MJI
Media Jurnal Informatika
Special Issue Information System September 2025
p-issn : 2088-2114, e-issn : 2477-2542

Optimizing Function Point Analysis in Moodle: A Plugin-Based Solution for Automated Grading and Student Assessment

Lukmannul Hakim Firdaus[1*], Bambang Wisnuadhi[2], Ani Rahmani[3]
[1,2,3]Politeknik Negeri Bandung
lukmannul.hakim@polban.ac.id[1], bwisnuadhi@polban.ac.id[2], anirahma@polban.ac.id[3]

ARTICLE INFO

Article history
Received : 2025/09/06
Revised : 2025/09/28
Accepted : 2025/09/30

Keywords
Learning Management System (LMS), Moodle, Function Point, Software Estimation

ABSTRACT

Learning Management Systems (LMS), especially Moodle, have become integral tools in modern education, enabling efficient management of teaching materials, assignments, and assessments. This study aimed to develop and integrate a Moodle plugin for automating Function Point (FP) calculations, a method used in software engineering education to estimate development complexity. The research focused on creating an Activity Module plugin that automates the FP calculation process, reduces manual grading efforts, and provides immediate feedback to students. The plugin underwent functional testing, including Requirement Verification, UI Testing, Integration Testing, and Output Validation, to assess its performance. The results demonstrated that the plugin successfully automated the calculation of Unadjusted Function Points (UFP) and Value Adjustment Factor (VAF), with accurate results and seamless integration into Moodle's core modules such as assignment creation, grading, and user management. Testing confirmed that the plugin met all functional requirements, and the UI and integration worked as intended. The study concluded that the FP plugin is a viable alternative to traditional FP instruction, delivering time saved, error reduction, and reduced manual grading effort. Limitations of the study include the small-scale testing, and future research should focus on evaluating scalability for larger classes and assessing its impact on student learning outcomes. Further development could explore integrating advanced features like analytics and AI-assisted feedback to enhance the learning experience.

1. Introduction

Learning Management Systems (LMS) have become integral to modern education, facilitating the entire learning process for both instructors and students[1]. Recent studies have highlighted the effectiveness of LMS in enhancing student engagement and academic performance [2], [3], [4]. Moodle remains one of the world's most widely used learning management systems, offering extensive features for course authoring and classroom management [5], [6]. It also facilitates evaluating student performance in an unbiased manner, filling knowledge gaps, and simplifying the assessment procedure [7]. Additionally, it provides constant access to learning materials, opportunities for additional online activities, and convenient assessment methods for students [8]

While Moodle offers rich features, research shows that Moodle is mainly used within University STEM disciplines and effectively improves student performance, satisfaction, and engagement, but more qualitative research is needed on the use of Moodle, particularly investigating educators perspectives [9], [10]. Since Moodle is structured with an application core surrounded by various plugins, it is possible to add new plugins without modifying the core application [11]. This architectural advantage of Moodle ensures that adding plugins does not interfere with future upgrades to the system.

The development and customization of Moodle plugins have received significant attention in the field of educational technology. Several studies have examined the implementation of custom plugins to enhance the learning experience, improve assessment efficiency, and facilitate instructor-student interaction. For instance, Baneș developed a Moodle plugin to enable file submissions via the chat interface, demonstrating how plugin customization can extend the functionality of Moodle beyond its standard modules [12], [13].

Rahmani reported on the use of an automatic grading plugin for basic programming exercises, which allowed instructors to assess student submissions in real-time, reducing manual grading workload and improving feedback speed[14]. Similarly, Sáiz-Manzanares developed the "eOrientation" Moodle plugin enabled personalized monitoring and early detection of at-risk students, highlighting that the flexibility of Moodle's plugin architecture is crucial for implementing targeted instructional strategies [15]. Moreover, recent research has explored the integration of advanced analytics and interactivity through plugins. Zhao & Zhang proposed a design framework for interleaved learning in Moodle, demonstrating that plugins can not only automate administrative tasks but also support innovative learning strategies that enhance student engagement and learning outcomes [16]. These studies collectively underscore that the ability to develop and customize plugins is crucial for instructors who wish to adapt Moodle to their specific course requirements and improve overall learning effectiveness.

In the field of software engineering education, one of the essential competencies that students need to master is function point (FP) calculation. FP is a method used to estimate the complexity of software requirements, especially for estimating the cost or effort involved in software development. A recent study by Lavazza discusses the application of Function Point Analysis (FPA) in software development and maintenance, emphasizing its relevance in estimating development effort and cost [17]. Traditionally, FP learning involves a typical educational process: the instructor provides the material, conducts discussions, and evaluates exercises. However, the manual assessment of FP exercises can be time-consuming and prone to errors. However, automated methods, such as Automated Function Points (AFP), have been proposed to streamline the process and improve consistency [18]. To address this, integrating automated tools within LMS platforms like Moodle can streamline the assessment process.

Moodle already supports automated assessments for multiple-choice questions, true/false, or short answers. However, the addition of a plugin for FP calculations, which are beyond standard question types, is made possible through Moodle's plugin system.This plugin will help instructors assess each FP exercise or task automatically, saving time and effort. This article describes the technical steps involved in creating a plugin for FP calculations and integrating it into Moodle so that the assessment process can be performed directly and automatically within Moodle. It is hoped that with this feature, FP learning will become more engaging for students, as they will receive immediate feedback on their work.

2. Method

This section explains the activities involved in customizing Moodle to support Function Point (FP) learning.

2.1 Function Point Calculation Process

Function Point (FP) analysis is a method used to estimate the effort required for software development and maintenance, especially when software requirements change. One of the main challenges in FP is that the software artifacts may be inconsistent, such as some classes being fully developed while others are not [19]. FP has several advantages: (1) it is technology-independent, (2) it is simple to apply, (3) it can estimate based on software requirements, and (4) it is easy for end-users to understand.

Typically, FP calculation involves summing Unadjusted Function Points (UFP) and applying a Value Adjustment Factor (VAF) as shown in the following equation:

$$FPs = UFP * VAF \qquad 1)$$

UFP is the sum of all functions such as External Interface (EI), External Output (EO), External Query (EQ), Internal Logical Files (ILF), and External Interface Files (EIF), with complexity levels (low, average, high). VAF is calculated from 14 General System Characteristics (GSC) [20].

2.2 Exploring Moodle Architecture

Exploring Moodle's architecture is based on the official Moodle documentation ("MoodleDocumentation," n.d.). To understand Moodle's architecture, one must first understand the Moodle core. The Moodle core provides the infrastructure necessary to build an LMS in Moodle, including course and activity modules, user management, course enrollment, user roles, capabilities, permissions, and Moodle context.

A crucial element to understand in Moodle's architecture is the *Moodle Context*, which is the structure or layout of the Moodle system. It contains areas where features such as courses, activity blocks, and other resources are placed. Figure 1 below shows the Moodle context diagram.

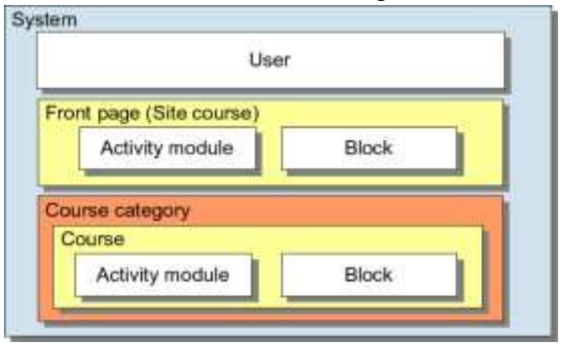


Fig 1. Moodle Context Diagram

The system diagram in Figure 1 represents the Moodle system. It shows that there must be at least one system administrator. The *Activity Module* is a component that supports user activities, whether static, dynamic, or interactive, such as quizzes, assignments, lessons, and chats. A *Block* is an area in a web page, like a sidebar, containing information related to the user's activities, such as a calendar or management tools for course administration.

Both *Activity Modules* and *Blocks* are only available within a course or site course. A *Course* is a page dedicated to conducting learning activities. It represents a specific subject or program. Each course is stored in a *course category*, while the *Site Course* is the landing page for the Moodle system, containing web information or announcements. Each context can be adjusted based on the roles of the users involved.

Technically, Moodle plugins are PHP script folders (including HTML, CSS, JavaScript, and others). Moodle core communicates with plugins by searching for specific entry points, often defined in the lib.php file within each plugin. The Moodle core provides all necessary infrastructure to build a content management system (CMS). By implementing the core concepts, Moodle plugins follow defined standards such as Course and Activities, Users, Course Enrollments, User Functionality, and more.

2.3 Integrating Function Point into Moodle

Moodle is a web-based application developed using PHP programming language with an SQL database for data storage. Several requirements must be met to install Moodle:

- The server must have a web server like Apache or Nginx installed.
- The web server must support PHP programming.
- The database can be MariaDB, PostgreSQL, Oracle, SQL Server, or MySQL.

The hardware specification used in the testing process is shown in Table 1.

Table 1. Hardware Specification Used

Num	Component	Specification
1	Operating System	Windows 10 x64
2	Moodle Version	3.11
3	Database	MariaDB
4	PHP Version	7.4.13

There are two ways to install Moodle: by downloading the source code from https://download.moodle.org or cloning it from the Git repository. Once the source code is downloaded or cloned, installation can be done via a browser by accessing http://<ip-address>/moodle. The installation process involves database configuration, checking device readiness, and creating the Moodle data storage directory on the device. Standard configuration settings are applied during installation.

For further customization or feature development, additional configuration adjustments may be necessary to simplify the development process. It is highly recommended that the installed Moodle system be configured to display PHP errors, warnings, or notices during development, as Moodle does not tolerate any PHP notice, even minor ones. Developers must ensure that their PHP code is correct to avoid such warnings.

For performance reasons, Moodle uses cache extensively, but in some cases, especially during development, cache may need to be purged manually or disabled to facilitate the display of changes instantly. The configuration for this is done in the config.php file.

2.4 Using the Moodle Framework

a. Moodle Modularity

Moodle has an architecture that supports modularity very well. Every feature available in Moodle is a standalone module known as a Moodle plugin. When Moodle is first downloaded and installed, it comes with standard modules. If further customization is required, Moodle plugins can be downloaded from the Moodle Plugin Directory, which can be accessed at https://moodle.org/plugins.

Half of Moodle's system consists of standard plugins, while the remaining half is made up of core subsystems. These subsystems provide Core API for plugin development. Figure 2 illustrates the relationship between the core subsystems, plugins, and the database API.

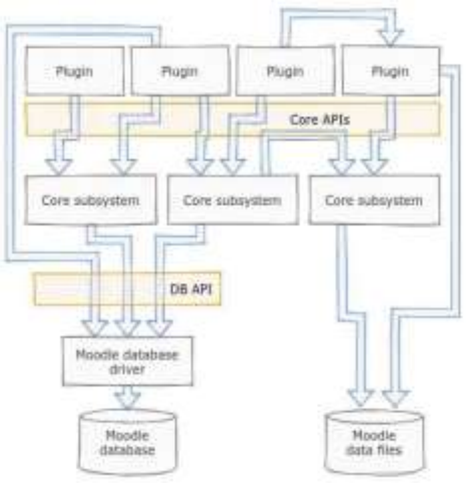


Fig 2. Relationship between Core Subsystems, Plugins, and Database API
(Sourced from https://learn.moodle.org/)

b. Moodle Plugins

There are various types of plugins in Moodle, each designed for specific functionality. It is essential to understand the purpose, characteristics, and limitations of each type of plugin to accurately select the appropriate one for a given need. Each plugin is stored in a specific directory according to its type, which enables the creation of various plugins with different purposes. Moodle's database structure is defined in the install.xml file within each plugin's db folder. For example, the mod/forum/db/install.xml file defines the database used by the Forum module.

When creating a plugin, developers must understand the structure of Moodle plugins. Moodle provides comprehensive documentation for plugin development, and every contributor must follow the guidelines. Additionally, understanding the Moodle coding style is crucial for developers.

2.5 Creating a Function Point Plugin

The appropriate plugin type for developing the Function Point plugin is an *Activity Module*. This type allows more flexible presentation of the learning process. An Activity Module has common features such as:

a. **Grading**: This feature allows an activity to have both automatic and manual grading, supported by the Gradebook API.
b. **Completion**: This feature enables an activity to be marked so that student progress can be monitored.
c. **Backup and Restore**: This feature enables activities to be included in the course's backup and restore process.
d. **Conditional Access**: This feature enables activities to be accessed based on specific criteria, such as user completion or prerequisites.

2.6 Testing

In this study, the testing was conducted to evaluate the Moodle plugin for Function Point (FP) through user acceptance testing (UAT) with 32 students. The testing focuses on verifying that each module and feature of the plugin performs according to the specified requirements. The testing process includes the following steps:

1. Requirement Verification : Ensure that all functional requirements for the FP plugin, including input handling, calculation of Unadjusted Function Points (UFP), Value Adjustment Factor (VAF), and final FP output, are correctly implemented.
2. User Interface Testing : Verify that all UI components, including forms, input fields, and result displays, work as intended and provide correct feedback to the user.
3. Integration Testing : Test the integration of the FP plugin with Moodle's core modules, including course activities, user roles, and gradebook functionalities, to ensure seamless operation without errors.
4. Output Validation : Check the correctness of FP calculations by comparing plugin outputs against manually calculated values or pre-defined answer keys with a ±2 tolerance for UFP differences.

3. Results and Discussion

3.1. Function Point Architechture

In this study, the function point module is implemented as an activity module plugin and is installed only within a single course. Consequently, several components created in this plugin heavily rely on the course module

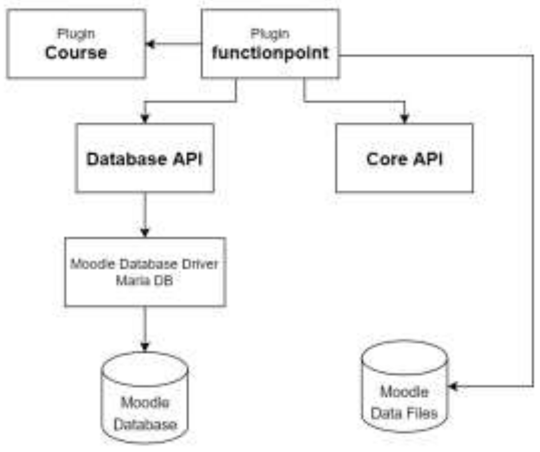


Fig 3. Function Point

Additionally, to support the functionality of the function point plugin, various Core APIs and Database APIs are utilized. Some of the Core APIs used in this plugin include the Page API, Moodlelib API, File API, Form API, Admin Setting API, and Data Manipulation API. Regarding data storage, there are two types of storage used: Moodle Data Files and the Moodle Database. Files uploaded by users during the creation of function point exercises are stored in the Moodle Data Files, while data related to the exercises, student answers, and grades are stored in the Moodle Database.

3.2. Function Point Plugin Implementation

The implementation process begins with the installation of the plugin in Moodle. Upon installation, Moodle automatically detects the plugin within the standard directory structure and verifies its compatibility by checking the version specified in the version.php file. If the specifications meet the requirements, the plugin is registered in the database after the "Upgrade Moodle Database Now" button is clicked. Once the installation is successful, the user is directed back to the Moodle dashboard.

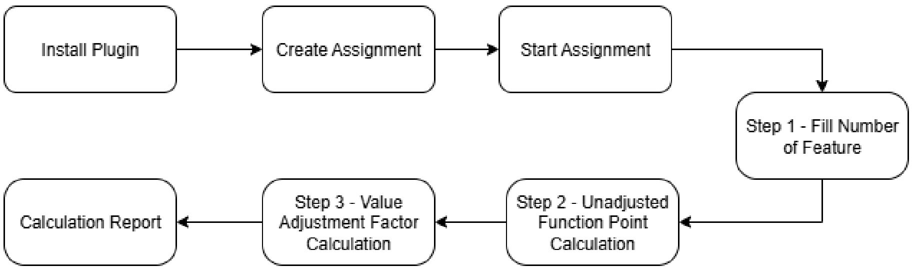
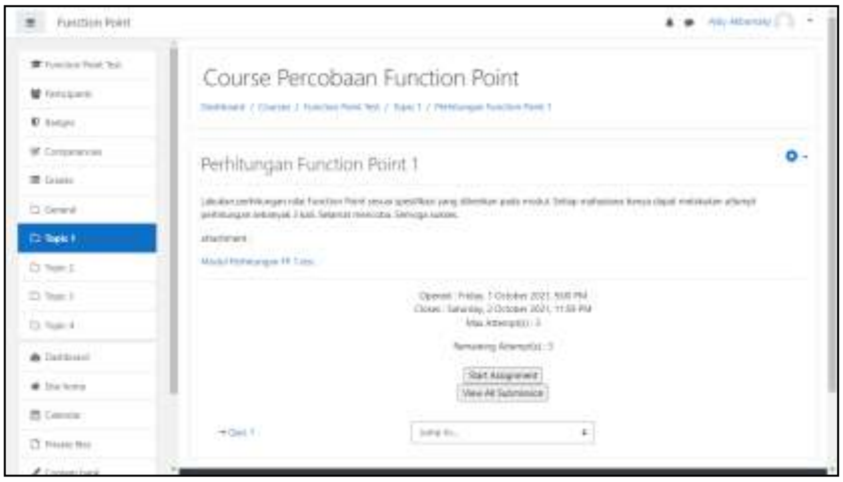

Install Plugin
Create Assignment
Start Assignment
Step 1 - Fill Number of Feature
Step 2 - Unadjusted Function Point Calculation
Step 3 - Value Adjustment Factor Calculation
Calculation Report

Fig 4. Function Point Flow Implementation

Following the installation, the instructor creates a new activity within the course, selecting the "Function Point" activity. The assignment description is provided by the instructor, detailing the case study that students are required to solve. Students can then begin the Function Point calculation exercise by selecting the assignment and clicking "Start Assignment."


Course Percobaan Function Point

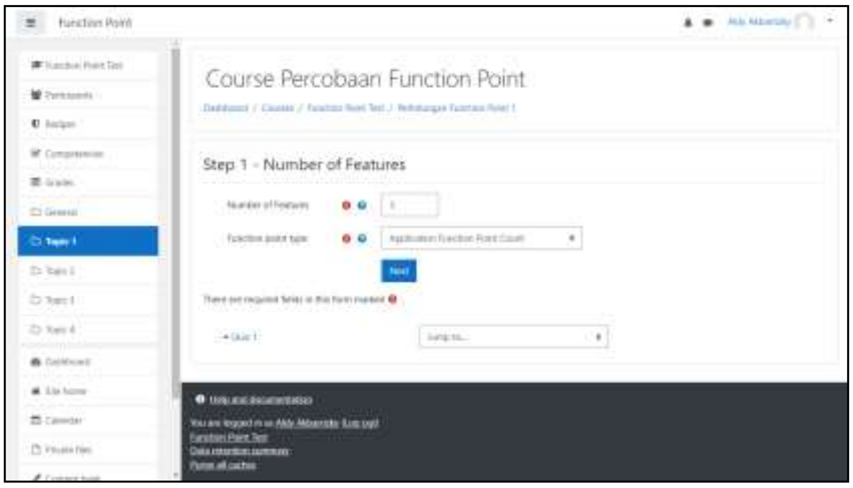
Perhitungan Function Point 1

[illegible]

Opened: Friday, 1 October 2021, 5:00 PM
Closes: Saturday, 2 October 2021, 11:59 PM

Fig 5. Assignment Display

The first step in the process involves the student entering the number of features for the application under evaluation. Additionally, the student selects the type of Function Point calculation to be performed, such as DFP (Development Project Function Point Count), AFP (Application Function Point Count), or EFP (Enhancement Project Function Point Count).



Course Percobaan Function Point

Dashboard / Courses / Function Point Test / Perhitungan Function Point 1

Step 1 - Number of Features

Number of Features: 1

Function point type: Application Function Point Count

Next

There are required fields in this form marked

Quiz 1 | Jump to...

Help and documentation

You are logged in as Aldy Akbarrizky (Log out)

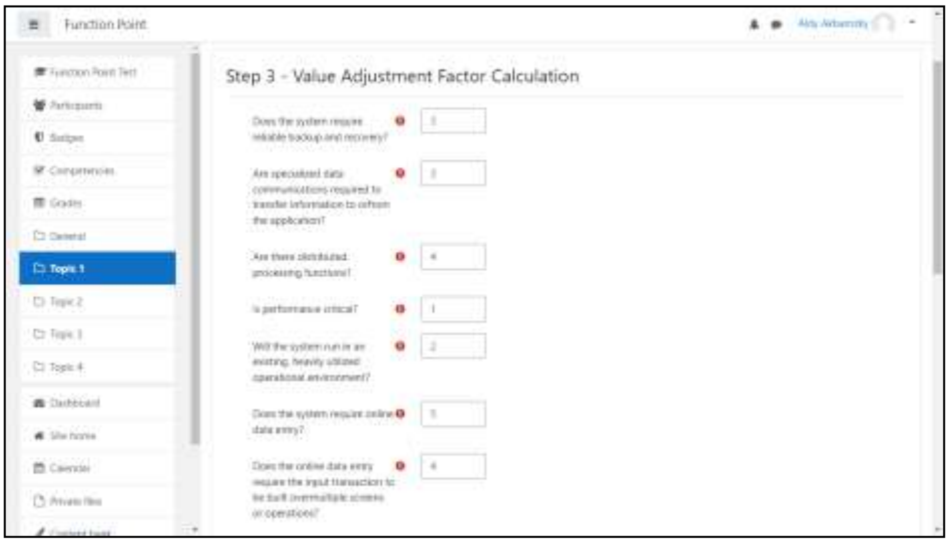
Function Point Test

Data retention summary

Purge all caches

Fig 6. Input Number of Features

In the second step, students are tasked with calculating the Unadjusted Function Points (UAF). This requires the student to input values for various data elements, including FTR (File Type Reference), DET (Data Element Type), RET (Record Element Type), and other necessary parameters for the processes under evaluation.

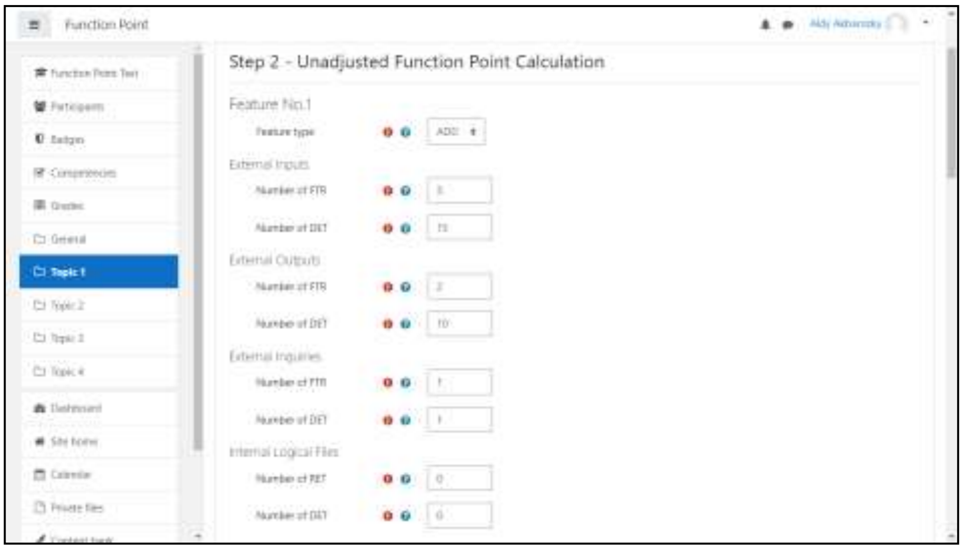


Step 3 - Value Adjustment Factor Calculation

Does the system require reliable backup and recovery?	3
Are specialized data communications required to transfer information to or from the application?	3
Are there distributed processing functions?	4
Is performance critical?	1
Will the system run in an existing, heavily utilized operational environment?	2
Does the system require online data entry?	5
Does the online data entry require the input transaction to be built over multiple screens or operations?	4

Fig 7. Input VAF Value

The third step involves the calculation of the Value Adjustment Factor (VAF). In this step, students enter values for the 14 General System Characteristics (GSC) of the application being evaluated. These characteristics influence the final calculation of the Function Points.

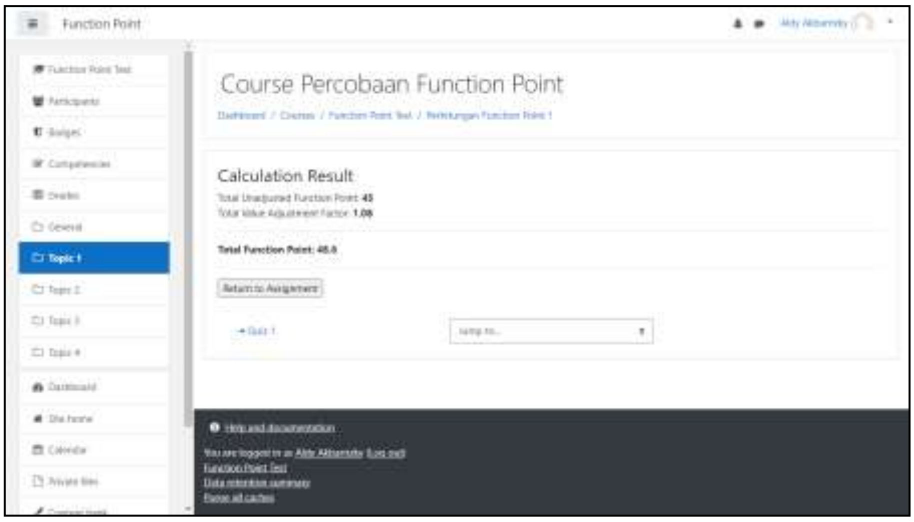


Step 2 - Unadjusted Function Point Calculation

Feature No.1
Feature type: ADD
External Inputs
Number of FTR: 3
Number of DET: 15
External Outputs
Number of FTR: 2
Number of DET: 10
External Inquiries
Number of FTR: 1
Number of DET: 1
Internal Logical Files
Number of RET: 0
Number of DET: 0

Fig 8. Input UAF Value

After completing these three steps, the Moodle LMS will automatically display the final Function Point value for the student, providing instant feedback on their calculation.



Course Percobaan Function Point

Calculation Result
Total Unadjusted Function Point: **45**
Total Value Adjustment Factor: **1.08**

Total Function Point: 48.6

Fig 9. Function Point Calculation Results

3.3. Testing Result

The testing process for the Moodle plugin focused on four key areas: Requirement Verification, UI Testing, Integration Testing, and Output Validation. Each of these testing areas was designed to ensure that the Function Point (FP) plugin functions correctly and integrates seamlessly within the Moodle LMS environment.

Table 2. Testing Result

#	Tasks	Requirement Verification	UI Testing	Integration Testing	Output Validation	Evidence/ Note
1	Moodle Instalation	✓	NA	✓	NA	Clean install, plugin activated
2	Create Assignment	✓	✓	✓	NA	Form loads, saved to course
3	Assignment Description	✓	✓	✓	✓	Matches key; text rendered
4	Input Number of Feature	✓	✓	✓	NA	Field validation OK
5	Input Value for Unadjusted Function Points (UAF)	✓	✓	✓	NA	Inputs persisted

#	Tasks	Requirement Verification	UI Testing	Integration Testing	Output Validation	Evidence/ Note
6	Input Value for Value Adjustment Factor (VAF)	✓	✓	✓	NA	Range/format validated
7	Calculation Result	✓	✓	✓	✓	Gradebook sync OK

Scenario-based user acceptance testing with 32 students covered the end-to-end workflow (attempting FP assignments, submitting answers, receiving automated computation, and viewing grades). All functional requirements listed in the Requirement Verification column were exercised and passed under these scenarios, confirming alignment with the intended design and functional objectives.

Regarding UI Testing, all user interface components performed as expected. Users were able to navigate through the workflow, including creating assignments, inputting feature counts, and entering values for both the Unadjusted Function Points (UAF) and the Value Adjustment Factor (VAF), without encountering any interface-related errors. This indicates that the system is user-friendly and responsive, providing a smooth interaction experience.

Integration Testing results show that all system modules interact correctly. Tasks involving multiple modules, such as assignment creation and calculation of function points, successfully transmitted and processed data across components. No integration conflicts were observed, demonstrating that the different parts of the plugin work seamlessly together.

Finally, Output Validation confirms that the system produces accurate and reliable results. Specifically, the assignment description and calculation results were verified as correct, ensuring that the computational logic, particularly for function point analysis, functions as intended. Other outputs, while not directly numerical, were consistently aligned with the expected system behavior, indicating overall system correctness and stability.

The testing process confirmed that the Function Point plugin was successfully integrated into Moodle and performed all required functions as specified. The plugin passed all testing categories, including Requirement Verification, UI Testing, Integration Testing, and Output Validation. The implementation of this plugin provides an efficient and automated solution for teaching and assessing Function Point calculations within Moodle, offering immediate feedback to students and reducing manual grading efforts for instructors. The successful integration of this plugin demonstrates the flexibility and effectiveness of Moodle's modular architecture for supporting customized educational tools.

3.4. Discussion and Fidings

1) Effectiveness of the FP Plugin Based on Functional Testing Results

The testing results have demonstrated that the Function Point (FP) plugin is highly effective in automating the FP calculation process within Moodle. Functional testing results confirmed that all key components, including input handling, calculation of Unadjusted Function Points (UFP), Value Adjustment Factor (VAF), and the final FP output, were correctly implemented. The plugin was able to process data efficiently and provide accurate results without any errors. Furthermore, the integration of the plugin within Moodle ensured that users—both instructors and students—had a smooth experience, with features like real-time calculation and immediate feedback contributing to the effectiveness of the tool in the learning process.

2) Modular Architecture and Its Facilitation of Implementation and Integration

Moodle's modular architecture played a crucial role in the successful implementation and integration of the FP plugin. Moodle's flexibility allows for the easy addition of plugins without altering the core system, which means that new functionalities like the FP calculation plugin can be introduced with minimal disruption. The plugin was developed as an Activity Module, which enabled it to seamlessly integrate with Moodle's existing course structure and grading system. This modularity not only facilitated a smooth implementation but also ensured that the plugin could be customized and expanded in the future without affecting Moodle's core functions. The use of Moodle's Core APIs and Database APIs further streamlined the development process, enabling the plugin to leverage the platform's existing capabilities while adding new features.

3) Comparison with Previous Approaches and Feedback

Based on user feedback collected through interviews after UAT, we found a clear shift in cognitive load from computation to analysis. Students reported that the system removes the calculation burden that previously led to inattentive errors from numerous intermediate tallies, allowing them to focus on selecting the appropriate classifications and complexity/weightings rather than recomputing formulas. Instructors noted time savings in marking and grade consolidation due to automatic calculations and gradebook synchronization, which lets them center feedback on the reasoning behind students' classifications and complexity choices instead of verifying formula correctness. Overall, compared with manual worksheets and external calculators, the plugin offers an integrated workflow that reduces arithmetic/transcription errors, shortens feedback turnaround, and recenters teaching and learning on substantive FP judgment.

When compared with traditional manual FP calculation methods, the automated FP plugin significantly improves the process's efficiency. Manual calculations often involve time-consuming and error-prone steps, which can lead to inconsistencies and delays in feedback. The plugin eliminates these issues by automating the entire process, providing immediate and accurate results to students. In comparison with other automated tools, the FP plugin offers a more integrated experience within Moodle, where assignments, grading, and feedback are centralized in one platform. Many external tools require manual data transfer and integration, whereas Moodle's plugin system ensures that the plugin works seamlessly with existing course management features, such as the gradebook and user management system. This reduces the need for additional third-party tools and improves the user experience.

4) Potential Improvements, Scalability, and Opportunities for Advanced Features

While the current implementation of the FP plugin has proven effective, there are several potential improvements that could enhance its functionality. One area of improvement is the ability to handle more complex FP calculation scenarios, such as incorporating additional calculation models or customizing the plugin to accommodate various software development methodologies. As the plugin is designed for Moodle, scalability for larger classes is a critical consideration. As student numbers increase, it would be essential to optimize the plugin's performance to ensure that calculations are processed efficiently without delays. This could involve performance enhancements, such as caching or load balancing, to handle high volumes of data and user interactions.

There are also opportunities to integrate advanced features such as analytics and AI-assisted feedback. For instance, incorporating data analytics could help track student progress and identify trends in FP calculations, providing valuable insights for instructors to tailor their teaching strategies. AI-assisted feedback could further enhance the learning experience by offering personalized suggestions based on students' performance, helping them improve their FP calculation skills over time. These advanced features would not only improve the usability of the plugin but also make the learning process more engaging and data-driven.

5) Limitations of the Current Testing and Future Research Directions

While the current testing of the FP plugin was thorough in terms of functional verification, there are notable limitations to consider. The testing was primarily conducted on a small scale, with a limited number of students involved in the process. Therefore, further testing is required to assess the plugin's performance and effectiveness in larger classroom settings. It is important to conduct tests involving a broader range of students to ensure that the plugin remains efficient and functional as class sizes increase.

Additionally, the effectiveness of the plugin in improving students' learning outcomes has not been fully measured. While the plugin successfully automates the FP calculation process, further research is needed to assess how it impacts students' understanding of Function Point Analysis (FPA), their ability to perform calculations independently, and their overall engagement with the learning material. This evaluation should include both qualitative and quantitative measures of learning effectiveness, such as student feedback, performance improvements, and retention of FP concepts. These areas of investigation will be the focus of future research, aimed at refining the plugin and evaluating its impact on the educational process.

4. Conclusion

The development and integration of the Function Point (FP) plugin within Moodle has proven to be a successful endeavor, enhancing the learning process by automating the calculation and assessment of Function Points. The plugin's ability to provide real-time feedback has significantly improved the efficiency of FP learning, reducing the manual workload for instructors while ensuring accurate and timely results for students. The use of Moodle's modular architecture has greatly facilitated the plugin's implementation, ensuring seamless integration with existing Moodle functionalities. The plugin's performance and usability were validated through extensive testing, confirming its effectiveness in terms of requirement verification, UI testing, integration with Moodle, and output accuracy. The primary limitation is the limited testing scope. Subsequent research should examine scalability to larger classes and assess its impact on students' learning. Future studies should focus on testing the plugin in larger classrooms to evaluate its scalability and performance under higher user loads. Additionally, the impact of the plugin on students' learning outcomes needs to be assessed through both qualitative and quantitative measures to determine its effectiveness in enhancing student engagement and understanding of Function Point Analysis. Advanced features, such as analytics and AI-assisted feedback, present exciting opportunities to further improve the learning experience. These potential improvements will be explored in future versions of the plugin, ensuring that it remains a valuable tool for both instructors and students.

Acknowledgment

This article was produced as part of research aimed at enhancing the capabilities of the Software Engineering Laboratory. We would like to express our sincere gratitude to P3M Politeknik Negeri Bandung for their invaluable support in facilitating this research. Special thanks are also extended to Aldy Albarrizky for his significant contribution to the exploration and integration of the Moodle plugin.

Declarations

Author contribution. The authors contributed equally to the design, development, and testing of the Moodle plugin for Function Point analysis. All authors were responsible for the plugin development, integration, and testing phases. All Authors contributed to the writing, data analysis, and reviewing of the paper. All authors approved the final manuscript.

Funding statement. This research was supported by [P3M Politeknik Negeri Bandung] under grant number [B/74.7/PL1.R7/PG.00.03/2021] in the year [2021].

Conflict of interest. The authors declare no conflict of interest.

Additional information. No additional information is available for this paper.

Data and Software Availability Statements

The data and software supporting the results of this study are available upon request. The datasets used during the study, including those related to testing results, are available from the corresponding author upon reasonable request..

References

[1] S. Simelane-Mnisi, "Effectiveness of LMS Digital Tools Used by the Academics to Foster Students' Engagement," *Educ Sci (Basel)*, vol. 13, no. 10, Oct. 2023, doi: 10.3390/educsci13100980.

[2] M. Furqon, P. Sinaga, L. Liliasari, and L. S. Riza, "The Impact of Learning Management System (LMS) Usage on Students," *TEM Journal*, vol. 12, no. 2, pp. 1082–1089, May 2023, doi: 10.18421/TEM122-54.

[3] D. Turnbull, R. Chugh, and J. Luck, "Learning management systems and social media: a case for their integration in higher education institutions," *Research in Learning Technology*, vol. 31, 2023, doi: 10.25304/rlt.v31.2814.

[4] T. Rahman and C. Nur Alfian, "Analisis Persepsi Mahasiswa Terhadap Teknologi Informasi dalam Pembelajaran Blended Learning Universitas Bina Sarana Informatika," *Media Jurnal Informatika*, vol. 16, no. 2, p. 154, Dec. 2024, doi: 10.35194/mji.v16i2.4694.

[5] R. R. Aljad, "Analysis of Development Trends and Experience of using LMS in Modern Education: An overview," *E-Learning Innovations Journal*, vol. 1, no. 2, pp. 86–104, Sep. 2023, doi: 10.57125/elij.2023.09.25.05.

[6] J. Bojiah, "Effectiveness of Moodle in Teaching and Learning," *Journal of Hunan University Natural Sciences*, vol. 49, no. 12, pp. 320–328, Dec. 2022, doi: 10.55463/issn.1674-2974.49.12.33.

[7] Natalia V. Belozertseva, Olga I. Vaganova, Irina V. Akimova, Anna V. Lapshova, and Roman A. Stepanov, "Monitoring and evaluation procedure with LMS Moodle," *Journal of the University of Zulia*, vol. 35, no. Special Issue, pp. 290–302, Nov. 2021, doi: 10.46925//rdluz.35.17.

[8] A. Abdula, H. Baluta, N. Kozachenko, D. Kassim, and F. Zhuravlev, "The Use of Moodle in the Teaching of Philosophy and Distance Learning," Scitepress, May 2022, pp. 616–630. doi: 10.5220/0010926600003364.

[9] S. H. P. W. Gamage, J. R. Ayres, and M. B. Behrend, "A systematic review on trends in using Moodle for teaching and learning," Dec. 01, 2022, *Springer Science and Business Media Deutschland GmbH*. doi: 10.1186/s40594-021-00323-x.

[10] A. Berggren *et al.*, "Practical and Pedagogical Issues for Teacher Adoption of IMS Learning Design Standards in Moodle LMS," 2005.

[11] Claudia Rodríguez Rodríguez, Roberto Vicente Rodríguez, Gisselle Cortés Moure, and Claudia León Pérez, "Personalization of Moodle with the integration of most used web technologies in higher education," *ITECKNE*, vol. 15, no. 2, pp. 131–142, Dec. 2018, doi: 10.15332/iteckne.v15i2.2074.

[12] V. Baneș, C. Ravariu, and A. Srinivasulu, "New Functionality for Moodle E-Learning Platform: Files Communication by Chat Window," *Applied Sciences (Switzerland)*, vol. 14, no. 18, Sep. 2024, doi: 10.3390/app14188569.

[13] M. Huerta, J. A. Caballero-Hernández, and M. A. Fernández-Ruiz, "Comparative Study of Moodle Plugins to Facilitate the Adoption of Computer-Based Assessments," *Applied Sciences (Switzerland)*, vol. 12, no. 18, Sep. 2022, doi: 10.3390/app12188996.

[14] A. Rahmani and J. Lian Min, "Automatic Grading System to Supporting Blended Learning in Basic Programming Practice-an Experience Report," 2020. [Online]. Available: https://github.com/hit-

[15] M. C. Sáiz-Manzanares, R. Marticorena-Sánchez, and C. I. García-Osorio, "Monitoring students at the university: Design and application of a moodle plugin," *Applied Sciences (Switzerland)*, vol. 10, no. 10, May 2020, doi: 10.3390/app10103469.

[16] X. Zhang, V. C. Lee, D. Xu, J. Chen, and M. S. Obaidat, "An Effective Learning Management System for Revealing Student Performance Attributes," Mar. 2024, [Online]. Available: http://arxiv.org/abs/2403.13822

[17] L. Lavazza, A. Locoro, and R. Meli, "Software Development and Maintenance Effort Estimation Using Function Points and Simpler Functional Measures," *Software*, vol. 3, no. 4, pp. 442–472, Oct. 2024, doi: 10.3390/software3040022.

[18] L. Lavazza, "Automated function points: Critical evaluation and discussion," in *International Workshop on Emerging Trends in Software Metrics, WETSoM*, IEEE Computer Society, Aug. 2015, pp. 35–43. doi: 10.1109/WETSoM.2015.13.

[19] J. Shah and N. Kama, "Extending function point analysis effort estimation method for software development phase," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Feb. 2018, pp. 77–81. doi: 10.1145/3185089.3185137.

[20] A. Yhurinda, P. Putri, and A. P. Subriadi, "Software Cost Estimation Using Function Point Analysis," 2018.