NFC INSTITUTE OF ENGINEERING & FERTILIZER RESEARCH
NFC-IEFR Journal of Engineering and Scientific Research
(NIJESR)
http://nijesr.com/
E-mail: editor.jesr@iefr.edu.pk

The Evolution of WordPress and Django Web Frameworks Using Lehman's Laws

M. Murad[1], M. W. Ashraf[1]

[1]*Department of Computing, Riphah International University, Faisalabad Campus, Faisalabad, Pakistan*

ARTICLEINFO

***CorrespondingAuthor:**
muraddw@gmail.com

DOI:
10.24081/nijesr.2019.1.0007

Keywords:
Common Ratio, LOC, Lehman's Laws Django,Software Evolution, WordPress.

ABSTRACT

In recent time, the evolution of web applications have gained importance over the web development process and the factor of web evolution cannot be ignored by web developers. Web development has become complex and challengeable for web developers. The process of software evolution played an important role during the development of the software. Millions of web application have been developed every year around the world It has included various approaches, tools, and frameworks to reorganize the web applications with an improved version. Research has been shown that there are no proper and systematic techniques is available for evolving web applications. This special article has been written to make a comparative analysis of WordPress and Django web framework using Lehman's laws of software evolution. It has been found that the six out of eight Lehman's laws found valid during the evolution process for web frameworks.

I. INTRODUCTION

World Wide Web (WWW) has been matured over the last couple of decades. In recent time, web development is the most popular field in computer sciences. Different programming languages are being utilized to develop web applications. In this special article, we have conducted a systematic study to evolve web applications using Lehmanlaws for evolving the software. In the past, onlydesktop applications were being evolved using Lehman's Laws, but in this case study, two popular web applications named WordPress and Django web framework have been analyzed using these laws. These two frameworks have strong competition in the web development environment. Both frameworks have their own features. [1]Wordpress provides dynamics platform for developing web application and blogs. It was firstly developed in the year 2003 by Matt Mullenwegand Mike Little. It is an open source and freely available at online repositories such as www.openhub.com and www.github.com. Research has shown that over 60 million websites including top 10 websites based on WordPress since 2018. Wordpress is a combination of different programming language. Table 1 showed detailed information about language breakdown of WordPress version 5.1[2].

Django is another popular and uprising framework for developing web applications. It is based on Python programming language. It was developed in the year 2005 by Adrain Holovaty and Simon Willison. It is open source and also freely available at online free repositories[2]. Interestingly, Django was named after a French-based Jazz guitarist Django Reinhardt. Django has been designed to provide a simplified approach for developing web applications. In the year 2018, it was competing with C++ and Java and was ranked at No. 4 with other popular languages. Django framework provided several features for developing web applications such as session, template forms, caching, user authentication, testing, etc. There are over 4000 packages available for Django for outlining, assessing and debugging. Django web framework is also a combination of different languages. Table 2 has contained information of language breakdown of the latest version of Django 2.2.0.

TABLE 1. Language Breakdown of WordPress Web Framework [2].

Languages	Line of Code	Comm. Lines	Comm. Ratio	Blank Lines	Total Lines	%
PHP	314780	146563	31.8%	66,352	527695	62.6%
Javascript	155460	30188	16.3%	21464	207112	24.6%
CSS	64018	3687	5.4%	12080	79785	9.5%
HTML	24043	138	0.6%	1,834	26,015	3.1%
XML	2372	170	6.8%	198	2700	0.3%

TABLE 2. Django Web Framework language Breakdown[2].

Languages	Line of Code	Comm. Lines	Comm. Ratio	Blank Lines	Total Lines	%
Python	248106	49736	16.745	59493	357335	89.4%
JavaScript	18,927	3934	17.2%	4380	27241	6.8%
CSS	4,690	150	0.5%	529	5843	1.2%
HTML	4099	19	0.5%	529	4647	0.8%

NFC INSTITUTE OF ENGINEERING & FERTILIZER RESEARCH
NFC-IEFR Journal of Engineering and Scientific Research
(NIJESR)
http://nijesr.com/
E-mail: editor.jesr@iefr.edu.pk

Other	**1450**	**18**	**3.8%**	**99**	**4573**	**1.1%**

It has been seen that there is tough competition between PHP and Python over the last few years. This special article has been written to check the progress of both web frameworks. For this purpose, we decide to implement famous Lehman laws for evolving the software [3]. The purposed study has the first footstep to evolve web applications using Lehman laws. These laws were first developed in 1976 by Meie Manny Lehman, a British software engineer. He designed a series of eight laws for software evolution. According to Lehman, it is important for a software to remain stable, popular among the software developer and the customer's or organizations. If the software has evolved on a regular basis so, it cannot be survived for a longer period of time. We will apply all eight laws on both web frameworks and will check the validity of each during the evolution process.

II. Related work

In this section, we will cover the literature related to web evolution.[3]examined 30 different PHP projects using Lehman's laws. They concluded that the factor of constant growth can cause the maintenance of software at a continuous level. They also concluded that the quality of web applications has not decreased over a period of time.[4] emphasized the evolution of WordPress. According to the author, WordPress can only stay at the top if it will evolve on a regular basis otherwise it will be lost its credibility among the web developers and users around the world. It can only be possible through contributing proper development, testing, training. The factor of growth rate and community have paid an important role during the evolution process.[5] explained the importance of Django and its architectural pattern and which is based on the model view controller. [6] shared the detail of web evolution. They adopted a model-based approach to analyze irregularities and also make sure to increase the quality of web applications. [7] highlighted the importance of web development and growth. They analyzed the factors of usability of web applications and user satisfaction through different evolution methods. They purposed a method for early deduction and also predict problems related to usability of web applications. [8]After explained the fact about web evolution system. The author has analyzed the web evolution system over the last 15 years. He further added that the future of web evolution is not yet over. He emphasized three factors, security, testing, and access.Web evolution can play a very important role in large scale data analytics. It can also solve certain problems and the limitations of web applications. They also emphasized on systematic web evolution because of better development of web applications[9]. [10] explained the factor for evolving web applications. They have focused on three important issues, Process development, maintenance and re-engineering the application. According to the author, new technologies for web development can play an important role in better development of the software.

[11] has focused on maintaining web applications. According to the authors, maintainability can play an important role because it may reduce maintenance cost, controlling quantitative metrics and model for future predictions.[12]have been reviewed different dimensions for the sustainable and longer term for the web application system. The factor of change played a critical role in any web application. The author has presented different statistical observations to measure the level of changes in web applications [13].[14] have analyzed locally customer's requirements with the requirement belongs to the global base environment. [15]have been described as the process of analyzing and combining the large size of the source program is not a proper way and systematic way to check the progress of the software. [16] have shared their experience in software evolution and its related problems. Further, it is an important idea to build a powerful model for evolving the software. [17] haveproposed a code flattering technique is used to remove the complexity from code and made the code simple. [18] has shared the data mining technique for analyzing the pattern of input request by the users. [19] explained the evolution process requires deep analysis of software. The factor of certain changes has directly affected the process of software evolution. The more changes in the software can cause to increase the complexity, cost, and behavior of the software[20]. [21] Emphasized on a deeper level of software evolution process because of only deep analysis enables developer better understanding about the software**.** [22] Has focused on the performance of the large size of software abs also analyzed the software to gain information about the software capability to handle certain challenges regarding the development of large size software. [23] has wanted to enhance the feasibility of software and shifted uncertain situations to well organize software development and its effective evolution. [24] have been found several factors for software evolution that must be investigated. These factors included, unused of code, removing functions, utilization outsourcing libraries, API's and complexity of API's. [25] concluded that the four out of eight laws found a true result for selected software projects for evolution and also found distinguishable distinctions. [26] explained the factors of software evolution. These factors included certain changes, costs, and efforts for developing software. [27]has initiated that the growth rate of Linux kernel has increased over a period of time because of increasing its size (source code). [28] have confirmed that the four out of eight laws are applicable for evolving the software system. [29] have described that the complexity can only be reduced through

proper teamwork of software developers and restructuring the source code from complexity to normality. [30]have explained their experiences of utilizing Lehman's law in their purposed work and found different results and variations in the conclusive results of each law during the evolution process. [31] shared their work by describing that the four out of eight laws have shown slight reflection in the proposed graph.[32] explained that the software quality has directly linked with software evolution. The change in the declining quality of particular software is a high-risk factor. When a software system deal with requirement changes the chances of software aging is also increased. The authors have highlighted the challenges that directly affect the evolution process.[33] described the software complexity factors. During the software maintenance process, complexity metrics have created an important role and also made the situation more complex and critical for software developers. The authors have utilized seven different software metrics for evolving three software. Their result indicated that the complexity factors directly affect the growth, integration, understanding, and design of the software.[34]examined the software complexity using Lehman Law. They have utilized two different complexity factor such as cyclomatic and interface complexity. To check the validity of law was their major aim in their empirical research. They applied the law on four open source software system. They conclusively found that the cyclomatic and interface complexity has increased by version to version.[35] expressed that the process of software evolution. They applied Lehman Law on two open source software system. They have concluded that few o laws have been determined their validity during the evolution process. Due to the open source nature of the software, it has a complex to evolve using Lehman's Laws.
[36]explained that software engineering has required a scientific based approach to evolving software. Lehman' law was the first footstep towards the evolution of software on scientific ground. These laws have been based on the factor of change in the software. [37]applied Lehman laws on mobileapplications. Three out of eight laws have been three out of eight laws on selected mobile applications. They also compared the results of a mobile application with the desktop application with the same features and data. They conclusively found that the law of continuing change has a similar result for both mobile and desktop application but found variations in the other two laws, increasing complexity and declining quality. [38]Compared to several scripting languages. They reviewed different characteristics of languages based on selected criteria and which has included defined applicability, popularity, users, learning and period of the language. [39] described that the software has the ability to accommodate certain changes. They applied Lehman's law and found that the factor of size and complexity have gradually increased over a period of time. Due to growth in the software, it has also been found that the quality of the software is also decreased. [40] software evolution process is a continuous process and which is directly connected with feedback of the software. The authors have described the challenges related to software evolution process in a real-world environment.[41] have explained that the software evolution process over the period of the last thirty years of time span. According to the author, the evolution process has played a significant role. The laws of evolving the software arethe first scientific study and also provide a basis for future development of rules for the software evolution process.

III. THE CONTRIBUTION OF MEIR MANNY LEHMAN FOR SOFTWARE EVOLUTION

The relationship between Meir Manny Lehman and the field of evolving the software is spanned over 30 years. He was the man who presented systematic laws for software evolution way back in 1974. [42] Brief summary of each individual law is described in table1. [43] investigated the effects in term of size, changing factor and growth rate of the module per release. [44] explained about critical situations that may arise during the evolution process of software. The authors have contributed their work in the field of software evolution.

TABLE 3. A brief description of Lehman's Rules for Software evolution.

Law	Name	Description
1	Continuing Change	Software changes over a period of time
2	Increasing Complexity	Complexity is also increased
3	Self-Regulation	Software evolution should be self-regulated.
4	Organizational Stability	During Software evolution, Organizational cannot lose its stability
5	Conservation of Familiarity	Evolution process cannot affect the familiarity of software.
6	Conservation of Growth	During evolution Process, it may increase the growth
7	Declining Quality	During the Evolution Process, it cannot lose its quality
8	Feedback System	The feedback system can decrease the growth of software system

NFC INSTITUTE OF ENGINEERING & FERTILIZER RESEARCH
NFC-IEFR Journal of Engineering and Scientific Research
(NIJESR)
http://nijesr.com/
E-mail: editor.jesr@iefr.edu.pk

IV. RESEARCH QUESTION

Research Question:*Is Lehman's laws can be validated for Django and Wordpress web framework*?

V. CRITERIA FOR SELECTING CASES

To select the projects for evolution, the following criterion has been set. It includes the following important points,

i. The source code should be available online.
ii. The object-oriented based project will be utilized for evolution.
iii. Projects should be varying in size and different versions.

VI. SOFTWARE METRICS FOR DATA ANALYSIS

To check the validity of each Lehman's laws, the following table indicated that each law has been enclosed with appropriate variable (Ei….Eviii).

TABLE4. Software Metrics for Data Analysis.

Law(s)	Specified Variable	Data Investigation
1	E_i = Days Between Releases (DBR)	Each Law will evolve using: • Trend Test • Excel Graph
2	E_{ii} = Size of applications (KB)	
3	E_{iii}= Incremental Changes in LOC	
4	E_{iv}= Number of Comments	
5	E_v=Rate of Modifications	
6	E_{vi}= Line of Code	
7	E_{vii}= Common Ratio	
8	E_{viii}= Line of Code	

VII. STATISTICAL ANALYSIS OF DATA

The purposed process for data analysis started from the extracted all files from selected projects (Django, Catalyst, and Wordpress). Then software metrics have been selected from the extracted files at class levels. Each metrics have been calculated and also been utilized to validated the Lehman's laws on each project. The whole process is drawn in the Fig. 1. It includes the following,

i. Each Lehman Laws have defined through. Variable ($E_{i...}E_{viii}$). Different software metrics are assigned to variables and calculated.
ii. The general hypothesis for the purposed study is:
H_0 = Software metric E has no trend.
H_1 = Software metric E has a trend.
iii. The results have been shown in MS-EXCEL based trend line analysis.

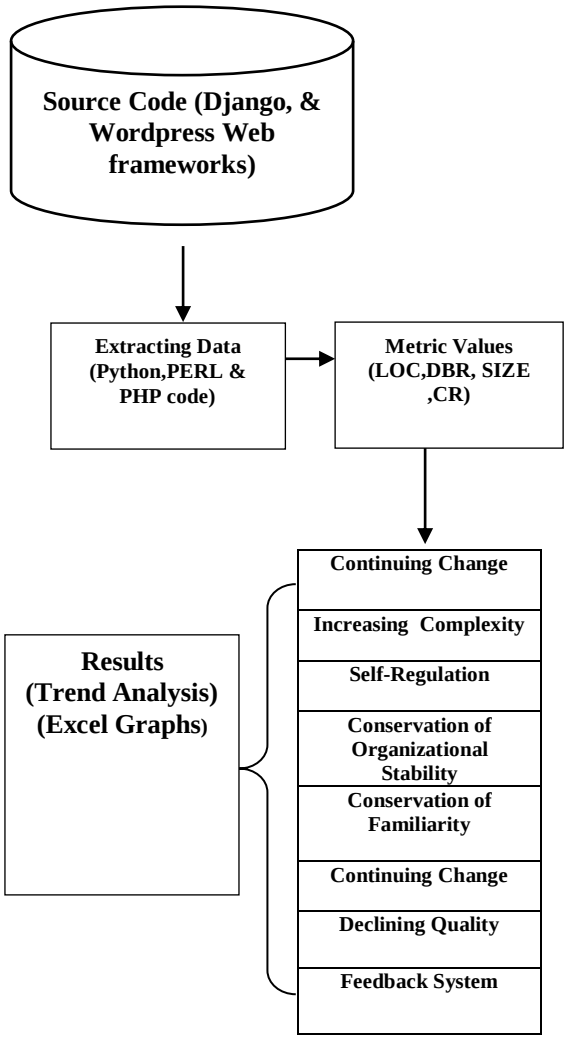


Figure 1. Process of Data Analysis.

VIII. RESULTS AND DISCUSSION

In this section, each framework has been evolved using different software metrics such as Line of code, Size,

NFC INSTITUTE OF ENGINEERING & FERTILIZER RESEARCH
NFC-IEFR Journal of Engineering and Scientific Research
(NIJESR)
http://nijesr.com/
E-mail: editor.jesr@iefr.edu.pk

Common Ratio. Each Lehman law has been examined against each individual framework (Django and WordPress.

i. Implementing laws of Continuing Change.

Software metrics Ei =DBR (Days Between Releases) has utilized to measure the validity of this law. Table 3 Contained the information of Days Between the release of each framework. Similarly,Fig. 4 and Fig.5 represented a graphical representation. Conclusively, both frameworks have shown a trend and new versions have been launched on a regular basis. It indicated the validated the law of continuing change.

TABLE 5. Days Between Release of Django and WordPress[2]

Django Version(s)	Days Between Release (DBR)		WordPress Version(s)	Days Between Release (DBR)
1.3	340		4.0	166
1.4	192		4.1	127
1.5	205		4.2	107
1.6	211		4.3	112
1.7	244		4.4	126
1.8	244		4.5	128
1.9	246		4.6	110
1.10	242		4.7	184
1.11	242		4.8	160
2.0	243		4.9	386

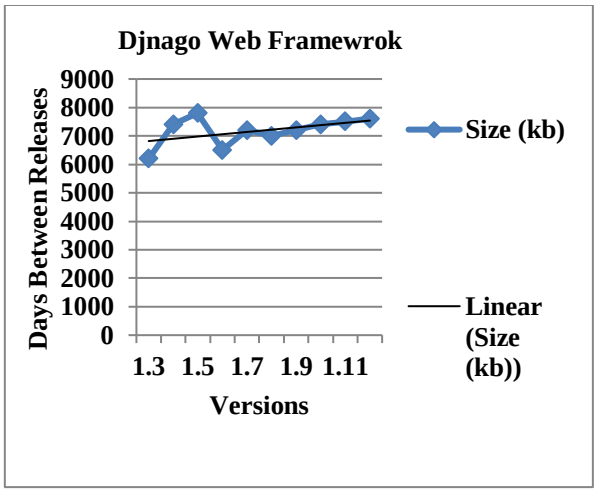


Figure 2. Trend Line Analysis of Django using DBR.

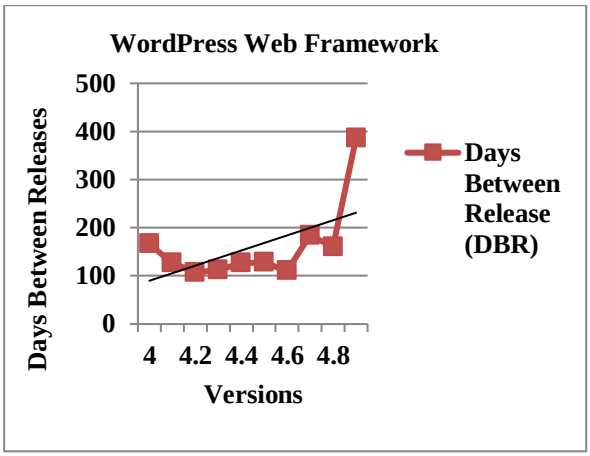


Figure 3. Trend Line Analysis of WordPress using DBR.

ii. Implementing the Law of Increasing Complexity

The variable Eii = Size of Application has utilized to check the validity of the law.Table6. has contained information about the size of both frameworks by year. Fig.4 and Fig. 5 have shown a positive trend. Therefore, it indicated that validityof the law.

Table6. Different Versions and Sizes of Django and WordPress [2]

Django Version(s)	Size (Kb)		WordPress Version(s)	Size (Kb)
1.3	6200		4.0	6300
1.4	7400		4.1	6400
1.5	7800		4.2	6600
1.6	6500		4.3	6800
1.7	7200		4.4	7300
1.8	7000		4.5	8000
1.9	7200		4.6	8200
1.10	7400		4.7	8300
1.11	7500		4.8	8500
2.0	7600		4.9	9900

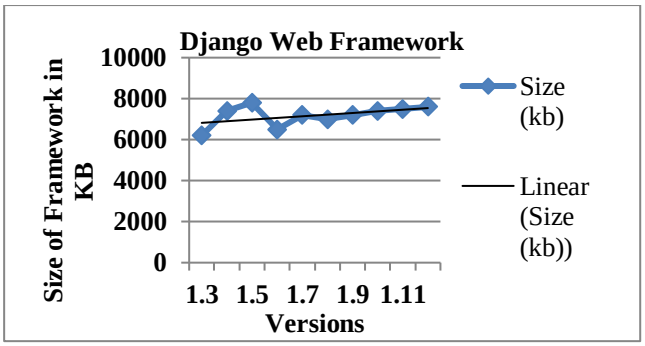


Figure 4. Trend Line Analysis of Django using Size of the framework.

NFC INSTITUTE OF ENGINEERING & FERTILIZER RESEARCH
NFC-IEFR Journal of Engineering and Scientific Research
(NIJESR)
http://nijesr.com/
E-mail: editor.jesr@iefr.edu.pk

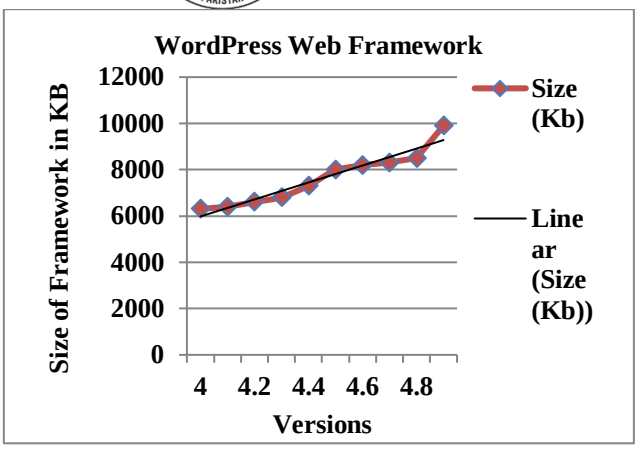


Figure5. Trend Line Analysis of WordPress using Size of the framework.

iii. Implementing the Law of Self-Regulation

The variable E_{iii}= Incremental Changes in LOC has utilized to check the validity of the laws.Table 7 contained the information of anIncremental Changes in the line of code per year and graphical illustration (Fig.6 and Fig. 7) indicated a positive trend which also indicated the validity of the law for both web framework.

TABLE 7. Incremental Changes in LOC of Django and WordPress [2].

Year	Incremental Changes in LOC of Django		Year	Incremental Changes in LOC of WordPress
2010	96091		2010	138849
2011	134958		2011	145460
2012	158140		2012	162917
2013	166569		2013	202699
2014	198748		2014	287255
2015	219740		2015	359046
2016	246711		2016	415831
2017	257201		2017	481668
2018	80288		2018	576357

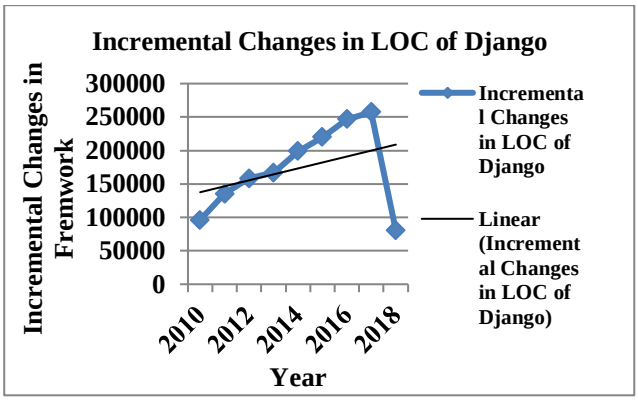


Figure 6. Trend Line Analysis of Django Incremental Changes.

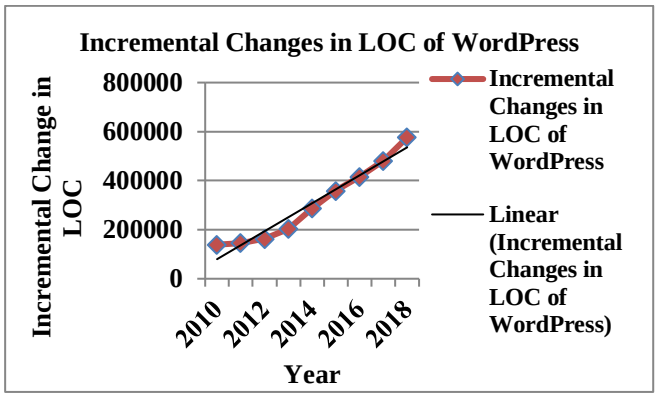


Figure 7. Trend Line Analysis of Django usingIncremental Changes

iv. Implementing the Law of Organizational Stability

The software metrics Number of Comments have utilized to check the validity of the law. Table8 analyzed through graphical illustration (Fig. 8 and Fig. 9) and a positive trend has been found and that validated the law.

TABLE 8.Number of Commentsin LOC of Django and WordPress [2].

Year	Number of Comments in Django		Year	Number of Comments in WordPress
2010	48594		2010	41239
2011	38260		2011	47154
2012	43119		2012	49953
2013	40525		2013	60936
2014	46879		2014	91947
2015	48283		2015	118036
2016	52697		2016	148600
2017	52816		2017	175528
2018	53476		2018	180761

NFC INSTITUTE OF ENGINEERING & FERTILIZER RESEARCH
NFC-IEFR Journal of Engineering and Scientific Research
(NIJESR)
http://nijesr.com/
E-mail: editor.jesr@iefr.edu.pk

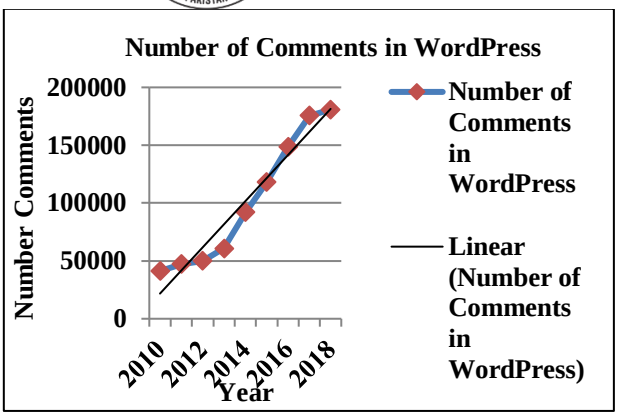


Figure 8. Trend Line Analysis of Django usingNumber of Comments.

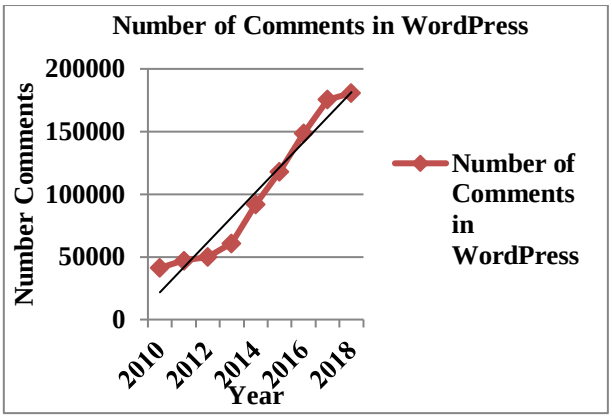


Figure 9. Trend Line Analysis of WordPress usingNumber of Comments.

v. Implementing the Law of Conservation of Familiarity

The software metrics "Ev =Modifications in files" has utilized to check the validity of the law. Table 16 contained the information all modification in number over a period of time for the Django and WordPress web frameworks. The graphical representation (Fig. 8 and Fig. 9) showed slightly inclination. This inclination did affect the validity of the law. The graphical illustration (Fig. 16) showed a positive trend and therefore it validated the law.

TABLE 9.Modifications in LOC for Django and WordPress [2].

Date	Modifications in files		Date	Modifications in files
05-April-2019	8		05-Oct-2018	3
03-April-2019	6		04-Oct-2018	1
02-April-2019	2		03-Oct-2018	2
01-April-2019	9		02-Oct-2018	6
31-Mar-2019	2		01-Oct-2018	2
30-Mar-2019	6		28-Sep-2018	3
29-Mar-2019	15		23-Sep-2018	4
28-Mar-2019	3		28-Sep-2018	1

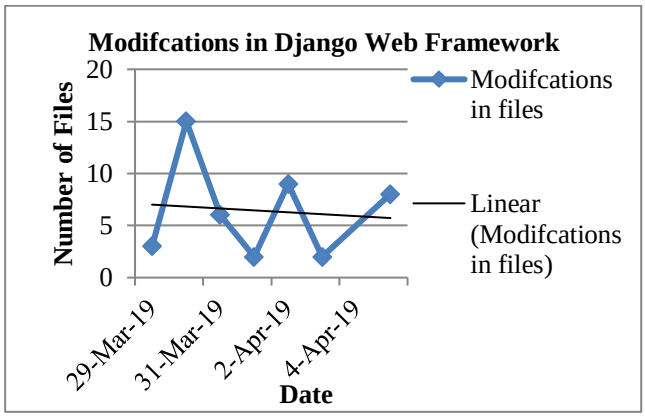


Figure.10. Trend Line Analysis of Django usingModifications in Files.

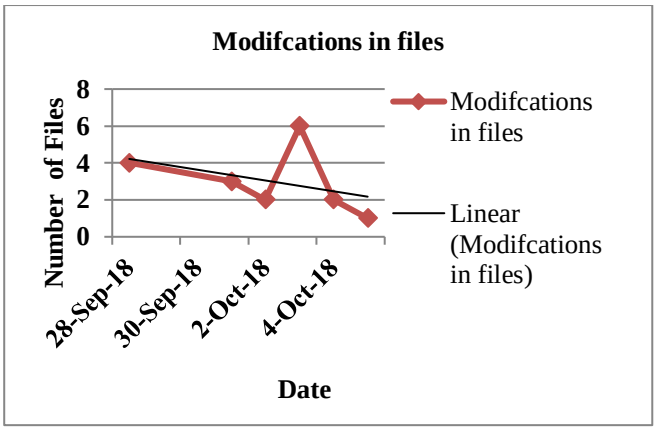


Figure 11. Trend Line Analysis of Django usingModifications in Files.

NFC INSTITUTE OF ENGINEERING & FERTILIZER RESEARCH
NFC-IEFR Journal of Engineering and Scientific Research
(NIJESR)
http://nijesr.com/
E-mail: editor.jesr@iefr.edu.pk

vi. Implementing the Law of Continuing Growth

The following variable is utilized to check the validity of the law. E_{vi} = LOC. Table10 has contained the information of Line of the code of both frameworks.

TABLE 10. LOC of Django and WordPress [2].

Year	Line of Code		Year	Line of Code
2012	158140		2012	154021
2013	166569		2013	177154
2014	198748		2014	250957
2015	219740		2015	340677
2016	246711		2016	406262
2017	257331		2017	476241
2018	274325		2018	562664

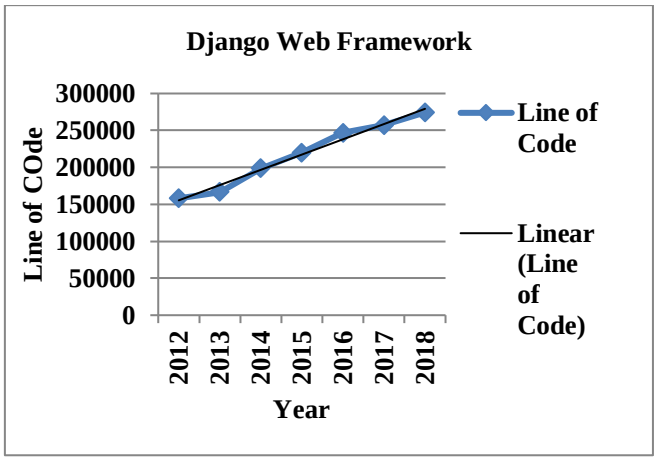


Figure 12. Trend Line Analysis of Django usingLOC.

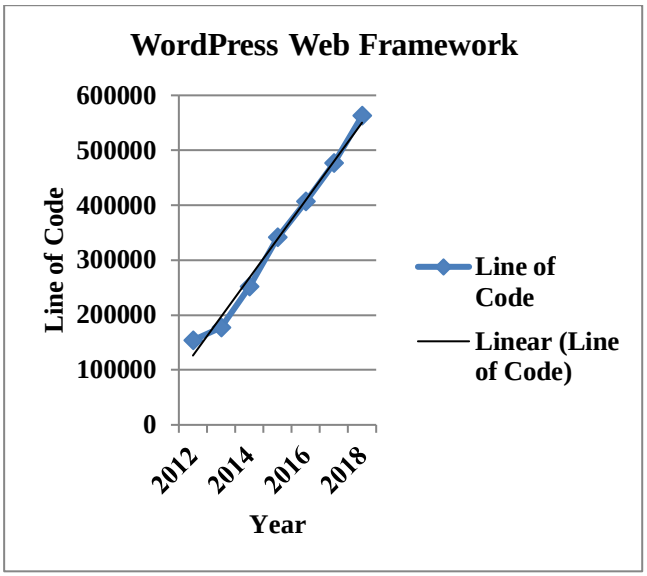


Figure 13. Trend Line Analysis of WordPress using LOC.

vii. Implementing the Law of Declining Quality

To check the validity of the law of declining quality, the following formula has been designed

E_{vii} = Number of comment ratio / Total Line of Code

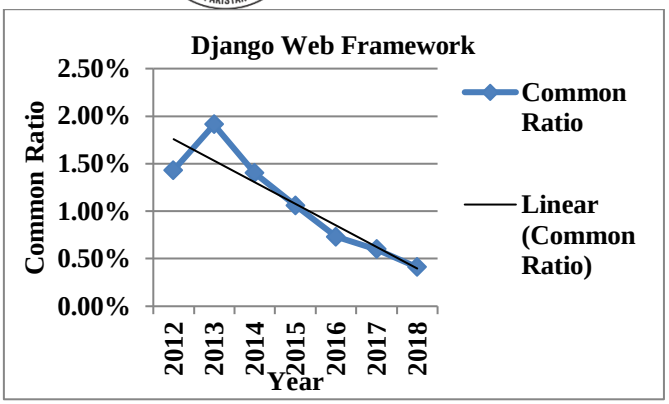
(or) E_{vii} = NCR/ TLOC

After calculating the value of Common Ratio which indicated in table 10. Then the graphical repsentations(Fig.14 and Fig. 15) have shown that slight inclination. Due to large size of applications, it can be assumed that the quality of both frameworks have not declined over period of time. Therefore, it did not validate the law of decelining quality.

Table10.Common Ratio of Django and WordPress Web Frameworks [2].

Year	LOC	Comments Line	Common Ratio		Year	LOC	Comments Line	Common Ratio
2012	158140	2126	1.43%		2012	154021	3559	2.31%
2013	166569	3187	1.91%		2013	177154	3405	1.92%
2014	198748	2792	1.40%		2014	250957	3805	1.51%
2015	219740	2346	1.06%		2015	340677	4636	1.36%
2016	246711	1825	0.73%		2016	406262	2967	0.73%
2017	257331	1555	0.60%		2017	476241	1731	0.36%
2018	274325	1145	0.41%		2018	562664	725	0.12%

NFC INSTITUTE OF ENGINEERING & FERTILIZER RESEARCH
NFC-IEFR Journal of Engineering and Scientific Research
(NIJESR)
http://nijesr.com/
E-mail: editor.jesr@iefr.edu.pk



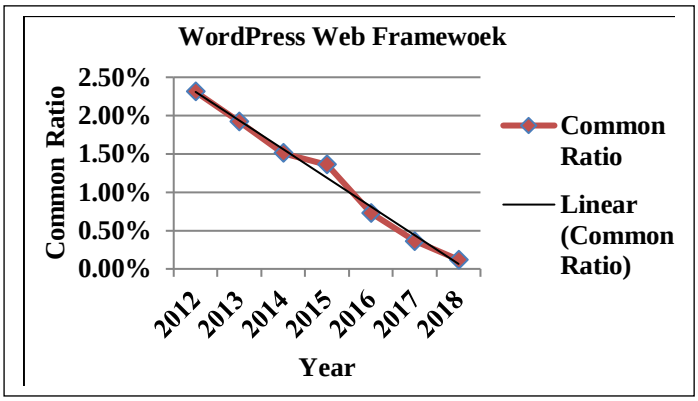


Figure 14. Trend Line Analysis of Django usingCommon Ratio.Figure 15. Trend Line Analysis of WordPressress usingCommon Ratio.

viii. Implementing the Law of Feedback System

The following variable has utilized to check the validity of the law.

E_{viii} = Growth rate Source Line of code.

Table 11 contains the information of changes in the source line of code over a period of time with respect to Change and Fig. 16 and Fig. 17 displayed variations in trend line analysis. Conclusively, It has been concluded that the feedback system did not affect the growth of both frameworks. Therefore, the law did not validate.

Table 11. Changes in the LOC for Django and WordPress [2].

Year	LOC	Increase/ Decrease LOC		Year	LOC	Increase/ Decrease in LOC
2012	158140	23518		2012	154021	15119
2013	166569	8429		2013	177154	23133
2014	198748	32215		2014	250957	73803
2015	219740	20992		2015	340677	89720
2016	246711	26971		2016	406262	65585
2017	257331	10620		2017	476241	69979
2018	274325	16994		2018	562664	86423

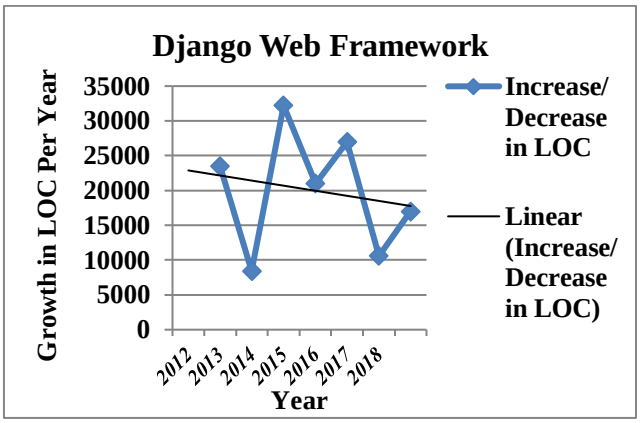


Figure 16. Trend Line Analysis of Django usingIncremental Changes.

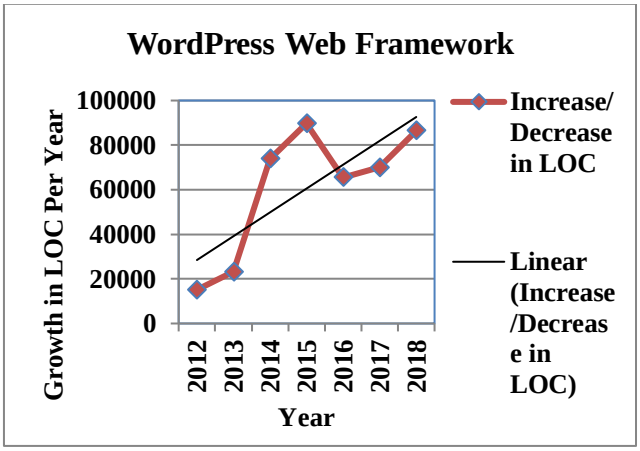


Figure 17. Trend Line Analysis of Django usingIncremental Changes.

Due to the large size of the application, it cannot assume that the feedback system creates a negative impact on the growth of the frameworks. Therefore, The Law is not considered to be validated.

The summary of findings against each Law on selected web applications is summarized here in table 12.

TABLE 12. Summary of findings against each law.

Law	Name	Description	Findings
1	Continuing Change	Software changes over a period of time	Validated
2	Increasing Complexity	Complexity is also increased	Validated
3	Self-Regulation	Software evolution should be self-regulated.	Validated.
4	Conservation of Organizational Stability.	During Software evolution, Organizational cannot lose its stability	Validated
5	Conservation of Familiarity	Evolution process cannot affect the familiarity of software.	Validated
6	Continuing Growth	During evolution Process, it may increase the growth	Validated
7	Declining Quality	During the Evolution Process, it cannot lose its quality	Not Validated
8	Feedback System	The feedback system can decrease the growth of software system	Not Validated

IX. CONCLUSION

The purposed study was designed to fulfill the gap between web evolution. Three different web applications Django, Catalyst, and WordPress have been evolved using Lehman's all eight laws. Convulsive results have been found after measuring the software metrics such as Source Line of code (SLOC), Days Between Releases (DBR), Comments line, Common Ratio (CR). It has been found that the six out of eight Lehman's laws have been validated by the selected web application. These laws include Law I (law of continuing change), Law II (Increasing Complexity), Law III (Self-Regulation), Law IV (Conservation of Organizational Stability) and Law V (Conservation of Familiarity) and Law VI (Continuing Growth). Two Laws such as Law VII (Declining Quality) and Law VIII (Feedback System) did not support the selected web applications.

ACKNOWLEDGMENT

This issue is conducted at the platform of Riphah International University Islamabad (Faisalabad Campus).

REFERENCES

[1] *WordPress Web Design For Dummies*. New York: John Wiley & Sons Inc, 2013.

[2] "Open Hub, the open source network", *Openhub.net*, 2019. [Online]. Available: https://www.openhub.net. [Accessed: 02- Apr- 2019].

[3] T. Amanatidis and A. Chatzigeorgiou, "Studying the evolution of PHP web applications", *Information and Software Technology*, vol. 72, pp. 48-67, 2016.

[4] L. Prechelt, I. Universit, and K. Germany, "Are Scripting Languages Any Good ? A Validation of Perl, Python, Rexx, and Tcl against C, C + +, and Java," n: Advances in Computers, Academic Press, San Diego, vol. 58, pp. 1–62, 2002.

[5] A. Ortiz, "Web development with Python and Django (abstract only)", *Proceedings of the 43rd ACM technical symposium on Computer Science Education - SIGCSE '12*, 2012.

[6] A. Rio and F. e Abreu, "Analyzing web applications quality evolution", *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*, 2017. Available: 10.23919/cisti.2017.7975959.
J.K. Ousterhout, "Scripting: higher level programming for the 21st century," Info.Softw. Technology. Vol. 72,pp.48-67,2016.

[7] S.Aprana, A.Mishra, "A Systematic Review on measuring and evaluating web usability in Model-Driven Web Development", International Journal of Engineering Development and Research (IJEDR), ISSN:2321-9939, Volume.2, Issue NCETSE Conference, pp.171-180, March 2014.

[8] .S. Tillsey, "15 Years of web systems evolution," *2013 15th IEEE International Symposium on Web Systems Evolution (WSE)*, Eindhoven, pp. 3-4,2011.

[9] W. Hall and T. Tiropanis, "Web evolution and Web Science", *Computer Networks*, vol. 56, no. 18, pp. 3859-3865, 2012. Available: 10.1016/j.comnet.2012.10.004.

[10] A. Baravalle, C. Boldyreff, A. Capiluppi, and R. Marques, "On the sustainability of web systems evolution," *2013 15th IEEE International Symposium on Web Systems Evolution (WSE)*, Eindhoven, 2013, pp. 31-34.

[11] G. Di Lucca, A. Fasolino and P. Tramontana, "Reverse engineering Web applications: the WARE approach", *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 16, no. 12, pp. 71-101, 2004.

[12] A. Baravalle, C. Boldyreff, A. Capiluppi, and R. Marques, "On the sustainability of web systems evolution," *2013 15th IEEE International Symposium on Web Systems Evolution (WSE)*, Eindhoven, 2013, pp. 31-34.

[13] D. Fetterly, M. Manasse, M. Najork and J. Wiener, "A large-scale study of the evolution of Web pages", *Software: Practice and Experience*, vol. 34, no. 2, pp. 213-237, 2004.

[14] J. R. M. Camilo, A. L. Erario, and J. A. Fabri, "A Process for Distributed Software Evolution A proprietary software case study," Proceedings of the 13th International Conference on Global Software Engineering, Gothenburg, Sweden, pp. 44–53, 2018

[15] A. Talai and Z. E. Bouras, "Software evolution based activity diagrams," *ICIT 2017 - 8th Int. Conf. Inf. Technol. Proc.*, no. 1995, pp. 82–88, 2017.

[16] M. Linares-Vásquez, K. Moran, and D. Poshyvanyk, "Continuous, evolutionary and large-scale: A new perspective for automated mobile app testing," *Proc. - 2017 IEEE Int. Conf. Softw. Maint. Evol. ICSME 2017*, pp. 399–410, 2017.

[17] Y. Higo and S. Kusumoto, "Flattening code for metrics measurement and analysis," *Proc. - 2017 IEEE Int. Conf. Softw. Maint. Evol. ICSME 2017*, pp. 494–498, 2017.

[18] M. Gupta, "Improving Software Maintenance Using Process Mining and Predictive Analytics," *2017 IEEE Int. Conf. Softw. Maint. Evol.*, pp. 681–686, 2017.

[19] G. Destefanis, M. Ortu, S. Porru, S. Swift, and M. Marchesi, A Statistical Comparison of Java and Python Software Metric Properties, the International Workshop on Emerging Trends in Software Metrics, WETSoM 2016. Association for Computing Machinery, Inc. 2016.

[20] K. Aggarwal, A. Hindle, and E. Stroulia, "GreenAdvisor : A Tool for Analyzing the Impact of Software Evolution on Energy Consumption," IEEE International Conference on Software Maintenance and Evolution (ICSME) pp. 311–320, 2015.

[21] T. Chaikalis, E. Ligu, G. Melas, and A. Chatzigeorgiou, SEAgle : Effortless Software Evolution Analysis, ICSME pp. 582–585, 2014.

[22] M. D. Syer, "The Maintenance and Evolution of Field-Representative Performance Tests," *2014 IEEE Int. Conf. Softw. Maint. Evol.*, pp. 665–665, 2014.

[23] V. Rajlich, Software evolution, and maintenance, *Proceedings of the on Future of Software Engineering*, pp. 133-144, 2014.

[24] P. Kyriakakis and A. Chatzigeorgiou, Maintenance Patterns of large-scale PHP Web Applications, IEEE International Conference on Software Maintenance and Evolution Maintenance, pp.381-390, 2014.

[25] K. Duran, G. Burns, and P. Snell, "Lehman's laws in agile and non-Agile projects," *Proc. - Work. Conf. Reverse Eng. WCRE*, pp. 292–300, 2013.

[26] C. F. Kemerer and S. Slaughter, "An empirical approach to studying software evolution," IEEE Trans. Softw. Eng., vol. 25, no. 4, pp. 493–509, 1999.

[27] M. W. Godfrey and Q. T. Q. Tu, "Evolution in open source software: a case study," Softw. Maintenance, 2000. Proceedings. Int. Conf., pp. 131–142, 2000.

[28] I. Neamtiu, G. Xie, and J. Chen, "Towards a better understanding of software evolution: An empirical study on open-source software," J. Softw. Evol. The process, vol. 25, no. 3, pp. 193–218, 2013.

[29] A. Capiluppi, "An Empirical Study of the Evolution of an Agile Developed Software System", *Proc. 29th Int'l Conf. Software Eng. (ICSE 07)*, pp. 511-518, 2007.

[30] R. Sindhgatta, N. C. Narendra, and B. Sengupta, "Software Evolution in Agile Development: A Case Study," *SBIE - Simpósio Bras. Informática na Educ.*, pp. 105–114, 2010.

[31] R.P. Oliveira, E.S. Almeida, G.S.S. Gomes, "Evaluating Lehman's Laws of Software Evolution within Software Product Lines: A Preliminary Empirical Study", *Proc. 14th Int'l Conf. Software Reuse*, pp. 42-57, 2015.

[32] B. Singh and P. Luthra, "Study of Lehman's Laws and Metrics during Software Evolution," *Int. J. Comput. Syst.*, vol. 226, no. 06, pp. 2394–1065, 2394.

[33] [34] T. Mens, S. Demeyer, M. Wermelinger, R. Hirschfeld, S. Ducasse, and M. Jazayeri, "Challenges in software evolution," *Int. Work. Princ. Softw. Evol.*, vol. 2005, pp. 13–22, 2005.

[34] D. Kafura and G. R. Reddy, "The Use of Software Complexity Metrics in Software Maintenance," *IEEE Trans. Softw. Eng.*, vol. SE-13, no. 3, pp. 335–343, 1987

[35] A. Michael, "Empirical Study of Cyclomatic Complexity and Interface Complexity of Evolving Open Source Systems," Daffodil International University Journal of Sciences and Technology, vol. 12, no. 1, 2017.

[36] T. Kaur, Applicability of Lehman Laws on Open Source Evolution : A Case study Applicability of Lehman Laws on Open Source Evolution : A Case study, International Journal of Computer Applications,93(18):40-46, May 2014.

[37] Skoulis, I., Vassiliadis, P., Zarras, A.: Open-source databases: within, outside, or beyond Lehman's laws of software evolution? In: Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., Horkoff, J. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 379–393. Springer, Heidelberg (2014).

[38] Herraiz, I., Rodriguez, D., Robles, G., Gonzalez-Barahona, J.M.: The evolution of the laws of software evolution: a discussion based on a systematic literature review. ACM Comput. Surv. 46(2), 1–28 (2013).

[39] J. Zhang, S. Sagar, and E. Shihab, "The evolution of mobile apps: an exploratory study," Proc. 2013 Int. Work. Softw. Dev. Lifecycle Mob. - DeMobile *2013*, pp. 1–8, 2013.

[40] O.Oluwagbemi, A. Adewumi, and L.Fernandez-sanz, An Analysis of Scripting Languages for Research in Applied Computing," Conference on Computational Sciences and Engineering, pp 174-179 (2013).

[41] L. Yu, A. Mishra, "An empirical study of Lehman's law on software quality evolution", *International Journal of Software & Informatics*, vol. 7, no. 3, pp. 469-481, 2013.

[42] L.A. Belday & M. M. Lehman, "A model of large program development," *BM Syst. J.*, vol. 15, pp. no. 3, pp. 225–252, 1976.

[43] M. M. Lehman, "Laws of software evolution revisited," *Lect. Notes Comput. Sci. (including Subsea. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 1149, pp. 108–124, 1996.

[44] 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," *Proc. Fourth Int. Softw. Metrics Symp.*, pp. 20–32, 1997.