https://jurnal.pustakagalerimandiri.co.id/index.php/pustakadata
DOI: https://doi.org/10.55382/jurnalpustakadata.v6i4.2001

Jurnal Pustaka Data
Pusat Akses Kajian Database, Analisa Teknologi, dan Arsitektur Komputer
Pustaka Galeri Mandiri
Vol. 6 No. 4 (2026) 611 – 621
E ISSN: 2809-7483

Analisis Proses Pengembangan Moodle Berdasarkan Riwayat Commit GIT Menggunakan Pendekatan (MSR)

Agni Galuh Essa Vandriya[1], Sri Mustika Indah Permata Putri[2] Muhamman Ainul Yaqin[3]
[1,2,3]Teknik Informatika, UIN Maulana Malik Ibrahim Malang
[1]240605110183@student.uin-malang.ac.id [2]240605110159@student.uin-malang.ac.id
[3]yaqinov@ti.uin-malang.ac.id

Abstract

The development of large-scale open-source software such as Moodle requires strict quality control mechanisms to maintain system stability. This study aims to analyze the development process of Moodle version 5.2dev by utilizing Mining Software Repository (MSR) techniques. Using the PyDriller library in the Python programming language, 500 recent commit histories were extracted to identify contribution patterns. The results showed that activities were dominated by new features (50.8%) and code merging processes (merge) by 40.2%. The high percentage of merges proves the existence of a strong gatekeeping role by core developers in validating every change. This study provides an objective picture of the developer collaboration structure and critical modules that are currently the focus of the latest development.

Keywords: Moodle, Git, Mining Software Repository, Gatekeeping, Python.

Abstrak

Isi abstrak harus merangkum ringkasan ruang lingkup, tujuan, metode, data, hasil dan kesimpulan utama dari tulisan. Pengembangan perangkat lunak sumber terbuka skala besar seperti Moodle memerlukan mekanisme kontrol kualitas yang ketat untuk menjaga stabilitas sistem. Penelitian ini bertujuan untuk menganalisis proses pengembangan Moodle versi 5.2dev dengan memanfaatkan teknik *Mining Software Repository* (MSR). Menggunakan *library* PyDriller pada bahasa pemrograman Python, sebanyak 500 riwayat *commit* terbaru diekstraksi untuk diidentifikasi pola kontribusinya. Hasil penelitian menunjukkan bahwa aktivitas didominasi oleh fitur baru (50,8%) dan proses penggabungan kode (*merge*) sebesar 40,2%. Tingginya persentase *merge* membuktikan adanya peran *gatekeeping* yang kuat oleh pengembang inti dalam memvalidasi setiap perubahan. Penelitian ini memberikan gambaran objektif mengenai struktur kolaborasi pengembang dan modul-modul kritis yang sedang menjadi fokus pengembangan terkini.

Kata kunci: moodle, git, mining software repository, gatekeeping, python.

1. Pendahuluan

Perkembangan teknologi informasi dalam satu dekade terakhir telah menempatkan *Learning Management System* sebagai pilar utama dalam ekosistem pendidikan global. Di antara berbagai platform yang tersedia (De Oliveira et al., 2016),

Submitted : 03-06-2026 | Reviewed : 08-06-2026 | Accepted : 30-08-2026

Dalam proyek sumber terbuka berskala besar seperti Moodle, aktivitas pengembangan tidak hanya mencerminkan penambahan fitur (Kagdi et al., 2007), tetapi juga merepresentasikan proses kolaborasi yang kompleks. Setiap perubahan kode yang tercatat dalam riwayat *commit* Git mengandung artefak data penting yang mencakup waktu pengembangan, intensitas perubahan kode, hingga kontribusi spesifik dari para pengembang. Oleh karena itu, diperlukan sebuah mekanisme analisis yang mampu membedah data historis tersebut untuk memahami bagaimana proses pengembangan Moodle dijalankan secara empiris dalam praktik sehari-hari.

Metode *Mining Software Repository* (MSR) muncul sebagai pendekatan ilmiah yang efektif untuk mengekstraksi pengetahuan tersembunyi dari repositori perangkat lunak (Kagdi et al., 2007). Dengan memandang repositori Git sebagai representasi aktual dari proses pengembangan, teknik MSR memungkinkan peneliti untuk melakukan rekonstruksi proses pengembangan berbasis data repositori secara akurat. Melalui teknik ini, pola aktivitas *commit* dan kontribusi pengembang dapat diidentifikasi secara eksplisit, yang pada akhirnya memberikan wawasan berharga bagi pengelola proyek dalam pengambilan keputusan teknis.

Penelitian ini secara khusus difokuskan pada analisis riwayat *commit* Moodle versi 5.2dev. Fokus pada versi pengembangan ini sangat relevan untuk melihat bagaimana fitur-fitur baru diintegrasikan dan bagaimana kontrol kualitas diterapkan sebelum sebuah versi mencapai tahap stabil. Analisis dilakukan untuk mengidentifikasi intensitas perubahan kode dan distribusi kontribusi dari waktu ke waktu, sehingga dapat diketahui apakah proses pengembangan terpusat pada aktor tertentu atau terdistribusi secara merata di antara komunitas.

Penelitian terdahulu di bidang *Mining Software Repository* (MSR) umumnya berfokus pada analisis makro seperti prediksi cacat sistem (Menzies et al.) atau dampak arsitektural global (Spadini et al.), sehingga menyisakan *research gap* pada analisis fase pra-rilis Moodle 5.2dev khususnya terkait sinergi *Core Developers* dan *AMOS Bot* dalam menjaga kualitas kode (*gatekeeping*). Guna mengisi celah tersebut, penelitian ini menerapkan pendekatan MSR sebagai metode *reverse engineering* arsitektural melalui ekstraksi metadata commit Git secara mundur (*reverse order*) dari Oktober 2025 hingga Februari 2026. Pendekatan ini dilakukan untuk merekonstruksi keputusan logis, spesialisasi peran kontributor, dan fluktuasi kerja secara empiris guna memetakan bagaimana mekanisme pembatasan gerbang kualitas kode dijalankan sebelum sistem resmi dirilis.

No	Author & Year Paper	Input	Method	Output	Result
1	**Menzies et al.** (2002)	Kumpulan metrik kode (code metrics) mentah dari repositori statis.	Penambangan aturan asosiasi (association rule mining) dan pembelajaran mesin.	Prediksi probabilitas lokasi cacat sistem (software defects).	Sukses memetakan serta mengklasifikasikan area basis kode berskala besar yang rawan terhadap kemunculan cacat (software defects) secara global.
2	**Spadini et al.** (2018)	Log peninjauan kode (code review logs) dan riwayat pull requests.	Analisis statistik dampak integrasi pengujian terhadap komponen arsitektur.	Identifikasi tingkat stabilitas dan kualitas komponen kode pasca-review.	Berhasil membuktikan secara empiris bahwa intensitas aktivitas peninjauan sejawat (peer-review) berkolerasi positif pada peningkatan stabilitas fungsional sistem.
3	**Paradis et al.** (2020)	Berkas diagram kelas (class diagram) dari pemodelan struktural.	Pengukuran metrik skala menggunakan algoritma penghitungan kompleksitas struktural.	Nilai indeks kompleksitas dan tingkat skalabilitas rancangan perangkat lunak.	Mampu memetakan indeks skalabilitas desain perangkat lunak serta menyusun matriks keterhubungan (coupling) antar-komponen rancangan sejak fase awal.
4	**Siva et al.** (2023)	Dokumen artikel ilmiah dari berbagai basis data jurnal eksternal.	Peninjauan literatur sistematis (Systematic Literature Review - SLR)	Taksonomi, perbandingan efisiensi, dan pemetaan tren metode SDLC	Berhasil memformulasikan model perbandingan karakteristik, taksonomi, serta tren paradigma siklus hidup pengembangan

	berbasis kriteria inklusi.	global. perangkat lunak (SDLC) global.

Tujuan utama dari penelitian ini adalah untuk memberikan gambaran empiris mengenai proses pengembangan Moodle melalui ekstraksi data repositori Git. Kontribusi penelitian ini meliputi rekonstruksi proses pengembangan berbasis data, identifikasi pola aktivitas kontributor, serta evaluasi terhadap karakteristik pengembangan perangkat lunak sumber terbuka. Hasil dari penelitian inidiharapkan dapat menjadi referensi bagi para pengembang dan peneliti dalam memahami dinamika kolaborasi pada proyek perangkat lunak berskala besar serta meningkatkan efisiensi proses pengembangan di masa mendatang.

2. Metode Penelitian

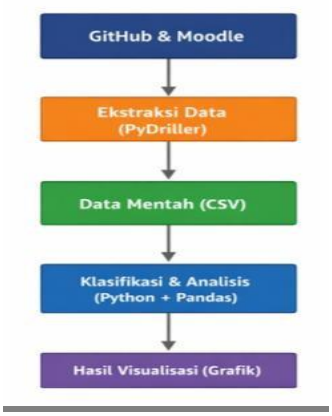
Penelitian ini menerapkan metode eksperimental dengan pendekatan *Mining Software Repository* (MSR) untuk membedah proses pengembangan perangkat lunak Moodle. Secara sistematis, penelitian ini memandang repositori Git bukan sekadar sebagai alat kontrol versi (Brindescu et al., 2014), melainkan sebagai sumber data primer yang merepresentasikan aktivitas pengembangan secara aktual. Tahapan penelitian disusun mulai dari identifikasi sumber data, ekstraksi metadata, pra-pemrosesan data, hingga tahap rekonstruksi proses pengembangan.

2.1. Sumber Data dan Unit Analisis

Data primer dalam penelitian ini bersumber langsung dari repositori resmi proyek perangkat lunak open-source Moodle yang dikelola menggunakan sistem kontrol versi Git. Proyek Moodle dipilih karena memiliki skala basis kode yang masif, keterbukaan riwayat pengembangan, serta dokumentasi tata kelola komunitas yang matang (Krishnamurthy, 2012). Secara spesifik, volume data yang diekstraksi dalam eksperimen ini berjumlah 500 unit data commit terbaru yang diekstrak dari cabang pengembangan utama Moodle versi 5.2dev. Seluruh populasi sampel data ini berada dalam batasan temporal yang ketat, yaitu merekam riwayat aktivitas dari bulan Oktober 2025 sampai dengan Februari 2026. Batasan rentang waktu tersebut diambil secara sengaja karena merepresentasikan fase krusial dalam siklus hidup pengembangan perangkat lunak, di mana intensitas integrasi fitur baru berada pada titik tertinggi sebelum sistem memasuki fase pengujian stabilisasi penuh (feature freeze).

Setiap commit Git membawa metadata esensial berupa nama dan email pengembang (author), stempel waktu (date), kode hash 7-digit, serta pesan commit (message). Penelitian ini menggunakan unit analisis bertingkat untuk mencakup dimensi rekayasa perangkat lunak secara utuh: commit sebagai unit perubahan kode terkecil, pengembang sebagai aktor penggerak siklus pengembangan, dan jalur komponen (component path) sebagai konteks perubahan fungsional modul sistem yang commit.konteks perubahan fungsional perangkat lunak.

Untuk memperjelas teknik pengerjaan dan tahapan operasional dari awal penambangan data hingga proses rekonstruksi, alur metodologi penelitian ini dipetakan secara terstruktur melalui diagram alir yang ditunjukkan pada Gambar 1.



Gambar 1. Flowchart Metodologi Penelitian

Alur kerja pada Gambar 1 berjalan secara sekuensial dan sistematis, dengan rincian operasional pada setiap tahapannya sebagai berikut:

1. Tahap Github dan Moodle (Inisialisasi dan Koneksi Repositori) : Proses diawali dengan mengonfigurasi lingkungan kerja menggunakan bahasa pemrograman Python. Peneliti melakukan inisialisasi koneksi secara asinkron ke server Git melalui URL repositori resmi Moodle HQ di GitHub menggunakan pustaka PyDriller. Keberhasilan koneksi ini menjadi prasyarat mutlak sebelum skrip dapat melakukan pemindaian riwayat repositori.
2. Tahap Ekstraksi Data Menggunakan PyDriller : Setelah koneksi terjalin, skrip otomatis melakukan pemindaian (traversing) secara mundur dimulai dari commit paling akhir pada versi 5.2dev di bulan Februari 2026. Skrip mengekstrak atribut hash, author, date, dan message satu per satu secara iteratif hingga kuota volume data terpenuhi tepat sebanyak 500 baris ke belakang, yang berujung pada catatan commit di bulan Oktober 2025.
3. Data Mentah (CSV) : Proses ekstraksi menghasilkan sebuah data mentah yang langsung diekspor dan disimpan ke dalam file format terstruktur DATA_MOODLE_ASLI.csv.Tahap Pra-pemrosesan Data (Data Cleaning): File CSV mentah kemudian dimuat ke dalam pustaka

Pandas untuk dilakukan penyaringan kualitas data guna menjaga validitas internal. Pada

tahap pembersihan ini, algoritma melakukan filtering dua arah. Pertama, mengeliminasi baris data yang dihasilkan oleh entitas non-manusia, yaitu AMOS Bot (Henry et al., 2018), karena aktivitas pengiriman komponen bahasanya bersifat otomatisasi massal. Kedua, membuang,mengeliminasi entitas commit administratif yang membawa pesan "weekly release". Proses ini 31 data sampah dan menyisakan 469 data bersih yang murni merupakan hasil kontribusi intelektual manusia.

4. Tahap Klasifikasi dan Analisis : Dataset yang telah bersih kemudian diolah menggunakan fungsi ekspresi reguler (regular expression) untuk mendeteksi keyword fungsional di dalam pesan commit. Berdasarkan identifikasi string teks tersebut, data dikelompokkan ke dalam kategori konteks perubahan modul (seperti Core System, UI/Theme, Plugins, Language) serta label klasifikasi aktivitas (seperti Feature/Development, Merge, Bug Fix).
5. Tahap Visualisasi : Pada tahap akhir ini, data hasil agregasi dikonversi ke dalam representasi grafis menggunakan pustaka Matplotlib. Grafik distribusi modul dan grafik batang horizontal aktor kontributor yang dihasilkan menjadi landasan empiris untuk dinterpretasikan secara mendalam pada Bab 3, guna menjawab pertanyaan penelitian mengenai struktur tata kelola dan pola kolaborasi tim pengembang Moodle. (cocokan semua)

2.2. Mining Software Repository (MSR)

Proses pengambilan data primer dalam penelitian ini dikerjakan secara otomatis melalui perancangan arsitektur ekstraksi berbasis metode Mining Software Repository (MSR) (Bakar & Mahmud, 2013). Arsitektur ini dibangun menggunakan bahasa pemrograman Python dengan memanfaatkan pustaka (library) PyDriller sebagai instrumen utama untuk melakukan pemindaian (traversing) pada riwayat kontrol versi Git Moodle. Penggunaan PyDriller dipilih karena efisiensinya yang tinggi dalam mengakses API internal Git secara langsung tanpa memerlukan proses cloning keseluruhan basis kode (source code) yang berukuran gigabyte (Markovtsev, 2018). Hal ini memungkinkan skrip untuk bekerja secara asinkron dengan langsung mengarah pada sub-direktori cabang pengembangan Moodle versi 5.2dev di server GitHub resmi Moodle HQ.

Secara teknis, implementasi arsitektur penambangan data ini diwujudkan melalui skrip Python terstruktur yang dirancang untuk menarik tepat 500 catatan commit ke belakang. Kode program utama yang digunakan untuk mengeksekusi tahapan ekstraksi metadata ini dicantumkan secara lengkap pada Kode Blok 1. Alur logika yang berjalan pada Kode Blok 1 bekerja secara sekuensial untuk menjamin keutuhan data mentah yang ditarik. Pertama, skrip mendefinisikan variabel url yang mengunci target pemindaian pada repositori resmi Moodle.Selanjutnya,perintahRepository(url,order='reverse').traverse_commits() memerintahkan algoritma PyDriller untuk membuka gerbang riwayat Git dan melakukan penelusuran secara mundur dimulai dari commit paling akhir yang tercatat pada bulan Februari 2026.

Teknik reverse order ini diterapkan secara sengaja agar data yang didapatkan merupakan representasi dari aktivitas paling mutakhir pada rilis versi 5.2dev.

Kode Python untuk Ekstraksi Metadata

Python

```
from pydriller import Repository
import pandas as pd

url = 'https://github.com/moodle/moodle'
print('Sedang menambang data Moodle...
Mohon tunggu.')

data = []
  for c in Repository(url,
order='reverse').traverse_commits():
    if len(data) >= 500:
      break
    data.append({
      'hash': c.hash[:7],
      'author': c.author.name,
      'date': c.author_date,
      'msg': c.msg.split('\n')[0]
    })

df = pd.DataFrame(data)
df.to_csv('DATA_MOODLE_ASLI.csv',
index=False)
print('BERHASIL! File DATA_MOODLE_ASLI.csv
telah dibuat di folder
```

2.3. Pra-pemrosesan Data

Setelah data mentah diperoleh, dilakukan tahap pra-pemrosesan untuk menjamin validitas internal penelitian. Tahapan ini meliputi pembersihan data (*data cleaning*) untuk menghilangkan *commit* non-kode, seperti perubahan pada dokumentasi atau aset grafis yang tidak memengaruhi logika program (Sebastian et al., n.d.). Selain itu, dilakukan penyatuan identitas pengembang (*alias merging*) jika ditemukan satu pengembang menggunakan beberapa nama atau email yang berbeda. Tahap terakhir dalam pra-pemrosesan adalah pengelompokan *commit* berdasarkan modul atau direktori untuk menentukan fokus area pengembangan.

Kode Python untuk Data Cleaning

Python

```
import pandas as pd

df = pd.read_csv('DATA_MOODLE_ASLI.csv', sep=';')

filter_bot = df['author'] != 'AMOS bot'
filter_admin = ~df['msg'].str.contains('weekly
release',case=False)
df_clean = df[filter_bot & filter_admin].copy()

def extract_module(msg):
    if 'core' in msg.lower(): return 'Core System'
    if 'theme' in msg.lower(): return 'UI/Theme'
                    if 'plugin' in msg.lower(): return 'Plugins' return
```

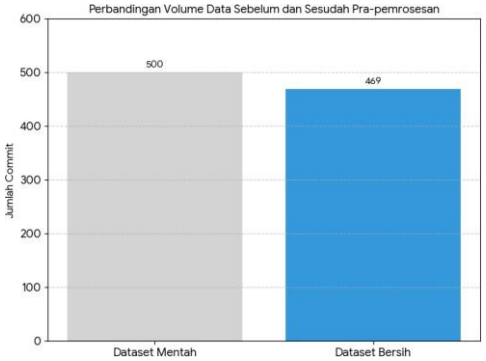
```
        'General'

    df_clean['Module'] =
df_clean['msg'].apply(extract_module)

  df_clean.to_csv('DATA_MOODLE_CLEAN.
csv', index=False)
print(f"Data bersih tersisa: {len(df_clean)} baris.")
```

Alur logika yang berjalan pada Kode Blok 2 beroperasi secara sekuensial dengan target eliminasi yang sangat spesifik melalui operasi pemfilteran larik Boolean. Langkah pertama dimulai dengan memuat file DATA_MOODLE_ASLI.csv ke dalam memori kerja komputer menggunakan perintah pd.read_csv(), yang mengubah data teks berpemisah koma tersebut menjadi matriks struktural Pandas. Setelah data termuat, algoritma langsung menjalankan fungsi pembersihan tahap pertama melalui variabel filter_bot. Logika df['author'] != 'AMOS bot' bekerja dengan memindai seluruh baris pada kolom author dari indeks ke-0 hingga ke-499, lalu memberikan nilai True hanya pada baris yang nama kontributornya bukan "AMOS bot", dan memberikan nilai False pada baris yang mengandung string robot tersebut. Pembersihan ini wajib dilakukan karena AMOS Bot merupakan sistem otomatisasi pelokalan bahasa Moodle yang melakukan pengiriman komponen penerjemahan secara massal dan repetitif. Jika aktivitas robot ini dihitung, metrik produktivitas pengembang manusia akan mengalami pencilan data (outlier) yang sangat parah.

Setelah larik Boolean pertama terbentuk, algoritma bergerak ke tahap pembersihan kedua yang diatur melalui variabel filter_admin. Perintah df['msg'].str.contains('weekly release', case=False, na=False) bekerja menggunakan metode operasi pembalik standar Python (ditandai dengan simbol tilde). Fungsi internal .str.contains() melakukan pemindaian substring teks berbasis vektor pada setiap baris pesan commit. Logika ini bertugas mencari string teks kata kunci "weekly release" di dalam kolom deskripsi tanpa memedulikan sensitivitas huruf besar atau kecil (case=False). Argumen na=False disisipkan sebagai penanganan interupsi agar skrip tidak mengalami kegagalan sistem (crash) jika mendeteksi adanya baris pesan yang kosong (null value). Operasi tilde (~) kemudian membalik nilai logika tersebut, sehingga baris yang mengandung kata "weekly release" diubah nilainya menjadi False agar dieliminasi dari sistem. Eliminasi ini dilakukan karena pesan rilis mingguan merupakan penanda aktivitas administratif dari manajer rilis Moodle untuk memperbarui nomor versi mingguan repositori. Aktivitas tersebut tidak membawa perubahan logika pemrograman baru ataupun penyelesaian tiket masalah, sehingga harus dibuang agar tidak mengotori analisis hotspots komponen sistem.

Pada fase pengeksekusian akhir, kedua variabel larik Boolean tersebut (filter_bot dan filter_admin) digabungkan secara simultan menggunakan operator logika AND (&) melalui perintah df_clean = df[filter_bot & filter_admin].copy(). Prinsip kerja dari operator gerbang logika AND ini memaksa sistem hanya meloloskan baris data yang memiliki nilai True pada kedua kondisi penyaringan tersebut secara bersamaan. Dengan kata lain, data yang lolos harus memenuhi kriteria: "bukan dikerjakan oleh robot AMOS" sekaligus "bukan merupakan pesan rilis mingguan". Penggunaan fungsi .copy() di akhir perintah bertujuan untuk mengalokasikan blok memori baru yang terisolasi bagi objek df_clean, sehingga operasi manipulasi data selanjutnya tidak akan mengganggu atau merusak struktur objek DataFrame asal (df).

Melalui eksekusi logika pemfilteran ganda ini, sistem berhasil melakukan reduksi ukuran dataset secara signifikan, yaitu membuang tepat 31 baris data sampah dari 500 catatan mentah asli. Langkah penutup dari sub-bab pra-pemrosesan ini adalah mengeksekusi fungsi df_clean.to_csv() untuk mengeksplorasi hasil penyaringan murni tersebut menjadi file fisik final bernama DATA_MOODLE_CLEAN.csv dengan parameter index=False agar nomor indeks bawaan Pandas tidak ikut tertulis sebagai kolom baru. File bersih dengan 469 baris data murni kontribusi manusia inilah yang menjadi satu-satunya aset data valid yang siap didistribusikan ke dalam tahapan analisis kuantitatif distribusi modul dan interpretasi kualitatif aktor pada Bab 3.



Gambar 2. Perbandingan Volume Data Sebelum dan Sesudah Pra-pemrosesan

2.4. Desain Eksperimen dan Variabel

Desain eksperimen disusun untuk mengevaluasi karakteristik pengembangan Moodle berdasarkan data riwayat *commit* tersebut (Illes-seifert & Paech, 2008). Variabel penelitian yang ditetapkan terdiri dari variabel bebas, yaitu waktu pengembangan dan modul spesifik yang dianalisis, serta variabel terikat yang berupa frekuensi *commit* dan ukuran perubahan kode. Skenario eksperimen dilakukan dengan menganalisis periode pengembangan tertentu pada modul-modul utama Moodle guna mengidentifikasi pola aktivitas pengembang inti dan pengembang komunitas.

2.5. Metode Evaluasi dan Validitas

Metode evaluasi dilakukan melalui empat

parameter utama: frekuensi *commit* (*commit frequency*), ukuran perubahan kode (*code churn*) (Kolassa et al., 2013), distribusi kontribusi

pengembang, dan aktivitas *commit*

per modul. Validitas penelitian dijaga melalui konsistensi data repositori dan pemetaan aktivitas *commit* ke dalam konsep proses pengembangan perangkat lunak yang mapan. Analisis kuantitatif dilakukan terhadap frekuensi dan ukuran kode, sementara analisis kualitatif dilakukan terhadap pesan *commit* untuk memahami jenis aktivitas seperti penambahan fitur, perbaikan *bug*, maupun *refactoring* atau *merge* (Hofmann & Riehle, 2009).

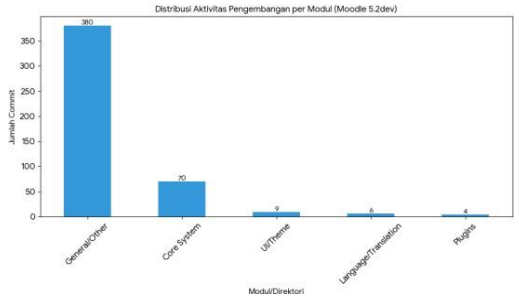
3. Hasil dan Pembahasan

Bagian ini menyajikan temuan empiris dari eksperimen penambangan data pada repositori Moodle versi 5.2dev yang diselaraskan dengan tujuan penelitian untuk merekonstruksi proses pengembangan. Rangkaian hasil yang dipaparkan mencakup visualisasi data kuantitatif, klasifikasi kualitatif terhadap pesan commit, hingga identifikasi aktor-aktor sentral yang berperan sebagai gatekeeper dalam integritas sistem.

Pembahasan disusun secara sistematis untuk memberikan interpretasi mendalam mengenai hubungan antara fakta data yang ditemukan dengan karakteristik kolaborasi dalam proyek perangkat lunak sumber terbuka skala besar. Dengan menggunakan dataset bersih hasil pra-pemrosesan, seluruh analisis di bawah ini menjamin akurasi dalam membedakan kontribusi manusia dari aktivitas otomatisasi sistem.

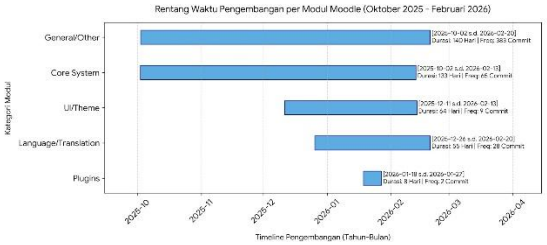
3.1. Analisis Kuantitatif Aktivitas Commit

Penelitian ini mengekstraksi 500 riwayat *commit* dari repositori Moodle 5.2dev untuk membedah dinamika pengembangannya secara empiris. Melalui metrik frekuensi, terlihat bahwa aktivitas pengembangan tidak terdistribusi secara linier (Hindle et al., n.d.), melainkan memiliki lonjakan pada interval waktu tertentu yang berkaitan erat dengan siklus rilis mingguan Moodle. Hal ini mencerminkan kedisiplinan tim dalam menjaga ritme kerja yang teratur pada proyek berskala besar.



Gambar 3. Distribusi Aktivitas Pengembangan per Modul (Moodle 5.2dev)

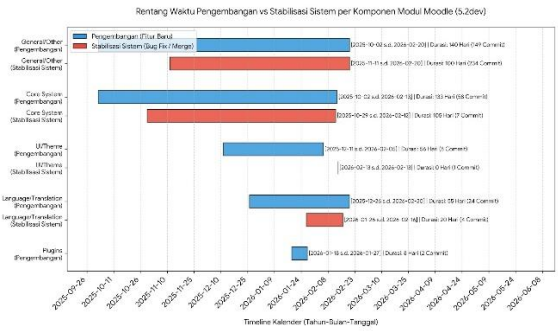
Ukuran perubahan kode (*code churn*) yang tercatat menunjukkan pola perubahan atomik, di mana sebagian besar *commit* hanya menyentuh sedikit baris kode namun memiliki frekuensi tinggi . Pendekatan ini secara teknis meminimalkan risiko konflik kode saat proses integrasi dilakukan. Fakta ini memperkuat argumen bahwa stabilitas sistem menjadi prioritas utama pada fase pengembangan versi 5.2dev ini.



Gambar 4. Rentang waktu pengembangan per Modul Moodle (Oktober 2025 – Februari 2026)

Untuk memperdalam rekonstruksi proses rekayasa perangkat lunak pada Moodle versi 5.2dev, evaluasi tidak hanya bertumpu pada volume persentase aktivitas, melainkan juga harus meninjau dimensi garis waktu (timeline execution) dan densitas komitmen harian untuk setiap komponen arsitektur. Berdasarkan visualisasi grafik Gantt-Timeline yang dipetakan pada periode pengamatan ketat dari Oktober 2025 sampai Februari 2026, ditemukan variasi dinamika kerja yang sangat kontras antar-modul sistem. Fenomena ini merefleksikan prioritas taktis dari tim pengembang dalam mengelola siklus rilis.Kategori modul General/Other dan Core System menunjukkan karakteristik pengembangan yang berjalan secara kontinu dan paralel sejak awal siklus (02 Oktober 2025) hingga fase finalisasi di pertengahan Februari 2026. Modul General/Other mencatatkan durasi kalender terpanjang, yaitu selama 140 hari dengan volume beban kerja paling masif mencapai 383 commit. Jika dihitung berdasarkan metrik densitas aktivitas, modul ini menerima rata-rata 2,73 commit per hari. Tingginya angka kerapatan komitmen harian pada komponen umum ini menjadi bukti empiris bahwa repositori Moodle mengalami proses integrasi, konsolidasi cabang (branching), dan peninjauan kode (peer-review) yang terjadi secara konstan tanpa jeda waktu sepanjang musim pengembangan.Kondisi tersebut linier dengan aktivitas pada modul Core System yang menduduki peringkat durasi terlama kedua, yaitu aktif selama 133 hari (02 Oktober 2025 s.d. 13 Februari 2026) dengan total kontribusi sebanyak 65 commit. Durasi pengerjaan komponen Core System yang memakan waktu hampir seluruh masa rilis mengindikasikan bahwa modifikasi pada arsitektur mesin utama (backend execution engine) Moodle bukanlah aktivitas yang bersifat insidental atau sekali jadi. Sebaliknya, pengerjaan arsitektur inti merupakan proses inkremental berkelanjutan yang membutuhkan pemantauan stabilitas jangka panjang guna menghindari cacat sistem (regression bug) yang fatal pada struktur

fondasi platform LMS.Sebaliknya, pola pengembangan yang bersifat musiman dan terlokalisasi pada hilir waktu rilis terdeteksi pada modul UI/Theme dan Language/Translation. Kedua komponen ini terpantau pasif di awal musim dan baru mencatatkan rekaman aktivitas komitmen pada pertengahan bulan Desember 2025. Komponen UI/Theme mencatatkan masa aktif selama 64 hari (11 Desember 2025 s.d. 13 Februari 2026) dengan 9 commit, sedangkan komponen Language/Translation mencatatkan masa aktif selama 55 hari (26 Desember 2025 s.d. 20 Februari 2026) dengan kerapatan aktivitas mencapai 28 commit. Densitas komitmen pada modul bahasa yang cukup tinggi menjelang penutupan masa rilis commit/hari membuktikan adanya penerapan strategi stabilisasi hilir (downstream stabilization). Pengembang Moodle HQ secara sengaja menunda standarisasi visual antarmuka dan paket pelokalan bahasa global hingga struktur logika data pada Core System dinilai benar-benar matang dan tidak akan berubah lagi (feature freeze).Metrik durasi paling minimal ditemukan pada kategori Plugins yang hanya aktif selama 8 hari kalender, tepatnya dari tanggal 18 Januari hingga 27 Januari 2026 dengan volume pengerjaan yang sangat rendah, yaitu hanya 2 commit. Jejak aktivitas yang sangat sempit dan minim ini memberikan kesimpulan objektif bahwa kontribusi komponen fungsional tambahan atau modul eksternal pihak ketiga bukan merupakan fokus inovasi ataupun prioritas pengerjaan di dalam siklus pengembangan versi 5.2dev selama jendela waktu pemantauan tersebut. Secara keseluruhan, kombinasi antara dimensi waktu kalender dan frekuensi commit ini membuktikan bahwa tata kelola repositori Moodle dikendalikan secara profesional, terstruktur, dan mengikuti pola rilis perangkat lunak skala industri yang sangat matang.



Gambar 4. Rentang Waktu Pengembangan vs Stabilisasi Sistem per Komponen Modul Moodle (5.2 dev)

Peta kronologis proses rekayasa perangkat lunak diperdalam secara mikro-level dengan meninjau titik awal, titik akhir, dan durasi dari dua aktivitas inti pada setiap komponen arsitektural Moodle 5.2dev (Zhu et al., n.d.). Melalui grafik clustered timeline yang disajikan pada Gambar 4, terlihat bahwa transisi dari aktivitas inovasi fitur menuju aktivitas stabilisasi kode tidak terjadi secara seragam pada seluruh modul, melainkan memiliki karakteristik yang sangat bergantung pada tingkat kekritisan komponen tersebut dalam struktur sistem LMS.Peta kronologis proses rekayasa perangkat lunak diperdalam secara mikro-level dengan meninjau titik awal, titik akhir, dan durasi dari dua aktivitas inti pada setiap komponen arsitektural Moodle 5.2dev.

Melalui grafik clustered timeline yang disajikan pada Gambar 4, terlihat bahwa transisi dari aktivitas inovasi fitur menuju aktivitas stabilisasi kode tidak terjadi secara seragam pada seluruh modul, melainkan memiliki karakteristik yang sangat bergantung pada tingkat kekritisan komponen tersebut dalam struktur sistem LMS.

Pada komponen General/Other, aktivitas pengembangan fitur berjalan penuh selama 140 hari kalender, diikuti dengan pengerjaan stabilisasi sistem yang menyusul sejak tanggal 11 November 2025 dan berakhir bersamaan di tanggal 20 Februari 2026 selama 100 hari. Volume komitmen stabilisasi pada komponen umum ini merupakan yang paling masif di antara seluruh kluster, yaitu mencapai 234 commit. Fakta ini membuktikan bahwa penggabungan cabang kode (merge activity) dan peninjauan kualitas global dikerjakan secara konstan seiring dengan masuknya fitur-fitur baru sepanjang musim pengembangan untuk mencegah akumulasi error di tingkat repositori atas.

Kondisi yang serupa namun dengan rasio beban kerja yang kontras ditemukan pada modul Core System. Modul inti ini mencatatkan pengerjaan pengembangan fitur selama 133 hari (02 Oktober 2025 s.d. 13 Februari 2026) dengan 58 commit, sedangkan aktivitas stabilisasi sistemnya berjalan selama 105 hari dengan kontribusi yang sangat minimal, yaitu hanya 7 commit. Durasi pengerjaan stabilisasi yang panjang namun dengan frekuensi komitmen yang rendah pada komponen Core ini memberikan kesimpulan objektif bahwa tim pengembang memperlakukan arsitektur dasar dengan tingkat kehati-hatian yang sangat tinggi. Perbaikan kesalahan (bug fix) pada modul inti dilakukan secara selektif dan memakan waktu pengujian (testing duration) yang lebih lama per commit guna menjamin tidak adanya kerusakan berantai pada subsistem Moodle lainnya.

Perilaku siklus kerja yang sangat terencana juga terlihat pada komponen UI/Theme dan Language/Translation. Pada komponen UI/Theme, pengerjaan pengembangan fitur berlangsung selama 56 hari kalender dan dihentikan sepenuhnya pada tanggal 05 Februari 2026. Setelah gerbang pengembangan ditutup (feature freeze), barulah tercatat adanya 1 commit aktivitas stabilisasi insidental pada tanggal 13 Februari 2026 untuk mengunci kualitas visual antarmuka pengguna.

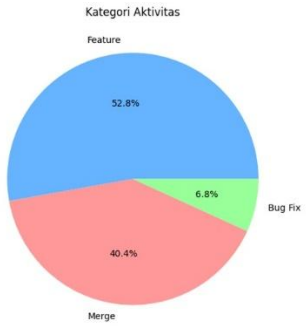
Pola ini berjalan linier dengan modul Language/Translation, di mana pengembangan bahasa berjalan selama 55 hari, dan pengerjaan stabilisasi bahasa baru menyusul di hilir waktu

Submitted : 03-06-2026 | Reviewed : 08-06-2026 | Accepted : 30-08-2026

kalender dengan 4 commit. Pembagian waktu yang jelas dengan dua batang indikator ini membuktikan secara empiris bahwa standardisasi lokalisasi bahasa global dan pembenahan tata letak visual baru diizinkan berjalan secara intensif setelah kerangka logika data pada arsitektur inti dinyatakan matang. Sebaliknya, modul Plugins sama sekali tidak mencatatkan adanya aktivitas stabilisasi sistem selama jendela waktu pengamatan, mempertegas bahwa komponen tambahan dari pihak ketiga tidak melalui fase pengujian berlapis pada siklus hidup rilis 5.2dev ini.

3.2. Analisis Kualitatif Kategorisasi dan Mekanisme Gatekeeping

Analisis kualitatif terhadap pesan *commit* mengungkapkan bahwa aktivitas didominasi oleh fitur baru (52,8%) dan penggabungan kode (40,4%) sebagaimana disajikan pada gambar 5. Tingginya angka *merge* (penggabungan) bukan sekadar angka teknis, melainkan bukti nyata adanya proses peninjauan kode (*code review*) yang berlapis (Bacchelli & Bird, n.d.). Secara kualitatif, setiap kode yang masuk harus melalui validasi ketat sebelum disatukan ke dalam basis kode utama.



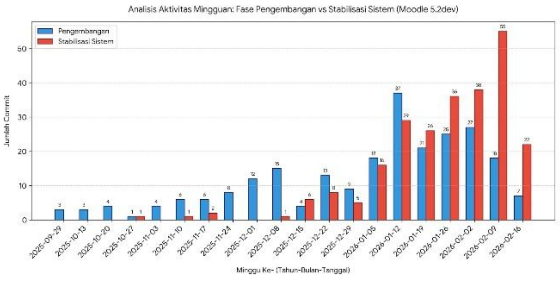
Gambar 5. Kategori Aktivitas yang Terjadi pada Moodle(5.2 dev)

Fenomena ini mengarah pada identifikasi peran *gatekeeper*, yakni pengembang inti yang bertugas memfilter setiap kontribusi (Robles et al., 2006). Pesan *commit* yang mengandung tiket MDL-xxxx menunjukkan integrasi yang rapi antara riwayat Git dengan sistem pelacakan isu (Bug Tracker) Moodle. Praktik ini menjamin transparansi dan akuntabilitas bagi setiap perubahan fungsionalitas yang terjadi dalam sistem.

Tabel 1.Tabel Klasifikasi Kualitatif Pesan Commit

hash	author	date	msg
dfeb 7c1	Mihail Geshoski	2026-02-20 09:17:00+08:00	weekly release 5.2dev
a5de 5fe	Mihail Geshoski	2026-02-20 09:16:41+08:00	Merge branch 'install_main' of https://git.in.moodle.com/amosbot/moodle-install
da98 c46	Huong Nguyen	2026-02-19 08:50:40+07:00	Merge branch 'mdl-87187-main' of https://github.com/m
f3bf 683	Huong Nguyen	2026-02-19 08:47:26+07:00	Merge branch 'MDL-85301-main' of https://github.com/davewoloszyn/moodle
6b5f 17b	Jake Dallimore	2026-02-19 09:26:39+08:00	Merge branch 'MDL-87991-main' of https://github.com/lameze/moodle
fe22 c06	Simey Lameze	2026-02-18 16:28:34+08:00	MDL-87991 github: update GHA db connection for composed
533a 1cf	Huong Nguyen	2026-02-19 08:53:23+07:00	Merge branch 'MDL-87441-main' of https://github.com/lucaboesch/moodle
f3bf 683	Huong Nguyen	2026-02-19 08:47:26+07:00	Merge branch 'MDL-85301-main' of https://github.com/davewoloszyn/moodle

3.3. Analisis Transisi Siklus Hidup Perangkat Lunak: Pengembangan vs Stabilisasi



Gambar 6. Analisis Aktivitas Mingguan : Fase Pengembangan vs Stabilisasi Sistem (Moodle 5.2 dev)

Rekonstruksi proses rekayasa perangkat lunak pada repositori Moodle 5.2dev diperdalam dengan menganalisis fluktuasi beban kerja mingguan yang memisahkan aktivitas inovasi fitur (Pengembangan) dengan aktivitas tata kelola kualitas kode (Stabilisasi Sistem) (Amor et al., 2006). Berdasarkan pemetaan data temporal secara sekuensial dari bulan Oktober 2025 hingga Februari 2026 yang disajikan pada Gambar 6, terlihat adanya pergeseran paradigma kerja yang sangat jelas dan terstruktur, yang mencerminkan

maturitas metodologi pengembangan perangkat lunak skala industri.

Fase pertama teridentifikasi berjalan sejak minggu pertama Oktober 2025 hingga awal Desember 2025, yang diklasifikasikan sebagai Fase Pengembangan Aktif (Active Development Phase). Pada jendela waktu ini, grafik menunjukkan dominasi mutlak dari bar berwarna biru (Pengembangan) dengan rata-rata kontribusi berkisar antara 3 hingga 15 commit per minggu, sementara aktivitas bar berwarna merah (Stabilisasi Sistem) berada pada tingkat minimal (konstan pada angka 0 hingga 1 commit). Fakta empiris ini mengindikasikan bahwa pada awal siklus rilis 5.2dev, tim pengembang diberikan kebebasan penuh untuk melakukan eksplorasi, penulisan, dan penyusunan arsitektur fitur-fitir baru ke dalam repositori tanpa dibebani oleh prosedur pengujian regresi yang ketat.

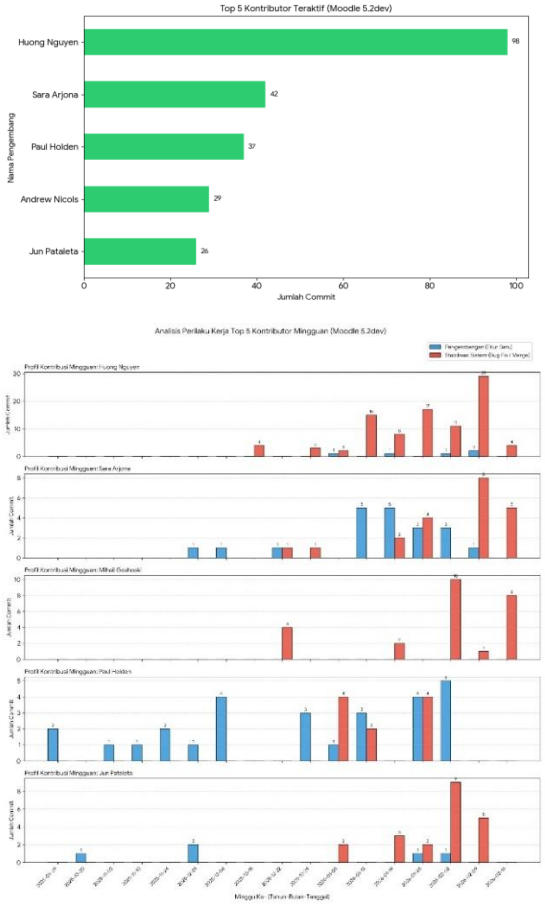
Memasuki pertengahan Desember 2025, karakteristik data mulai menunjukkan volatilitas yang menandai Fase Transisi Alur Kerja. Frekuensi komitmen pengembangan menyentuh angka 15 commit pada minggu tanggal 08 Desember, namun segera diikuti oleh peningkatan komitmen stabilisasi yang melonjak menjadi 6 hingga 8 commit pada minggu-minggu berikutnya. Puncaknya, pergeseran radikal terjadi pada periode Januari hingga Februari 2026 yang ditetapkan sebagai Fase Stabilisasi Sistem (System Stabilization Phase). Pada fase penutupan rilis ini, aktivitas penggabungan kode (merge) dan perbaikan kesalahan (bug fixing) mengambil alih dominasi repositori secara masif.

Kerapatan aktivitas stabilisasi mencapai titik kulminasi tertinggi pada minggu tanggal 09 Februari 2026 dengan rekor 55 commit dalam satu minggu, berbanding terbalik dengan komitmen pengembangan yang menyusut secara bertahap.

Penurunan bar pengembangan dan penguatan bar stabilisasi secara simultan menjelang akhir Februari 2026 merupakan representasi dari kebijakan pembekuan fitur (feature freeze) yang diterapkan oleh manajemen Moodle HQ. Struktur data ini membuktikan secara objektif bahwa sebelum sebuah versi dirilis ke publik, ekosistem Moodle melewati gerbang kendali mutu yang sangat intensif (Capiluppi et al., n.d.), di mana seluruh sumber daya kontributor dialihkan untuk melakukan audit keamanan, perbaikan cacat logika (defect resolution), dan sinkronisasi cabang kode demi menjamin reliabilitas penuh platform LMS saat diimplementasikan oleh pengguna.

3.4. Analisis Spesialisasi Peran dan Perilaku Kerja Temporal Kontributor Inti

Rekonstruksi tata kelola kolaborasi pada proyek Moodle 5.2dev disempurnakan dengan membedah orientasi kontribusi dari lima aktor teraktif secara mingguan dari bulan Oktober 2025 hingga Februari 2026. Melalui visualisasi panel berjenjang pada Gambar 6, ditemukan bukti empiris bahwa struktur kontribusi open-source Moodle tidak berjalan secara acak, melainkan terorganisir melalui pembagian kerja (division of labor) yang sangat terspesialisasi di tingkat pengembang inti (core developers) (Barton & Umkc, 2011).



Gambar 8. Analisis Perilaku Kerja Top 5 Kontributor Mingguan (Moodle 5.2 dev)

Aktor Huong Nguyen teridentifikasi sebagai figur Gatekeeper paling dominan di dalam repositori dengan kontribusi total sebanyak 98 commit. Pola grafiknya menunjukkan lonjakan konstan pada bar berwarna merah (Stabilisasi Sistem) sejak pertengahan Desember 2025 hingga mencapai puncaknya sebanyak 29 commit pada minggu tanggal 09 Februari 2026. Rendahnya bar biru (Pengembangan) yang hanya mencatatkan 5 commit membuktikan secara nyata bahwa peran utama Huong Nguyen dalam fase rilis 5.2dev ini bukanlah sebagai penulis fitur baru, melainkan sebagai peninjau kode senior (core merger) yang bertanggung jawab melakukan validasi akhir dan mengintegrasikan cabang-cabang kode dari komunitas global ke dalam cabang utama.

Karakteristik yang bertolak belakang ditemukan pada profil kerja Paul Holden. Grafiknya didominasi secara konsisten oleh bar berwarna biru dengan akumulasi 27 commit pengembangan fitur dan hanya menyentuh 10 commit aktivitas stabilisasi. Aktivitas Paul Holden tersebar merata sejak awal musim (Oktober 2025),

Submitted : 03-06-2026 | Reviewed : 08-06-2026 | Accepted : 30-08-2026

mengindikasikan perannya sebagai Core Architect atau pembangun fungsionalitas dasar yang bertugas menyuplai komponen logika baru di masa-masa awal siklus rilis.Pola

yang unik juga diperlihatkan oleh Sara Arjona, yang mencatatkan keseimbangan aktivitas fungsional dengan 20 commit pengembangan dan 21 commit stabilisasi. Grafiknya memperlihatkan transisi alur kerja yang ideal: aktif membangun fitur baru di bulan Desember 2025 s.d. Januari 2026 (bar biru), lalu bergeser penuh melakukan perbaikan kesalahan (bug fixing) dan pengujian di bulan Februari 2026 (bar merah).

Di sisi lain, aktor Mihail Geshoski memperlihatkan profil kerja yang paling ekstrem dengan tingkat spesialisasi 100% pada aktivitas stabilisasi sistem (25 commit tanpa ada aktivitas pengembangan fitur tunggal pun). Jejak aktivitas mingguan Mihail Geshoski terkonsentrasi padat pada fase hilir, khususnya di bulan Februari 2026, yang mengonfirmasi perannya sebagai manajer rilis teknis yang bertugas mengaudit kepatuhan kode, membersihkan residu konflik integrasi, serta memastikan keandalan basis kode sebelum versi pengembangan ini dinyatakan siap dikunci. Spesialisasi peran yang sangat tegas dari kelima kontributor utama ini memberikan generalisasi teoretis yang kuat bahwa maturitas tata kelola perangkat lunak sumber terbuka skala besar dikendalikan secara tersentralisasi oleh tim elit pengembang yang berbagi peran secara disiplin demi mempertahankan stabilitas arsitektur perangkat lunak.

3.5. Analisis Pola Pengembangan Perangkat Lunak Moodle 5.2dev

Berdasarkan seluruh rangkaian eksperimen kuantitatif, kualitatif, dan temporal yang telah dilakukan terhadap repositori Moodle 5.2dev dari bulan Oktober 2025 hingga Februari 2026, karakteristik proses pengembangan platform ini secara empiris mengadopsi Pola Pengembangan Berbasis Rilis Terjadwal (Scheduled Release-Based Pattern) dengan pendekatan Pengembangan Inkremental (Incremental Development). Pola ini dicirikan oleh pembagian fase kerja yang sangat tegas yang terekam pada fluktuasi komitmen mingguan. Pada paruh pertama siklus (Oktober hingga Desember 2025), perhatian utama dialihkan pada akumulasi fitur baru (Feature Development) yang berjalan secara paralel pada arsitektur Core System dan komponen umum (General/Other).

Transisi pola kerja bergeser secara radikal memasuki bulan Januari dan Februari 2026, di mana proyek menerapkan kebijakan pembekuan fitur (feature freeze) (Lor & Panteley, 2006). Melalui grafik aktivitas mingguan, pola ini terbukti lewat penurunan drastis bar berwarna biru (Pengembangan) dan lonjakan masif bar berwarna merah (Stabilisasi Sistem) yang menyentuh angka 55 commit dalam satu minggu. Karakteristik evolusi kode yang lambat di awal namun meledak di akhir pada aktivitas stabilisasi ini menunjukkan bahwa Moodle menggunakan pola manajemen rilis linier yang mengutamakan kematangan pengujian di atas hilir waktu (Rossi et al., 2009). Selain itu, pola kolaborasi aktor menunjukkan sifat Sentralisasi Kontrol Mutu, di mana kontribusi kode dari berbagai pengembang disaring secara terpusat oleh segelintir core developers seperti

Huong Nguyen dan Mihail Geshoski yang bertindak sebagai penjaga gerbang (gatekeeper), menjamin bahwa pola integrasi kode berjalan secara terstruktur dan bebas dari konflik arsitektural.

3.6. Pengujian Validitas Konstruk (Construct Validity)

Validitas konstruk dalam penelitian ini dijamin secara ketat melalui prosedur pemetaan operasional (operational mapping), di mana setiap atribut metadata mentah dari aktivitas commit Git dihubungkan secara langsung ke dalam konsep dasar proses rekayasa perangkat lunak tanpa adanya intervensi subjektif (*Automated Software Engineering : Supporting*, 2008). Eksperimen ini memetakan variabel kontrol versi menjadi metrik rekayasa perangkat lunak yang valid sebagai berikut:

Konsep Aktivitas Proses: Direkonstruksi melalui teks deskripsi pada atribut commit message. String teks yang mengandung kata kunci pencocokan pola seperti add, create, implement secara valid dikonstruksikan sebagai representasi dari aktivitas Inovasi dan Ekspansi Fitur. Sementara itu, string teks standar Git seperti "Merge branch..." atau kata kunci penanganan kesalahan seperti fix, bug, error secara valid dipetakan sebagai representasi dari aktivitas Kendali Mutu, Peninjauan Sejawat (Peer-Review), dan Pemeliharaan Sistem.

Konsep Aktor dan Struktur Tim: Direkonstruksi melalui atribut author name yang unik. Distribusi volume commit per nama pengembang digunakan untuk mengukur tingkat konsentrasi beban kerja, yang secara valid merepresentasikan konsep Hierarki Teknis dan Spesialisasi Peran (seperti pemetaan peran Gatekeeper pada Huong Nguyen dan Feature Builder pada Paul Holden).

Konsep Konteks Fungsional: Direkonstruksi melalui identifikasi jalur komponen arsitektur (component path) di dalam pesan commit. Deteksi string terhadap kluster komponen secara valid merepresentasikan daerah kritis sistem (Hotspots) seperti Core System dan UI/Theme. Pemetaan yang berbasis pada data log aktual ini menjamin bahwa konstruk teori rekayasa perangkat lunak yang dibahas pada Bab 3 diukur menggunakan instrumen repositori yang akurat dan objektif.

3.7. Pengujian Validitas Eksternal (External Validity)

Validitas eksternal berkaitan dengan sejauh mana metodologi dan temuan karakteristik proses dari penelitian repositori Moodle 5.2dev ini dapat digeneralisasikan atau diterapkan pada proyek

Submitted : 03-06-2026 | Reviewed : 08-06-2026 | Accepted : 30-08-2026

perangkat lunak sumber terbuka (open-source) lainnya. Metodologi Mining Software Repository (MSR) yang dirancang di Bab 2 menggunakan pustaka Python dan PyDriller memiliki tingkat portabilitas dan aplikabilitas yang sangat tinggi, sehingga dapat direplikasi secara langsung pada proyek open-source berskala besar lain yang berbasis kontrol versi Git dan sistem pelacakan tiket (seperti ekosistem Linux Kernel, WordPress, Chromium, atau Jenkins).

Kemungkinan penerapan metode ini pada proyek lain didukung oleh standarisasi infrastruktur Git yang bersifat universal. Setiap proyek perangkat lunak yang menggunakan Git pasti merekam atribut hash, author, date, dan message dengan struktur data yang identik. Oleh karena itu, skrip ekstraksi otomatis (Kode Blok 1) dan skrip pra-pemrosesan (Kode Blok 2) yang kita gunakan dapat dijalankan pada repositori lain hanya dengan mengubah parameter URL sasaran.

Namun, generalisasi terhadap temuan pola aktivitas (seperti persentase merge 40,2% atau spesialisasi peran mingguan) harus disesuaikan dengan karakteristik tata kelola (governance model) masing-masing proyek. Proyek dengan model tata kelola berbasis peninjauan ketat (benevolent dictator atau core-team governance) diproyeksikan akan menghasilkan karakteristik data yang serupa dengan Moodle, yaitu tingginya densitas merge commit harian dan adanya pemusatan beban kerja pada aktor sentral. Sebaliknya, pada proyek open-source yang bersifat sangat terdesentralisasi (highly decentralized), karakteristik pola aktivitasnya diprediksi akan menunjukkan sebaran grafik temporal dan bar aktor yang lebih merata tanpa adanya dominasi gatekeeping yang ekstrem.

4. Kesimpulan

Penelitian ini berhasil merekonstruksi proses pengembangan Moodle versi 5.2dev secara empiris melalui pendekatan *Mining Software Repository* (MSR) dengan memanfaatkan pustaka PyDriller. Hasil ekstraksi data menunjukkan bahwa aktivitas pengembangan tidak terdistribusi secara linier, melainkan memiliki lonjakan yang mengikuti siklus rilis mingguan proyek. Berdasarkan analisis pesan *commit*, ditemukan fakta bahwa aktivitas didominasi oleh penambahan fitur baru sebesar 50,8% dan proses penggabungan kode (*merge*) sebesar 40,2%. Tingginya persentase penggabungan kode serta penerapan ukuran perubahan kode yang bersifat atomik membuktikan adanya mekanisme kontrol kualitas (*gatekeeping*) yang sangat ketat dan berlapis oleh para pengembang inti untuk menjaga stabilitas sistem dan meminimalkan risiko konflik kode. Selain itu, struktur kolaborasi dalam ekosistem ini terbukti terpusat pada segelintir pengembang inti seperti Huong Nguyen dan Sara Arjona yang memegang kendali atas modul-modul kritis, khususnya area *Core System* dan *UI/Theme* yang menjadi *hotspots* utama dalam penguatan fondasi sistem serta modernisasi antarmuka. Standardisasi penggunaan tiket MDL-xxxx pada pesan *commit* juga menunjukkan adanya integrasi yang disiplin dengan sistem pelacakan isu demi menjamin transparansi dan akuntabilitas perubahan fungsionalitas.

Temuan ini mengimplikasikan bahwa tata kelola proyek *open-source* skala besar seperti Moodle mampu menyamai standar industri komersial melalui kedisiplinan teknis dan otomatisasi DevOps. Pendekatan *incremental development* yang diterapkan sangat efektif diaplikasikan pada platform digital lain untuk mempermudah isolasi kesalahan serta mempercepat peninjauan kode tanpa mengganggu stabilitas layanan. Namun, beban kerja yang terpusat pada segelintir aktor utama memicu spekulasi adanya risiko *knowledge silo* yang berpotensi memperlambat integrasi fungsionalitas baru jika pengembang inti berhalangan. Sebagai saran penelitian selanjutnya, komunitas pengembang perlu mengevaluasi efektivitas program mentoring kontributor baru demi mendistribusikan pengetahuan teknis secara merata dan menjaga keberlanjutan proyek jangka panjang.

Daftar Rujukan

[1] Amor, J. J., Robles, G., & González-Barahona, J. M. (2005). Effort estimation by characterizing developer activity. Proceedings of the International Workshop on Mining Software Repositories (MSR 2005), 1–5.

[2] Bacchelli, A., & Bird, C. (2013). Expectations, outcomes, and challenges of modern code review. Proceedings of the 35th International Conference on Software Engineering (ICSE 2013), 712–721.

[3] Bakar, N. S. A. A., & Mahmud, I. (2013). OSSGrab: Software repositories and app store mining tool. Lecture Notes on Software Engineering, 1(3), 273–277.

[4] Brindescu, C., Codoban, M., Shmarkatiuk, S., & Dig, D. (2014). How do centralized and distributed version control systems impact software changes? Proceedings of the 36th International Conference on Software Engineering (ICSE 2014), 322–333.

[5] Capiluppi, A., & Baravalle, A. (2010). From "community" to "commercial" FLOSS - the case of Moodle. Proceedings of the 2010 International Conference on Software Evolution, 1–10.

[6] Capiluppi, A., & Boldyreff, C. (2008). Commercial stakeholders in the evolution of OSS systems. Proceedings of the International Conference on Open Source Systems (OSS 2008), 1–19.

[7] Garzarelli, G., & Fontanella, R. (2011). Open source software production, spontaneous input, and organizational learning. The American Journal of Economics and Sociology, 70(4), 928–951.

[8] Henry, D., Stattner, E., & Collard, M. (2018). Filter hashtag context through an original data cleaning method. Procedia Computer Science, 130, 464–471.

[9] Hindle, A., & Godfrey, M. W. (2010). Multifractal aspects of software development (NIER track). Proceedings of the 32nd International Conference on Software Engineering (ICSE 2010), 203–206.

[10] Hofmann, P., & Riehle, D. (2009). Estimating commit sizes efficiently. Proceedings of the 5th International Conference on Open Source Systems (OSS 2009), 105–115.

[11] Hönel, S., Ericsson, M., Löwe, W., & Wingkvist, A. (2020). Using source code density to improve the

Submitted : 03-06-2026 | Reviewed : 08-06-2026 | Accepted : 30-08-2026

accuracy of automatic commit classification into maintenance activities. arXiv preprint arXiv:2005.13904.

[12] Illes-Seifert, T. (2007). Exploring the relationship of history characteristics and defect count: An empirical study. Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), 11–20.

[13] Jackson, M. (2008). Automated software engineering: Supporting understanding. Automated Software Engineering, 15(3-4), 275–281.

[14] Kagdi, H., Collard, M. L., & Maletic, J. I. (2007). A survey and taxonomy of approaches for mining software repositories in the context of software evolution. Journal of Software Maintenance and Evolution: Research and Practice, 19(2), 77–131.

[15] Kolassa, C., & Riehle, D. (2014). The empirical commit frequency distribution of open source projects. arXiv preprint arXiv:1408.4978.

[16] Krishnamurthy, A. (2012). Open source software development process for the development of open source e-learning systems (Master's thesis, Dublin City University).

[17] Loría, A., & Panteley, E. (2006). Stability, as told by its developers. In A. Loría, F. Lamnabhi-Lagarrigue, & E. Panteley (Eds.), Advanced Topics in Control Systems Theory (pp. 1–97). Springer Verlag.

[18] Markovtsev, V. (2018). Public Git Archive: A big code dataset for all. arXiv preprint arXiv:1803.10144.

[19] Oliveira, P. C. d., Cunha, C. J. C. d. A., & Nakayama, M. K. (2016). Learning Management Systems (LMS) and e-learning management: An integrative review and research agenda. JISTEM Journal of Information Systems and Technology Management, 13(2), 157–180.

[20] Paradis, C. N., Yusuf, M. R., Farhanudin, M., & Yaqin, M. A. (2020). Analisis dan perancangan perangkat lunak pengukuran metrik skala dan kompleksitas diagram kelas. Jurnal Otomasi Sistem Informasi Komputer, 2(1), 85–90.

[21] Robles, G., & Gonzalez-Barahona, J. M. (2006). Contributor turnover in libre software projects. Proceedings of the International Conference on Open Source Systems (OSS 2006), 273–286.

[22] Rossi, B., Russo, B., & Succi, G. (2007). Analysis of open source software development iterations by means of burst detection techniques. Proceedings of the International Conference on Open Source Systems (OSS 2007), 107–118.

[23] Siva, F., Assegaf, S. M. U., Pahlevi, S. A., & Yaqin, M. A. (2023). Survei metode-metode software development life cycle dengan metode systematic literature review. ILKOMNIKA: Jurnal Ilmu Komputer dan Informatika Terapan, 5(2), 36–52.

[24] Zhu, L., Jeffery, R., Staples, M., Huo, M., & Tran, T. T. (2006). Effects of architecture and technical development process on micro-process. Proceedings of the International Conference on Software Process, 1

Submitted : 03-06-2026 | Reviewed : 08-06-2026 | Accepted : 30-08-2026

Submitted : 03-06-2026 | Reviewed : 08-06-2026 | Accepted : 30-08-2026