Validation of Lehman Laws of Growth and Familiarity for Open Source Java Databases

Arvinder Kaur and Vidhi Vig

Abstract Lehman's laws of software evolution have been widely researched and validated but there exists very few studies that verified these laws for databases in open source. Database evolution jeopardize the semantical and syntactical cogency of an applications, but, is their incremental augmentation restrained by the growth and familiarity? To verify this, the current study explores the properties of growth for database evolution by analyzing Lehman's fifth and sixth law of software evolution: Law of Conservation of Familiarity and Continuous Growth on three open source Java databases spread across 63 releases for 17774 number of bugs. The study found that Lehman's laws of growth and familiarity applies on Open Source Java databases also and laws were validated by all the datasets.

Keywords Open source · Databases · Lehman laws of software evolution · Bugs

1 Introduction

Lehman laws of software evolution [1–4] were defined and redefined from late 1960s to early 1990s. Lehman laid three Laws of software evolution

1. Law of Continuing Change
2. Law of Increasing Entropy
3. Law of Statistically Smooth Growth.

A. Kaur · V. Vig (✉)
USICT, Guru Gobind Singh Indraprastha University, Sec 16-C Dwarka, New Delhi, India
e-mail: vidhi.ipu@gmail.com

A. Kaur
e-mail: arvinderkaurtakkar@yahoo.com

S.C. Satapathy et al. (eds.), *Computer Communication, Networking and Internet Security*, Lecture Notes in Networks and Systems 5,
DOI 10.1007/978-981-10-3226-4_43

In 1978, modification of earlier laws and redesigning resulted in five laws, three of which were the old ones only. The new laws introduced were Law of Invariant Work Rate and Law of Incremented Growth.

Though, the first three laws remained same, only their definitions changed. These laws were validated specifically on E (embedded) type of systems which involved human interception. Unlike the other two types of programs (S (specified) type and P (problem solving) type), E type programs showed evolutionary pattern and changed as a result of change in its environment. Laws 3–5 experienced a new version in year 1980 and were stated as:

3. Law of fundamental law of program evolution
4. Law of conservation of organizational stability
5. Law of conservation of familiarity

Continuous uncertainty in the software led to reformulations of laws in 1996. The laws now increased from 5 to 8 and since then, are used as it is till date by the researchers to prove or refute the laws for their study. The reason behind revision of these laws was change in development and maintenance standards of the software, with time. Thought of global users coming together to work and discuss software was totally unimaginable until internet came into existence. Lehman totally abandoned Free Libre Softwares while formulating the earlier laws resulting in formulation and reformulation of laws as the time progressed. Law VIII is one such example, where libre softwares strongly hold and progress.

1. Law of Continuing Change
2. Law of Increasing Complexity
3. Law of Self-Regulation
4. Law of Conservation of Organizational Stability
5. Law of Conservation of Familiarity
6. Law of Continuing Growth
7. Law of Declining Quality
8. Law of Feedback Systems

Databases resembles a lot to E-type systems as they revolve around a community (developers and users) that solve real world problems (use of queries) [5] therefore, they must apply to Laws of Software evolution also. Evolution of databases has been rarely studied for its entire lifecycle in spite of the fact that alteration in schema of databases may lead an application to crash or behave abnormally [5]. In fact, no study till date has explored the incremental growth and familiarity of these databases for Open Source Java applications. The current study therefore specifically explores the Law of Continuous Growth and Conservation of Familiarity for three open source Java databases spread across 63 releases for 17774 number of bugs.

The related work is presented in Sect. 2 followed by Data Collection Methodology in Sect. 3. Validation of Law of Evolution for Growth Attributes of Databases is presented in Sect. 5 and Conclusion in Sect. 5.

2 Related Work

Lehman laws of software evolution have been studied and validated widely. Many researchers verified these laws on closed source and industrial projects. Lehman himself proposed these laws for closed source applications. [6] in their study gave an elaborate description on categorization of projects in S, P and E type. [7] on the other hand, explained the methodology to perform empirical research in this field for software evolution. Consequently, these laws were explored on different platforms and programming languages by various studies [8].

A schema evolution jeopardize the semantical and syntactical validity of the related applications and tremendously affect the users as well as developers [5]. Despite this fact, only four studies have verified evolution on databases [5, 6, 9, 10]. [6, 9, 10] studied only the statistical attributes of the evolution and fail to render details of formal mechanism of database evolution. [5] on the other hand studied evolution on open source databases and found that the evolution of databases augmented in controlled manner. [11, 12] analyzed schema evolution for databases too.

The current study on the other hand, focuses on growth and change brought in open source Java databases and verifies fifth law, Law of Conservation of familiarity and sixth law, Law of Continuous Growth only. Till date no study has explored these laws individually and specifically for Java databases in open source. Moreover, this study verifies these laws for every major and minor releases in details unlike [5] study, who just explored the ripples brought in by releases as a whole.

3 Data Collection Methodology

The all-volunteer Apache Server Foundation (ASF) develops, stewards, and incubates more than 350 Open Source projects and initiatives that cover a wide range of technologies [13]. Apache Server Foundation was mined for database projects whose bugs were tracked under Jira [14] repository to maintain uniformity of data. These projects were then checked for their status. Projects under status ‘Active’ were kept in the pool for selection and randomly three projects were chosen for the study.

These datasets were then mined for their artefacts and used in the study. It must be noted that Versions with status ‘Released’ and Bugs with status ‘Closed and Resolved’ were considered for the study in order to avoid spurious results. This process is described briefly in Fig. 1 and the datasets thus collected are presented in Table 1 given.

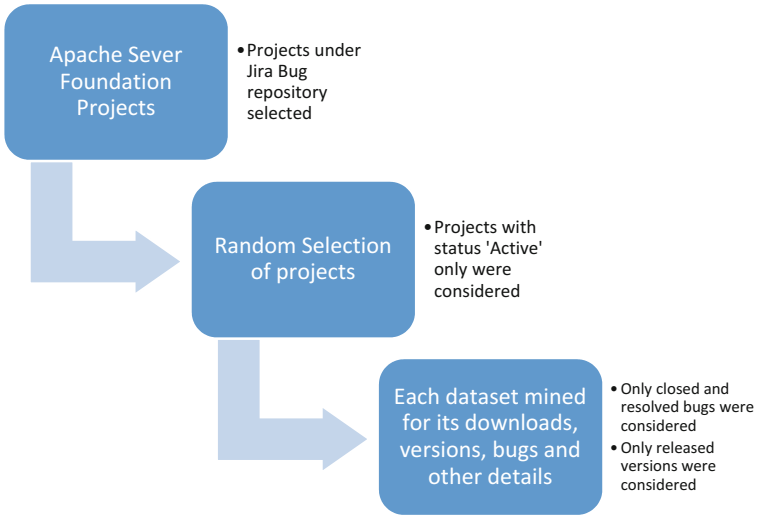

Fig. 1 Data collection process

Table 1 Datasets selected for the study

Datasets	URL	Category	Brief description	Date of first release	Date of last release	No of versions
Pig	www.pig.apache.org	Hadoop, database	Platform for analyzing large data sets on Hadoop*	29/10/2007	06/06/2015	22
Hive	www.hive.apache.org	Hadoop database	Data warehouse software facilitates querying and managing large datasets residing in distributed storage*	30/04/2009	15/02/2016	16
Zookeeper	www.zookeeper.apache.org	Database	Distributed computing platform*	13/11/2007	02/09/2015	25

*Information gathered from https://projects.apache.org/ *Note* The artefacts were last updated in May 2016

4 Validation of Law of Evolution for Growth Attributes of Databases

In Jira repository, every issues was categorized into five categories: Bug, Wish, Task, Improvement and New feature [14]. The current study explored these issues for the amount of change brought by each one of them. It was observed that bugs contributed to more than 60% of change in all the datasets. Figure 2 given below presents the details of the issues of datasets selected for the study.

To further statistically validate this behavior, single factor ANOVA was applied. To verify this the following null and alternative hypothesis were laid.

H_0: All the issues brings in equal change.
H_0: At least one issue (Bugs) brings in more change than others.

The null hypothesis states that all the issues bring in equal change in the application but since, p value is less than 0.05 (Table 2), we can reject the null hypothesis and support the alternate hypothesis. Consequently, the study mined and verified bugs for each version in order to identify their trend major and minor versions. The study gathered details of 17774 bugs in 3 open source Java databases for 63 versions.

Law of Conservation of Familiarity: This law proposes that system's incremental growth tends to remain statistically invariant or to decline. This is so, because the developers need to understand the program's source code and behavior. A corollary is often presented, stating that releases that introduce many changes will be

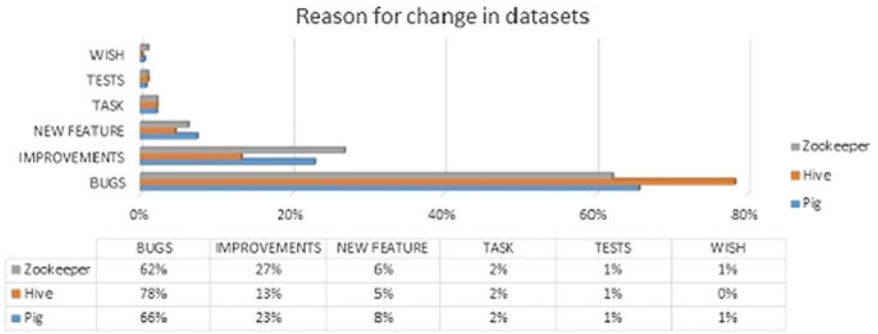

Fig. 2 Graph displaying various reasons for change in datasets

Table 2 Result of single factor ANOVA

Source of Variation	SS	Df	Ms	F	P-value	F crit
Between Groups	7484626	5	1496925	3.74666	0.028252	3.105875
Within Groups	4794431	12	399535.9			
Total	12279057	17				

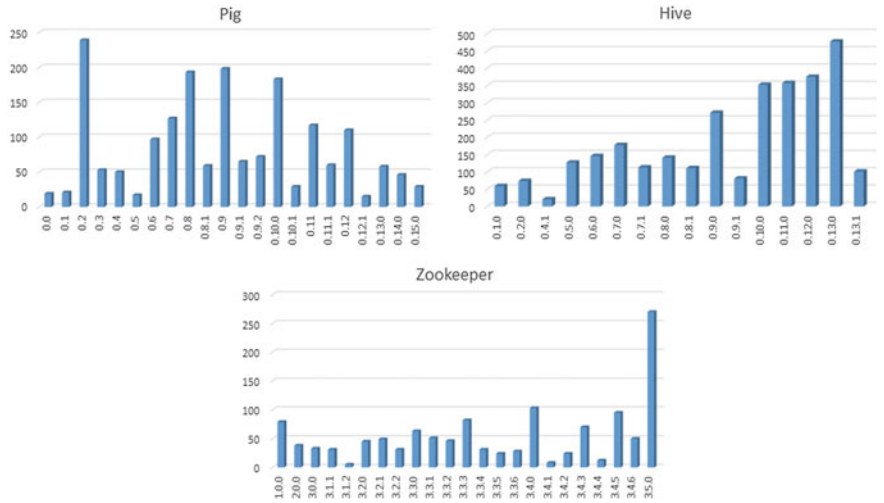

Fig. 3 Graphs displaying the distribution of bugs in Pig, Hive and Zookeeper dataset with Bugs on x axis and versions on y axis

followed by smaller releases that correct problems introduced in the prior release, or restructure the software to make it easier to maintain [1–4].

To further understand the behavior of changes every datasets was mined for its each and every versions. Since, it has already been observed that bugs were the biggest reason for change (Fig. 2), bugs for every version of the datasets were collected. Note: The study uses the word release and version synonymously in the chapter.

While analyzing of these bugs, a trend was observed. Every major version had a huge number of bugs followed by their minor versions with comparatively smaller number of bugs. This trend was observed uniformly by all the datasets and are presented in Fig. 3.

Law of continuous growth: Line of Code (LOC) for all the versions of the datasets were gathered from their very first releases till their last release before May 2016 (because after this we started analysing the results). The study found and increasing trend in LOC of all the databases validating the Law of Continuous Growth for all the datasets. These graphs are not presented in the chapter since graphs of change in LOC are presented in Fig. 4 given below. Increase in LOC can be observed from these graphs also.

Interestingly, the study found that change in LOC (presented in Fig. 4) of the major versions were far more greater than the change in LOC of the minor versions. Further investigations revealed that every major version in the selected dataset either brought a 'New Feature' or an 'Improvement' resulting in a sudden increase in LOC. This sudden increase in LOC further brought an increase in bugs (in major versions) also. The minor versions, on the other hand, were maintenance releases or releases launched after unit testing and rectification of bugs.

Fig. 4 Change in LOC of each version of selected datasets

Hence, it can be stated after the analysis that Law of Continuous growth and Conservation of Familiarity hold for the selected datasets. It can also be observed that the releases (Versions) that bring maximum change are the "major versions" of the software system and these releases are often followed by the "minor versions" that bring smaller changes wherein bugs are the leading factor of change.

5 Conclusion

The study explored the laws concerning the growth attributes of database evolution which is an indispensable and almost untouched field of software evolution. It was observed that the databases continuously grow and expand periodically in the system validating the Law of Continuous Growth in the system. The study discovered that bugs were unanimously the biggest reason for change in the system and should be paid due attention while validation of Laws of Software Evolution. Law of Conservation of Familiarity was also validated by all the datasets and it was found that releases that bring maximum change are the "major versions" and these releases are often followed by the "minor versions" that bring smaller changes wherein bugs are the leading factor of change.

Further research is going on to explore the generalizability of these results on all the Open Source Java databases. Other laws of evolution are also under study for this domain.

References

1. Lehman, M.M., Perry, D.E., Ramil, J.F.: On evidence supporting the FEAST hypothesis and the laws of software evolution, 5th International Software Metrics Symposium, 84–88 (1998)
2. Lehman, M.M., Ramil, J.F., Wernick, P.D., Perry, D.E., Turski, W.M.: Metrics and laws of software evolution—the nineties view, International Software Metrics Symposium, 0–3 (1997)
3. Lehman, M.M., Ramil, J.F.: Rules and tools for software evolution planning and management, Annals of Software Engineering, 11 (1), 15–44 (2001)
4. Lehman, M.M.: Laws of software evolution revisited, European Workshop on Software Process Technology (1996)
5. Skoulis, I., Vassiliadis, P., Zarras, A.: Open-Source Databases: Within, Outside, or Beyond Lehman's Laws of Software Evolution?. In International Conference on Advanced Information Systems Engineering, 379–393 (2014)
6. 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, 18(1), 1–35 (2006)
7. Kemerer, C.F., Slaughter, S.: An empirical approach to studying software evolution, IEEE Transactions on Software Engineering, 25(4), 493–509 (1999)
8. Kaur, A., Vig, V.: Mining software repositories for empirical validation of laws of software evolution for Java projects, International Journal of Computational Systems Engineering, 3, 155–173 (2016)
9. Papastefanatos, G., Vassiliadis, P., Simitsis, A., Vassiliou, Y.,: Metrics for the prediction of evolution impact in etl ecosystems: A case study. Journal on Data Semantics, 1(2), 75–97 (2012)
10. Sjøberg, D.: Quantifying schema evolution. Information and Software Technology, 35(1), 35–44 (1993)
11. Manousis, P., Panos V., Apostolos Z., George, P.: Schema Evolution for Databases and Data Warehouses, In European Business Intelligence Summer School, 1–31 (2015)
12. Cleve, A., Maxime, G., Loup, M., Jerome, M., Jens, W.: Understanding database schema evolution: A case study, Science of Computer Programming 113–121 (2015)
13. Apache Server Foundation, http://www.apache.org
14. Jira, https://www.atlassian.com/software/jira