory, processes and perspectives of software evolution. The emerging trends and challenges are handled in chapter four, while chapter five is the conclusion.

II. REVIEW OF SOFTWARE DEVELOPMENT, MAINTENANCE AND EVOLUTION

Usually, during the lifecycle of the software, it passes through a number of distinctive stages: initial development, evolution, servicing, phase out and close down. It is during the initial development that the system develops its first functioning version whereas the evolution stage offers the engineers opportunity to extend the system's capabilities and functionalities. As the software enters the servicing stage, it is subjected to minor

defect repairs. However, at the phase out stage there is no more servicing; while at close down, the software is withdrawn from the market. [15]

2.1 Software Development

Simply put, software development is the planned process of the development of a software product which involves the conception of the desired software through to the final release of the software. Of the several purposes for developing software, the three most common are: meeting specific needs of a specific client/business, meeting perceived need of potential users or for personal use. Also, in developing software one can adopt the engineering-based approach or the incremental approach. No matter the approach, the stages in software development are virtually the same though the order in which they are carried out may vary. [16].

2.1.1 Software Development Process

Also called software development life-cycle and software process, it is simply a structure imposed on the development of a software product. In this process, information systems, models and methodologies that are used to develop these systems are created or altered. [16][17] While developing the software, it will go through such activities as planning, implementation, testing, documentation, deployment and maintenance. Several models are employed in the course of software development and they are meant to streamline the development process.

2.2 Software Development Models

During the development process of software, several development approaches are defined and designed. [19] Actually, a software development model refers to a strategy organised for executing the various steps in the life cycle of a system and is designed to take place in an efficient, repeatable and predictable manner. [18] Some of the common models include: build and fix, water fall, spiral, rapid prototyping, incremental, and iterative development.

2.2.1 Build and Fix Model

This model which works best in a monopolistic or semi-monopolistic environment took its bearing from an earlier and simpler age of hardware product development. [18] In this type, new models are brought out and old models updated and sold without testing. Here, testing is left to the customer to carry out. Of course this approach does not address the product's overall quality and provides no feedback mechanism.

2.2.2 Waterfall Model

Here is a sequence of stages in which the output of each stage becomes the input for the next stage. It was introduced by Win Royce in 1970s and derives its name from the fact that it can be represented in a cascade from defining requirements, to design creation, to program implementation, to system test and to release to customers. [18] The original waterfall model had no feedback paths; feedback was introduced in later versions. Due to complexity issues, this model has been unsatisfactory for software products since it is practically impossible to perfectly address every stage before proceeding to the next.

2.2.3 Spiral Model

This model can be drawn in a spiral form and is visualised as a process passing through four phases in iterations. The phases are: planning, risk analysis, engineering and evaluation and this model is viewed as an enhancement of the waterfall model. It does not depend on ability to assess risks during development accurately. [18] Here, feedback is added to earlier stages just like other improvements of waterfall model. It is good for large and mission-critical projects and can be a costly model and therefore unsuitable for smaller projects.

2.2.4 Rapid Prototyping Model

The activity of producing prototypes of software applications is known as software prototyping. [20] There are two major types of prototyping: throwaway prototyping and evolutionary prototyping. Throwaway prototyping also known as rapid prototyping is the creation of a model that will be eventually discarded rather than becoming part of the final delivered software. In rapid prototyping, a working model of the various parts of the system is created at the early stage after a relatively short investigation.

Among the benefits of rapid prototyping is that the designer and the implementer can get valuable feedback from the users early in the project. Therefore, a very important factor is the speed with which the model is provided. Simply put, the main reason for using Rapid prototyping is that it can be done quickly. The rapidity and iteration involved in the process generates feedback very early and leads to improvement in the final design. [21]

2.2.5 Incremental Model

This model recognises that the whole requirement in the software development steps are not discrete but are instead divided into a number of builds. Here, there are several development cycles with each cycle divided into smaller easily managed modules. Each build goes through the phases of requirements, design, implementation and testing. Usually the first build or the working version is improved upon and functionality is added until it becomes second build; the third build is generated from the second build, and so on. Thus, each subsequent release of the module adds a function to the previous release and the process continues till the complete system is achieved. The advantage of this model is that is that it is flexible enough to respond to critical specific changes as the development progresses. [18][22]

2.2.6 Iterative Development Model

This is also called evolutionary model and maintains a continuous feedback between each stage and the previous one and at times there is a feedback across several stages. It is different from build-and-fix and the original waterfall that operate on open loop.

2.3 Software Maintenance

Simply put, maintenance work refers to any work done to change the software after it is in operation. It is a broad activity which includes the following; error corrections, enhancements of capabilities, deletion of obsolete capabilities and optimisation. The essence of software product modification after delivery is to correct faults and to improve performance or other attributes. Usually, it consumes a large part of the overall lifecycle costs. [23] There are three types of software maintenance: maintenance to repair software faults, maintenance to repair software to a different operating environment and maintenance to add to or modify the system's function. [4]

The impetus for software change comes from dissatisfaction with a current system and the driver of the change is innovation and not preservation. The idea is that there should be a new system that is better adapted to its environment. For instance, there was a rapid growth of software maintenance at the end of the twentieth century as a result of two massive software updates:

- ❖ Set of changes that were needed to support the new unified European currency; that is the Euro. Close to 10% of the total volume of the world software needed to be updated in support of the Euro.[25]
- ❖ The Y2K or year 2000 problem was caused by the use of only two digits for storing calendar dates. For instance, the year 1997was stored as 97. The use 00 for the year 2000 would violate normal sorting rules. The effect was the failure of many software applications and in some cases production of incorrect results. Globally, 75% of the installed software applications were adversely affected by Y2K problem.

The double impact of the Euro conversion work and the Y2K repair work resulted in the engagement of more than 65% of professional software engineering population in various maintenance activities during the period 1999 and 2000. [25]

2.4 Laws of software evolution

These refer to a number of observations of how software evolves over time which Meir Lehman identified in the course of his study of the evolution of software systems. Really, the laws were not presented as laws of nature; instead they are general observations that are expected to hold for all E-type systems irrespective of specific programming or management practices. The eight laws and their years of formulation are summarised in Table1. [7] [8] [13][24][26][27]

Table 1

S/N	Name of Law	Statement/Description	Year
1	Continuity Change	An E-type system must be continuously adapted else it becomes progressively less satisfactory.	1974
2	Increasing Complexity	As software develops, its complexity also increases with it, except when efforts are made to maintain or reduce it.	1974
3	Self-regulation	E-type system evolution process is self regulating with measures of product and process attributes close to normal distribution.	1974
4	Conservation of organisational stability (Invariant work rate)	The normal efficient global activity rate on a developing as well as growing system over the product life time is invariant.	1978
5	Conservation of Familiarity	The average effective global activity rate of an evolving system is invariant over the product life time.	1978
6	Continuing	The functional content of a program must be continually increased	1991

	Growth	to maintain user satisfaction over their lifetime.	
7	Declining Quality	The quality of the software will decline unless steps are taken to keep it in accord with operational changes.	1996
8	Feedback System	E-type evolution processes constitute multi-level, multi-loop. Multi-agent feedback systems and must be treated as such to achieve significant improvement over any reasonable base.	1996

The above laws apply mainly to monolithic, proprietary software. Also, the laws are applicable in large and tailored systems developed by large organisations. Actually, the laws represent an emerging theory of software process and software evolution based on many inputs which include software development. [25]

2.5 Applications of software evolution

According to the Third International, ERCIM (European Research Consortium for Informatics and Mathematics) Symposium on Software Evolution held on 5th October, 2007 in Paris, software evolution is applied in the following areas: software architectures, real-time systems, entrenched systems, web servers and mobile and ambient computing. [7]

III. THEORY, PROCESSES AND PERSPECTIVES OF SOFTWARE EVOLUTION

Now that the society has become ever dependent on computers and consequently on software, it is imperative that real world software be changed and adapted in response to computer usage and changes in the application. Also, the functionalities and behaviours of systems are expected to keep pace with all changes. [10]

3.1 Theory of Software evolution

For the purpose of ensuring reliable planning and management of software evolution, it is essential that a theory of software evolution be established so as to facilitate understanding, refinement and extension. [6] There have been several ways of achieving an empirically grounded theory in science and they are applicable to natural, artificial and hybrid phenomena. The relevant approach to the formulation of software evolution is based on observation, hypothesis and assumptions. [10] The next is to look for patterns, regularities and trends which will provide inputs for the development of hypotheses. As the confidence in hypothesis builds up, relationships are looked for and this will help in developing the seeds of a theory from a collection of observations and inferences. [10]The development of theory follows from here.

In systematic study of software evolution, two aspects are brought to the fore: nounal view and the verbal view. The nounal view of evolution also called evolution as a noun focuses on the nature of evolution, its causes, properties, characteristics, consequences, impact, management, control and exploitation. Thus, it studies the nature of evolution, its process and products. Equally it is called the why and what of software evolution. For instance, it asks such questions as: what evolves? Why does it evolve? What drives, controls, constrains evolution process? [9]

On the other hand, there is the verbal view which is also called evolution as a verb. Any person who adopts this view concerns himself with providing and improving means, processes activities, languages and tools. Also called the how view, it considers implementation of evolution process. It asks the questions what is the goal of evolution activity? Where or which part must be evolved to attain goal. [9][10]

3.2 Software Evolution Process

Recently, software evolution has become a subject of serious academic study due to the increasing strategic value of software. [28] According to Lehman and Ramil, software evolution involves all activities that change a software system. In fact, the activities of evolution of software go on with the preliminary development and also continue after the delivery of the initial version of the system. On the other hand software evolution process refers to all programming activities that are proposed to produce new software version from an earlier operational version. It is the process by which programs transform the shape, adjust the market situations and take over some characteristics fro some pre-existing programs. .[7]

Therefore, in discussing evolution process, the following objectives are borne in mind:

- Explanation of the reason for change being predictable if software structures are useful
- Discussion on maintenance of software, preservation of cost factors and approaches that are used to access revolution strategies for altering software system. [7]

3.3 Perspectives of software evolution

There are different types of software systems and each serves different purposes. In the same vein, the rate and scope of evolution varies depending on the nature of the system. For instance, business systems are

embedded within an organisation. On its own part, evolution can vary both in scope and in frequency depending on the system type. The concept of software evolution is now viewed from several perspectives.

3.3.1 Evolution as a perspective on change

Software maintenance and evolution offer different perspectives on the nature of change. Whereas research on software maintenance addresses practical engineering goals, research on software evolution asks questions of a broader and more scientific nature. For instance, in maintenance, the questions that one may be concerned with are those have to do with next thing to-do, the risk involved and work validation. On the other hand, the researcher on software evolution handles questions on system growth speed, emergence of internal module boundaries and the difference between open source development and industrial source development. [5]

3.3.2 The staged model of the software lifecycle: a new perspective on software evolution

Software evolution is now viewed from the perspective that it is totally different from servicing, phase out and close- down. This perspective is useful in planning and helps to stimulate a set of research issues. Furthermore, there has been an assumption that maintenance phase is uniform over time in terms of activities undertaken, the process and tools used. The new perspective is that maintenance is not single uniform phase; instead, it consists of several distinct stages, with each phase having different technical and business perspective. This perspective is known as staged model. [15]

IV. TRENDS AND CHALLENGES

Nowadays, there is a steady increase of reliance of ICT on software and consequently, virtually all sectors of both our private and public lives are dependent on software. It is therefore expedient that software evolves in order to be able to adapt to the real –world environment and the speed of evolution is a function of the feedback loop structure and other characteristics of the global system. [13]

4.1 Trends

When Lehman made his observations on software evolution, he was not given attention initially. But now, software evolution is gradually gaining attention in the area of research, open software and unanticipated software evolution.

4.1.1Research

Generally, research in software evolution is in infancy stage and thus it is an area for exploration. [30] The whole idea of software evolution research, of course, is to use history of software system to analyse its present state and to predict its future development. For evolution research to be meaningful, it is required that there be accurate data concerning the system being studied. At present, the research is limited by the amount of information available to researchers. [33]

4.1.2 Open source software evolution

Software evolution has led to open source software evolution resulting in the discovery of new ideas that will lead to system improvement. [13] Open source software is the type of software in which users are given licences that allow them to use, modify and redistribute the original source code. With the licence, the user may decide to review the software, add features to it or fix bug or even bring out a new version. Examples of open source software are Linux which is a UNIX-like operating system and Apache open source program which is designed for Web servers. [29]

Recently, the empirical study of open source software evolution has not only become a topic of interest but its development has virtually disseminated throughout the software world. The evolution of open source software is different from that of proprietary or closed source software. [30][31]

4.1.3 Unanticipated software evolution

In evolving software, in spite of the fact that unexpected requirement changes account for most of the technical problems, they are not well supported. [7] As a matter of fact, most of the technical complications and expenses suffered by software evolution result from changes that were not anticipated in the original design. [35] Of course predicting the future is usually difficult and the techniques and technologies that support software evolution are never ideal.

There is now an emerging technology known as Unanticipated Software Evolution (USE). USE technology recognises the fact that unanticipated changes in software systems are inevitable. Therefore, it explores mechanisms in such a way as to accommodate the unanticipated needs in the software systems. This technology is a big asset to software engineers because it works more automatically, dynamically and has a greater assurance of success. [7][35]

To explore the USE technology, the First Workshop on Unanticipated Software Evolution was held on 10-14 June 2002 at Malaga, Spain and was dedicated to towards better support for unanticipated software related runtime infrastructures.[36]

4.2 Challenges

There are several challenges that are associated with software evolution; some are discussed in this paper.

4.2.1 Software quality improvement and preservation

As the software system changes, its quality slowly and steadily degrades as a result of a slow decline in quality. The negative social and economic effects of aging can be overcome by the provision of more tools and techniques which will eventually ensure development of the quality characteristics of a system irrespective of its complexity and size. [7]

4.2.2 Multi- language support

There is an increase in the number of languages that are used in the software systems. In order to tackle this crucial challenge of multi-programming languages, software evolution techniques that support multi-language are provided.[7]

4.3.3 Huge amounts of data analysis

The study of software evolution is challenging due to difficulties in collecting empirical data. Here, for proper manipulation of large quantities of data, new techniques and tools such as data mining techniques used by the database community are considered. [7][34]

V. CONCLUSION

. The characteristic of a software system to easily accommodate changes over time is referred to as software evolution. This feature enables software systems to incorporate new requirements in order to meet business objectives of the stakeholders. [32] In software evolution, emphasis is on preserving and improving software quality and minimizing complexities. The laws of software evolution as proposed by Lehman and his colleagues are now major players in the field of software engineering and computer science. There is the need to increase the awareness of the importance of software evolution since evolution research is still a young field. Greater attention should be given to such areas as open source software evolution and unanticipated software evolution.

REFERENCES

[1] Heraclitus [2014] [Online] Available : http://en.wikiquote.org/wiki/Heraclitus

[2] [2013] The Evolution [Online] Available: http://www.viewpoints-and-perspectives.info/home/perspectives/evolution/

[3] D. Mcnaughton [2014] Evolution vs. Revolution: Do You Know the Difference? [Online] Available: http://organizationevolution.com/evolution-vs-revolution-do-you-know-the-difference/

[4] (2013) Software change [Online] Available: http://ifs.host.cs.st-andrews.ac.uk/Resources/Notes/Evolution/SWChange.pdf

[5] M.W. Godfrey, D.M. German [2013] The Past, Present and Future of Software Evolution [Online] Available: http://libra.msra.cn/Publication/5131328/the-past-present-and-future-of-software-evolution

[6] (2014) SETh- Approach to a Theory of Software Evolution Case for support Part 2: proposed research and context [Online] Available: https://homepages.fhv.at/hv/Semester4/OOAD/seth_p2.pdf

[7] [2013] Software Evolution Process [Online] Available: http://www.ukessays.com/essays/information-systems/software-evolution-process.php

[8] (2013) Lehman's Laws of Software Evolution [Online] Available: http://en.wikipedia.org/wiki/Lehman%27s_laws_of_software_evolution

[9] M.M. Lehman, J.F. Ramil (2001) Software Evolution (Part 1) Towards a Theory of the Phenomenon STRL Annual Distinguished Lecture De Montfort University, Leicester, U.K. 20 Dec 2001 Change [Online] Available:http://www.eis.mdx.ac.uk/staffpages/mml/feast2/papers/pdf/690c.pdf

[10] Society for Design and Process Science (2003) Software Evolution – Background, Theory, Practice, Integrated Design and Process Technology, United States of America, 2003 [Online] Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.98.5547&rep=rep1&type=pdf

[11] (2013) M.M. Lehman Rules and Tools for Software Evolution planning and management [Online] Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.33.6261

[12] (2013) The Evolution of the laws of software evolution. A discussion based on systematic literature review. ACM Computing Surveys, vol.1, no.1, article 1. [Online] Available: http://www.cc.uah.es/drg/jif/2013HerraizRRG_CSUR.pdf

[13] (2013) Software Evolution [Online] Available: http://en.wikipedia.org/wiki/Software_evolution
[14] (2013) T. Mens and J. Klein, Evolving software- Introduction to the special theme [Online] Available:http://ercim-news.ercim.eu/en88/special/evolving-software-introduction-to-the-special-theme
[15] (2013) K. H. Bennett, V. T. Rajlich, The staged model of software lifecycle: a new perspective on software evolution [Online] Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.43.4782&rep=rep1&type=pdf
[16] (2013) Software development [Online] Available: http://en.wikipedia.org/wiki/Software_development
[17] (2013) Software development process [Online] Available: http://en.wikipedia.org/wiki/Software_development_process
[18] Pearson Education, Informit (2013) Software Development Strategies and Life-Cycle Models [Online] Available: http://www.informit.com/articles/article.aspx?p=605374&seqNum=2
[19] ISTQB Guide in testing throughout the testing life cycle (2013) What are the Software Development Life Cycle phases? [Online] Available: http://istqbexamcertification.com/what-are-the-software-development-life-cycle-phases/#.UUxgolFdBGc
[20] (2014) Software prototyping [Online] Available: http://en.wikipedia.org/wiki/Software_prototyping
[21] L. Cerejo, (2014) Design better and faster with rapid prototyping? [Online] Available: http://www.smashingmagazine.com/2010/06/16/design-better-faster-with-rapid-prototyping/
[22] (2014) What is Incremental model- advantages, disadvantages and when to use it? [Online] Available: http://istqbexamcertification.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/
[23] (2013) Software maintenance [Online] Available:http://en.wikipedia.org/wiki/Software_maintenance
[24] (2013) The laws of software evolution [Online] Available: http://www.databorough.com/laws-of-software-evolution.html
[25] (2006) The economics of software maintenance in the twenty first century [Online] Available: http://www.compaid.com/caiinternet/ezine/capersjones-maintenance.pdf
[26] (2014) Laws of software evolution revisited [Online] Available: http://www.rose-hulman.edu/Users/faculty/young/CS-Classes/OldFiles/csse575/Resources/Lehman-LawsRevisited-96.pdf
[27] E. Karch (2014) Lehman's Laws of Software Evolution and the Staged-Model [Online] Available: http://blogs.msdn.com/b/karchworld_identity/archive/2011/04/01/lehman-s-laws-of-software-evolution-and-the-staged-model.aspx
[28] S. Williams (2013) A unified theory of software evolution [Online] Available: http://www.salon.com/2002/04/08/lehman_2/
[29] Microsoft Encarta Encyclopaedia (2008) Open source software
[30] W. Scacchi (2013) Understanding open source software evolution [Online] Available: http://www.ics.uci.edu/~wscacchi/Papers/New/Understanding-OSS-Evolution.pdf
[31] G. Xie, J.Chen, I.Neamtiu (2013) Towards a Better Understanding of Software Evolution: An Empirical Study on Open Source Software [Online] Available: http://www.cs.ucr.edu/~neamtiu/pubs/icsm09xie.pdf
[32] M.A. Chauhan (2014) A Survey of Open Source Software Evolution Studies [Online] Available: http://www.idt.mdh.se/kurser/ct3340/ht10/FinalPapers/5-Chauhan.pdf
[33] R. Robbes and M. Lanza (2014) Change-based Software Evolution [Online] Available: http://users.dcc.uchile.cl/~rrobbes/p/ENTCS-changebased.pdf
[34] C.F Kemerer and S. Slaughter (2014) An empirical approach to studying software evolution [Online] Available: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=799945&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D799945
[35] Wiley InterScience (2005) Unanticipated software evolution J. Softw. Maint. Evol.: Res. Pract.2005; 17 :307–308 [Online] Available: http://onlinelibrary.wiley.com/doi/10.1002/smr.318/pdf
[36] G. Kniesel, J.A.R. Noppen, T. Mens, and J. Buckley (2002) The First International Workshop on Unanticipated Software Evolution [Online] Available: http://doc.utwente.nl/61729/