D Prasad, Dr.Ch.Satyananda Reddy / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 4, Jul-Aug 2013, pp.2295-2299

Understanding the Differences between Proprietary & Free and Open Source Software

D Prasad[1]and Dr.Ch.Satyananda Reddy[2]
1. Department of Computer Science & Engineering, DVR & Dr HS MIC College of Technology, Kanchikacherla,.Andhra Pradesh-521180, India.
2. Department of CS & SE, Andhra University, Visakhapatnam, Andhra Pradesh-530003, India.

ABSTRACT

Proprietary software is the one which is owned by an individual or a company. They are almost always major restrictions on its use, and its source code is always kept secret. Examples include the products delivered by Microsoft, the Windows family of operating systems and Microsoft office. On the other hand Open source software has recently become a major interest in the software industry. Open source software includes free software and public domain software. Open source software generally same as free software is available at no or with limited cost to everyone along with the source code, and it can be used by any one for any purpose and with minimal restrictions. In this paper we would like to focus some of the differences between the Proprietary and Open source software in general terms and in particular in terms of their evolution, verifiability of Lehman laws of software evolution, Style of development, the participants in the development and evolution and the differences in the users' perspective.

I. INTRODUCTION

Proprietary software systems are those developed under the supervision of closed set of people, with defined set of requirements. The term proprietary is derived from the Latin word '*proprietas*' which mean property. There is no concept of freedom in any aspect of the development process. Every activity is guided by the project management team and is closely monitored. Any change or change request in the product shall pass through a series of milestones and reviews to get incorporated into the product. Developers work on man-hours basis for the development. The sense of ownership of the software lies with the organization but not with the developer. The distribution of the software purely lies in the control of the projects top management team. The Model of deployment is the Cathedral in the case of Proprietary software, is developed with limited set of resources provided by the top management. The Users of Proprietary Software are the users who pay for the software. The Proprietary software development is targeted towards fulfillment of the contract with strong discipline. Above all the development is performed secret assuming this secrecy improves reliability of the software with the communication between the developers being face to face. As the Proprietary software development teams involve strong discipline and work under the top management, Quality is ensured by the projects Management.

When it comes to the Open Source Software in which the code is open and available to the users. The concept of Open Source Software is based on the concept of free software foundation. The basis of this concept is reflected in the user's freedom to use, cope, distribute study, make changes, and improves it. More precisely, the concept of free software means four kinds of freedom for the software users [6].

Freedom0: Freedom for the user to run the program for any purpose.
Freedom1: Freedom to study how the program works, as well as the freedom to make it to his use.
Freedom2: Freedom to redistribute copies.
Freedom3: Freedom to improve the program and show to the public as the source code is accessible to the public.

The Open Source Software Development model is based on the Bazaar model where the resources to the development are uncertain and unknown. Apart from the developers, the users, the visitors to the software's web sites are also the participants in this type of software development. The intended target of this sort of development process is to solve the problem but not with the contract as in the case of Proprietary software. The Development is public and all the source code is visible to the public, allowing any one to contribute to the software development. The cooperation between the individuals or the development teams that are sparsely distributed around the world are connected via the internet. Talking about the discipline, it is weak as the developers are not closely located, and there are change requests from different sources. Though the only quality warranty in the case of Open Source Software is the competition among the peer products to provide better features and characteristics in the software they develop, many results have proved that this approach of Open Source Development has many potential advantages.

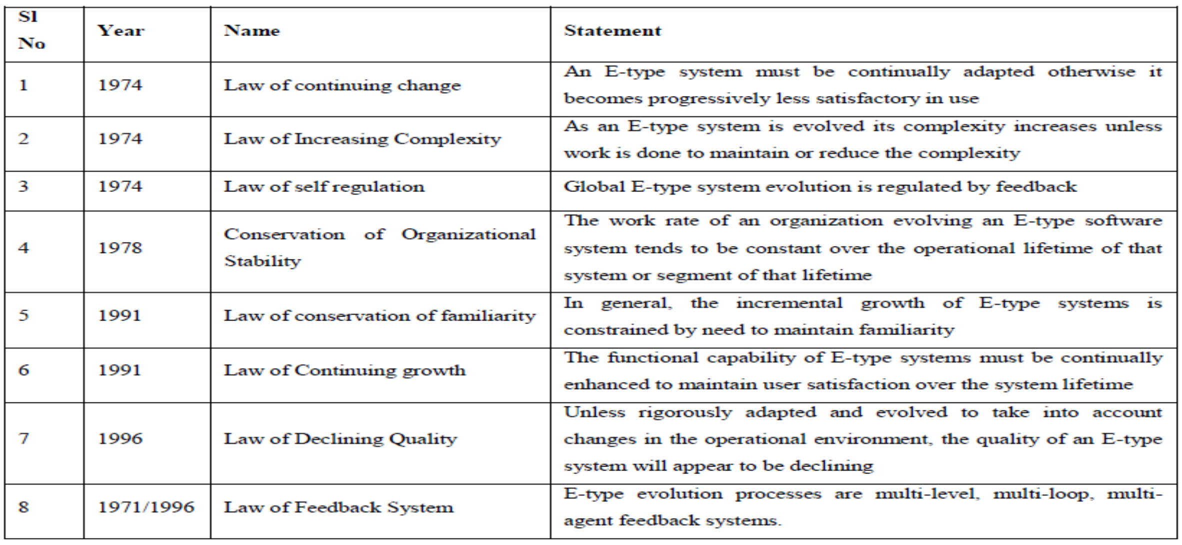
The following sections of the paper discuss the differences between the Proprietary software and Open Source Software with respect to their evolution, verifiability of Lehman's laws of software Evolution, Style of development, participants in the development process and from the perspective of the users in particular.

II. PROPRIETARY SOFTWARE vs. OPEN SOURCE SOFTWARE WITH RESPECT TO EVOLUTION

Software evolution is the phenomenon of software change over years and releases since the first release of the software (Some argue it as from the concept formation) to the decommissioning of the software system. The following discussion briefs out the Software Evolution in the Proprietary context and the open source context with some focus on Lehman's laws of software evolution [11]. Change in the program is the constant entity which is inevitable, which ever the process model we take, which ever the domain we take. The phenomenon of continues change in the program is due to that change to keep the software up-to-date with the changing operational domain. This change is needed for stakeholders' satisfaction to remain at an acceptable level in the ever changing world. Lehman based on his studies classified the systems into three broad categories [11]
S-Type: Specified type Systems.
P- Type: Problem type Systems.
E-Type: Evolutionary type Systems.

In the term S-type systems "S" stands for specified. In S-type systems the specification is complete and can be formally expressed using Mathematics. S- type programs represent the domain within which the application of formal verification methods is more meaningful and likely to be effective. The second classification is the P-Type; here "P" stands for problem type. The P-type problems are usually well-defined and can be formally described. The problems addressing such problems are based on heuristics rather than mathematical proof. They are characterized by some tradeoffs. Examples of P-type systems can be the software for class schedule, train schedule…. The vital classification that Lehman has given is the third type is the E-type systems, where "E" stands for evolutionary. An E-type system is the one for which the problem being addressed cannot be fully defined. E-type systems are always to some degree incomplete and addresses "open" problems. The openness in that the boundaries of the system can change from time to time i.e. the requirements change time to time. The E-type systems always have a perceived need for change and improvement. Another characteristic of E-type system is that the installation of the program in its operational domain changes the domain. The evolution of an E-type system becomes a feedback system. After a series of studies by Lehman on various systems, proposed 8 laws which are called The Laws of Software Evolution. Table 1 shows the summary of the laws proposed by Lehman. The laws were proposed in the days where most of the development went in-house by dedicated closed set of developers working in the same place under some form of hierarchical management control and following a well defined model like the waterfall – like model. The software systems of 70s and 80s were in cases monolithic and there was little reuse from other systems. The present proprietary systems are less monolithic and there are serious alternatives to the waterfall model such as the agile development methodologies.

The laws are difficult to test empirically as they are informal statements. Qualitative simulations and multiagent simulations are promising techniques [1]. But due the unavailability of the data it was difficult to validate the laws.



Sl No	Year	Name	Statement
1	1974	Law of continuing change	An E-type system must be continually adapted otherwise it becomes progressively less satisfactory in use
2	1974	Law of Increasing Complexity	As an E-type system is evolved its complexity increases unless work is done to maintain or reduce the complexity
3	1974	Law of self regulation	Global E-type system evolution is regulated by feedback
4	1978	Conservation of Organizational Stability	The work rate of an organization evolving an E-type software system tends to be constant over the operational lifetime of that system or segment of that lifetime
5	1991	Law of conservation of familiarity	In general, the incremental growth of E-type systems is constrained by need to maintain familiarity
6	1991	Law of Continuing growth	The functional capability of E-type systems must be continually enhanced to maintain user satisfaction over the system lifetime
7	1996	Law of Declining Quality	Unless rigorously adapted and evolved to take into account changes in the operational environment, the quality of an E-type system will appear to be declining
8	1971/1996	Law of Feedback System	E-type evolution processes are multi-level, multi-loop, multi-agent feedback systems.

Table 1: Laws of E-type Software Evolution

When trying to compare proprietary and Free Open Source software, the access to data on proprietary systems is not permitted. But with the wide availability of the data by the emergence of the open source software, it made easier for the research community to validate these laws empirically, and make certain observations. Viz... The growth patterns of Open Source systems is less regular that those of the proprietary systems that were studied in the past.

The evolutionary trends of Open source software are more difficult to predict than those of traditional systems. The users and visitors of the software have variable degree of control over the evolution. With today's collaborative software development methodologies the role of users and other contributors is becoming critical in the evolution of the software. The users of Open Source Software can eventually implement their own features, fix defects or even create and evolve their own version if they need to.

With the large set of empirical data that is available open, by the open source software some of the laws proposed by Lehman need studies for further validity.

Table 2 describes the applicability of the laws of software evolution to some of the successful Open Source Projects.

Sl No	Name	Empirical Support	Statement
1	Law of continuing change	YES	Seems to apply well for Open Source software projects that have achieved maturity
2	Law of Increasing Complexity	Validated for some and not validated for some	There are some systems that show increasing complexity and others of decreasing complexity
3	Law of self regulation	Validated for some and not validated for some	Not clear whether this law applies to OSS or not.
4	Conservation of Organizational Stability	Validated for some and not validated for some	Variable degrees and types of control by small groups of lead developers and how they effect the organization
5	Law of conservation of familiarity	Validated for some and not validated for some	The need to familiarize with new release might be less relevant in Open source software than in proprietary systems
6	Law of Continuing growth	YES	Some successful Open source systems like Linux display superliner growth. Many other OSS display none or little growth
7	Law of Declining Quality	Possibly	This law is difficult to test, as there are no formally defined requirements available to measure the quality of the software
8	Law of Feedback System	YES, but different type of feedback system	This law applies well to the Open Source Software Evolution, but the nature of the feed back in both the cases is different.

Table 2:Applicability of Laws of Evolution to Open Source Software Projects

III. Proprietary Software Vs Open source software- Style of Development

When dealing with the style of development, the organizations choose the model based on the initial set of requirements they possess and the applications of the system. The evolution of the proprietary software evolves in the form of the staged model of the software lifecycle. The key idea contributed here is that the systems tend to go through distinctive phases viz.. Initial development, evolution, servicing, phase-out, and finally close-down where each of the phases involve specific set of management challenges. The models generally followed for proprietary development are "one-shot "models like the waterfall model or iterative models like the spiral model.

While free/Open Source Software is not totally free of constraints… it gives the user the flexibility to do so what they need in order to get work done. At the same time it protects the rights of the author. Now that's freedom. [6] i.e... The user has the flexibility to do necessary changes and customize the software and release it as a new version.

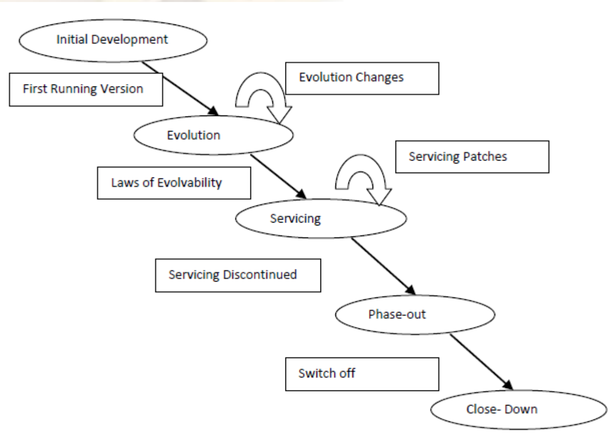


Fig 1: Staged model of the Software Life Cycle

This concept of releasing the constraints on the user to customize the existing version confirms to a life cycle model that may be called "Perpetual Development". The Perpetual development life cycle model comprises the following elements

1. Continuous and steady development of the system, adding new features all the time. Open source projects make this activity of adding new features public. The development is done based on anticipated user needs and explicit user feedback, rather than the specifications of how the system should be used that are predefined or preconceived.
2. Significantly when new functionality is accumulated, the continuous development is interrupted to prepare a major release of a new production version. For example prior to the release of version 2.6 series of the Open Source Operating system Linux the interval between such releases used to be more than a year, this was reduced to 2 to 3 months in 2.6 series.
3. There are more common minor releases of an existing production version, reflecting bug fixes and security patches. This enables production support of several production versions in parallel.

IV. Proprietary software Vs Open Source Software –The Participants

The main participants in the development and evolution to the proprietary software are group of closed set of developers that are known to each other. Any improvement or change to the existing software can be incorporated only after thorough discussion and series of meetings with the project management team. Any request for change will only come through proper channel through various roles in the project management. Incorporating the change requires proper plan and action. This also should take into account the budget considerations from the project management team.

On the other hand Free and open source software projects comprise groups of developers and users that are geographically separated. But these are connected together through shared values and the internet. Though in literature developers are coined out as the main participants to the success of Open Source Software, the users even play an important role and are critical in the innovation of Open Source Software. Another source of contribution is the input from the visitors who visit the web pages of the Open Source Software. Visitors who frequently visit the websites report various problems such as broken links, installation problems, missing features… These requests are treated well by the Open Source Developer Community of the software and start taking up the suggested changes, which facilitates the evolution of the Open Source Software. The contributors that include developers, users, and visitors contribute in different ways to the shared-innovation process.

Further the source code of the Open Source Software itself could be the contributor for the evolution of the open source software. Since the source code is available for free and open to the teams of developers that are sparsely located the developer teams may be inclined to peep into the code and induce their ideas to improve the functionality, reduce the complexity, and reduce the bugs… in the source code. Particularly in the case of reducing the bugs, more people tend to locate and fix more bugs, request and develop more features, creating and environment, where innovations are more likely to occur and higher-quality software generated. Sometimes it appears that the bug is instantly corrected even when before the bug is reported by the users, due to the collaborative nature of the Open Source Software development.

V. Open Source Vs Proprietary Software- The developer and user perspective

From the perspective of the developer the software is an entity that he is developing for the payment that he is receiving. The developers task is complete the code module that the project management team has assigned to him. Many commercials are involved in the additional characteristic that is to be incorporated if any. But this is not the case when dealing with the Open Source Software. The developers, with their interest volunteer the community of the software. Each of the developers investigates the code, the characteristics and modifies the software where ever he feels necessary. The involvement of commercials is limited in the case of Open Source development. Each of the developers has the sense of ownership towards that particular software product. The developers with that sense of ownership realizes the importance of bug fixing and fixes the bugs even when they are not reported by the user community. Each of the collaborative developers sparsely distributed throughout the globe feel equally responsible for the Open Source Product. They share the pride of the projects' success and penetration in the market.

From the users perspective the Closed Source software is only the entity that he uses are the features provided in the software. The user has nothing to do with missing features, bugs reported. The user shall be satisfied with the available features of the product. Any bug identified shall be reported to the development organization and has to wait until it is fixed and the updated version is released. There is also the probability that the user getting locked up by a particular product and invest on recurring cost to upgrade the software product. Where as in the case of Open Source Software the user can be contributor for the refinement of the software. The perpetual development model which the Open source development follows encourages feedback from the user community who are spread throughout the world, the collaborative developer community incorporates the changes and suggestions, release the new version.. This new version with all new characteristics again is validated by the user and this follows a cyclic process. Further the users can incorporate necessary changes, customize the software according to their needs and use it. This feature of Open Source Software has been vital and has drawn attention in number of dimensions.

The Users in Open source software development are treated as one of the primary contributors where as the Proprietary software the user has only a limited role in its development.

VI. Conclusion

The paper discusses some of the key differences between the proprietary and open source software with respect to various parameters. The discussion is limited to the evolution, participants, developers & users, and style of development. These studies even need further investigation and refinement. In addition to the above we would like to further investigate the differences between Proprietary Software and Open Source Software with respect to the usability scenarios, the stability parameter. Today with thousands of Open Source Software projects being announced, judging the best Open Source Software is the major concern. Identifying such measures could help the Open Source Community to take up appropriate decisions to make their projects successful. The Survivability and stability of the Open Source Software is another aspect which needs further study.

References

[1] *Software Evolution – Tom Mens, Serge Demeyer-Springer-2008*

[2] *Three Strategies for Open Source Deployment: Substitution, Innovation and Knowledge Reuse- Jonathan P.Allen: IFIP International Federation for Information Processing*

[3] *Open-Source Vs Proprietary Software Jean-Michel Dalle, Nicolar Jullien*

[4] *The Linux kernel as a case study in software evolution: Ayelet Israeli,Dror G. Feitelson*

[5] *The Past Present and Future of Software Evolution- Michael W.Godfrey, Daniel M. German*

[6] *Open Source Approach in Software Development- Advantages and Disadvantages- Jovica Durkovic, Vuk Vukovic. Lazar Rakovic- Management information systems Vo.3 No.2*

[7] *" On Understanding Laws, Evolution and conservation in the large program Life cycle", Journal of Systems and Software.*

[8] *Bennet. K. H Rajlich, V.T: Software Maintenance and Evoluton: A Roadmap In: The Future of Software Engineering.*

[9] *Raymond. E.S: The Cathedra and the bazaar: musings on Linux and open source by an accidental revolutionary.*

[10] *Software Evolution: An Empirical study of Mozilla Firefox: Anita Ganpati, Dr. Arvind Kalia, Dr. Hardeep Singh. Int. J. Computer Technology & Applications, Vol 3.*

[11] *M.M. Lehman, "Programs, Life Cycles and Laws of Software Evolution", Proceedings of the special Issue of Software Engineering, IEEE, Vol.68, No.9, 1980*

[12] *Has open source software been institutionalized in organizations or not? Josianne Marsan, Guy Pare, Michael D. Wybo*

[13] *From Proprietary to open source-Growing an open source ecosystem Terhi Kilamo, Imed Hammoda, Tommi Mikkonen, Timo Aaltonen- Journal of Systems and Software -2011.*