Open Source Software Evolution And Metric

Sandeep Singh[1], Prithvipal Singh[2]

[1]*Assistant Professor, CSE, G.N.D.U RC Gurdaspur, Punjab*
[2]*Research Scholar, CS, G.N.D.U, Amritsar, Punjab*

---***---

Abstract - *Open Source software evolution (OSS) has becoming widely adopted by Commercial, Public and Academic organizations. OSS has increased in abundance in few years due to the success of well known OSS such as VLC, Mozilla, Linux, Apache etc. This paper presents an analysis of the evolution of various open source software's with the help of SDMetric tool that used to obtain the metric and observe the quality change along the evolution of various versions of OSS. In addition, this paper gives a brief literature survey on OSS evolution and Lehman's seven laws of software evolution. SDMetric tool captures the combined effect of multiple influencing factors such as Coupling between components, Size of packages, Complexity of classes in all in one cohesive model and Translates non-interpretable internal quality data to easily interpreted external quality data in form of Metric Data Tables, Histograms, Kiviat Diagrams.The metric data output shows the evolution affect on various released versions of OSS.*

Key Words: **Software evolution, SDMetric, Lehman's Laws, *Histograms, Kiviat Diagrams*.**

1. INTRODUCTION

Software evolution reflects "a process of progressive change in the attributes of the evolving entity or that of one or more of its constituent elements" [1]. Specifically, software evolution is related to how software systems evolve over time [2]. Computer software can be generally divided into two development models, which are proprietary and OSS. Proprietary is closed software and the source code is not typically made public, and this kind is owned by an individual or company. OSS is the acronym for Open Source Software, which is becoming widely adopted by commercial, public and academic organizations. OSS uses open source codes that are unrestricted and freely available by downloading from the Internet. There are a number of OSSs available, ranging from simple email software to Internet servers such as Apache, full operating systems such as Linux, and Java's type safe nature Guice [3,4].

The evolution of OSS has two shapes, which are the development of the requirement of the application and the maintenance of the code of software. The main aim of this paper is to discuss and focus on the evolution of OSS. It analyses OSS and monitor the evolution using metric technology

2. Software Evolution

Open Source Software (OSS)[10] has becoming widely adopted by commercial, public and academic organisations. Currently, there is increasing interest and demand in the existing applications of OSS in all fields all over the world. OSS has increased in prominence in the last decade, mostly due to the success of well-known software organisations such as Apache, Mozilla, Linux and Guice. As these organisations have become more dependent on software, the effective management of Software Evolution (SE) becomes more critical to an organization's success. OSS firstly evolved throughout the 1970s. Richard Stallman, who is an American software developer, proposed that sharing source codes and ideas is essential to developing a free edition of UNIX. The GNU program was released under the newly created General Public License (GNU GPL).

Controlling SE for huge OSS is a most important challenge in these days. There are many factors that build software are hard to maintain, distributed, and easy modify and explicit project planning. Therefore, the big challenge of OSS is how to evolve its environment, especially the improvement of the Design of these systems. Therefore, the provision of well-evolved OSS has become an urgent issue in these days and will be so in the future. Therefore, the major challenge in OSS is how to evolve its environment, especially improvements in the security and quality of these systems.

Lehman's eight laws of software evolution first formulated in the early 1970s, in Belady and Lehman's study on the evolution of OS/360 [5], these laws essentially characterize the software evolution process as a self-regulating and self-stabilizing system, subject to continuing growth and change [6,7,8]. The laws are named after traits of the software evolution process: "I - Continuing Change", "II - Increasing Complexity", "III - Self Regulation", "IV - Conservation of Organizational Stability","V - Conservation of Familiarity", "VI – Continuing Growth", "VII - Declining Quality", and "VIII – Feedback System".

Some Examples of Open Source Software are: Linux, GNOME, KDE, Apache, Firefox, Dovecot, Postfix, Squirrel mail, Thunderbird, Open office, K office, Asterisk, Free switch, Gnugk.

3. OSS: Related Work

To paint a clearer image of the software evolution process[9], Researchers has performed an empirical study on long spans in the lifetime of seven open source projects. Their analysis covers 653 official releases, and a combined 69 years of evolution. Where they first tried to verify Lehman's laws of software evolution and analyse the growth rate for projects' development and maintenance branches, and the distribution of software changes. They have find similarities in the evolution patterns of the programs they studied, which brings them closer to constructing rigorous models for software evolution. The results indicate that Continuing Change, Increasing Complexity, Self-Regulation, and Continuing Growth are still applicable to the evolution of today's open source software.

For real-world software to remain satisfactory to its stake-holders requires its continual enhancement and adaptation [10]. Acceptance of this phenomenon, termed software evolution, as intrinsic to real world software has led to an increasing interest in disciplined and systematic planning, management and improvement of the evolution process. This paper presents an analysis of the evolution behavior of two small size open source software systems, the Barcode Library and Zlib. Surprisingly, unlike large scale open source software systems, the evolution behavior of these small size open source software systems appears to follow Lehman's laws for software evolution.

The main aim of this paper is to measure the evolution of OSS using, Eclipse Metrics (EM), with Guice software (GS) as a case study. An analysis of OSSE by using Eclipse Metrics (EMs) with Guice Software (GS) as a case study. The study depends on two versions of GS and it will discover the difference between these versions through five EMs, and it also will examine three areas in the GS code, which are: 1) package metrics, 2) type metrics 3) and method metrics. This study found that a number of different concepts on SE drives the OSS industry, such as security, quality, and the reliability of reusability. Therefore, the future of SE should consist of industrial rules for OSS. In addition, some OSSs are still being challenged by closed software because their evolution and development are often faster than those of OSSs[10].

Software systems increasingly require to deal with continuous evolution [11]. In this paper we present the EVOSS tool that has been defined to support the upgrade of free and open source software systems. EVOSS is composed of a simulator and of a fault detector component. The simulator is able to predict failures before they can affect the real system. The fault detector component has been defined to discover inconsistencies in the system configuration model. EVOSS improves the state of the art of current tools, which are able to predict a very limited set of upgrade faults, while they leave a wide range of faults unpredicted. we proposed the EVOSS tool for managing the evolution of free and open source software. A simulator and a fault detector are the main components of EVOSS and they have been defined to predict upgrade failures before they can affect the real system. EVOSS has been experimented in real Linux distribution installations and these experiences show that EVOSS improves the state of the art of package managers.

Source code analysis is important for software management [12]. It enables us recognize strengths and weaknesses in our earlier projects or releases. We developed a source code analysis tool. This tool gathers several metrics from C/C++, C# or Java source codes. In this paper, we will use the tool to analyze some of the open source code projects. We will study the selected projects release evolutions and compare some characteristics between the same project releases, as well as among different projects. Different programming language code and development styles will be studied through those open source projects. SWMetric is a tool we developed to gather metrics on the function and the class level.

While many theoretical arguments against or in favour of open source and closed source software development have been presented [14], the empirical basis for the assessment of arguments and the development of models is still weak. Addressing this research gap, this paper presents the first comprehensive empirical investigation of published vulnerabilities and patches of 17 widely deployed open source and closed source software packages, including operating systems, database systems, web browsers, email clients, and office systems. The empirical analysis uses comprehensive vulnerability data contained in the NIST National Vulnerability Database and a newly compiled data set of vulnerability patches. The results suggest that it is not the particular software development style that determines the severity of vulnerabilities and vendors' patching behaviour, but rather the specific application type and the policy of the particular development community, respectively.

This paper presents an analysis of the evolution of an open source software system, JFreeChart, which is an open source charting library [13], based on its size, fan-in/out coupling, and cohesion metrics. We developed JamTool, a Java Automated Measurement Tool to obtain the metrics and to observe the quality change along the evolution of the twenty-two released versions of JFreeChart. The empirical study clearly indicates that there are positive relations between the number of classes and the fan-in/out coupling, and the added class group has better software quality than the removed class group.

4. Metric

SDMetric tool captures the combined effect of multiple influencing factors such as Coupling between components, Size of packages, Complexity of classes in all in one cohesive model and Translates non-interpretable internal quality data to easily interpreted external quality data in form of Metric Data Tables, Histograms, Kiviat Diagrams. It takes input as output generates by UML software with xmi extension. SDMetric Captures the combined effect of multiple influencing factors in one cohesive model and Translates non-interpretable internal quality data to easily interpreted external quality data

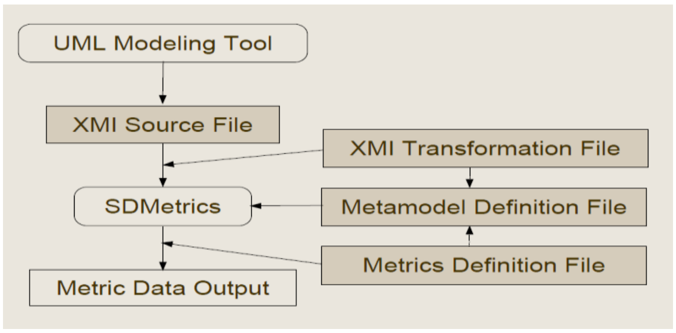
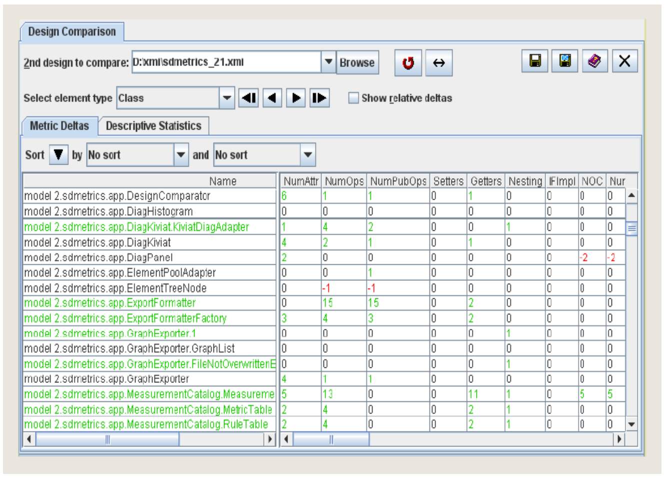


Fig -1: Implementation of SDMetric Tool.

Calculate design metrics for UML designs, such as coupling between components, Size of packages, Complexity of classes, etc. SDMetric tool will show the various views such as Metric, Element, and Table that reflects the whole changes in that software clearly.

Random Result:


Design Comparison
2nd design to compare: D:\xmi\sdmetrics_21.xmi
Browse
Select element type Class
Show relative deltas
Metric Deltas
Descriptive Statistics
Sort by No sort and No sort

Name	NumAttr	NumOps	NumPubOps	Setters	Getters	Nesting	IFImpl	NOC	Nur
model 2.sdmetrics.app.DesignComparator	6	1	1	0	1	0	0	0	0
model 2.sdmetrics.app.DiagHistogram	0	0	0	0	0	0	0	0	0
model 2.sdmetrics.app.DiagKiviat.KiviatDiagAdapter	1	4	2	0	0	1	0	0	0
model 2.sdmetrics.app.DiagKiviat	4	2	1	0	1	0	0	0	0
model 2.sdmetrics.app.DiagPanel	2	0	0	0	0	0	0	-2	-2
model 2.sdmetrics.app.ElementPoolAdapter	0	0	1	0	0	0	0	0	0
model 2.sdmetrics.app.ElementTreeNode	0	-1	-1	0	0	0	0	0	0
model 2.sdmetrics.app.ExportFormatter	0	15	15	0	2	0	0	0	0
model 2.sdmetrics.app.ExportFormatterFactory	3	4	3	0	2	0	0	0	0
model 2.sdmetrics.app.GraphExporter.1	0	0	0	0	0	1	0	0	0
model 2.sdmetrics.app.GraphExporter.GraphList	0	0	0	0	0	0	0	0	0
model 2.sdmetrics.app.GraphExporter.FileNotOverwrittenE	0	0	0	0	0	1	0	0	0
model 2.sdmetrics.app.GraphExporter	4	1	1	0	0	0	0	0	0
model 2.sdmetrics.app.MeasurementCatalog.Measureme	5	13	0	0	11	1	0	5	5
model 2.sdmetrics.app.MeasurementCatalog.MetricTable	2	4	0	0	2	1	0	0	0
model 2.sdmetrics.app.MeasurementCatalog.RuleTable	2	4	0	0	2	1	0	0	0

Fig -2: Metric data values of a class.

5. Proposed work:

We will do an analysis of the evolution of various open source software's with the help of Software Design Metric (SDMetric) Tool that used to obtain the metric values and after that we will observe the quality change along the evolution of various versions of OSS. Tool will captures the combined effect of multiple influencing factors such as Coupling between components, size, and complexity of classes in one cohesive model and translates into metric data table values.

6. CONCLUSIONS

From research work we intend to determine the design of various versions formed due to evolution in Open source software with SDMetric tool. Over 120 design metrics, 130 design rules. It Cover all diagram types of the UML i.e. Users can define new metrics and rules that Works with all UML tools with XMI export (Customizable XMI import) and Batch processing via command line interface with Fast execution that Analyses large designs with hundreds of thousands of model elements within seconds.

REFERENCES

[1] Madhavji, N.H., Fernandez-Ramil, J., and Perry, D.:'Software Evolution and Feedback: Theory and Practice', John Wiley & Sons, 2006.

[2] Yu, L., Ramaswamy, S., and Bush, J.: 'Symbiosis and Software Evolvability', IT Professional, 10, (4), pp. 56-62, 2008.

[3] Guice software, retrieved on 15/3/2009 from http://code.google.com/p/google-Guice.

[4] Y. Wang, et al. Measuring the evolution of open source software systems with their communities. SIGSOFT Softw.Eng. Notes, ACM, New York, USA, vol. 32(6), pp. 7, 2007.

[5] L. A. Belady and M. M. Lehman. A model of large program development. IBM Systems Journal, 15(3):225–252, 1976.

[6] M. Lehman. Laws of Software Evolution Revisited. In European Workshop on Software Process Technology, 1996.

[7] M. Lehman and J. Ramil. Rules and Tools for Software Evolution Planning and Management. Annals of Software Engineering, 11(1):15–44, 2001.

[8] M. M. Lehman, J. F. Ramil, P. D. Wernick, D. E. Perry, and W. M. Turski. Metrics and laws of software evolution – the nineties view. In METRICS '97, pages 20–32, 1997.

[9] S. L. Pfleeger. Software Engineering—Theory and Practice. Prentice Hall,1998.

[10] The Evolution of Open Source Software using Eclipse Metrics Ajlan Al-Ajlan Software Technology Research Laboratory (STRL)De Montfort University The Gateway ,Leicester,LE19BH,K.ajlan@dmu.ca.uk 2009 International Conference on New Trends in Information and Service Science.

[11] Evolution in Open Source Software: A Case Study Michael W. Godfrey and Qiang Tu Software Architecture Group (SWAG) Department of Computer Science, University of Waterloo email: fmigod, qtug@ swag.uwaterloo.ca

[12] A Survey of Open Source Software Evolution Studies Muhammad Aufeef Chauhan M.Sc. Software Engineering Mälardalen University 721 78 Västerås, Sweden.

[13] "Open Source Evolution Analysis" Izzat Alsmadi ,Kenneth Magel 22nd IEEE International Conference on Software Maintenance (ICSM'06) 0-7695-2354-4/06 © 2006.

[14] "The Future of Software Engineering", Anthony Finkelstein (Ed.), ACM Press 2000 Order number is 592000-1, ISBN 1-58113-253-0.