Journal of Electrical Vocational Education
Vol. 01, No. 02, November, pp. 153~160
E-ISSN: 3110-486X
DOI : https://doi.org/10.24036/jeve.v1i2.47

Analysis of the Evolution Patterns of Student Software Development through GitHub Repository Activity in Project-Based Learning

Mutia Rahmi Dewi[1*], Defni[2], Yoliza Erwanda[3]
[1,2,3]Information Technology Department, Politeknik Negeri Padang, Padang, Indonesia
Corresponding Author mail* : mutiarahmi@pnp.ac.id

Article Info

Article history:

Received 29 December 2025
Revised 30 December 2025
Accepted 15 January 2025

Keywords:

software evolution
software construction
github
Project Based Learning
commit analysis

ABSTRACT

Software evolution is a continuous process of change that occurs throughout the system development life cycle. In the context of higher education, understanding the dynamics of software change is essential for assessing students' competencies in managing team-based projects. This study aimed to analyze software evolution patterns in student projects developed using the Project-Based Learning (PBL) approach. The research data were obtained from 12 public GitHub repositories belonging to students who developed applications based on the Laravel framework. The study employs a descriptive quantitative approach by analyzing commit log data using automated Python-based scripts. The analysis focuses on activity metrics such as the number of commits, the average number of files changed per commit, and the identification of the most frequently modified files. The results indicate that each group exhibits distinct construction and evolution patterns. Several groups exhibited extremely high numbers of files changed per commit, indicating large-scale commits and suboptimal version control practices, such as improper use of .gitignore. The most frequently modified files were controllers, views, configuration files, and database migration files, reflecting a strong focus on application logic and interface development. These findings demonstrate that PBL effectively supports iterative and collaborative software development practices, but also reveal gaps in students' understanding of structured change management and semantic commit discipline. This study provides empirical evidence of student software evolution behavior and offers a foundation for developing automated, data-driven evaluation systems to support software engineering education, particularly in vocational contexts.

Corresponding Author:

Mutia Rahmi Dewi
Information Technology Department, Politeknik Negeri Padang, Padang, Indonesia

Email: mutiarahmi@pnp.ac.id

A. INTRODUCTION

The construction and evolution of software are fundamental aspects of the modern software development life cycle. This process focuses not only on programming activities but also on the software's ability to adapt to changes in user needs, technological advancements, and collaboration dynamics within development team[1]. As the complexity of systems and the expectation for software resilience increase, there is a need for adaptive, iterative, and collaborative development approaches. In the context of higher education, Project-Based Learning (PBL) has been recognized as an effective pedagogical approach to integrating technical skills and collaborative skills through hands-on experience in software development projects[2].

The implementation of PBL in software engineering provides students with the opportunity to be fully involved in the development cycle, from planning and construction to maintenance and system evolution. However, the effectiveness of this approach cannot only be assessed based on the final product, but also needs to be viewed in terms of the dynamics of the development process that occurs within the project repository. Activities such as commit frequency, code change complexity, and integration patterns reflect the students' understanding of modern software engineering principles[3].

Various previous studies have shown that analyzing historical data from software repositories, particularly through GitHub commit logs, can provide valuable insights into the construction and evolution patterns of features in collaborative projects[4]. These commit activities represent processes such as feature additions, bug fixes, and code refactoring that contribute to the quality of the system[5]. This analytical approach allows for a quantitative evaluation of developer behavior, including students, which can be used to assess the development of technical competencies and team effectiveness[6].

In recent years, GitHub has become the primary platform supporting students' collaborative activities in software development. Features such as version control, branching, and pull requests provide a rich empirical data source for tracing the dynamics of system construction and evolution[7]. Analyzing commit-based metrics (commit-level metrics) can reveal development characteristics, the complexity of changes, and the relationship between the frequency and depth of code modification[8]. However, there remains a significant research gap in understanding how the software features developed by students evolve throughout the PBL process. Most previous studies have focused on the final product, without considering the feature change dynamics during the development process.

Based on this background, this study focuses on analyzing the evolution patterns of software features in student projects based on Project-Based Learning, using historical data from 12 GitHub repositories. Each repository represents one Laravel-based development project, which implements approximately 30 Product Backlog Items (PBIs) over one semester. This research aimed to identify the construction and evolution patterns of features, code change complexity, and interrelationships among development activities within the context of project-based learning. Academically, this study is expected to broaden the understanding of software evolution behavior in the context of information technology education. Practically, the findings of this study are expected to contribute to the development of data-based evaluation instruments to assess the effectiveness of Project-Based Learning implementation.

B. METHOD

2.1. Literature Review and Hypothesis Development

2.1.1. Software Construction and Feature Evolution

Software construction is an implementational phase in the Software Development Life Cycle (SDLC) involving coding, debugging, and module integration activities[9]. In an educational context, this phase becomes an indicator of students' ability to implement designs into functional systems. On the other hand, software evolution focuses on continuous changes to the system to adapt to user needs and technological advancements[10].

Recent studies highlight the importance of historical documentation in understanding software evolution. Historical data in version control systems can be used to predict change trends and detect areas with high technical risks (technical debt)[11]. Meanwhile, analyzing commit patterns in student projects can serve as an objective and measurable tool for evaluating project-based learning[12].

2.1.2. Project-Based Learning in Software Engineering Education

Project-Based Learning has been shown to improve students' problem-solving abilities, collaboration, and conceptual understanding in the fields of engineering and informatics[13]. In the field of software engineering, this approach is implemented through team-based projects that reflect actual industry practices. Research by Shahid et al. found that using PBL in programming courses helps students understand the entire software life cycle and internalize Agile Development practices[14].

Moreover, several studies also highlight the importance of learning analytics from GitHub platforms as tools for assessing student engagement. Student commit patterns can be used to assess participation and collaboration in group projects[15]. Thus, the integration of PBL and repository data analysis could represent a new approach for evaluating learning effectiveness in software engineering.This study uses a descriptive quantitative approach, utilizing data-driven software analytics

2.2. Research Design

This study uses a descriptive quantitative approach, utilizing data-driven software analytics from the students' GitHub repositories. The goal is to identify the evolution patterns of software features based on the commit history performed by each project team. This approach aligns with the mining software repositories (MSR) methodology, commonly used in software evolution research to understand the dynamics of collaboration and source code changes[16].

2.3. Data Sources

The data used in this study comes from 12 public GitHub repositories owned by students in the Software Construction and Evolution course. Each repository represents one project group in Project-Based Learning. Each group develops a web application with features equivalent to about 30 Product Backlog Items (PBIs).

The following table presents a list of the repositories analyzed:

Table 1. List of Repositories

No	Group	Github Repository
1	Kelompok 1	furqonaugust17/PBL2D-Kel6-Project
2	Kelompok 2	rmaisshadiq/fabulous-five
3	Kelompok 3	DikaJefrianto/Agile_D4
4	Kelompok 4	Anlaharpanda2/AgileD3_2025
5	Kelompok 5	ReykelRaflen/PBL
6	Kelompok 6	nauvalalpen/Agile-D1
7	Kelompok 7	gioaprilino/Refive-PBL
8	Kelompok 8	Gioezzy/SIGAP
9	Kelompok 9	DarulFebri/Sistem-Informasi-Tugas-Akhir-dan-Praktek-Kerja-Lapangan
10	Kelompok 10	Cukurukuk-TI/GreonePBL
11	Kelompok 11	deanzyyy/Englicious
12	Kelompok 12	pbl-1b/pbl-1b-code

From each repository, data extraction is performed, including:

- Group name and repository URL
- Total number of commits
- Average number of files changed per commit
- List of the ten most frequently changed files

The data extraction process is automated using a Python script based on GitPython to clone the repositories and read commit metadata. The extracted data is stored in .csv format for further analysis.

2.4. Analysis Process

The analysis is performed in three main stages:

1. Data Cleaning and Normalization — Commits with empty or non-informative messages are removed. Only the main branch (main/master) is analyzed to ensure data consistency.
2. Descriptive Statistical Analysis — Defining the research variables to be used in analyzing the data. These variables are:

Table 2. Research Variables

Variable	Indicator	Description
Construction Activity	Total Commit	Total number of commits in a repository
Change Complexity	Avg Files Changed per Commit	Average number of files changed per commit
Focus of Changes	Top 10 Files	Files most frequently modified
Evolution Pattern	Commit vs File Changed	Relationship between the frequency of changes and the depth of modifications

Once the research variables are determined, the frequency of each indicator is calculated and analyzed. This analysis aims to examine the construction intensity and stability of development within each group.

3. Evolution Pattern Visualization — The analysis results are visualized using bar charts and distribution diagrams to facilitate interpretation and comparison between groups. Visualization helps identify development hotspots and the frequency of changes to specific features.

2.5. Validity and Reliability

To ensure data validity, only repositories with a public commit history and complete project structures are included. Reliability is maintained by using an automated script for all groups, so the extraction process does not rely on subjective judgment. Additionally, the research steps are recorded in a Jupyter notebook, which allows for full replication by other researchers.

C. RESULTS DISCUSSION

3.1. Overview of Construction Activity

Based on the extraction results from 12 student GitHub repositories, a total of 2.274 commits were obtained during one semester of Project-Based Learning (PBL). The number of commits varied across groups, ranging from 15 to 510 commits per project. This variation shows differing levels of development activity between teams, which may be influenced by the collaboration strategies and task division applied by each group.

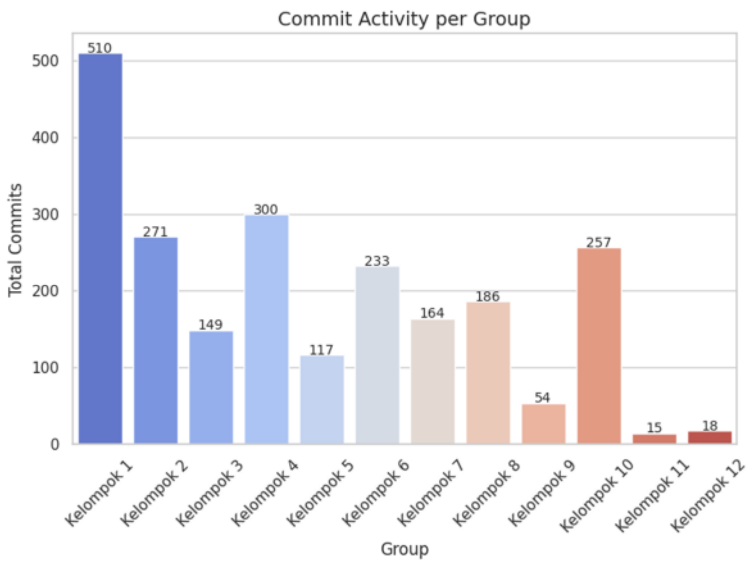

Figure 1. Commit activity of each student group during the software construction and evolution process

Figure 1 shows the distribution of commits across the 12 project groups. Groups with higher commit numbers exhibit iterative and incremental development patterns. It means that continuous integration practices encourage teams to commit more frequently in order to minimize code conflicts and improve software stability.

3.2. Average Number of Files Changed per Commit

Figure 2 shows the average number of files changed per commit. The analysis results indicate that the average number of files changed per commit reaches 101.46 files. However, this value is heavily influenced by several outliers. In particular, Group 11 and Group 9 exhibit extremely high average numbers of files changed per commit (more than 300 files per commit). This condition indicates the presence of large-scale commits that include many build artifacts or dependencies, such as vendor/ directories, node_modules/, compiled files, or auto-generated files. This phenomenon also reflects students' lack of understanding of good version control practices, especially in the application of the .gitignore file, resulting in directories that should be excluded being uploaded to the repository. This finding suggests that students have not yet fully grasped the principles of efficient and structured change management in software development.

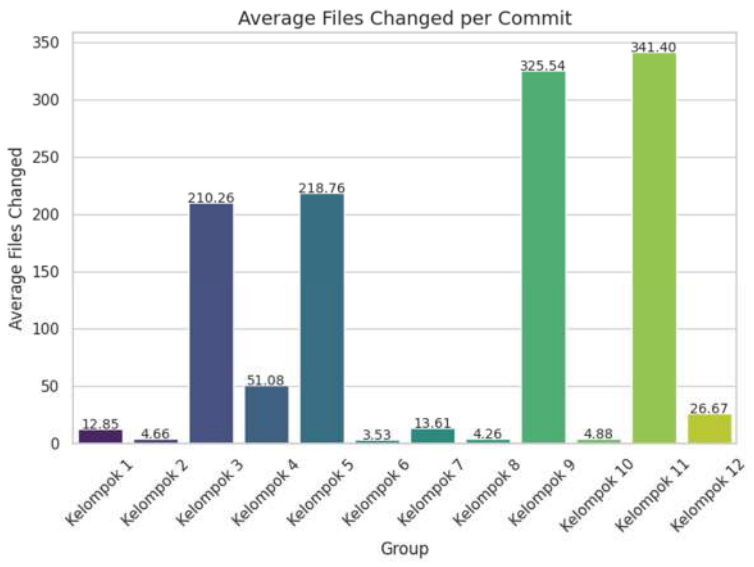

Figure 2. Average number of files changed per commit by each student group

3.3. Most Frequently Changed Files

Figure 3 illustrates the ten files most frequently changed across all repositories. The majority of changes occurred in:

- Controller and view files (particularly in Laravel),
- Configuration files (.env, web.php),
- Files related to database migration.

This indicates that students' primary focus was on constructing application logic and user interfaces. The frequency of changes to configuration files also suggests repeated adjustments to the testing and deployment environments. These findings affirm that the early phase of a project is typically dominated by structural and architectural changes, while feature evolution becomes more dominant in the middle and later stages.

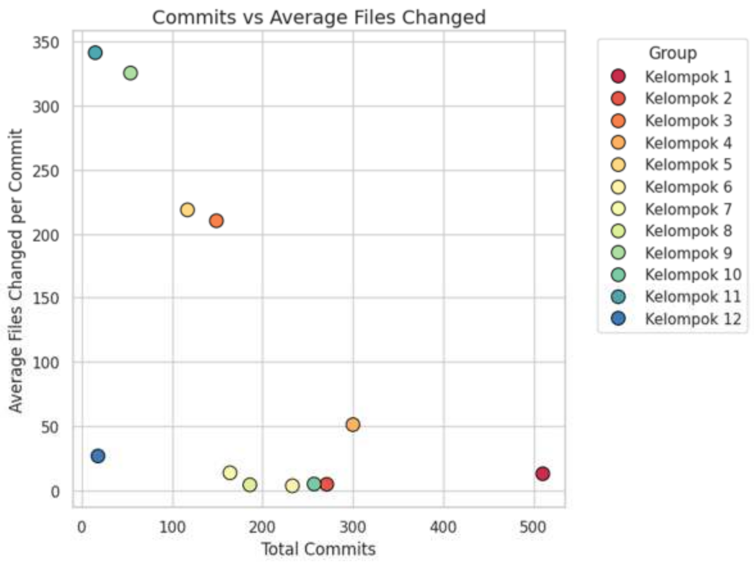

Figure 3. Correlation between the number of commits and the average files changed per commit

3.4. Discussion

Overall, the results of this study support the view that Project-Based Learning can be an effective means of practicing modern software engineering principles. The empirical data from GitHub shows that students are not only coding but also internalizing collaborative practices such as branching, pull requests, and commit discipline. These findings emphasize the importance of using version control platforms in information technology education.

However, there are indications of low consistency in commit messages and insufficient documentation, which hinder the semantic interpretation of feature changes. This suggests the need for further guidance on semantic commit practices and integration with automatic code quality analysis tools such as SonarQube or GitHub Actions for continuous evaluation.

D. CONCLUSION

This study provides empirical insights into the construction and evolution patterns of software in student projects within the context of Project-Based Learning (PBL). The analysis of 12 GitHub repositories shows that some groups experienced intensive construction and stabilization phases, while others showed relatively low activity, indicating differences in team working strategies and project management practices. The variability in the number of files changed per commit also suggests differences in students' understanding of version control practices and dependency management.

However, this study has some limitations that need to be noted. First, the data only includes commit history without considering the semantic context of each code change, which reduces the depth of the analysis regarding the type and purpose of changes made. Second, the variation in technical capabilities between groups and differences in project complexity may affect the frequency and patterns of commits, so the results do not fully reflect individual learning effectiveness. Third, this study has not directly evaluated the relationship between construction activity and the final product quality, such as system performance or user satisfaction.

These limitations open opportunities for further research to develop more comprehensive approaches, such as integrating semantic commit message analysis, issue tracking, and pull requests to map software feature evolution in more detail. A mixed-methods approach combining quantitative repository analysis with student interviews or reflections could also enhance the validity of the findings. Thus, the results of this study are expected to serve as a foundation for understanding how effective software construction practices can contribute to the success of Project-Based Learning in software engineering education.

REFERENCES

[1] A. Obaid, "Using prototypes in agile software development," *IJCI*, vol. 3, no. 2, pp. 23–38, 2024, doi: 10.59992/ijci.2024.v3n2p2.

[2] G. Guo, "Enhancing project-based manufacturing education with integrated engineering software tools," *Comput. Appl. Eng. Educ.*, vol. 33, no. 2, 2025, doi: 10.1002/cae.70012.

[3] C. Connolly and G. Meiselwitz, "integrating software engineering in computer programming education," pp. 50–54, 2009, doi: 10.1145/1631728.1631745.

[4] Y. Golubev, J. Li, T. Bryksin, V. Bushev, and I. Ahmed, "Changes from the trenches: should we automate them?," 2021, doi: 10.48550/arxiv.2105.10157.

[5] E. Zabardast, J. González-Huerta, and D. Šmite, "refactoring, bug fixing, and new development effect on technical debt: an industrial case study," pp. 376–384, 2020, doi: 10.1109/seaa51224.2020.00068.

[6] S. Bonesso, F. Gerli, and E. Bruni, "The emotional and social side of analytics professionals: an exploratory study of the behavioral profile of data scientists and data analysts," *Int. J. Manpow.*, vol. 43, no. 9, pp. 19–41, 2022, doi: 10.1108/ijm-07-2020-0342.

[7] M. AlMarzouq, A. AlZaidan, and J. Dallal, "Mining github for research and education: challenges and opportunities," *Int. J. Web Inf. Syst.*, vol. 16, no. 4, pp. 451–473, 2020, doi: 10.1108/ijwis-03-2020-0016.

[8] H. Toba, T. K. Gautama, J. Narabel, A. Widjaja, and S. F. Sujadi, "Evaluasi metodologi ci/cd untuk pengembangan perangkat lunak dalam perkuliahan," *JEPIN (Jurnal Edukasi dan Penelit. Inform.*, vol. 8, no. 2, pp. 227–234, 2022.

[9] M. Chan and S. Yazid, "A novel framework for information security during the sdlc implementation stage: a systematic literature review," *J. Resti (Rekayasa Sist. Dan Teknol. Informasi)*, vol. 8, no. 1, pp. 88–99, 2024, doi: 10.29207/resti.v8i1.5403.

[10] A. Alkhalil, "evolution of existing software to mobile computing platforms: framework support and case study," *Int. J. Adv. Appl. Sci.*, vol. 8, no. 3, pp. 100–111, 2021, doi: 10.21833/ijaas.2021.03.013.

[11] M. Murillo, G. López, R. Spínola, J. Guzmán, N. Rios, and A. Pacheco, "Identification and management of technical debt," *J. Softw. Eng. Res. Dev.*, 2023, doi: 10.5753/jserd.2023.2671.

[12] S. Hamer, C. Quesada-López, A. Martínez, and M. Jenkins, "Using git metrics to measure students' and teams' code contributions in software development projects," *Clei Electron. J.*, vol. 24, no. 2, 2021, doi: 10.19153/cleiej.24.2.8.

[13] M. Yunus, P. Setyosari, S. Utaya, and D. Kuswandi, "The influence of online project collaborative learning and achievement motivation on problem-solving ability," *Eur. J. Educ. Res.*, vol. volume-10-2021, no. volume-10-issue-2-april-2021, pp. 813–823, 2021, doi: 10.12973/eu-jer.10.2.813.

[14] M. Shahid, K. Pervaiz, M. Awais, and S. Khurshid, "Project-based iterative teaching model for introductory programming course," *Nile J. Commun. Comput. Sci.*, vol. 5, no. 1, pp. 10–41, 2023, doi: 10.21608/njccs.2023.321167.

[15] M. Trujillo, "The penumbra of open source: projects outside of centralized platforms are longer maintained, more academic and more collaborative," 2021, doi: 10.48550/arxiv.2106.15611.

[16] M. Scheidgen and J. Fischer, "Model-based mining of source code repositories," pp. 239–254, 2014, doi: 10.1007/978-3-319-11743-0_17.

[17] 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, p. 1012, 2024, doi: 10.3390/math12071012.

[18] R. Alfayez and A. Alazba, "merge conflict prediction using feature selection and stacking heterogeneous ensembles: an empirical investigation," *J. Softw. Evol. Process*, vol. 37, no. 9, 2025, doi: 10.1002/smr.70047.