Peningkatan Kualitas *Efficiency* Dan *Mainttainability* Pada Sistem Informasi Web Sekolah Dengan Menggunakan Metode *Refactoring*

Muhammad Ihsan Maulana[1], Puspita Nurus Sabrina[2], Edvin Ramadhan[3]
Fakultas Sains dan Informatika/Program Studi Informatika
Universitas Jenderal Achmad Yani
Cimahi, Indonesia
e-mail: [1]sansgrids5601@gmail.com, [2]puspita.sabrina@lecture.unjani.ac.id,
[3]edvin.ramadhan@lecture.unjani.ac.id

Correspondence : *e-mail*: mihsanmaulana20@if.unjani.ac.id

Diajukan: 14 Agustus 2024; Direvisi: 23 Agustus 2024; Diterima: 24 Agustus 2024

Abstrak

Perkembangan teknologi sistem informasi telah mengalami kemajuan pesat dalam beberapa dekade terakhir. Sistem informasi web sekolah hadir sebagai sistem pengelolaan data sekolah, mulai dari data siswa, guru, hingga akademik. Namun, sistem ini memiliki kekurangan dalam hal efisiensi, dengan waktu respon yang lambat, serta maintainability yang rendah, membuat perbaikan dan pengembangan lebih sulit. Penelitian ini membahas perbaikan efisiensi dan maintainability pada sistem informasi web sekolah melalui teknik refactoring, khususnya dengan mendeteksi code smell dan menerapkan metode extract class serta extract method. Extract class diterapkan untuk memecah kelas yang terlalu kompleks, sesuai dengan prinsip Single Responsibility Principle (SRP), sedangkan extract method digunakan untuk membagi metode yang terlalu besar menjadi metode yang lebih spesifik. Evaluasi dilakukan terhadap keterbacaan, konsistensi, kualitas, dan struktur kode. Hasil refactoring menunjukkan peningkatan signifikan, termasuk waktu muat halaman yang lebih cepat hingga 77%, peningkatan kohesi sebesar 94%, penurunan kopling sebesar 15%, dan pengurangan jumlah bug sebesar 28%. Secara keseluruhan, sistem menjadi lebih responsif, mudah diskalakan, dan efisien dalam penggunaan sumber daya, yang meningkatkan pengalaman pengguna serta mengurangi biaya operasional. Penelitian ini menekankan pentingnya refactoring dalam menjaga keandalan dan kemudahan pengembangan sistem informasi web sekolah di tengah pertumbuhan jumlah pengguna.

Kata kunci: *code smell, extract class, extract method, McCall, refactoring.*

Abstract

The development of information system technology has progressed rapidly in the last few decades. School web information systems are present as school data management systems, ranging from student, teacher, to academic data. However, this system has shortcomings in terms of efficiency, with slow response time, as well as low maintainability, making improvement and development more difficult. This research discusses improving the efficiency and maintainability of the school web information system through refactoring techniques, specifically by detecting code smell and applying the extract class and extract method methods. Extract class is applied to break down classes that are too complex, in accordance with the Single Responsibility Principle (SRP), while extract method is used to divide methods that are too large into more specific methods. Evaluation was conducted on readability, consistency, quality, and code structure. The refactoring results showed significant improvements, including faster page load times by 77%, improved cohesion by 94%, decreased coupling by 15%, and a reduction in the number of bugs by 28%. Overall, the system became more responsive, scalable, and efficient in resource usage, which improved user experience and reduced operational costs. This research emphasizes the importance of refactoring in maintaining the reliability and ease of development of school web information systems amidst a growing number of users.

Keywords*: code smell, extract class, extract method, McCall, refactoring*

1. Pendahuluan

Perkembangan teknologi sistem informasi telah mengalami kemajuan pesat dalam beberapa dekade terakhir. Transformasi ini mencakup berbagai aspek, mulai dari perangkat keras dan perangkat lunak hingga infrastruktur jaringan dan metode pengelolaan data. Peningkatan kecepatan pemrosesan, kapasitas penyimpanan, dan efisiensi operasional telah menjadi pendorong utama dalam perkembangan ini.

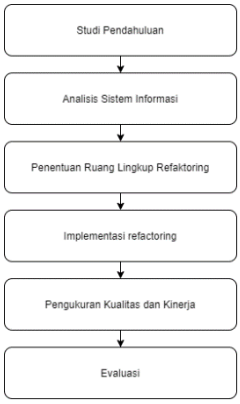
Tahapan untuk menguji dan menjamin kualitas sistem informasi web sekolah tersebut, dapat diukur dengan beberapa faktor kualitas seperti *correcnest, usability, reliability, integrity* dan *efficiency*. Faktor ini dikenal sebagai *Software Quality Factor* atau yang lebih dikenal dengan teori *McCall Factor* [1]. Pada penelitian terdahulu menjelaskan bahwa penentuan kualitas sendiri dapat dilakukan berdasarkan faktor yang mempengaruhi sistem, dimana pemilihan faktornya ini merujuk pada relevansi dengan dengan objek atau perangkat lunak yang digunakan [2]. Berdasarkan penjelasan penelitian sebelumnya, maka analisis terhadap sistem yang akan dilakukan *refactoring* yang pada kali ini faktor *efficiency* dan *maintainability* menjadi dua faktor yang paling relevan dengan sistem yang akan di*refactoring*. Efisiensi sendiri merujuk pada sejauh mana perangkat lunak dapat menggunakan sumber daya secara efisien termasuk kedalamnya penggunaan jaringan [3]. Sedangkan maintainability merujuk pada sejauh mana perangkat lunak dapat mudah dipelihara dan dikembangkan [4]. Untuk melihat kekurangan dari sebuah program metode *code smell* merupakan salah satu dari metode pendeteksi kesalahan *source code* yang nantinya dilakukan *refactoring*, dimana aspek aspek yang dilihat dalam *code smell* seperti keterbacaan kode, konsistensi kode, kualitas kode, struktur kode dan penggunaan bahasa pemrograman.

Refactoring sendiri merupakan proses mengubah struktur internal dari kode tanpa mengubah fungsionalitas eksternalnya [5]. Untuk melihat struktur internal kode metode *code smell* merupakan salah satu dari metode pendeteksi kesalahan *source code* yang nantinya dilakukan *refactoring*, dimana aspek aspek yang dilihat dalam *code smell* seperti keterbacaan kode, konsistensi kode, kualitas kode dan struktur kode [6]. Pada penelitian terdahulu menjelaskan bagaimana teknik *refactoring*, khususnya ekstraksi kelas (*extract class*), dapat digunakan untuk mengatasi permasalahan khususnya pada kelas besar yang memiliki banyak tanggung jawab dimana ini bertentangan dengan konsep *Single Responsibility Principle (SRP)* [7]. Selain *extract class*, *Extract method* juga merupakan sebuah metode *refactoring* yang dapat digunakan untuk menangani beberapa kekecatan kode yang memungkinkan pemecahan metode berdasarkan relevansi fungsional. Dengan dasar *Single Responsibility Principle (SRP)*, pendekatan ini mengidentifikasi set instruksi yang paling kohesif dan menyarankan ekstraksi mereka ke dalam metode terpisah [8].

Dengan demikian penelitian ini akan menggunakan extract class dan extract method sebagai metode *refactoring* untuk meningkatkan faktor *efficiency* dan *maintainability* pada sistem informasi web sekolah.

2. Metode Penelitian

Tahapan penelitian ini terdiri dari enam tahapan yaitu studi pendahuluan, analisa sistem informasi, penentuan ruang lingkup *refactoring*, implementasi *refactoring* pengukuran kualitas dan kinerja dan evaluasi.



Gambar 1. Metode penelitian

2.1. Studi Pendahuluan

Pada tahap awal penelitian, dilakukan identifikasi masalah dan kekurangan yang muncul dalam sistem informasi web sekolah. Temuan ini didokumentasikan secara rinci, memfokuskan perhatian pada kelemahan fungsionalitas dan tingkat kompleksitas kode dalam perangkat lunak yang menjadi pusat perhatian. Terutama yang mencakup dalam faktor *efficiency* dan *maintainability*.

2.2. Analisis Sistem Informasi

Fase analisis sistem informasi melibatkan dua hal, pertama pemeriksaan mendalam terhadap struktur dan logika *source code*. Identifikasi *source code* berdasarkan faktor SRP (*Single Responsibility Principle*), *code smells*, memungkinkan penentuan potensi perbaikan dan optimasi yang dibutuhkan untuk meningkatkan kualitas serta efisiensi sistem. Kedua menguji waktu muat halaman pada sistem informasi dengan menggunakan *tools* K6.io [9]. Analisis juga melibatkan perhitungan kualitas kode sebelum *refactoring* dengan menghitung nilai kohesi berdasarkan LCOM2, kopling berdasarkan nilai *Instability*, perhitungan kompleksitas kode berdasarkan nilai *Weighted Methods Per Class* (WMC) dan bug dengan menggunakan *tools* PhpMetrix.

2.2.1. Kohesi

Kohesi adalah ukuran sejauh mana metode dan atribut dalam sebuah kelas saling berinteraksi dan berhubungan [10]. *Metrix* yang digunakan untuk mengukur kohesi yaitu *Lack of Cohesion in Methods* (LCOM), dimana nilai kohesi diukur berdasarkan interaksi antara metode dan atribut dalam kelas dengan melihat total metode yang berbagi atribut yaitu nilai Q dengan total metode yang tidak berbagi atribut yaitu nilai P. yang dalam penelitian ini menggunakan LCOM2 dengan rumus

$$\text{LCOM2} = \begin{cases} P - Q, & if\ P \geq Q \\ 0, & Otherwisw \end{cases}$$

2.2.2. Kopling

Kopling dalam konteks perangkat lunak merujuk pada hubungan antara modul-modul dalam sistem, yang diukur melalui dua jenis pengukuran yaitu *Afferent coupling (Ca)* dan *Efferent Coupling (Ce)*. Kedua pengukuran ini digunakan untuk menghitung *instabilitas (I)* dari perangkat lunak. modul dengan kopling aferen tinggi dan kopling eferen rendah dianggap lebih stabil, sedangkan modul dengan kopling aferen rendah dan kopling eferen tinggi lebih rentan terhadap perubahan, yang dapat mempengaruhi keseluruhan sistem perangkat lunak, dimana jika nilai I lebih kecil atau menjadi 0 dari yang sebelumnya maka nilai kopling nya sendiri menjadi lebih baik [11].

$$\text{I} = (\text{Ce} / (\text{Ce}+\text{Ca}))$$

2.2.3. Class Complexity

Class complexity merujuk pada ukuran yang mengindikasikan seberapa kompleks suatu kelas dalam pemrograman berorientasi objek. Dimana untuk pengukuran nya sendiri menggunakan *Weighted Methods Per Class (WMC)* dengan menghitung jumlah keseluruhan dari kompleksitas setiap metode [12].

$$\text{WMC} = \sum ci$$

2.3. Penentuan Ruang Lingkup *Refactoring*

Berdasarkan hasil analisis, ditentukan ruang lingkup *refactoring* dengan memberikan prioritas pada area-area yang dianggap kritis, seperti kompleksitas kode tinggi, kelas yang tidak memenuhi standar SRP, dan keberadaan code smells yang merugikan.

2.4. Implementasi *Refactoring*

Langkah implementasi *refactoring* melibatkan penggunaan dua metode yaitu *Extract Class* dan *Extract Method* yang disesuaikan dalam sistem informasi web sekolah. Fokus utama adalah mengatasi kompleksitas kode dengan memperbaiki code smells yang berada dalam modul tersebut.

2.5. Pengukuran Kualitas dan Kinerja

Setelah implementasi *refactoring*, dilakukan pengujian untuk mengukur kualitas dan kinerja perangkat lunak. Evaluasi mencakup pengukuran waktu respons yang telah dioptimalkan, memberikan gambaran perbandingan dengan kondisi sebelumnya.

2.6. Evaluasi

Terakhir, hasil *refactoring* dievaluasi berdasarkan kriteria keberhasilan yang telah ditetapkan sebelumnya. Tinjauan mencakup peningkatan kualitas, kinerja, dan ketepatan sistem informasi web sekolah setelah penerapan *refactoring*. Hasil evaluasi memberikan dasar untuk mengidentifikasi pelajaran yang dapat dipetik dan memberikan rekomendasi untuk pengembangan selanjutnya.

3. Hasil dan Pengujian

Pada penelitian ini, *refactoring* dilakukan dengan menggunakan metode *Extract Class* dan *Extract Method* untuk meningkatkan efisiensi dan *maintainability* dari sistem informasi web sekolah. Hasil *refactoring* menunjukkan peningkatan signifikan dalam hal efisiensi dan *maintainability*. Dengan penerapan metode yang serupa, hasil ini berpotensi untuk diimplementasikan pada skala yang lebih besar, seperti pada sistem informasi di sekolah-sekolah lain atau bahkan di institusi yang lebih besar. Implementasi ini dapat dilakukan dengan menyesuaikan ruang lingkup *refactoring* berdasarkan kompleksitas dan kebutuhan spesifik dari setiap sistem, sehingga memungkinkan peningkatan kinerja dan kualitas secara keseluruhan di berbagai institusi.

3.1. Ruang Lingkup *Refactoring*

Ruang lingkup *refactoring* dalam penelitian ini mencakup penerapan metode *Extract Class* dan *Extract Method* pada modul-modul yang memiliki kompleksitas tinggi serta kelas-kelas dengan kohesi rendah dan kopling tinggi. Fokus *refactoring* ditujukan pada bagian-bagian kode yang kritis dalam hal maintainability dan kinerja sistem. Modul-modul yang dipilih untuk *refactoring* merupakan modul yang paling sering mengalami perubahan dan memiliki dampak terbesar terhadap stabilitas sistem secara keseluruhan.

Tabel 1. Ruang Lingkup *Refactoring*.

No	Class	Metode
1	ConfigHome	Extract Class
2	SiswaMgMt	Extract Class dan Extract Method
3	GuruMgMt	Extract Class dan Extract Method
4	WaliKelas	Extract Class dan Extract Method
5	Daftar	Extract Class dan Extract Method
6	AbsenStaff	Extract Method
7	Absen Guru	Extract Method
8	AgendaKelas	Extract Method
9	Agenda	Extract Method

3.2. Pengujian

Pengujian berisi perbandingan pada class atau pada page sebelum dan sesudah *refactoring* untuk melihat apakah metode yang digunakan dapat meningkatkan pada faktor *efficiency* dan *maintainability.*

3.2.1. Pengujian Kualitas Kode

Pengujian kualitas kode mengacu pada evaluasi berbagai aspek yang mempengaruhi kehandalan, efisiensi, dan keterbacaan kode. Dalam konteks ini, pengujian dengan melihat kompleksitas kelas, kohesi, kopling dan bug yang merupakan bagian penting dari analisis kualitas kode. Kohesi mengukur seberapa erat fungsi-fungsi dalam sebuah modul saling berkaitan, sementara kopling mengukur seberapa banyak sebuah modul bergantung pada modul-modul lain. Pengujian kohesi berfokus pada memastikan bahwa setiap modul atau kelas dalam kode memiliki tanggung jawab tunggal dan jelas, sehingga lebih mudah dipahami, diuji, dan dipelihara. Di sisi lain, pengujian kopling bertujuan untuk meminimalkan ketergantungan antara modul-modul yang berbeda, yang dapat menyebabkan masalah ketika satu modul diubah. Dimana pada sebelum *refactoring* terdapat kelas dengan tanda (-) dimana ini dimaksudkan bahwa sebelum melakukan *refactoring* kelas kelas tersebut belum ada dan setelah di *refactoring* dengan menggunakan extract class kelas tersebut mempunyai nilai yang harus diperhitungkan.

Tabel 2. Pengujian Kualitas Kode

No	Class	Sebelum				Sesudah			
		WMC	LCOM	I	Bug	WMC	LCOM	I	Bug
1	ConfigHome	17	104	1	0.62	2	0	1	0.06
2	SliderManager	-	-	-	-	6	5	0,85	0.16
3	JurusanManager	-	-	-	-	6	5	0,85	0.13
4	GalleryManager	-	-	-	-	6	5	0,85	0.17
5	SiswaMgmnt	11	1	1	0.55	14	0	1	0.48
6	GuruMgmnt	8	4	1	0.33	11	0	1	0.26
7	WaliKelas	8	4	1	0.42	13	0	1	0.42

8	UserCreationService	-	-	-	-	3	0	0,40	0.03
9	StudentCreationService	-	-	-	-	2	0	0,33	0.04
10	TeacherCreationService	-	-	-	-	2	0	0,50	0.04
11	Daftar	14	30	1	0.62	35	0	1	0.63
12	StudentClassService	-	-	-	-	2	0	0,50	0.01
13	StudentPaymentService	-	-	-	-	5	0	0,67	0.04
14	AbsenStaff	6	0	1	0	17	0	1	0.25
15	AbsenGuru	6	0	1	0	17	0	1	0.25
16	AgendaKelas	7	0	1	0.3	7	0	1	0.36
17	Agenda	9	1	1	0.33	17	0	1	0.32

Pengujian menunjukkan variasi dalam perubahan nilai MWC setelah *refactoring*. Penggunaan metode extract method menyebabkan beberapa nilai MWC meningkat, karena kode kompleks dipecah menjadi metode yang lebih kecil dan mudah dikelola, sehingga meningkatkan kualitas perangkat lunak. Sebaliknya, metode extract class tidak menunjukkan perubahan signifikan pada nilai MWC, menunjukkan bahwa pemisahan kode ke dalam kelas baru tidak berdampak besar pada metrik yang diuji.

Kelas seperti "Config Home" awalnya memiliki nilai LCOM tinggi (104), menunjukkan kohesi rendah karena banyak metode tidak berbagi atribut. Setelah *refactoring*, nilai LCOM turun signifikan menjadi 0, menunjukkan peningkatan fokus dan tanggung jawab yang lebih jelas dalam kelas. Penurunan LCOM sebesar 94% ini meningkatkan kemudahan pemahaman, pengujian, dan pemeliharaan kode.

Pengukuran kopling melalui metrik instability menunjukkan penurunan dari 1 menjadi 0,85 setelah *refactoring*, menandakan penurunan 15%. Ini menunjukkan bahwa ketergantungan antar modul berkurang, membuat sistem lebih stabil, mudah dipelihara, dan memudahkan pengembangan fitur baru tanpa mengganggu fungsionalitas yang ada.

Pengujian bug menggunakan PHPMetrix menunjukkan penurunan signifikan jumlah bug setelah *refactoring*. Misalnya, kelas "Config Home" mengalami penurunan nilai bug dari 0,62 menjadi 0,06. Kelas lain seperti "Slider Manager", "Jurusan Manager", dan "GalleryManager" yang sebelumnya tidak memiliki data bug juga menunjukkan penurunan nilai bug masing-masing menjadi 0,16, 0,13, dan 0,17. Penurunan bug sebesar 28% ini menunjukkan peningkatan stabilitas dan kehandalan sistem.

3.2.2. Pengujian Waktu Muat Halaman

Pengujian waktu muat halaman bertujuan untuk memastikan sistem informasi berjalan dengan cepat, sehingga meningkatkan pengalaman pengguna dan mengurangi risiko pengabaian halaman. Pengujian ini menggunakan *tools* K6.io dengan melihat data average sebelum dan sesudah di *refactoring*

Tabel 3. Pengujian waktu muat halaman.

No	Class	Sebelum	Sesudah	Peningkatan	Persentase
1	Config Home	964.85 ms	217.46 ms	-747.39	77.46%
2	Absen Tendik	999.65 ms	225 ms	-774.65	77.49%
3	Absen Guru	958.96 ms	230.57 ms	-728,39	75.96%
4	Pendaftaran Siswa	1.02 s	217.84 ms	-782.18	78.22%
5	Agenda Kelas	993.05 ms	216.42 ms	-776.63	78.21%
6	Wali Kelas	966.97 ms	217.91 ms	-749.06	77.46%
7	Management Siswa	916.35 ms	213.11 ms	-703.24	76.74%
8	Management Guru	899.53 ms	208.47 ms	-691.06	76.82%
9	Agenda	1.01 s	273.39 ms	-726.62	72.66%

Setelah implementasi *refactoring* pada sistem informasi, dilakukan pengujian waktu muat halaman untuk menilai perbaikan kinerja. Pengujian ini menggunakan alat bantu K6.io untuk mengukur waktu muat rata-rata sebelum dan sesudah *refactoring* pada berbagai halaman sistem dengan peningkatan sebesar 77%.

4. Kesimpulan

Hasil pengujian waktu muat halaman, kohesi, kopling, dan bug pada sistem informasi yang telah di *refactoring* menunjukkan peningkatan. *Refactoring* dengan metode extract class dan extract method telah membawa banyak manfaat bagi sistem informasi, termasuk peningkatan kohesi dan penurunan kopling antar kelas, yang berdampak positif pada pengurangan jumlah bug dan peningkatan kinerja sistem secara

keseluruhan. Peningkatan waktu muat halaman sebesar 77%, peningkatan kohesi sebesar 94%, penurunan kopling sebesar 15%, dan pengurangan bug sebesar 28% menunjukkan bahwa *refactoring* ini sangat efektif dalam meningkatkan performa dan kualitas sistem informasi. Peningkatan ini juga secara langsung berkontribusi pada efisiensi dan maintainability sistem, karena dengan struktur kode yang lebih rapi dan modular, pengembang dapat lebih mudah memahami, memelihara, dan mengembangkan sistem di masa depan, yang pada gilirannya menghemat waktu dan sumber daya. Dengan peningkatan nilai maintainability, efisiensi juga meningkat, yang terlihat dari waktu muat halaman yang lebih cepat, sehingga memberikan pengalaman pengguna yang lebih baik.

Pengembangan lebih lanjut, disarankan untuk mengeksplorasi penerapan metode lain yang dapat melengkapi *refactoring*, seperti teknik Continuous *Refactoring* atau penerapan metode *Refactoring* berbasis AI untuk peningkatan maintainability. Selain itu, penelitian lebih lanjut diperlukan untuk menguji efektivitas metode ini dalam meningkatkan maintainability pada sistem informasi yang lebih kompleks, dengan mempertimbangkan berbagai faktor seperti skala sistem, jumlah pengguna, dan integrasi dengan sistem yang lain.

Daftar Pustaka

[1] M. Purnasari and Z. Karman, "Pengukuran Kualitas Perangkat Lunak PosPay 5000 Menggunakan Metode McCall," vol. 4, no. 2, pp. 232–243, 2024.

[2] S. H. Sadiyah and A. Qoiriah, "Pengaruh *Refactoring* Terhadap Tingkat Pemeliharaan Perangkat Lunak Pada Aplikasi Permainan 'Infection Defender,'" *Journal of Informatics and Computer Science*, vol. 05, 2024.

[3] Salamudin, sri hartiati, haris saputro, and dian meilantika, "Effisiensi," *JURNAL NASIONAL ILMU KOMPUTER*, vol. 5, no. 1, pp. 16–24, 2024.

[4] Y. Fitrisia Politeknik Caltex Riau, "Pengembangan Object Oriented Software Metric *Tools* Untuk Evaluasi Maintainability," 2015. [Online]. Available: https://www.researchgate.net/publication/324909118

[5] R. Mufrizal and D. Indarti, "*Refactoring* Arsitektur Microservice Pada Aplikasi Absensi PT. Graha Usaha Teknik," *Jurnal Nasional Teknologi dan Sistem Informasi*, vol. 5, no. 1, pp. 57–68, Apr. 2019, doi: 10.25077/teknosi.v5i1.2019.57-68.

[6] A. Hutami Endang, D. Aprilia Khairunnisa, and A. Jamiati Paramita, "Pendeteksian Code Smell pada Website Perusahaan," vol. 7, no. 1, 2022, doi: 10.36418/Syntax.

[7] M. Alzahrani, "Extract Class *Refactoring* Based on Cohesion and Coupling: A Greedy Approach," *Computers*, vol. 11, no. 8, Aug. 2022, doi: 10.3390/computers11080123.

[8] S. Charalampidou, A. Ampatzoglou, A. Chatzigeorgiou, A. Gkortzis, and P. Avgeriou, "Identifying Extract Method *Refactoring* Opportunities Based on Functional Relevance," *IEEE Transactions on Software Engineering*, vol. 43, no. 10, pp. 954–974, Oct. 2017, doi: 10.1109/TSE.2016.2645572.

[9] A. Suprapto and D. Sasongko, "EVALUASI PERFORMA WEBSITE BERDASARKAN PENGUJIAN BEBAN DAN STRESS MENGGUNAKAN LOADIMPACT (STUDI KASUS WEBSITE IAIN SALATIGA)," *Jurnal Ilmiah NERO*, vol. 6, no. 1, pp. 31–37, 2021, [Online]. Available: https://iainsalatiga.ac.id

[10] E. N. H. Kirgil and T. E. Ayyildiz, "Analysis of Lack of Cohesion in Methods (LCOM): A Case Study," in *2nd International Informatics and Software Engineering Conference, IISEC 2021*, Institute of Electrical and Electronics Engineers Inc., 2021. doi: 10.1109/IISEC54230.2021.9672419.

[11] D. B. Santos, A. M. P. Resende, E. C. Lima, and A. P. Freire, "Software Instability Analysis Based on Afferent and Efferent Coupling Measures," *Journal of Software*, vol. 12, no. 1, pp. 19–34, Jan. 2017, doi: 10.17706/jsw.12.1.19-34.

[12] N. Qamar and A. A. Malik, "Impact of Design Patterns on Software Complexity and Size," *Mehran University Research Journal of Engineering and Technology*, vol. 39, no. 2, pp. 342–352, Apr. 2020, doi: 10.22581/muet1982.2002.10.