Refaktoring Untuk Meningkatkan Kualitas Reliabilitas Dan *Correctness* Pada Sistem Manajemen Sekolah Berbasis Web

Ryad Helmi Fadila[1], Puspita Nurul Sabrina[2], Herdi Ashaury[3]
Program Studi Informatika, Fakultas Sains & Informatika,
Universitas Jenderal Achmad Yani,
Cimahi, Indonesia
e-mail: [1]Ryadhelmi@gmail.com, [2]puspita.sabrina@lecture.unjani.ac.id,
[3]herdi.ashaury@lecture.unjani.ac.id

Correspondence : *e-mail*: Ryadhelmi@gmail.com

Diajukan: 20 Agustus 2024; Direvisi: 23 Agustus 2024; Diterima: 24 Agustus 2024

Abstrak

Pada Perkembangan teknologi sistem informasi terutama web, telah menghasilkan sebuah sistem yang dapat memudahkan pengelolaan sekolah. sistem informasi tersebut berisikan tentang apapun yang berkaitan dengan pengolahan data sekolah, yang mana sebelumnya dilakukan secara fisik sekarang dapat dilaksanakan secara digital, seperti mengelola data siswa, data guru dan sebagainya. sehingga jumlah data yang dapat diakses sistem sangat banyak, maka penting untuk sistem informasi tersebut reliabel dan fungsi–fungsinya telah sesuai dan minim kesalahan, dengan menggunakan metrik bugs dapat membantu mencari kelas yang terdapat banyak kesalahan dan penggunaan metrik seperti defect rate dan kesesuaian input output untuk menilai kehandalan dengan menghitung rata - rata kegagalan yang terjadi, perhitungan tersebut juga dapat membantu menilai perubahan agar mengetahui proses yang dilakukan dapat memperburuk atau meningkatkan kualitas sistem tersebut. Dari kesalahan–kesalahan yang ditemukan dilakukanlah refaktoring untuk memperbaiki dan meningkatkan kualitasnya tersebut, setelah pelaksanaan refaktoring dapat dilakukan perhitungan terhadap perubahan yang telah terjadi, adapun peningkatan pada metrik defect rate sebesar 87%, adapun metrik consistency dan kesesuaian input output seperti Rate Occurance Of Failure sebesar 81% dan metrik Mean Time Between Failure sebesar 88.9%, berdasarkan peningkatan tersebut dengan melakukan refaktoring dengan tahapan tersebut terbukti dapat membantu meningkatkan aspek reliabilitas maupun correctness pada sistem informasi yang digunakan.

Kata Kunci: *McCall's quality model, reliabilitas, correctness, code smells, refactoring.*

Abstract

In the development of information system technology, especially the web, has produced a system that can facilitate school management. the information system contains anything related to data processing in schools, which was previously done physically can now be carried out digitally, such as managing student data, teacher data and so on. so the amount of data, or users that can be accessed or anyone accessing the system must be very large, it is very important for the information system to be reliable and its functions are appropriate and minimal errors. so the amount of data, or users that can be accessed or anyone who accesses the system must be very large, so it is very important for the information system to be reliable and its functions are appropriate and minimal errors, using the bugs metric can help find classes that have many errors and the use of metrics such as defect rate and input output conformity to assess reliability by calculating the errors that occur to help calculate the assessment of changes in order to find out the processes carried out can worsen or improve the quality of the system. then from the errors found it can help find the class that has many errors. then from the errors found, refactoring is carried out to repair and improve its quality, after the implementation of refactoring, calculations can be made of the changes that have occurred, as for the increase in the defect rate metric of 87% as well as the consistency metric and the suitability of input and output such as the Occurance Rate Of Failure of 81% and the Mean Time Between Failure metric of 88.9%, based on this increase by refactoring with these stages it is proven that it can help improve the reliability and correctness aspects of the information system used.

Keywords*: McCall's quality model, reliability, correctness, code smells, refactoring.*

1. Pendahuluan

Dewasa ini aplikasi berbasis web memiliki peran penting dalam kehidupan sehari hari, mudahnya pengaksesan serta rendahnya *resource* yang dibutuhkan menjadikan aplikasi web ini dapat memudahkan pengguna untuk mendapatkan informasi. salah satu jenis aplikasi yang dapat membantu untuk mendapatkan informasi adalah sistem informasi sekolah. Sistem Informasi Sekolah dapat membantu dalam pengelolaan data siswa, data pegawai, jadwal pelajaran, profil sekolah, fasilitas sekolah, dan informasi-informasi penting lainnya yang berkaitan dengan kegiatan sehari-hari di sekolah. Sistem Informasi Sekolah juga dapat membantu dalam pengambilan keputusan, pemantauan kinerja, dan peningkatan efisiensi dalam pengelolaan sekolah secara keseluruhan[1].

Terdapat masalah yang mungkin dapat terjadi pada saat penggunaan sistem tersebut, seperti pada penelitian [2] terdapat Masalah – Masalah seperti keamanan data Setelah penerapan, risiko keamanan seperti peretasan, pencurian data, atau kebocoran informasi pribadi siswa dapat meningkat, terutama jika sistem tidak dilindungi dengan baik. Selain itu ada Kendala Teknis, Pengguna mungkin mengalami masalah teknis seperti kesalahan sistem, *bug*, atau *downtime* yang dapat mengganggu akses ke sistem dan menghambat proses akademik [2]. Oleh karena itu agar perangkat lunak dapat berjalan dengan baik, minim terjadinya error, serta minim terjadinya masalah keamanan yang dapat menggangu sistem, digunakanlah pendekatan model faktor kualitas yaitu *Mc'Calls Quality Factor*. karakteristik yang digunakan dalam penelitian ini adalah *Reliability* sebagai fokus utama perubahan dan *correctness* untuk mengecek kesesuaian hasil dari perubahan kode. Pada model kualitas *McCall's* Reliabilitas adalah konsep krusial yang merujuk pada tingkat kepercayaan yang dapat diberikan kepada suatu sistem, proses, atau produk untuk berfungsi dengan benar dan konsisten sebagaimana yang diharapkan selama periode waktu tertentu [3]. Sedangkan *Correctness* adalah salah satu faktor kualitas utama dalam model *McCall*'s yang menunjukkan keandalan perangkat lunak dari perspektif fungsionalitasnya [4].

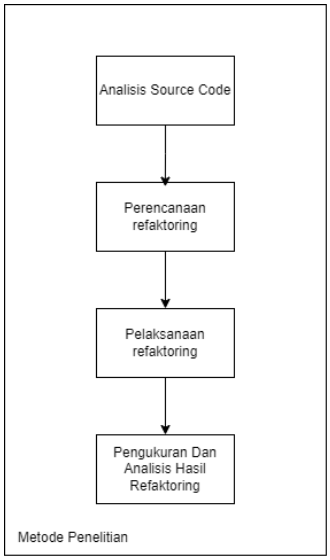
Pada penelitian ini dilakukan pendeteksian kode yang bermasalah dengan kriteria yang berpengaruh terhadap masalah reliabilitas dan *correctness* itu sendiri [5]. Adapun kriteria yang digunakan untuk mengukur hasil dari refaktoring yang dilakukan adalah *error tolerance*, dan *consistency* [6], untuk metrik yang digunakan adalah *defect rate* dan kesesuaian *input output*. Pada penelitian ini juga dilakukan Refaktoring. Refaktoring merupakan proses mengubah sistem perangkat lunak sehingga perubahan tersebut meningkatkan kualitas perangkat lunak tanpa mengubah perilaku perangkat lunak [7]. Untuk metrik metrik tersebut dapat dihitung dengan menggunakan rumus *Rate Occurrence Of Failure* (ROCOF) [8] dan *Mean Time Between Failure* (MTBF)[9].

Pada contoh Penelitian [10] menemukan bahwa mekanisme untuk menerapkan metode refaktoring memainkan peran penting dalam menentukan efeknya terhadap karakteristik kualitas perangkat lunak. Berbagai mekanisme ditemukan untuk setiap teknik refaktoring, tergantung pada desain internal sistem perangkat lunak, yang mengakibatkan variasi dalam dampak metode refaktoring terhadap karakteristik kualitas. Studi ini menyoroti bahwa pengaruh dari setiap metode refaktoring melalui mekanisme yang berbeda dapat bervariasi, menekankan pentingnya memahami mekanisme spesifik dan efeknya saat menerapkan metode refaktoring untuk meningkatkan kualitas perangkat lunak. Dengan mengidentifikasi mekanisme dan dampaknya, praktisi perangkat lunak dapat membuat pilihan yang terinformasi dalam memilih metode refaktoring yang sesuai untuk mengatasi cacat desain dan meningkatkan karakteristik kualitas secara efektif.

Berdasarkan Latar belakang yang diuraikan penting untuk menentukan alur untuk pelaksanaan refactoring agar dapat meningkatkan kualitas secara efektif, seperti menentukan faktor kualitas yang akan ditingkatkan sehingga dapat mengetahui bagian apa saja yang harus dilakukan refactor tersebut, setelah itu melakukan pencarian masalah pada kode per kelas terlebih dahulu, setelah itu pada kelas tersebut dilakukan pencarian *code smells* yang berkaitan dengan masalah faktor kualitas, dengan hal tersebut dapat diketahui masalah apa saja yang dapat menyebabkan error, dilakukanlah perhitungan ROCOF untuk menghitung peningkatan faktor kualitas *correctness* dengan melakukan perhitungan perkiraaan rata-rata kesalahan berdasarkan fungsi dan jumlah masalah yang ada pada setiap kelas tersebut, untuk *consistency* faktor kualitas reliabilitas dilakukanlah perhitungan rata-rata kesalahan berdasarkan jumlah operasi yang dilakukan dengan menggunakan perhitungan *Mean Time Between Failure* (MTBF) yang dilakukan dengan melakukan test otomatis menggunakan selenium, setelah melakukan perhitungan dilakukanlah refactoring pada masalah masalah tersebut, dan terakhir adalah memastikan kode yang telah di refactoring tidak menambah *bug* dilakukan pengujian *black box* testing secara manual, apabila sudah dilakukan pengujian dilakukanlah perbandingan perhitungan *Bugs* (*halstead*), ROCOF dan MTBF tersebut agar terlihat perkembangan ataupun penurunan kualitas yang terjadi setelah pelaksanaan refactoring tersebut.

2. Metode Penelitian

Adapun alur proses penelitian yang dilakukan dapat dilihat pada Gambar 1.



Gambar 1 – Alur Penelitian

2.1. Analisis *Source Code*

Analisis *Source code* dilakukan untuk memilih bagian - bagian kode yang akan diperbaiki. Ini bisa berupa kelas, metode, atau bagian kode lain yang dianggap perlu diperbaiki. Pemilihan target ini biasanya didasarkan pada beberapa faktor seperti kompleksitas kode, frekuensi perubahan, dan tingkat kesulitan dalam pemahaman dan pemeliharaan. Setelah target dipilih, selanjutnya melakukan pengukuran terhadap kelas kelas tersebut. Hal Ini melibatkan berapa rata rata kesalahan yang mungkin terjadi pada sistem dan mengidentifikasi area yang memiliki potensi untuk perbaikan. Adapun tahapan yang dilakukan pada analisis *source code* yaitu dengan melakukan identifikasi menggunakan metrik *bugs* (*halstead*) dengan menggunakan PHPmetrics, pada php metrics terdapat beberapa metrik tentang pengukuran pada kode seperti metrik kompleksitas & *bugs, coupling, & size pada* kelas, serta metrik *objek oriented*, Pada penelitian ini dipilih pengukuran metrik komplekistas & *bugs* yang digunakan untuk pemilihan kelas dengan *bugs* diatas 0.5 (lebih besar diantara kelas lain pada sistem yang digunakan) yang mana menandakan bahwa kelas lebih kompleks dan terdapat *bugs* yang lebih banyak dibandikan dengan kelas lain, digunakannya metrik ini pula dikarenakan keterkaitan faktor kualitas dengan metrik tersebut dikarenakan dapat mengurangi kompleksitas yang mungkin dapat menyebabkan sulitnya perbaikan system serta dapat mengurangi *bugs* yang dapat menyebabkan *error*, yang dilanjutkan dengan melakukan pencarian *Code Smells* yang berkaitan dengan masalah reliabilitas maupun *correctness* dengan tahap terakhir adalah melakukan perhitungan terhadap ROCOF dan MTBF sebelum dilakukannya refactoring untuk nanti dilakukan perbandingan dengan perhitungannya setelah refactoring. Untuk perhitungan ROCOF dapat dihitung dengan menggunakan rumus sebagai berikut

$$\text{ROCOF} = \frac{\text{kegagalan}}{\text{Unajusted Function Point}} \quad (1)$$

Pada perhitungan ROCOF diperlukan terlebih dahulu melakukan pengamatan pada setiap fungsi agar dapat menghitung *Unajusted Function Point* (UFP). Setelah mendapatkan nilai pada perhitungan ROCOF maka dapat menghitung relibilitas (rata – rata kesalahan) per fungsi setiap kelas yang dipilih dengan menggunakan rumus sebagai berikut

$$\text{Reliabilitas} = 1 - \text{ROCOF} \quad (2)$$

MTBF digunakan untuk mengukur kesalahan pada kelas untuk periode waktu tertentu, dalam penelitian ini dilakukan perhitungan per 1000 kali penggunaan dengan menggunakan skrip pengujian selenium, dengan menggunakan rumus sebagai berikut

$$\text{MTBF} = \frac{\text{total operasi}}{\text{jumlah kegagalan}} \quad (3)$$

2.2. Perencanaan *Refaktoring*

Perencanaan *Refaktoring* dilakukan untuk melakukan pemilihan metode ataupun cara yang akan dilakukan agar dapat menyelesaikan masalah yang terdapat pada kode sehingga proses *refaktoring* tersebut dapat berjalan dengan efisien dan tidak mengenalkan *bug* baru yang mungkin terjadi setelah refaktoring.

2.3. Pelaksanaan Refaktoring

Pelaksanaan refaktoring dilakukan dengan mengimplementasikan perubahan sesuai dengan rencana yang telah disusun. Ini bisa melibatkan memisahkan fungsi yang terlalu besar, menggabungkan kelas yang terlalu banyak, atau melakukan perubahan struktural lainnya. Selama tahap ini, penting untuk tetap konsisten dengan prinsip-prinsip desain yang baik.

2.4. Pengukuran Dan Analisis Hasil Refaktoring

Pengukuran dilalukan dengan tujuan untuk melakukan perbandingan sebelum dan sesudah dilakukannya refaktoring pada perhitungan ROCOF dan MTBF sehingga dapat membantu memudahkan analisis perkembangan dan pengurangan kualitas yang terjadi pada kelas yang telah dilakukan refaktoring tersebut.

3. Hasil dan Pembahasan

3.1. Analisis Source Code

Dari pengamatan terhadap metrik *bugs* yang didapat terdapat 4 kelas yang memenuhi kriteria diatas 0.5 yaitu kelas kelas yang dapat dilihat pada Tabel 1.

Tabel 1 Bugs PHPmetrics

Class	Bugs	Defects
\Ppdb\Daftar	0.62	0.22
\Admin\ConfigHome	0.61	0.15
\Guru\Walikelas	0.58	0.43
\Kurikulum\SiswaMgmt	0.55	0.22

Setelah melakukan pencarian bugs yang paling besar, dilakukanlah pencarian terhadap *code smells* yang terkandung pada kelas kelas tersebut yang berkaitan dengan masalah faktor kualitas *reliability* dan *correctness* seperti *Duplicated Code*, *Improper Error Handling*, SQL Injection Potential, God Class, *Missing Validation* Dan *Broken Authentication*. Dari kesalahan kesalahan tersebut yang dapat menybabkan error secara langsung adalah *missing validation*, dan *improper error handling* sehingga untuk menghitung UFP pada setiap kelasnya dapat dengan kesalahan yang menyebabkan *error*.

3.2. Pelaksanaan Refaktoring

Dari hasil analisis tersebut dapat dilakukan refaktoring pada kelas kelas tersebut dan menentukan teknik atau metode yang digunakan untuk menanggulangi kesalahan seperti yang terdapat pada Tabel 2.

Tabel 2 *Code Smells dan* penyelesaiannya

Code Smell	Contoh Lokasi dalam Kode	Metode Penyelesianan
Duplicate Code	metode *CRUD* pada walikelas.	*Extract Method*
Improper Error Handling	Metode Render dan *CRUD* pada confighome	*Introduce Try - Catch block*
SQL Injection Potential	Fungsi *search* pada render walikelas	*Introduce Prepared Statement*
God Class	Keseluruhan ConfigHome	*Extract Class*
Missing Validation	Kurang jelasnya validasi pada *input create* di Daftar	Melengkapi Validasi yang kurang pada metode
Broken Authentication	Metode *CRUD* SiswaMgmt	Menambahkan Autentikasi pada metode sensitif

3.3. Pengukuran Dan Analisis Hasil Refaktoring

Pada pengukuran ini diawali dengan melakukan pengukuran terhadap kriteria *error tolerance* dengan menggunakan *defect rate* setelah dilakukannya refactoring yang dapat dilihat pada Tabel 3.

Tabel 3 *Defect Rate*

		Sebelum			**Sesudah**		
NO	Nama Kelas	Line Code	Jumlah Cacat	Defect rate	Line Code	Jumlah Cacat	Defect rate
1.	Daftar	197	6	0.0304	254	1	0.003
2.	Confighome	249	10	0.028	44	0	0
2.1	Slidermanager	-	-	-	125	1	0.008
2.2	Galerymanager	-	-	-	122	1	0.008
2.3	Jurusanmanager	-	-	-	113	1	0.008
3.	Walikelas	172	8	0.046	247	0	0
4.	Siswamgmt	188	6	0.031	237	1	0.004

Pada pengukuran tersebut terdapat peningkatan sebesar 87% untuk *defect rate* dimana terdapat pengurangan kecacatan yang terdapat pada setiap kelas. Setelah mengukur *error tolerance* selanjutnya adalah mengukur perbandingan ROCOF dan MTBF untuk mengetahui perkembangan kriteria *consistency* yang terdapat pada kelas, untuk menilai metrik kesesuaian *input output* yang dapat dilihat pada Tabel 4, Tabel 5 dan Tabel 6.

Tabel 4 perbandingan ROCOF

NO	**NAMA KELAS**	**ROCOF sebelum**	**Reliabilitas sebelum**	**ROCOF sesudah**	**Reliabilitas sesudah**
1	Daftar	0.113	887x	0.019	981x
2	Confighome	0.093	903x	0.028	983x
3	Walikelas	0.103	897x	0	1000x
4	Siswamgmt	0.09	910x	0	1000x

Pada Tabel 4 dapat dilihat bahwa terdapat perkembangan pada perhitungan ROCOF itu sendiri. Namun pada kelas daftar dan confighome masih terdapat kesalahan pada *error handling* yang dapat menyebabkan kegagalan sistem setelah dilakukan pengujian dengan menggunakan *blackbox testing* sehingga rata – rata peningkatan ROCOF setelah dilakukan refaktoring adalah sebesar 88.2%.

Tabel 5 MTBF sebelum

NO	**NAMA KELAS**	**ITERASI SUKSES**	**ITERASI GAGAL**	**MTBF**
1	Daftar.php	716	284	3.52
2	Confighome.php	903	97	3.26
3	Walikelas.php	548	452	2.21
4	Siswamgmt.php	820	180	5.55

Tabel 6 MTBF sesudah

NO	NAMA KELAS	ITERASI SUKSES	ITERASI GAGAL	MTBF
1	Daftar.php	994	6	166.6
2	Confighome.php	960	40	25
3	Walikelas.php	1000	0	1000
4	Siswamgmt.php	1000	0	1000

Pada Tabel 5 dan 6 dapat dilihat bahwa setelah pengujian terdapat pengurangan kegagalan yang terjadi pada setiap kelas untuk rata - rata perkembangan yang terjadi setelah dilakukannya refaktoring adalah sebesar 88.9%.

4. Kesimpulan dan saran

Berdasarkan penelitian, refaktoring yang dilakukan berhasil meningkatkan reliabilitas dan keakuratan sistem dengan peningkatan rata-rata sebesar 88%. Peningkatan ini terlihat pada kriteria *error tolerance* (87%), *consistency* untuk faktor kualitas *correctness* (88.2% berdasarkan ROCOF), dan *consistency* untuk faktor kualitas reliabilitas(88.9% berdasarkan MTBF). Hasil ini menunjukkan bahwa refaktoring manual dengan pengujian terbatas dapat memberikan hasil yang memuaskan. Untuk penelitian selanjutnya, disarankan untuk fokus pada faktor kualitas lain, seperti maintainability, agar metrik lain seperti *object-oriented*, *size*, dan *complexity* dari tools seperti phpmetric dapat digunakan secara menyeluruh. Dengan digunakannya keseluruhan metrik tersebut dapat ditambahkan juga metrik yang berkaitan dengan masalah reliabilitas lainya seperti *Mean Time Between Repair* (MTBR) yang dapat membantu mengukur kecepatan perbaikan sistem. Penggunaan *Function Point* juga direkomendasikan untuk mendapatkan gambaran yang lebih lengkap tentang ukuran dan kompleksitas sistem dibandingkan menggunakan UFP yang menghitung kemungkinan kasar terjadinya kegagalan. Disarankan pula untuk menggunakan tools yang dapat membantu dalam pencarian kesalahan dan pengujian agar dapat mendeteksi kesalahan yang lebih mendalam seperti kesalahan pada saat proses penggunaan dibandingkan dengan melihat kesalahan pada kode sumber.

Daftar Pustaka

[1] S. Laugi, “Use of Websites in School Management…. Use of Websites in School Management: An Effort to Build School Readiness in the Era of 4.0,” 2020.

[2] R. F. Syafariani and A. Devi, “Web-Based Academic Information System,” in IOP Conference Series: Materials Science and Engineering, Institute of Physics Publishing, Nov. 2019. doi: 10.1088/1757-899X/662/2/022042.

[3] N. Bitew and J. Singh, “A Study on Software Quality Factors and Metrics to Enhance Software Quality Assurance,” International Journal of Productivity and Quality Management, vol. 1, no. 1, p. 1, 2019, doi: 10.1504/ijpqm.2019.10025496.

[4] E. A. AlOmar, M. W. Mkaouer, and A. Ouni, “Behind the Intent of Extract Method Refactoring: A Systematic Literature Review,” Dec. 2023, doi: 10.1109/TSE.2023.3345800.

[5] R. E. Al-Qutaish and A. Ain, “Quality Models in Software Engineering Literature: An Analytical and Comparative Study,” 2010.

[6] F. P. Juniawan et al., “E-Voting Software Quality Analysis with McCall’s Method,” in 2020 8th International Conference on Cyber and IT Service Management, CITSM 2020, Institute of Electrical and Electronics Engineers Inc., Oct. 2020. doi: 10.1109/CITSM50537.2020.9268854.

[7] E. Fernandes et al., “Refactoring effect on internal quality attributes: What haven’t they told you yet?,” Inf Softw Technol, vol. 126, Oct. 2020, doi: 10.1016/j.infsof.2020.106347.

[8] Ritzkal, G. Arief, and H. P. Eko, “PENGUKURAN KUALITAS PERANGKAT LUNAK SISTEM E-LEARNING MENGGUNAKAN METRIC FUNCTION ORIENTED,” Prosiding SNATIF Ke -4, 2017.

[9] G. Kaur and K. Bahl, “Software Reliability, Metrics, Reliability Improvement Using Agile Process,” 2014. [Online]. Available: www.ijiset.com

[10] A. Almogahed, H. Mahdin, M. Omar, N. H. Zakaria, G. Muhammad, and Z. Ali, “Optimized Refactoring Mechanisms to Improve Quality Characteristics in Object-Oriented Systems,” IEEE Access, vol. 11, pp. 99143–99158, 2023, doi: 10.1109/ACCESS.2023.3313186.