*International Journal of Computer Applications (0975 – 8887)*
*Volume 93 – No. 18, May 2014*

# Applicability of Lehman Laws on Open Source Evolution: A Case study

Taranjeet Kaur
M.Tech. Student,
Dept. of CSE & IT,
RIEIT, Railmajra, Punjab, India

Nisha Ratti
Assistant Professor,
Dept. of CSE & IT,
RIEIT, Railmajra, Punjab, India

Parminder Kaur
Assistant Professor
Dept. of CSE
GNDU,Amritsar, India

## ABSTRACT

Software evolution is an essential characteristic of real world software ,as the user requirements changes, software needs to change otherwise it becomes less useful. In order to be used for a longer time period, software needs to evolve. Software evolution can be a result of software maintenance. An effort is made to find the applicability of Lehman Laws on different releases of two software developed in C++ using object-oriented metrics In this paper, a study has been conducted on 10 versions of Graphics Layout Engine and Flight Gear Simulator evolved over the period of eight years.. The laws of continuous change, growth and complexity are found applicable according to the data collected.

## General Terms

Validation

## Keywords

Software evolution, Lehman Laws of evolution, open source, revisions, object-oriented metrics, complexity.

## 1. INTRODUCTION

In response to changes in the environment or user requirement software continues to evolve after the release of first version. Software evolution is necessity of real world software. In order to be used for a longer time period, software need to evolve otherwise it will become less useful. Software evolution was first addressed by MM Lehman in 1978[1], while studying the software process within IBM. Lehman demonstrated that software continues to evolve over time. Due to changes and growth of software, it becomes complex. It has been more than three decades, since the Lehman's laws were proposed but there are very few empirical studies to support their applicability.

The work reported in this paper is based on the analysis of two open source applications" i.e. GLE and FGS" both developed in object- oriented language, C++. Different releases have been examined to find the applicability of Lehman's laws of evolution on object-oriented software. The work is based on the computation of object oriented metrics proposed by Chidamber et al [2].

The rest of the paper is organized as: section 2 provides the brief background about the Lehman Laws of evolution and the metrics used. Section 3 contains the introduction to two case studies, section 4 contains the data computed, section 5 presents the findings and interpretations on Lehman laws of software evolution, section 6 presents the related works and section 7 discusses conclusions followed by future work.

## 2. BACKGROUND

This section briefly describes the Lehman laws of software evolution and object-oriented metrics used.

### 2.1 Lehman Laws [3]

- Continuing Change (1974)
- Increasing Complexity (1974)
- Self-Regulation (1974)
- Conservation of Organizational Stability (invariant work rate) (1980)
- Conservation of Familiarity (1980)
- Continuing Growth (1980)
- Declining Quality (1996)
- Feedback system(1996)

### 2.2 Metrics Used:

This contains the brief description of metrics used to compute data required for study. A number of object-oriented metrics (CBO, WMC, RFC, DIT, NOC, and LCOM) are proposed by Chidamber et al [2].

•**WMC** (Weighted Methods per Class): The WMC metrics is the sum of the complexities of its methods. Consider a class Ci with methods $M_1$, ---- $M_n$ that are defined in the class, let $C_1$ -----$C_n$ be complexity of methods then WMC= $\sum_{i=1}^{n} C\,i$

•**DIT** (Depth of Inheritance Tree): DIT is used for a class involving multiple inheritance. The DIT will be maximum length from the node to the root of the tree.

•**NOC** (Number of Children): Number of immediate subclasses subordinated to a class in class hierarchy.

•**CBO** (Coupling Between Objects): CBO classes is the count of number of classes coupled .Two classes are coupled when the methods declared in one class are used by methods of other class.

•**RFC** (Response For Class): RFC metrics measured the number of methods being invoked in response to the message received by an object of that class.

•**LCOM** (Lack of Cohesion of Methods): Number of methods in a class that are disjoint with respect to the member of class being accessed by them.

•**LOC** (Lines of Code): LOC includes lines of source code except commented and blank lines.

•**NOM** (Number of Methods): NOM for class defines the number of methods defined in the class.

•**NOV** (Number of variables): NOV is the number of variables used in source code.

•**NOF** (Number of Functions): NOF is the count of the number of functions in the source code.

## 3. INTRODUCTION TO CASE STUDIES

This section introduces two open source software's under consideration namely GLE and FGS, developed in C++. The revisions details and source code is available online [5-8].

### 3.1 Case Study1: Graphics Layout Engine (GLE)

GLE is a graphics scripting language designed for creating publication quality graphs, plots, diagrams, figures and slides. GLE relies on LaTeX for text output. Its output formats include EPS, PS, PDF, JPEG, and PNG. It is developed in C++ language. The source code is available on SourceForge [4]. Since registration on source forge on 11-9-2000, 23 releases are available till date. Out of the 23 releases, we have considered latest 10 releases from 25-12-2007 to 14-03-12. The details of releases of Graphics Layout Engine are given in Table1.

**Table 1. Revision Details of GLE**

| VERSION (GLE) | RELEASE DATE | LOC | NO. OF FUNCTIONS | NO. OF VARIABLES | NO. OF CLASSES |
|---|---|---|---|---|---|
| 4.1.0 | 25/12/2007 | 88115 | 4725 | 12078 | 366 |
| 4.1.1 | 5/1/2008 | 88155 | 4722 | 12076 | 366 |
| 4.1.0 | 9/2/2009 | 88770 | 4724 | 12076 | 366 |
| 4.2.0 | 20/4/2009 | 95293 | 5477 | 13409 | 414 |
| 4.2.1 | 5/9/2009 | 94524 | 9472 | 24570 | 495 |
| 4.2.2 | 6/10/2010 | 94749 | 5510 | 13493 | 417 |
| 4.2.3 | 23/10/2010 | 96943 | 5739 | 13938 | 435 |
| 4.2.4 | 1/1/2012 | 100812 | 6074 | 14630 | 482 |
| 4.2.4b | 14/1/2012 | 100851 | 6076 | 14635 | 482 |
| 4.2.4c | 14/3/2012 | 100741 | 6079 | 14635 | 492 |

### 3.2 Case study 2: Flight Gear Simulator (FGS)

The goal of the Flight Gear project is to create a sophisticated and open flight simulator framework for use in research or academic environments, pilot training, as an industry engineering tool, for DIY-ers to pursue their favorite interesting flight simulation idea, and last but certainly not least as a fun, realistic, and challenging desktop flight simulator. It is developed in C++ language and is and open source. Source code of 20 releases is available online [5]. 10 latest releases of the software are observed. Table 2 gives the revision details of the Flight Gear simulator.

## 4. METHODOLOGY USED FOR DATA COLLECTION

Since software used, are open source, the source code required for study is available online. Various tools are used accordingly to compute different metrics .the revision details of Graphics Layout Engine are available on [6] and Fight Gear Simulator are available on [7].

**Table 2. Revision Details of FGS**

| VERSION (FGS) | RELEASE DATE | LOC | NO. OF FUNCTIONS | NO. OF VARIABLES | NO. OF CLASSES |
|---|---|---|---|---|---|
| 0.9.9 | 18/11/2005 | 70112 | 3291 | 9344 | 182 |
| 1.0.0 | 15/12/2007 | 80326 | 3705 | 10632 | 216 |
| 1.9.0 | 20/12/2008 | 85413 | 3395 | 10974 | 207 |
| 1.9.1 | 26/1/2009 | 85413 | 3418 | 10974 | 207 |
| 2.0.0 | 17/2/2010 | 88000 | 3420 | 11375 | 213 |
| 2.4.0 | 16/8/2011 | 91883 | 3681 | 11896 | 217 |
| 2.6.0 | 17/2/2012 | 99060 | 3879 | 12896 | 242 |
| 2.8.0 | 16/8/2012 | 99043 | 4137 | 14102 | 258 |
| 2.10.0 | 18/2/2013 | 103771 | 5436 | 21402 | 494 |
| 2.12.0 | 16/9/2013 | 108772 | 5919 | 23734 | 550 |

## 5. OBSERVATIONS AND ANALYSIS OF SOFTWARE EVOLUTION

In this section the evolution of two software is observed to find the applicability of Lehman laws on the bases of computed data.

### 5.1 Law 1: Continuing Change:

Law of continuing change states that in order to use the software for longer period , it should change continuously according the user and environment needs. The change can be due to some bug fixing activity or the change can be due to addition of new function or class to the software.

In case of GLE and FGS, the size and functionalities are changed in each successive release of the software. In FGS the number of functions and classes are increased in each successive release except in release 1.9.1 as it was a bug fix release. In GLE, the number of functions and number of classes are changed in each successive release .The law of continuing change is reflected by both the software.

### 5.2 Law 6: Continuing growth:

Law of continuous growth states that the software should grows continuously in order to satisfy the requirements of user. The growth of the software can be measured in terms of its size and functionality. LOC of each version is computed to analyze the growth in terms of size and computed number of classes and number of functions to observe the growth in terms of functionality.

#### *5.2.1 Size Metrics:*

The LOC growth for GLE and FGS is showed in Fig 1&2. The growth of lines of code is the measure of lines in source code. LOC curve for FGS showed the increase in LOC in each release except in 2.8.0 (figure 2), few lines were removed may be in response to the bug fix activity. LOC graph GLE showed the increase in lines of code. In release 4.2.0 (figure 1), the increase was sharp, which further decreases in next version. But in later versions, it again shows the increase in LOC. So the law of continuing growth is reflected in terms of size metrics.

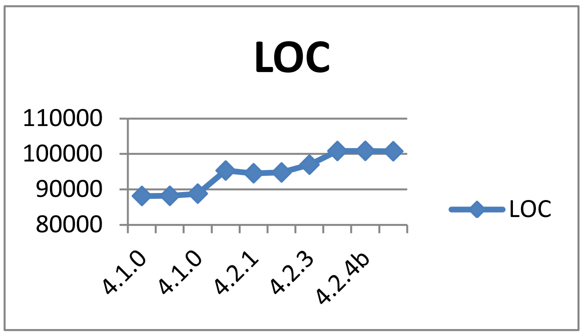


**Fig1: Growth curve of LOC for GLE**

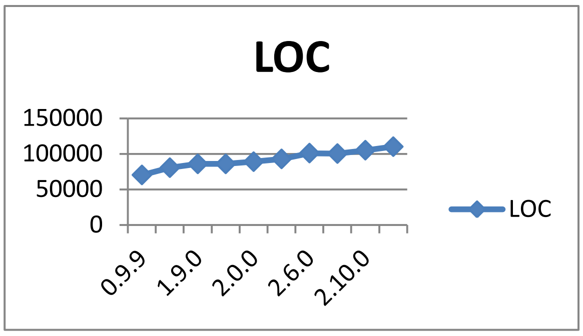


**Fig2: Growth curve of LOC for FGS**

### *5.2.2 Function Metrics*

The growth of the software in terms of functionality can be measured as number of classes and number of functions. Functionality of software can be increased by addition of class or functions figure 3&4 shows the number of classes for different releases of GLE and FGS and figure 5&6 represent the number functions in different releases of GLE and FGS. In case of the FGS, the number of classes and functions are increased in each successive release of a software. In case of GLE, the number of classes and functions are increased in linear fashion from version 4.1.0 to 4.1.2. In version 4.2.0, the number of functions and classes are increased sharply but in next version these are decreased. The later versions showed the increase in the number of classes and number of functions.

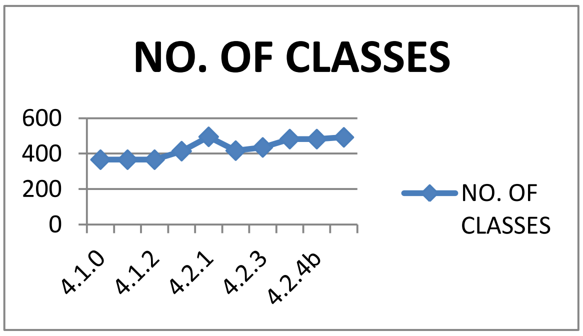


**Fig3: Growth curve of number of classes for GLE**

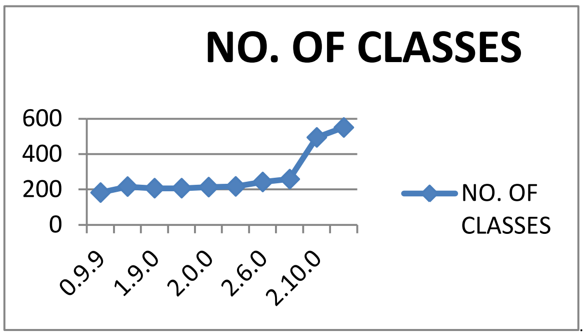


**Fig4: Growth curve of number of classes for FGS**

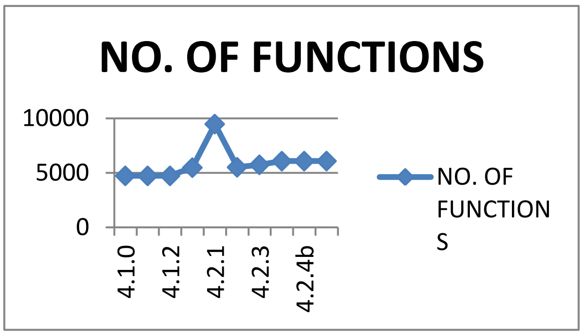


**Fig5: Growth curve of number of functions for GLE**

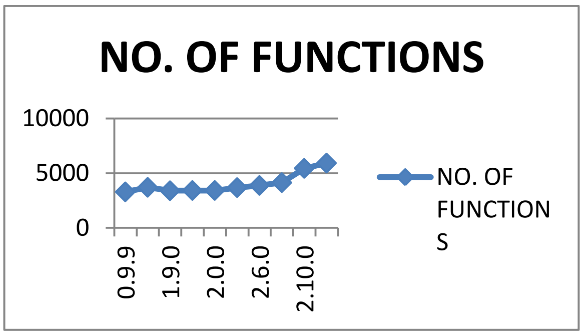


**Fig6: Growth curve of number of functions for FGS**

**Table 3 Object-oriented Metrics computed for GLE**

| VERSION (GLE) | Mc Cabe Complexity | DIT | NOC | CBO | RFC | WM | LOCH |
|---|---|---|---|---|---|---|---|
| 4.1.0 | 8900 | 104 | 76 | 423 | 3461 | 3547 | 18128 |
| 4.1.1 | 8898 | 104 | 64 | 433 | 3487 | 3493 | 17997 |
| 4.1.0 | 8908 | 104 | 80 | 424 | 3716 | 3902 | 18197 |
| 4.2.0 | 10061 | 148 | 98 | 482 | 5138 | 5145 | 28598 |
| 4.2.1 | 9905 | 147 | 102 | 990 | 6189 | 5835 | 32648 |
| 4.2.2 | 9942 | 149 | 102 | 575 | 4777 | 4943 | 27524 |
| 4.2.3 | 10187 | 178 | 114 | 724 | 5419 | 5494 | 24518 |
| 4.2.4 | 10692 | 224 | 146 | 886 | 6385 | 6369 | 30921 |
| 4.2.4b | 10702 | 234 | 146 | 897 | 6431 | 6120 | 30656 |
| 4.2.4c | 10698 | 210 | 143 | 976 | 6446 | 6326 | 29894 |

**Table 4 Object-oriented Metrics computed for FGS**

| VERSION (FGS) | Mc Cabe Complexity | DIT | NOC | CBO | RFC | WMC | LOCH |
|---|---|---|---|---|---|---|---|
| 0.9.9 | 4766 | 86 | 57 | 347 | 3905 | 4139 | 44121 |
| 1.0.0 | 5532 | 113 | 75 | 433 | 4295 | 5162 | 45751 |
| 1.9.0 | 6000 | 120 | 82 | 545 | 4298 | 5245 | 39941 |
| 1.9.1 | 6000 | 120 | 73 | 558 | 4233 | 5258 | 40489 |
| 2.0.0 | 6240 | 131 | 90 | 558 | 4243 | 5746 | 41868 |
| 2.4.0 | 6401 | 150 | 87 | 522 | 4241 | 5873 | 43023 |
| 2.6.0 | 6692 | 160 | 92 | 572 | 4246 | 5982 | 41652 |
| 2.8.0 | 6774 | 162 | 90 | 569 | 4647 | 6004 | 44878 |
| 2.10.0 | 6718 | 162 | 90 | 688 | 4796 | 5948 | 41843 |
| 2.12.0 | 7051 | 213 | 119 | 689 | 5381 | 5782 | 40269 |

## 5.3 Law 2: Increasing Complexity:

Law of increasing complexity states that as the software grows, its complexity increases unless certain measures are taken to keep the complexity under check. Complexity may arise due to changes or addition of functions. The object-oriented metrics is used for object-oriented software to determine the complexity, CBO, RFC, WMC ,DIT, LOCH metrics are used to determine the complexity of GLE and FGS, represented in Table 3&4 .WMC for GLE and FGS is shown by fig 7&8. The FGS shows the increase in WMC in first 8 versions but in later two versions it shows the decrease, this can be due to some bug fixing activity .In case of GLE the WMC showed the increasing trend in first in 4 versions but in 5th version it increased sharply, which were decreased in next successive version. In later versions, the WMC are increased, again in version 4.2.4b, the WMC are decreased, due to bug fix activity.

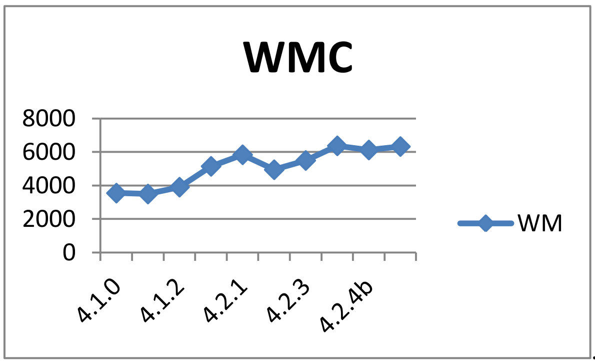


**Fig7: Growth curve for the average weighted methods for different releases of GLE**

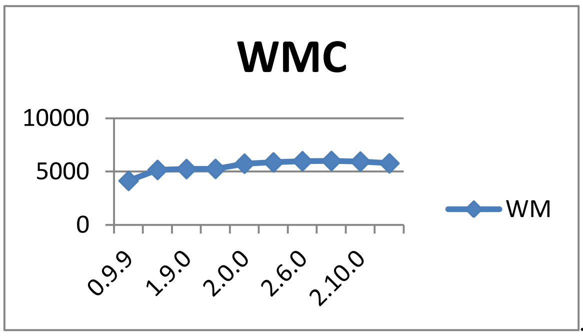


**Fig8: Growth curve for the average weighted methods for different releases of FGS**

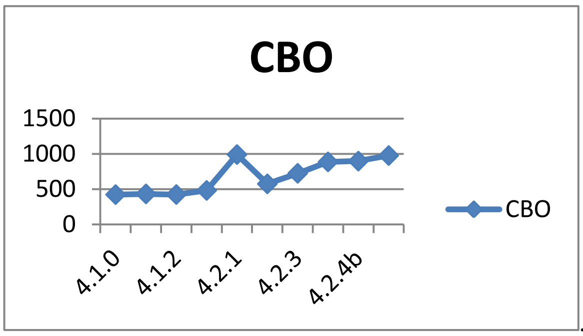


**Fig9: Growth curve for average coupling of objects of all classes for different releases of GLE**

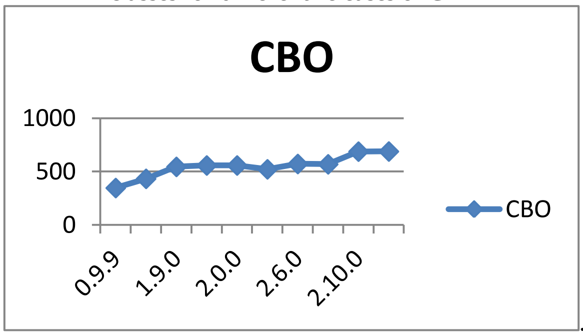


**Fig10: Growth curve for average coupling of objects of all classes for different releases of FGS**

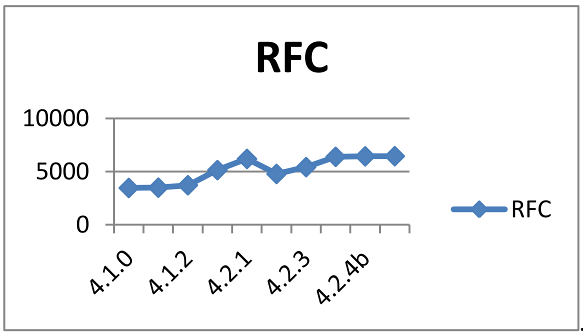


**Fig11: Growth curve for the response for all the classes for different releases of GLE**

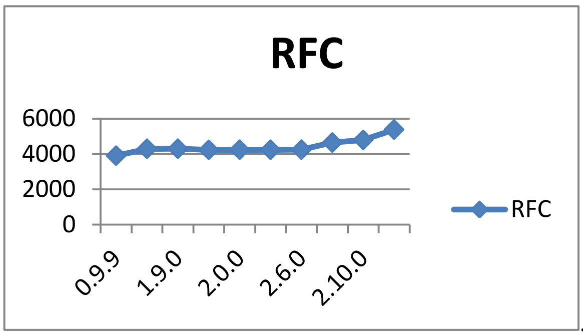


**Fig12: Growth curve for the response for all the classes for different releases of FGS**

The RFC and CBO are represented by Fig 9&11. For FGS , the RFC and CBO curve has shown the average increasing trend of complexity. The CBO and RFC for GLE is shown by Fig10&12, has shown the increase earlier releases before 4.2.1 after that it shows the downfall in release 4.2.2. but in later releases the RFC and CBO count in seen to be increased. DIT and NOC for FGS (Fig13 &15)and GLE(Fig14&16) are increased in successive releases of FGS and GLE ,which showed the increase in complexity. LOCH (Fig19 & 20) for the successive releases of FGS and GLE are increased in the mid bit showed the downfall in later releases which indicates the increase in complexity. Mc cabe complexity (Fig19 & 20) for both the software is increased in each successive release. From above interpretations the law of increasing complexity is reflected by FGS and GLE.

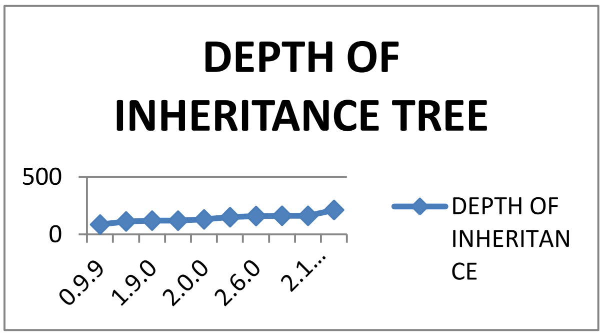
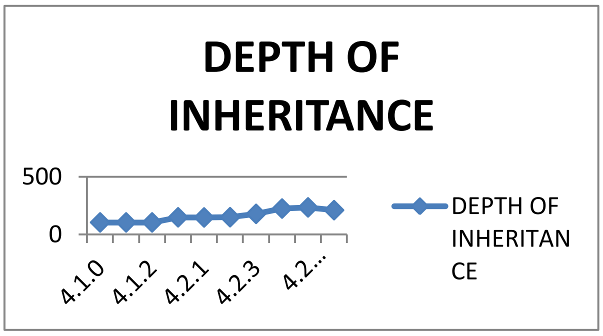


**Fig13: Growth curve for total depth of inheritance tree for all classes for different releases of FGS**



. **Fig 14: Growth curve for total depth of inheritance tree for all classes for different releases of GLE**

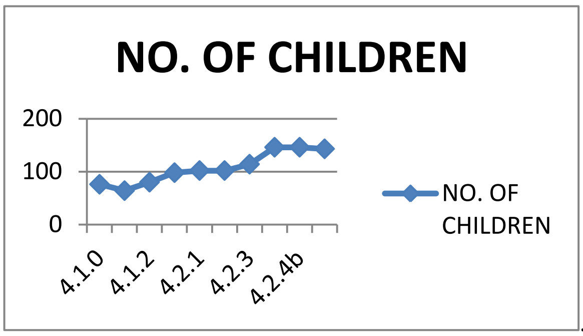


**Fig15: Growth curve for total number of direct descendent for all classes for different releases of GLE**

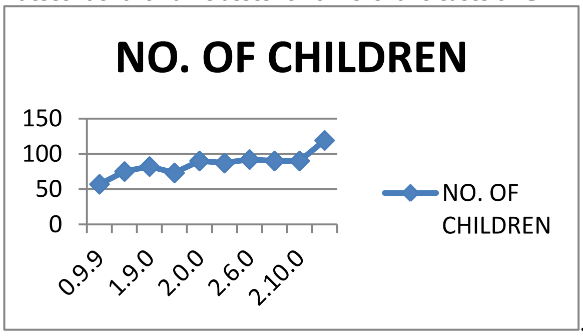


**Fig16: Growth curve for total number of direct descendent for all classes for different releases of FGS**

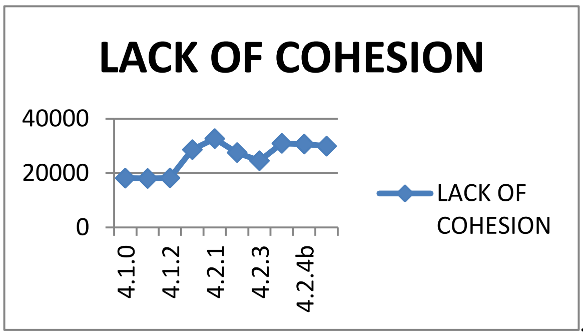


**Fig17: Growth curve for showing cohesion for all classes for different releases of GLE**

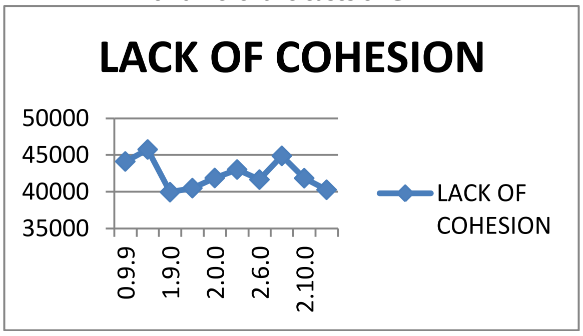


**Fig18: Growth curve showing the cohesion for all classes for different releases of GLE**

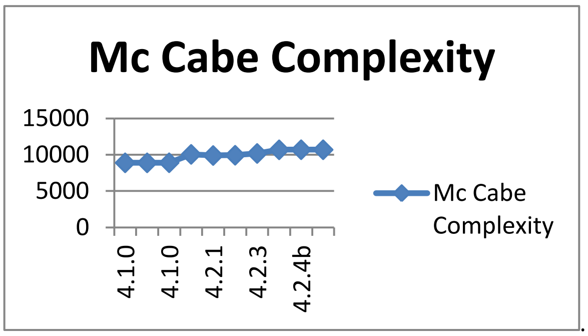


**Fig19: Growth of Mc cabe Cyclomatic complexity for GLE**

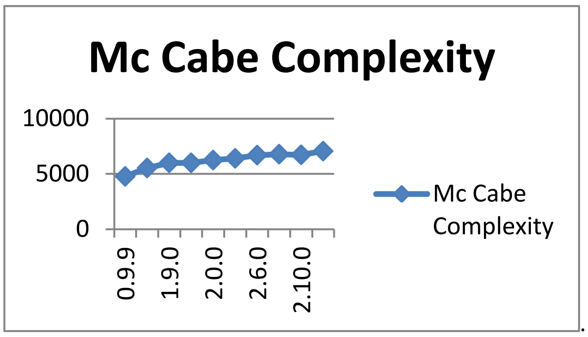


**Fig20: Growth of Mc cabe Cyclomatic complexity for FGS**

### 5.4 Law 7 : Declining Quality:

Law of declining quality, states that the software system evolving will decline with time unless it is rigorously maintained. The law is similar to the law2 increasing complexity. As the complexity of the software increases, its quality is decreased. Open source developers are free from any restrictions and pressures; they work according to their own interest which results in the declining quality of open source software. The decline of the quality of GLE and FGS is reflected by law 2.

### 5.5 Law 8: Feedback System:

The law of feedback system states that the evolution of the software is multi-level, multi agent and multi - loop feedback system. The existence of feedback system in case of open source software is reflected as the feature request and bugs are reported by the user community. In case of open source multi agent and multi loop system is difficult to determine.

### 5.6 Law 4: Conservation of organizational stability:

Law of conservation of organizational stability states that the average global rate of activity on evolving software is invariant over the product life i.e. over the product lifetime, the amount of work that goes into evolution, is fixed. But the measure of the work done in case of open source system is extremely difficult to determine as in overall development of software includes the community efforts and community size increases in case of open source systems.

### 5.7 Law 5: Conservation of organizational familiarity:

Law of conservation of organizational familiarity, states that familiarity with the evolving software is conserved. The average incremental growth rate should be constant as the software evolves i.e. to properly evolve the software the team should do it in fixed increments or there will be risk of losing the understanding of the software. By observing the revision details of Flight Gear Simulator, it is found that the changes to the number of classes or functions were few thereby the familiarity is maintained .In GLE, there is sharp change in release 2.1.0 (fig 20) which were decreased in next successive version and later versions shows the regular change and increase in size.

### 5.8 Law 3: Self-Regulation

Law of self-regulation, states that the evolution process is self-regulating leading to steady process. There is the balance between what is desired to change and what is actually achieved. The growth curve of Flight Gear Simulator shows the regular change and increase in size but in GLE there is sharp change in release 2.1.0 (fig 20) but in later shows the regular change and increase in size.

## 6. RELATED WORKS

The study carried out in this paper is closely related to some prior studies carried out by various researchers , includes Johari et al [8] find the applicability of Lehman laws on two open source software by considering 12 releases of Jhot and 16 releases of Rhino. Chris .J.Arges[9] evaluates Lehman laws on Linux kernel by studying 810 versions of linux kernel over the period on 14 years. Godfrey and Tu [10] examined the 96 version of linux kernel to find applicability of laws.

The pioneer work in software evolution field was done by Belady and Lehman [3] by analyzing the 20 release of OS/360 which led to the development of laws and published in [14][15]. Robles.G et al [11] conducted detailed literature review and given the description of the state of research in the area of open source software. Cook et al [12] has given the detailed explanation of SPE classification.

## 7. CONCLUSIONS AND FUTURE WORKS

In this paper, a study has been conducted on two open source software to find the applicability of Lehman laws. It is found that the law 1, 2 & 6 can be determined using different metrics but law 3, 4 &5 are difficult to determine in case of open source software and law 7&8 are related by hypotheses based on the study of change log of two software. To determine these laws more effectively, it requires deeper empirical studies in the field of open source software. Contributions of this work are:

1. The study is carried out in field of open source software.
2. The study encourages new opportunities of research in the field open source software evolution.
3. The study conducted may contain some anomalies and is ascertained as the material used is available on internet.

## 8. REFERENCES:

[1] id., Programs, Cities, Students, Limits to Growth?, Inaugural Lecture, May 1974. Publ. in Imp. Col of Sc. Tech. Inaug.l Lect. Ser., vol 9, 1970, 1974, pp. 211 - 229. Also in Programming Methodology, (D Gries ed.), Springer, Verlag, 1978, pp. 42 – 62

[2] Chidamber, S.R. and Kemerer, C. F. (1994), A metrics suite for object-oriented design. IEEE Transaction on. Software Engineering. Vol 20 No.6,June 1994,Page No. 476–493.

[3] Belady ,L.A. and Lehman M.M. 1976. A model of large program development. IBM Syst .J. 15 225-252

[4] GLE source code available[online] http://sourceforge.net/projects/glx/?source=navbar.

[5] Flight Gear source code available [online] http://fgfs.physra.net/ftp/Source/.

[6] GLE revision details [online] http://www.gle-graphics.org/main/changes.html.

[7] Flight Gear Revision details [online] http://www.flightgear.org/category/news/.

[8] Johari K and Kaur A, Effect of software evolution on software metrics: An open source case study ACM SIGSOFT Software Engineering Notes September 2011 Volume 36 Number 5.

[9] Arges C.J., Linux and Lehman- Literature Review of Open Source Evolution Analysis. http://chrisarges.net/files/arges_linux_evolution.pdf.

[10] Godfrey M.W. and Tu Q., Evolution in open source software: A case study. In Software Maintenance, 2000. Proceedings. International Conference on, pages 131 142. IEEE, 2000.

[11] Robles.G, Amor.J, Barahona.G.J and Herrariz.I, The evolution of the laws of software evolution. A discussion based on a systematic literature review.ACM Computing Surveys, Vol. 1, No. 1, Article 1, Publication date: June 2013.

[12] Cook S, Harrison R, Lehman M.M., Wernick P, Evolution in software systems: foundations of the SPE classification scheme, Journal of Software Maintenance and Evolution: Research and Practice, Volume 18, Issue 1, January/February 2006, Pages: 1–35.

[13] Godfrey M.W., German D.M., The Past, Present, and Future of Software Evolution Proc. ICSM 2008.

[14] Lehman .M.M. On Understanding Laws, Evolution and Conservation in the Large Program Life Cycle, J. of Sys. and Software, v. 1, n. 3, 1980, pp. 213 – 221.

[15] Lehman .M.M.(1980), Programs , life cycles and the laws of software evolution. In Proceedings of the IEEE (Special issue for Software Engineering, 68(9), pp1060-1076).