ISNNI: 3007-3391 HEC Category: Y

Journal of Computers and Intelligent Systems
Vol. 4, No. 3, 2026, pages 393 - 406
Journal Homepage:
https://journals.iub.edu.pk/index.php/JCIS/

Evaluation of Web Application Evolution Using Lehman's Laws: A Comparative Study of Python, Perl, and PHP Frameworks

Kanza Nasim[1], Syed Mudassar Alam[2], Hamid Nawaz[3], Naila Batool[3], Muhammad Usman[4]

[1]Department of Computer Information Systems Engineering, NED University of Engineering and Technology, Karachi, Pakistan
[2]Department of Computer Science, National University of Modern Languages, Islamabad, Pakistan
[3]Department of Software Engineering, National University of Modern Languages, Islamabad, Pakistan
[4]Department of Computer Science, The University of Faisalabad

ARTICLE INFO

Article History:

Received: July 14, 2026
Revised: August 24, 2026
Accepted: September 23, 2026
Available Online: September 30, 2026

Keywords:

Lehman's Laws
Software Evolution
Web Applications
Programming Languages
Software Metrices

Classification Codes:

Funding:

This research received no specific grant from any funding agency in the public or not-for-profit sector.

ABSTRACT

Web applications are always evolving in accordance with new requirements from users and technological innovations. It is necessary to evaluate this evolution since uncontrolled changes can add complexity to the system, reduce its quality and affect its sustainability. This study explores the evolution of chosen web applications by employing Lehman's Laws of Software Evolution as the core theoretical concept for the analysis. This study focuses on three popular open-source web applications: Django for Python, Catalyst for Perl, and WordPress for PHP. The versions of each application were analyzed using several software metrics, such as time between releases, number of lines of code, cyclomatic complexity, number of comments, rate of modifications, common ratio, and growth of source lines of code. The metrics were examined in relation to Lehman's eight laws of software evolution. Estimated regression analysis was also used to examine the observed trends. These findings show that the observed trends provide support for some of Lehman's Laws, mainly those connected with software growth and change. On the other hand, the laws which describe decreasing quality and feedback systems have not been confirmed for all the chosen projects. The general conclusion from these findings is that Lehman's Laws can be used as an effective way to analyze the process of evolution of web applications. However, their applicability may vary depending on a particular project, metrics, and development techniques used. Some metrics also provide only indirect evidence of the corresponding laws. Thus, this research contributes to software evolution studies by providing a comparative and metrics-based analysis of web projects created in PHP, Python and Perl.

OPEN ACCESS

© 2026 The authors published by JCIS. This is an Open Access Article under the Creative Common Attribution Non-Commercial 4.0

Corresponding Author's Email: mudassar.alam@numl.edu.pk

1. Introduction

The rapid growth of web applications has significantly transformed the way individuals, businesses, and organizations interact with digital technologies. Web-based applications are increasingly used for purposes such as communication, trade, learning, health care, finance, governance, and management of organizations. With continuous change in the needs of users and technology, web applications have to develop as well. Therefore, examining how web applications evolve has become an important research issue in software engineering. Software evolution refers to the continuous modification and maintenance of a software system after its initial development and deployment. Web applications that do not evolve to meet user expectations as well as new technology trends may gradually show poor performance. In practical software development environments, there is evidence that

various attributes like modifications in the source code, size and complexity of software, documentation, and maintenance directly affect the longevity of a software system. Some recent studies on software maintainability have also indicated that maintainability metrics continue to be relevant for software behavior [1], [2]. Lehman's Laws of Software Evolution [3] provide a foundational framework for studying the growth and transformation of software systems over time. Applying Lehman's laws allows researchers and developers to assess performance, anticipate problems, and design systems that will manage the change efficiently in web applications. This study aims of link theoretical knowledge to practical concepts for improving software quality and maintainability by employing Lehman law. Scripting languages such as Python, Perl, and PHP have been widely used in web application development. These languages are preferred due to quick development, code reusability, flexibility, and interaction with servers and databases. Previous studies have also indicated that the evolution of web-based scripting languages can be evaluated using software metrics [4], [5].

Django is a Python web framework, Catalyst is a Perl framework, and WordPress is a PHP-based platform. These frameworks are appropriate for analysis since they are open source, popularly adopted, and exist in many versions. The use of metrics in software engineering is quite common and they have been extensively used to examine various aspects of software systems including the quality, complexity, and evolution of software products. However, most previous metric-based studies have focused on studying the software for general purpose software, or desktop applications. Previous studies have used metrics such as cyclomatic complexity, coupling, and lines of code to inspect changes in software quality across different versions [6].

The aim of the current study is to investigate how chosen web projects have evolved using the laws of software evolution of Lehman. The three web projects that are chosen for investigation are Django, Catalyst, and WordPress, where Django is associated with Python web development, Catalyst is related to Perl web development, and WordPress is concerned with PHP web development. Several software metrics were used to examine their evolution. The main contribution of this study is the comparison of web applications developed using Python, Perl, and PHP. This study provides insights into the ways in which the selected web applications evolve over time, how the complexity of the chosen web applications changes, and if they satisfy the rules of software evolution proposed by Lehman. These findings may be useful for developers and researchers.

2. Related Work

Software evolution is an important field of software engineering because software products continue to change after their release into the market. According to Lehman, there exist some laws which can help to describe the behavior of software products from a long-term perspective. It is necessary to admit that these laws can be considered as an adequate theoretical basis of investigation into expanding and adapting software systems while maintaining its quality. A number of studies have used Lehman's Laws for investigating the evolutionary behavior of various software systems. Kaur et al. [7] have examined the applicability of these laws to open-source software systems and concluded that all the laws cannot be applicable to every system. Software evolution is usually measured with various types of metrics such as lines of code (LOC), cyclomatic complexity (CC) and number of comments (NOC). Othres metrics include release frequency, modification rate, and source code growth. Destefanis et al. [5] used Java and Python project and compared software metrics. Their study found that examine the software metrics is necessary across programming language. Nowadays, the source code, version history and release information are publicly available. These are the key factors that are widely used in software evolution research. Recent studies have focused on open-source software's project comparison and examined how these projects compared with each other [14], [15].

Pair programming and software refactoring also paly a vital role in software quality. Web applications evolve because they are frequently updated in response to user, security update and changed business requirements.
Modern internet features and semantic web components, mobile support and intelligent features make this more complex [16], [17]. Traditional methods may not meet the modern requirements, so adaptive web engineering approaches are needed.
Recent studies have also highlighted the importance of systematic approaches for handling architectural changes and recurring evolution problems [18]. Software evolution has also been studied in different application domains. Recent research has highlighted the effects of code smells on software quality and maintainability [21]. Studies on Android applications have focused on compatibility problems resulting from software evolution [10]. Maintenance-related studies have introduced classification models for software activities and changes [11]. Automated test repair for visual tests can

help maintain web applications that undergo the evolution process. Although numerous studies have examined software evolution, few have conducted comparative analyses of web-based applications developed in different scripting languages using Lehman's Laws [12]. Popular scripting languages such as Python, Perl, and PHP can be used to develop web applications, but their evolution processes may differ due to differences in communities, project structures, release practices, and maintainability. Therefore, this study aims to fill this research gap by evaluating Python-based, Perl-based, and PHP-based projects, represented by Django, Catalyst, and WordPress, respectively, using software metrics.

3. Problem Statement and Proposed Solution

Open-source web applications were selected because their version history and source code files can be analyzed from the perspective of software evolution. The selected web applications had to meet the following conditions:

1. The source code has to be publicly available.
2. The selected application has to be related to web application development.
3. The application has to have multiple versions/releases.
4. There should be enough source code to extract software metrics.
5. The selected application should use any of the scripting languages used in web application development.
6. The selected application should be object-oriented wherever applicable.

These projects were chosen because they are publicly available in repositories and have version histories. So, these web applications are enough for software evolution analysis.

3.1 Data Analysis Metrics

In order to assess the selected projects using Lehman's Laws, various software metrics were identified for each of the eight laws of software evolution. These metrics have been selected due to their ability for change, maintainability and feedback behaviors. Table 1 below presents the relationship between the Lehman's laws, the metrics and the analysis method.

Table 1: Data Analysis Metrics Used for Lehman's Laws Evaluation

Evolution Law	Measured Metric	Analysis Method
Continuing Change	Number of days between consecutive releases	Trend Evaluation using release history
Increasing Complexity	Cyclomatic Complexity Number (CCN) divided by Lines of Code (LOC)	Emerging Trends and Patterns using software metric values
Self-Regulation	Incremental growth observed between software versions	Temporal Evolution Analysis using version-wise growth patterns
Conservation of Organizational Stability	Number of code comments	Trend Evaluation using comment-line count
Conservation of Familiarity	Rate of modifications per year	Trend Evaluation using yearly modification rate
Continuing Growth	Total lines of source code	Trend analysis using LOC growth

Declining Quality	Common ratio based on code and comment relationship	Analysis using quality-related ratio
Feedback System	Growth of source lines of code in response to project evolution	Trend analysis using code growth behavior

3.1.1 Justification of Software Metrics

The selected software metrics were used to analyze the growth in software, complexity, release history, and code modifications. NOC records changes in code comments, and Number of Modifications records code modifications. Comment Ratio gives the proportion of comments in the source code. These software metrics were not used to measure Lehman's Laws directly; they were used to find trends in software evolution.

3.1.2 Reliability of Metric-Based Evaluation

One software metric was used to study each Lehman Law separately. This will help to find the change in software over the period of time. However, one single metric may not fully represent a law, so the results were taken with great attention.

3.2 Classification of Software Measurement Standards

The measurement standards for software provide a quantifiable basis for the analysis of the software system. Three types of measurement can be used in the web-based and object-oriented based project. This measurement can be applied at system level, process level and product level. The system level measures size and structure of the software, while the process level measures define the development and maintenance process. The product level measures feature aspects of the source code including complexity, comments, and size. This can be used for the comparison and for the trend evolution.

3.3 Tools and Techniques for Data Analysis

Various methods have been used to gather and analyze the necessary metric. Trend analysis was used to study how the metric values varied over versions of the chosen projects. Radon was used for Python as it uses source lines of code, comment ratios, and cyclomatic complexity metrics. Perl lint-type programs were used as they have the capability of static code analysis. PHP Depend was used for PHP as it provides various software metrics like cyclomatic complexity and code size. To analyze the slop trend for the collected data, Microsoft Excel was used to draw graphs and trend analysis.

3.4 Selection of Projects for Evolution Analysis

Django (Python), Catalyst (Perl), and WordPress (PHP) were selected as case studies by using Lehman's Laws of Software Evolution. These were selected as all open-source projects. Django and Catalyst are famous web frameworks with excellent technical foundations. The WordPress project has also become very popular owing to its user-friendliness and number of websites supported by it. Table 2 summarizes the selected project characteristics.

Table 2: Selected Projects for Lehman's Law Evaluation

Project Name	Representative Language	Lines of Code (LOC)	Comment Lines	Blank Lines
Django 4.0	Python	395,000	50,000	65,010

Catalyst 3.0.0	Perl	540,000	157,220	121,100
WordPress 6.7	PHP	6,500,000	185,100	111,300

3.5 Data Analysis Framework

Analysis of the data began with extraction of all files in the selected object-oriented projects and calculating software metrics at class level for different types of source code. The metrics were then analyzed for each project as shown in Figure 1. Each law was associated with the metrics generated for the selected projects and tested using trend line slope test as below:

H0: No trend is found in the data.

H1: There is a trend in the data

Linear regression was used to examine trends in the selected metrics. The numerical values were estimated from the existing graphs. Regression slopes, R^2 values, and p-values were calculated using Python. The results are approximate and require verification using the original data.

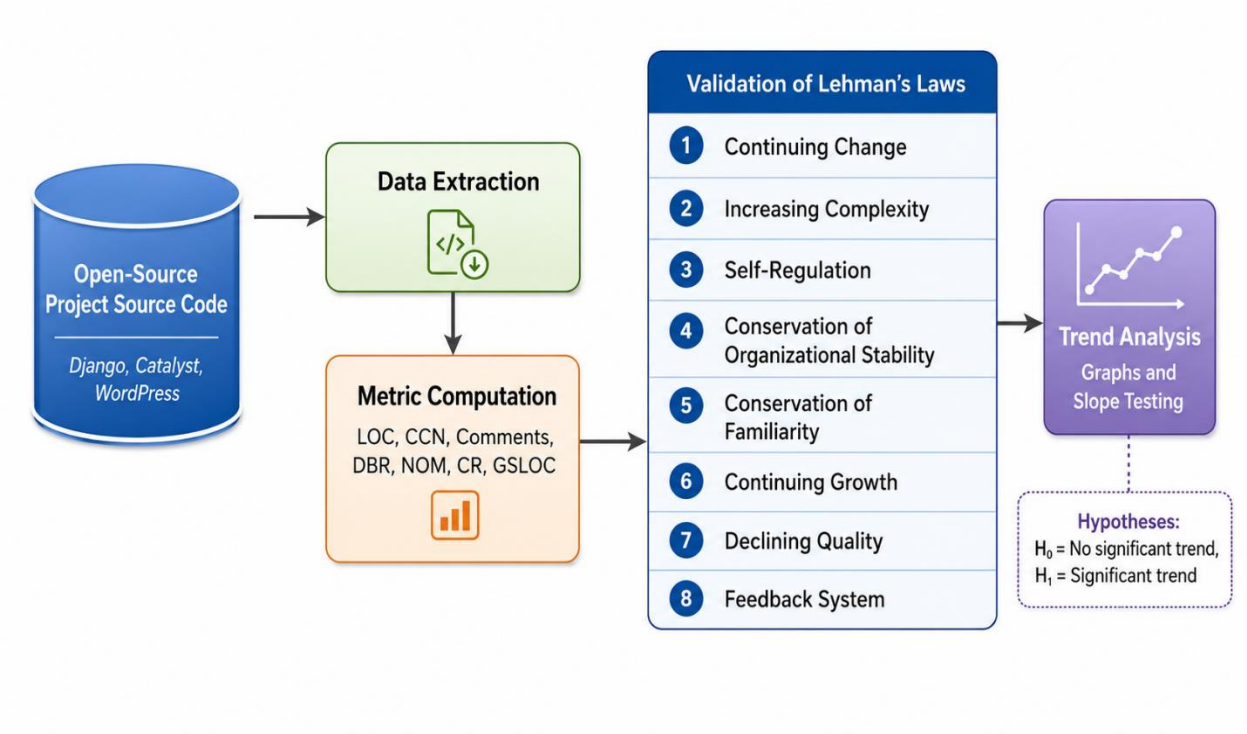


Figure 1. Proposed workflow for evaluating web application evolution using Lehman's Laws

4. Experimental Results

This study examined Django, Catalyst, and WordPress based on Lehman's eight laws of software evolution. Several software metrics were used to measure changes in software size, complexity, and development over time. The results indicate that some metrics follow the expected trends, while others

show different results. Graphs and estimated statistical results were used to study these changes. The following sections discuss the findings for each law.

4.1 Statistical Trend Analysis of Software Evolution Metrics

Linear regression analysis was employed in order to determine the software evolution in terms of the considered framework. The regression slope (β_1), coefficient of determination (R^2), and p-value were calculated for each metric. A significance level of 0.05 was used to examine the experimental linear trends. Table 3 summarizes the statistical results against different measures.

Table 3: Statistical Trend Analysis of Software Evolution Metrics Across Django, Catalyst, and WordPress.

Lehman's Law	Project	n	Slope (β_1)	R^2	p-value	Statistical Result
1. Continuing Change	Django	10	-16.61	0.349	0.0724	Not significant
	Catalyst	10	-4.30	0.386	0.0551	Not significant
	WordPress	10	20.12	0.396	0.0512	Not significant
2. Increasing Complexity	Django	10	79.70	0.302	0.1001	Not significant
	Catalyst	11	22.45	0.789	0.0003	Significant
	WordPress	8	1,189.29	0.909	0.0003	Significant
3. Self-Regulation	Django	8	37,678.57	0.993	<0.0001	Significant
	Catalyst	8	39,702.38	0.986	<0.0001	Significant
	WordPress	8	34,880.95	0.882	0.0005	Significant
4. Organizational Stability	Django	15	3,428.57	0.772	<0.0001	Significant
	Catalyst	16	13,767.65	0.817	<0.0001	Significant
	WordPress	14	14,072.53	0.872	<0.0001	Significant
5. Conservation of Familiarity	Django	14	-6.38	0.074	0.3478	Not significant
	Catalyst	4	-6.00	0.047	0.7824	Not significant
	WordPress	11	0.09	0.000	0.9974	Not significant
6. Continuing Growth	Django	10	22,303.03	0.990	<0.0001	Significant
	Catalyst	11	5,236.36	0.384	0.0419	Significant
	WordPress	11	49,409.09	0.953	<0.0001	Significant
7. Declining Quality	Django	12	-0.025	0.493	0.0109	Significant
	Catalyst	12	-0.015	0.191	0.1551	Not significant
	WordPress	13	0.024	0.716	0.0003	Significant
8. Feedback System	Django	13	-49.45	0.001	0.9371	Not significant
	Catalyst	11	-8,363.64	0.174	0.2020	Not significant
	WordPress	13	233.52	0.088	0.3250	Not significant

4.2 Laws of Continuing Change

The study applied Lehman's Laws to three web frameworks and found that some software metrics support the laws, while others do not show a direct relationship. The law states that software must continuously evolve to meet user needs. The Days Between Releases (DBR) metric is used for trend analysis and is applied separately to each project to check whether the law is being followed. Hypotheses are created to determine whether a trend exists between versions. The results are shown in Figures 2, 3, and 4. The figures show that each framework exhibited some growth. Django shows some variation over time, Catalyst has shorter intervals between releases, and WordPress shows an increasing trend. These differences may reflect different development practices. However, regression results show no substantial linear trends.

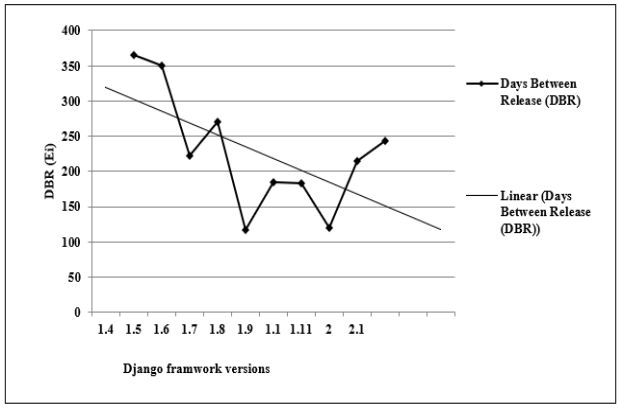


Figure 2: DBR Analysis of Django

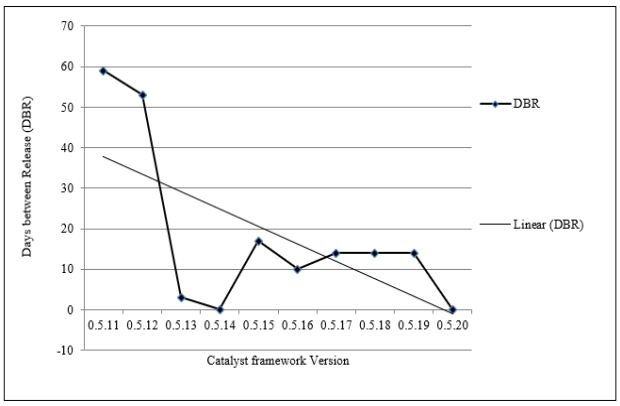


Figure 3: DBR Analysis of Catalyst

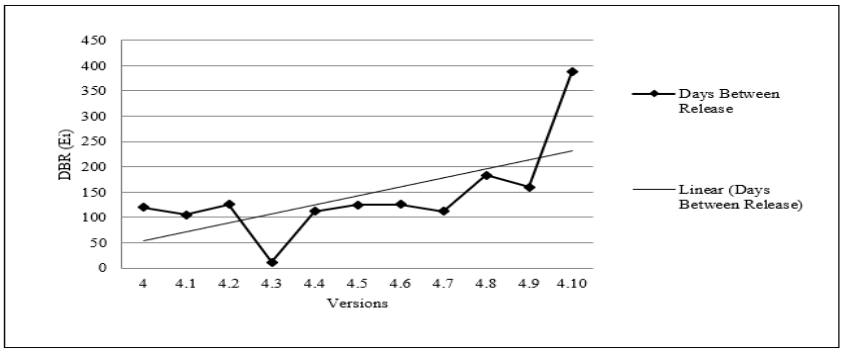


Figure 4: DBR Analysis of WordPress

4.3 Law of Increasing Complexity

As software size increases, software complexity also increases due to added features. Framework Size (SB) was tested for all projects, including Django, Catalyst and WordPress. The gradual increase in software size shows that software complexity increases along with software size, thus supporting Lehman's Law. Figures 5, 6, and 7 show these results. These statistical findings consequently give support for this law.

4.4 Law of Self-Regulation

Software is expected to control its own growth over a given time period. For Incremental Changes (IC), the software was tested across three languages. The results show a positive trend, supporting this law. Figure 8 presents this result and shows how lines of code have increased over the years. These trends do not directly confirm self-regulation because the stability of growth also needs to be observed with other factors.

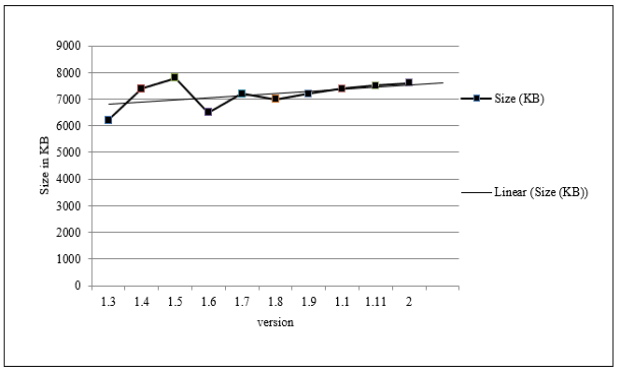


Figure 5: SB Analysis on Django

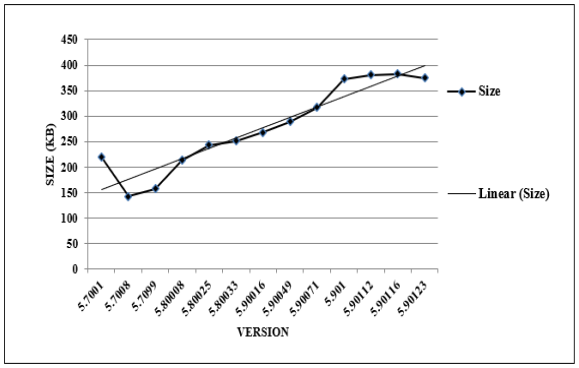


Figure 6: SB Analysis on Catalyst

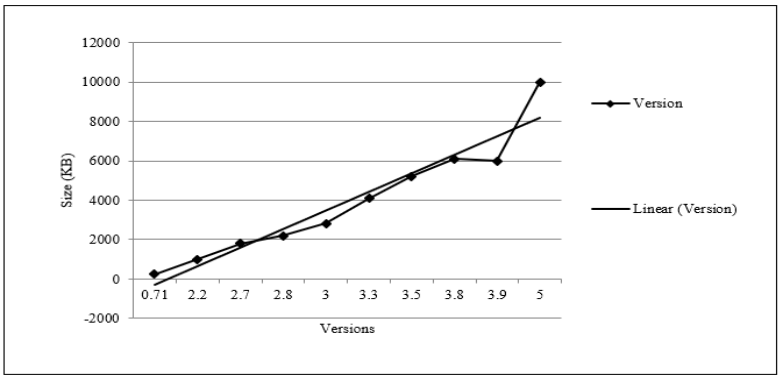


Figure 7: SB Analysis on WordPress

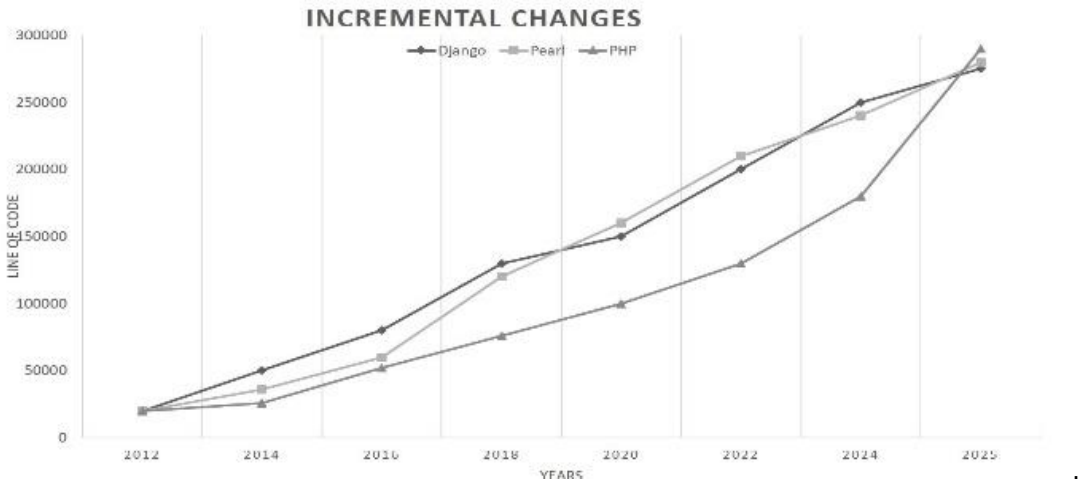


Figure 8: SB IC Analysis on Django, Catalyst, and WordPress

4.5 Law of Conservation of Organizational Stability

NOC is used as a metric to measure organizational stability in software development projects. In all three frameworks, there is a significant increase in comments, indicating a positive trend. This incremental growth in comments supports Lehman's Law for the selected frameworks, as shown in Figures 9, 10, and 11. The statistical analysis also indicates continued documentation activity, but it does not directly confirm organizational stability as other key factors also included in confirming the organization stability.

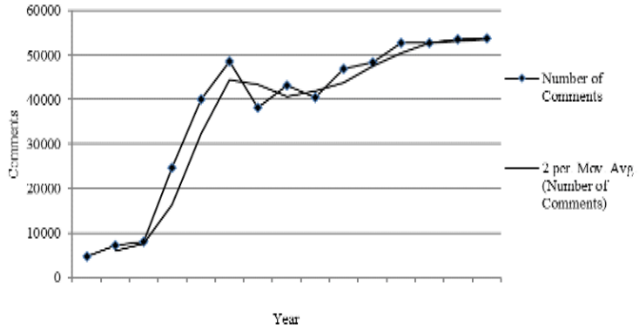


Figure 9: NOC for Django

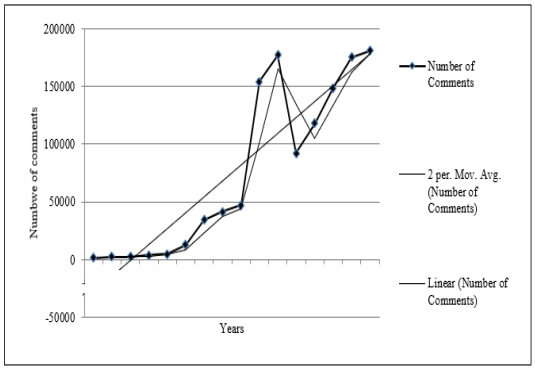


Figure 10: NOC for Catalyst

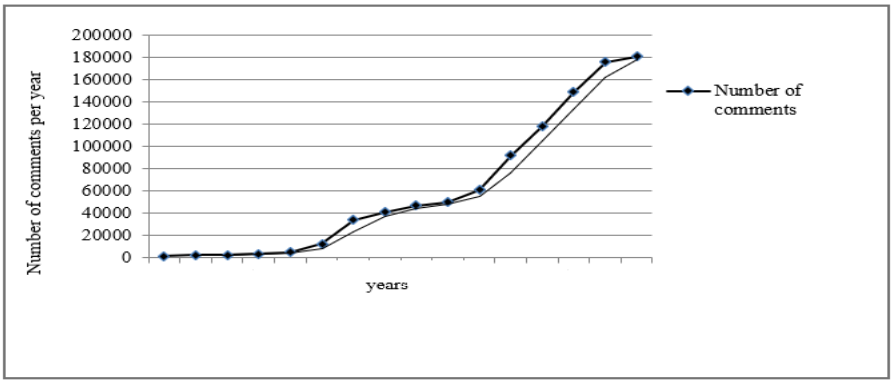


Figure 11: NOC for WordPress

4.6 Law of Conservation of Familiarity

Software should remain familiar to users despite changes. This means that developers should manage changes carefully to maintain quality and usability. This law also examines whether there is a trend in the Number of Modifications (NOM) in the code. Figures 12, 13, and 14 show a partial positive trend, supporting the law of conservation of familiarity.

4.7 Law of Continuing Growth

Lehman's law of continuing growth suggests that software systems continue to grow over time to meet increasing user needs. To validate this law, the trend in Lines of Code (LOC) was evaluated. Similar to the evaluation of the law of increasing complexity, this validation process considers all selected projects. As shown in Figures 15, 16, and 17, all frameworks show preliminary support growth, thereby supporting this law.

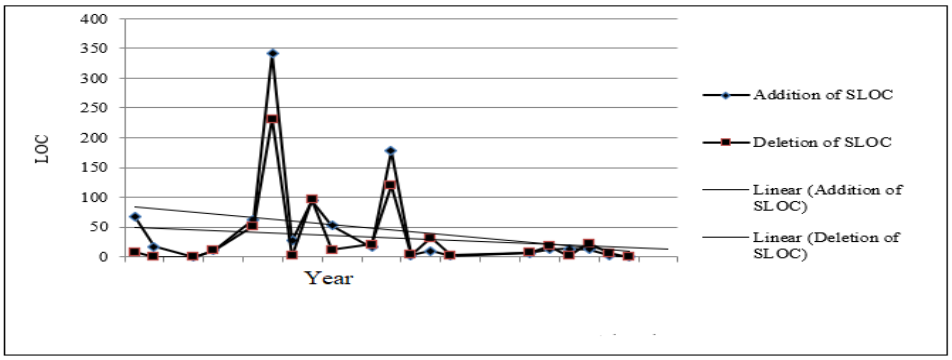


Figure 12: NOM for Django

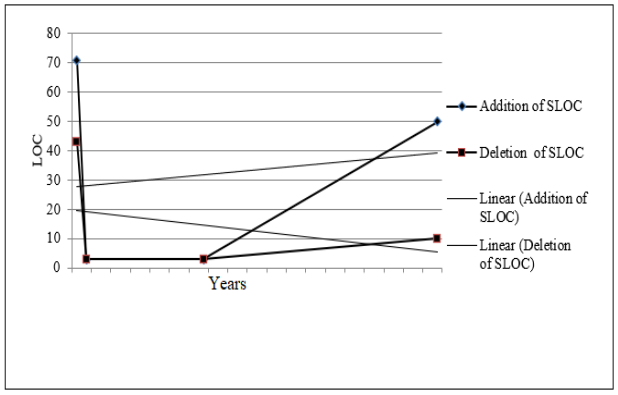


Figure 13: NOM for Catalyst

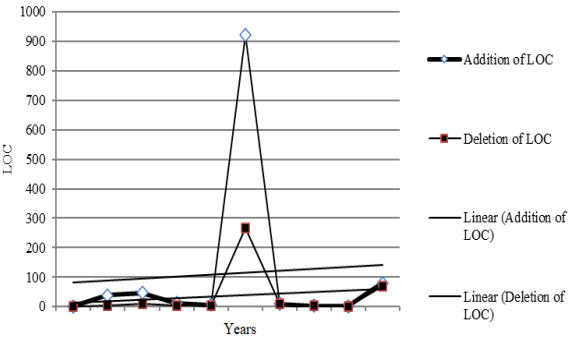


Figure 14: NOM for WordPress

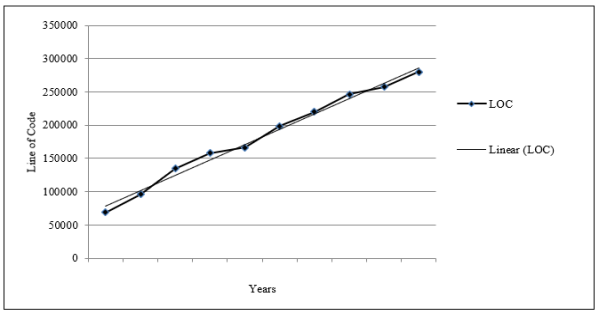


Figure 15: LOC for Django

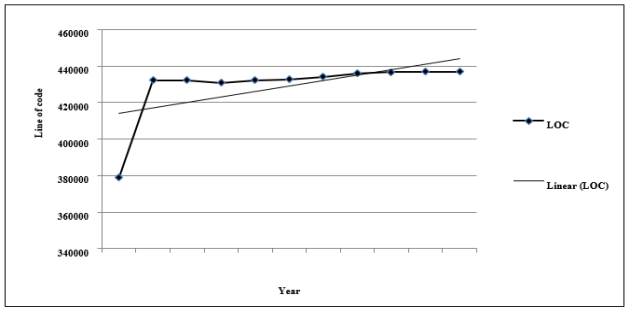


Figure 16: LOC for Catalyst

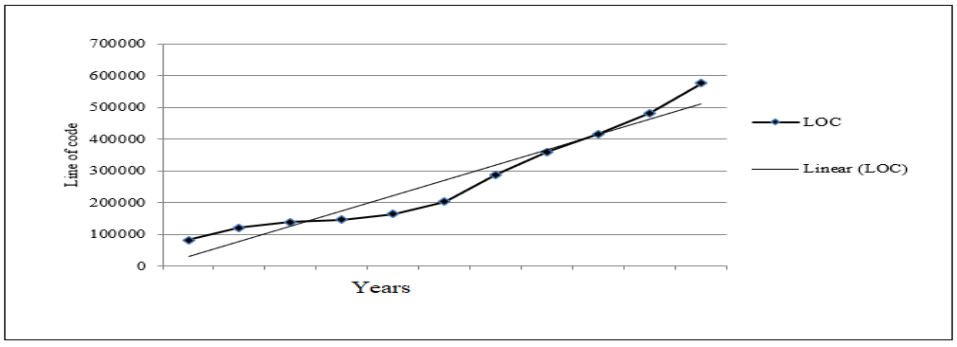


Figure 17: LOC for WordPress

4.8 Law of Declining Quality

According to this law, software quality may decline over time. This law can be validated by analyzing the Common Ratio (CR), which is the ratio of comment lines to the total lines of code, for the three web frameworks. Figures 18, 19, and 20 indicate that two of the three applications demonstrated a positive trend, whereas Django showed a continuous downward trend. Projects developed with Django may require special attention to address challenges effectively and on time.

4.9 Law of Feedback System

Lehman's law of feedback system states that software should handle customer feedback efficiently without compromising quality. To test this law, the growth rate of Source Lines of Code (GSLOC) was analyzed. Only Django shows some decline in the initial years, while the other two projects remain stable. Based on the results, this law is not fully applicable to these systems. Figures 21, 22, and 23 present the results Source code growth varied across the three projects. No significant linear trends were found in the estimated results ($p > 0.05$). These patterns do not provide sufficient evidence of feedback-driven evolution.

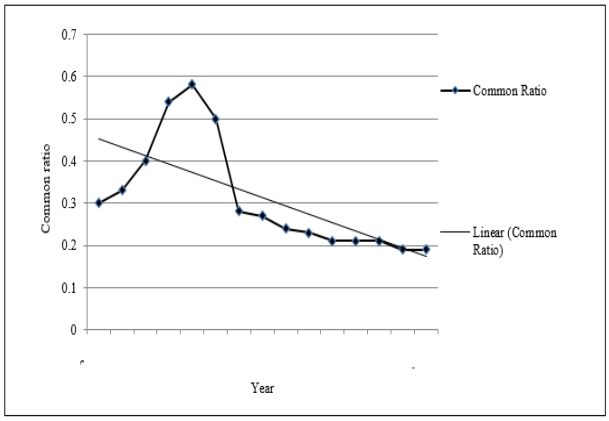


Figure 18: CR for Django

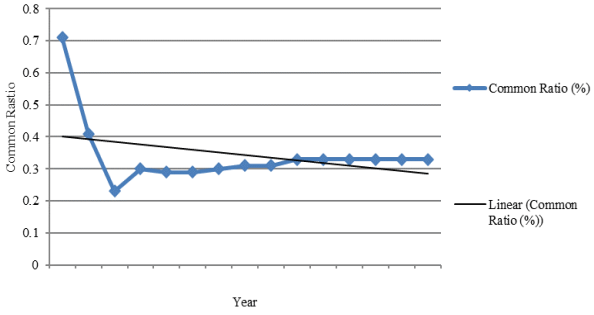


Figure 19: CR for Catalyst

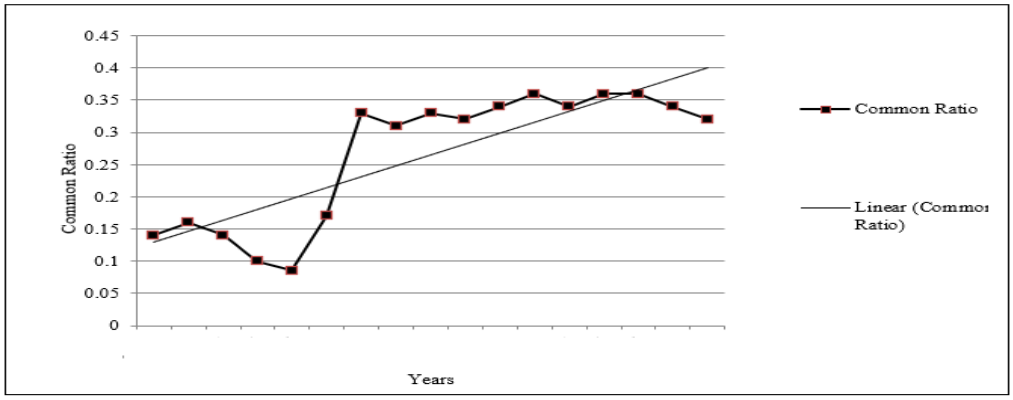


Figure 20: CR for WordPress

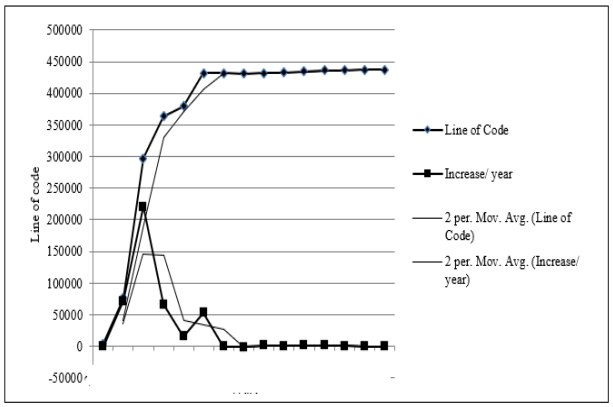


Figure 21: GSLOC for WordPress

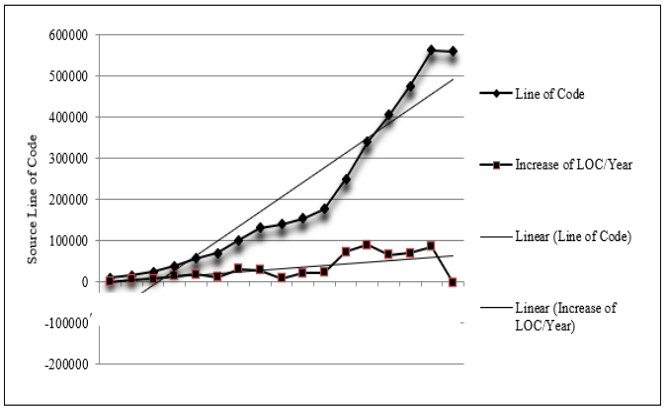


Figure 22: GSLOC for WordPress

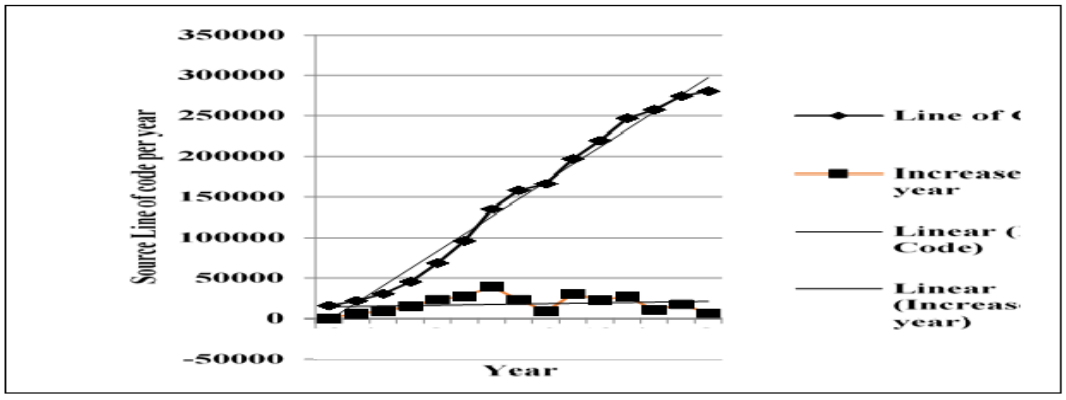


Figure 23: GSLOC for Django

4.10 Comparison with Previous Studies

The results obtained in the current study partially agree with some of the previous stuides. Kaur et al. [7] observed significant support for continuing change, growth, and complexity. Smajić et al. [8] observed continuous growth in open-source systems. Neamtiu et al. [19] observed continuing change and growth while the results for the other laws were inconsistent. Amanatidis and Chatzigeorgiou [4] observed significant support for continuing change and growth in PHP applications. Israeli and Feitelson [20] observed software growth but in cases where complexity did not necessarily increase. The present study shows as there was significant estimated growth trend in Django, Catalyst, and WordPress. There was no significance in the estimates for the release interval trend while the results for declining quality and feedback trend were inconclusive. It is therefore clear that Lehman's Laws can have different applicability among software projects. Table 4 shows the comparison of the results obtained from previous studies with the present study according to Lehman's laws.

All three projects differ in terms of size and development approaches. Whereas Django and Catalyst are frameworks for building web applications and WordPress is a Content Management System (CMS). These factors may influence the code base grows, releases and maintenance. The results should be taken in lieu of each project's characteristics and not only in terms of programming language. Ameer et al. [21] discussed the role of code smells in software quality. Their finding states that additional software quality metrics are need to measure the software quality as single measure cannot define the quality accurately.

Table 4: Comparison of Lehman's Laws of Software Evolution Across Previous Studies and Present Study.

Lehman's Law	Kaur et al. [7]	Smajić et al. [8]	Neamtiu et al. [19]	Amanatidis & Chatzigeorgiou [4]	Israeli & Feitelson [20]	Present Study
Continuing Change	●	●	●	●	●	●
Increasing Complexity	●	–	●	–	●	●
Self-Regulation	–	–	–	–	–	●
Conservation of Organizational Stability	–	–	–	–	–	●
Conservation of Familiarity	–	–	–	–	–	●
Continuing Growth	●	●	●	●	●	●
Declining Quality	–	–	●	–	–	●
Feedback Systems	–	–	●	–	–	●

● Reported support ● Mixed / partial findings – Not reported

5. Conclusion

The evolution of the selected web-based projects, evaluated according to the principles of the Lehman's Laws of Software Evolution. Software metrics Include release interval, lines of code, cyclomatic complexity, number of comments, modification rate, common ratio, and source code growth have been analyzed for the selected projects. The findings reveal a continuous growth and evolution of all the selected projects. Statistical analysis revealed significant growth trends; meanwhile, release intervals and modification rates did not reveal any linear trends. Declining quality and feedback system laws partly confirmed for the Selected projects. Based on the selected software metrics, the evolution of the chosen PHP-based project has revealed the most prominent features of growth and adaptation.

There are some limitations of this study Each law was analyzed using only one software metric. The study focusses on only the web applications. Future studies should on evolution of more applications of desktop and mobile platforms, increase the data set size and the number of programming languages used. In addition, it is also important to conduct an analysis based on source code metrics like git logs and bugs reports for software evolution.

6. Acknowledgements

Author is grateful to Mr. Abuzar Firdousi and his team for providing the necessary infrastructure to carry out this work.

7. References

[1] S. M. Syed-Mohamad, A. Ngah, A.-F. M. Ali, and P. Keikhosrokiani, "Measuring software maintainability: An exploration of metrics and continuous practices," *Journal of Advanced Research in Applied Sciences and Engineering Technology*, vol. 63, no. 2, pp. 181–195, 2025, doi: 10.37934/araset.63.2.181195.

[2] S. Rochimah, T. R. Hadiningrum, B. D. Mardiana, D. O. Siahaan, J. A. Rizky, and M. S. Ary, "A maintainability framework to ensure the software quality in object-oriented programming," *IEEE Access*, vol. 13, pp. 195796–195821, 2025, doi: 10.1109/ACCESS.2025.3633265.

[3] M. M. Lehman, “Programs, life cycles, and laws of software evolution,” *Proceedings of the IEEE*, vol. 68, no. 9, pp. 1060–1076, 1980, doi: 10.1109/PROC.1980.11805.

[4] T. Amanatidis and A. Chatzigeorgiou, “Studying the evolution of PHP web applications,” *Information and Software Technology*, vol. 72, pp. 48–67, 2016, doi: 10.1016/j.infsof.2015.11.009.

[5] G. Destefanis, M. Ortu, S. Porru, S. Swift, and M. Marchesi, “A statistical comparison of Java and Python software metric properties,” in *Proc. 7th International Workshop on Emerging Trends in Software Metrics (WETSoM)*, Austin, TX, USA, 2016, pp. 22–28, doi: 10.1145/2897695.2897697.

[6] V. S. S. Sambaturu, “Evolution analysis of software quality metrics in an open-source Java project: A case study on TestNG,” *arXiv preprint* arXiv:2505.22884, 2025, doi: 10.48550/arXiv.2505.22884.

[7] T. Kaur, N. Ratti, and P. Kaur, “Applicability of Lehman laws on open source evolution: A case study,” *International Journal of Computer Applications*, vol. 93, no. 18, pp. 40–46, 2014, doi: 10.5120/16438-6165.

[8] A. Smajić, M. ElZayyat, and E. Junuz, “Evolution of software through Lehman’s laws: Comparative analysis of open and closed source solutions,” in *Proc. 29th International Conference on Information Technology (IT)*, Žabljak, Montenegro, 2025.

[9] B. C. Matias, S. Freire, J. Freitas, F. Fronchetti, K. Damevski, and R. Spinola, “A survey on large language model impact on software evolvability and maintainability: The good, the bad, the ugly, and the remedy,” *arXiv preprint* arXiv:2601.20879, 2026, doi: 10.48550/arXiv.2601.20879.

[10] T. Mahmud, M. Che, and G. Yang, “Analyzing the impact of API changes on Android apps,” *Journal of Systems and Software*, vol. 200, Art. no. 111664, 2023, doi: 10.1016/j.jss.2023.111664.

[11] T. Heričko, B. Šumak, and S. Karakatič, “Commit-level software change intent classification using a pre-trained transformer-based code model,” *Mathematics*, vol. 12, no. 7, Art. no. 1012, 2024, doi: 10.3390/math12071012.

[12] X. Qi, X. Qian, and Y. Li, “Semantic test repair for web applications,” in *Proc. 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2023, pp. 1190–1202, doi: 10.1145/3611643.3616324.

[13] T. Heričko and B. Šumak, “Exploring maintainability index variants for software maintainability measurement in object-oriented systems,” *Applied Sciences*, vol. 13, no. 5, Art. no. 2972, 2023, doi: 10.3390/app13052972.

[14] R. Capilla, V. Salamanca, A. Valdezate, and G. Robles, “Can instability variations warn developers when open-source projects boost?” *Empirical Software Engineering*, vol. 29, Art. no. 97, 2024, doi: 10.1007/s10664-024-10482-4.

[15] A. Alami, R. Pardo, and J. Linåker, “Free open source communities’ sustainability: Does it make a difference in software quality?” *Empirical Software Engineering*, vol. 29, Art. no. 114, 2024, doi: 10.1007/s10664-024-10529-6.

[16] A. Rio and F. Brito e Abreu, “PHP code smells in web apps: Evolution, survival and anomalies,” *Journal of Systems and Software*, vol. 200, Art. no. 111644, 2023, doi: 10.1016/j.jss.2023.111644.

[17] A. Rio, F. Brito e Abreu, and D. Mendes, “Causal inference of server- and client-side code smells in web apps evolution,” *Empirical Software Engineering*, vol. 29, Art. no. 133, 2024, doi: 10.1007/s10664-024-10478-0.

[18] Z. Wan, Y. Zhang, X. Xia, Y. Jiang, and D. Lo, “Software architecture in practice: Challenges and opportunities,” in *Proc. 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2023, pp. 1457–1469, doi: 10.1145/3611643.3616367.

[19] I. Neamtiu, G. Xie, and J. Chen, “Towards a better understanding of software evolution: An empirical study on open-source software,” *Journal of Software: Evolution and Process*, vol. 25, no. 3, pp. 193–218, 2013, doi: 10.1002/smr.564.

[20] A. Israeli and D. G. Feitelson, “The Linux kernel as a case study in software evolution,” *Journal of Systems and Software*, vol. 83, no. 3, pp. 485–501, 2010, doi: 10.1016/j.jss.2009.09.042.

[21] Z. Ameer *et al.*, “Impact of code smells on software quality: A systematic literature review,” *Journal of Software: Evolution and Process*, vol. 38, no. 4, Art. no. e70091, 2026, doi: 10.1002/smr.70091.