Peningkatan Kualitas Maintainability Sistem PPID DINKES Cimahi Menggunakan Metode Refactoring

Rahmadi Dimas Wirianto[1], Puspita Nurul Sabrina[2], Herdi Ashaury[3]
[123]Fakultas Sains dan Informatika, Universitas Jenderal Achmad Yani, Cimahi, Indonesia
Email : rahmadidimasw20@if.unjani.ac.id, puspita.sabrina@lecture.unjani.ac.id,
herdi.ashaury@lecture.unjani.ac.id

Abstrak-Penelitian ini bertujuan untuk melakukan pengujian serta meningkatkan kualitas Maintainability pada perangkat lunak dengan menerapkan metode Refactoring. Pengujian kualitas dilakukan untuk menilai keakuratan, kelengkapan, dan kualitas perangkat lunak. Selanjutnya, melakukan analisis struktur kode untuk mengidentifikasi adanya masalah terhadap Object Oriented Metrics dan Code Smell untuk meningkatkan kualitas Maintainability pada sistem PPID Dinkes Cimahi. Hasil pengujian kualitas perangkat lunak akan menunjukkan hasil perhitungan Object Oriented Metrics. Menganalisis struktur kode mengidentifikasi kerentanan dalam bentuk pengujian Object Oriented Metrics, juga indikasi code smell. Selanjutnya dilakukan refactoring untuk memperbaiki struktur internal perangkat lunak, dengan tujuan untuk meningkatkan faktor kualitas pada McCalls quality. Penelitian ini memberikan kesimpulan berdasarkan hasil pengujian dan perbaikan kualitas perangkat lunak. Diharapkan bahwa hasil penelitian ini dapat memberikan pemahaman yang mendalam mengenai kondisi perangkat lunak dan memberikan solusi untuk meningkatkan kualitas perangkat lunak melalui refactoring. Hasil dari refactoring menunjukkan peningkatan yang signifikan dalam nilai Object Oriented Metrics. Sebagai contoh, nilai WMC pada kelas 'Admin_model' menurun dari 31 menjadi 15, nilai CBO menurun dari 15 menjadi 4, dan nilai RFC menurun dari 40 menjadi 23. Penurunan nilai-nilai ini menunjukkan bahwa kelas tersebut menjadi lebih sederhana, memiliki lebih sedikit ketergantungan dengan kelas lain, dan lebih mudah untuk diuji dan dipelihara. Selain itu, pada kelas 'Profile', nilai LCOM yang sebelumnya tinggi, yaitu 5, berhasil diturunkan menjadi 0 setelah refactoring, yang menunjukkan peningkatan kohesi antar metode dalam kelas tersebut. Kelas 'Banner' dan 'Auth' juga menunjukkan perbaikan, di mana nilai CBO dan LCOM berhasil diturunkan sehingga berada di bawah atau mendekati nilai threshold yang diinginkan. Misalnya, nilai CBO pada kelas 'Banner' menurun dari 9 menjadi 8, sedangkan pada kelas 'Auth', nilai CBO menurun dari 9 menjadi 5 dan nilai LCOM menurun dari 4 menjadi 2. Dengan demikian, dapat disimpulkan bahwa penerapan metode refactoring dalam penelitian ini berhasil mengoptimalkan nilai-nilai Object Oriented Metrics dan mengurangi code smell, yang pada akhirnya meningkatkan kualitas maintainability perangkat lunak sistem PPID Dinkes Cimahi secara keseluruhan. Hasil penelitian ini diharapkan dapat memberikan pemahaman yang lebih mendalam tentang kondisi perangkat lunak dan memberikan solusi yang efektif untuk meningkatkan kualitas perangkat lunak melalui metode refactoring, serta mempermudah keterbacaan dan pemeliharaan kode sumbernya.

Kata Kunci: *Maintainability, Object Oriented Metrics*, Refactoring, Kualitas, Sistem

Abstract-*This research aims to carry out testing and improve the maintainability quality of software by applying the Refactoring method. Quality testing is carried out to assess the accuracy, completeness and quality of the software. Next, carry out code structure analysis to identify problems with Object Oriented Metrics and Code Smell to improve the quality of Maintainability in the Cimahi Health Office PPID system. The results of software quality testing will show the results of Object Oriented Metrics calculations. Analyzing code structures identifies vulnerabilities in the form of Object Oriented Metrics testing, as well as code smell indications. Next, refactoring is carried out to improve the internal structure of the software, with the aim of increasing the quality factor in McCalls quality. This research provides conclusions based on the results of testing and improving software quality. It is hoped that the results of this research can provide a deep understanding of the condition of the software and provide solutions to improve software quality through refactoring. The results of refactoring show a significant increase in the value of Object Oriented Metrics. For example, the WMC value of the 'Admin_model' class decreased from 31 to 15, the CBO value decreased from 15 to 4, and the RFC value decreased from 40 to 23. The decrease in these values indicates that the class has become simpler, has fewer dependencies with other classes, and is easier to test and maintain. In addition, in the 'Profile' class, the previously high LCOM value of 5 was successfully reduced to 0 after refactoring, indicating increased cohesion between methods in the class. The 'Banner' and 'Auth' classes also showed improvements, where the CBO and LCOM values were successfully reduced to below or close to the desired threshold value. For example, the CBO value in the 'Banner' class decreased from 9 to 8, while in the 'Auth' class, the CBO value decreased from 9 to 5 and the LCOM value decreased from 4 to 2. Thus, it can be concluded that the application of the refactoring method in this study successfully optimized the values of Object Oriented Metrics and reduced code smell, which ultimately improved the maintainability quality of the Cimahi Health Office PPID system software as a whole. The results of this study are expected to provide a deeper understanding of the condition of the software and provide effective solutions to improve software quality through the refactoring method, as well as facilitate the readability and maintenance of its source code.*

Keywords: *Maintainability, Object Oriented Metrics, Refactoring, Quality, System*

1. PENDAHULUAN

Perangkat lunak adalah sekumpulan program komputer, prosedur, serta dokumen yang mengatur dan mengawasi fungsi-fungsi perangkat keras pada sistem komputer. Dalam proses pengembangan perangkat lunak, aspek jaminan kualitas menjadi hal yang penting pada setiap fase siklus hidup perangkat lunak. Terdapat beberapa ciri umum yang berkaitan dengan perlunya mengevaluasi kualitas perangkat lunak, salah satunya adalah bahwa setiap proyek perangkat lunak yang baik harus memenuhi dengan teliti perhitungan persyaratan dasar, dan setiap proyek perangkat lunak diharapkan memiliki kinerja yang baik, terutama dalam aspek pemeliharaan, keandalan, dan penggunaan kembali perangkat lunak yang krusial. Penyebab buruknya kinerja adalah kegagalan dalam menentukan persyaratan yang mendukung pembentukan pekerjaan dalam program [1].

Pengukuran perangkat lunak juga perlu dilakukan untuk memastikan tidak terdapat kekurangan pada hasil pengukuran dan untuk memastikan pengembangan sistem kedepannya lebih sesuai dengan kebutuhan. Salah satu metode pengukuran perangkat lunak yang dikenal adalah model McCall. Metode ini mencakup tiga kategori utama: Operasi Produk (kebenaran, keandalan, efisiensi, integritas, kemudahan penggunaan), Revisi Produk (kemudahan pemeliharaan, fleksibilitas, kemudahan pengujian), dan Transisi Produk (portabilitas, kemampuan penggunaan ulang, interoperabilitas). Inti dari konsep McCall adalah menilai hubungan antara elemen kualitas dan standar kualitas produk untuk meningkatkan mutu perangkat lunak [2]. Seperti yang dilakukan oleh peneliti [3] Penelitian ini menilai kualitas website SMKN 4 Bandung dengan menggunakan metode McCall. Metode ini bertujuan untuk meningkatkan mutu produk perangkat lunak dan situs web. hasil penelitian menunjukkan bahwa website sudah cukup baik terhadap faktor kualitas seperti correctness, reliability, usability, flexibility, dan portability. Hasil ini dapat menjadi pedoman dalam perbaikan dan pengembangan kualitas website.

Refactoring perangkat lunak adalah metode yang digunakan untuk mengubah struktur internal perangkat lunak tanpa mengubah fungsionalitasnya. Refactoring merupakan jenis perubahan yang bertujuan untuk meningkatkan kualitas perangkat lunak setelah dilakukan perbaikan, modifikasi, penambahan fitur, dan berbagai perubahan lainnya selama penggunaannya [4]. Code smell menandakan ada yang salah dengan kode yang perlu di refactoring. Hal ini merupakan teknik yang memodifikasi kode sumber menjadi bentuk yang lebih mudah dibaca dan dipelihara dengan menghilangkan code smell. Refactoring digunakan untuk meningkatkan kualitas perangkat lunak dengan mengurangi kompleksitasnya [5].

Pada penelitian [6] dengan menggunakan metode Refactoring untuk mengoptimalisasi perangkat lunak Klinik Utama Nur Khadijah dapat disimpulkan bahwa hasil refactoring perangkat lunak menunjukkan peningkatan kinerja dengan mengurangi kode duplikat dan menggunakan teknik extract method, hal ini mempengaruhi pada peningkatan kualitas maintainability, usability dan correctness. Pada penelitian [7] Tujuan utama penelitian ini adalah untuk mengevaluasi seberapa efektif metrik maintainability dalam memperkirakan usaha yang diperlukan untuk pemeliharaan perangkat lunak. Penelitian ini berfokus pada suite metrik yang dikembangkan oleh Chidamber dan Kemerer (CK-Metrics), yang mengukur berbagai aspek desain berorientasi objek, termasuk ukuran, kohesi, dan coupling. Penerapan metrik ini dapat mendukung pengelolaan kualitas perangkat lunak, khususnya dalam hal pemeliharaan dan pengurangan biaya. Studi ini menunjukkan bukti empiris bahwa penerapan metrik CK dapat berkontribusi pada peningkatan kualitas perangkat lunak serta efisiensi dalam pemeliharaannya. Pada penelitian [8] Penelitian ini mengukur kualitas perangkat lunak Jurnal Logic yang dibangun menggunakan framework Laravel pada aplikasi web Jurnal Logic. Metrik software digunakan untuk memahami karakteristik dan performa aplikasi web. Hasil penelitian menunjukkan bahwa nilai efisiensi, kompleksitas, keterpahaman, kemampuan penggunaan ulang, dan pemeliharaan produk dapat ditingkatkan melalui refactoring dan perbaikan yang didasarkan pada hasil pengukuran.

Sistem Pengelolaan Informasi dan Dokumentasi (PPID) di Dinas Kesehatan (Dinkes) Kota Cimahi dirancang untuk memfasilitasi akses informasi bagi masyarakat umum dan untuk pengelolaan data internal. Sistem ini memiliki dua tampilan utama: satu untuk pengguna umum, fungsi dari tampilan ini yaitu untuk memberikan akses informasi yang dibutuhkan oleh masyarakat umum, termasuk untuk mengajukan permohonan informasi. Tampilan satu lagi untuk pengelolaan data oleh admin dan staff, tampilan ini terbagi menjadi dua jenis user yaitu, admin dan staff. Admin memiliki kontrol penuh terhadap pengelolaan data dan pengguna sistem, sedangkan staff memiliki akses terbatas sesuai dengan peran mereka.

Pada sistem PPID Dinkes Cimahi ini memiliki kendala yaitu berupa nilai dari kriteria Object Oriented Metrics (CK-Metrics) memiliki nilai yang tinggi yang mempengaruhi kualitas maintainability. Object-oriented metrics adalah metrik perangkat lunak yang dirancang khusus untuk perangkat lunak berorientasi objek [9]. CK-Metric adalah kumpulan metrik kualitas yang digunakan untuk menilai kualitas perangkat lunak berorientasi objek, ada 6 kriteria pada CK-Metrics yaitu WMC, DIT, NOC, CBO, RFC, dan LCOM. Selain terdapat masalah pada metricsnya, sistem ini juga memiliki masalah pada kode sumbernya yaitu teridentifikasi adanya code smells. Code smells adalah tanda-tanda masalah dalam desain atau implementasi kode yang tidak menyebabkan kesalahan langsung, tetapi mempengaruhi kualitas, pemeliharaan, dan pengembangan sistem.

Untuk mengatasi masalah ini, penerapan metode refactoring sangat diperlukan. Refactoring adalah proses memperbaiki struktur internal kode tanpa mengubah perilaku eksternal dari perangkat lunak [5]. Tujuannya adalah untuk meningkatkan kualitas kode dan memudahkan tim atau pengembang baru untuk pemeliharaan serta pengembangan di masa depan yang mana hal ini sangat penting untuk proyek jangka panjang yang mana pergantian tim/pengembang tidak dapat terhindari. Dengan baiknya kualitas maintainability memastikan agar transfer pengetahuan berjalan lancar dan pengembang baru dapat cepat menjadi produktif.

2. METODE PENELITIAN

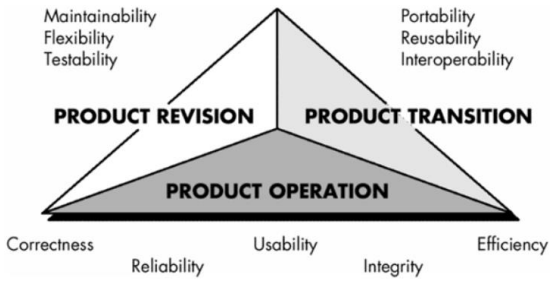
2.1 Tahapan Penelitian

Proses dan langkah-langkah yang dilakukan pada penelitian ini adalah :

1. Analisis Perangkat Lunak
   Pada tahap ini dilakukan analisis terhadap perangkat lunak dengan membuat *class diagram* sistem sekarang.
2. Identifikasi Masalah
   Mengidentifikasi perangkat lunak dengan menganalisa *Object Oriented Metrics* untuk mengetahui apa yang menjadi masalah pada perangkat lunak yang akan ditingkatkan kualitasnya.
3. Perencanaan Peningkatan Kualitas Perangkat Lunak
   Pada tahap ini dilakukan perencanaan untuk meningkatkan kualitas perangkat lunak dengan membuat *class diagram* yang baru yang akan dijadikan sebagai acuan untuk meningkatkan kualitas perangkat lunak.
4. Implementasi Refactoring
   Setelah melakukan analisa dan pengujian pada perangkat lunak, akan dilakukan optimalisasi dengan menggunakan metode Refactoring.
5. Pengujian Setelah Refactoring
   Setelah dilakukan optimalisasi perangkat lunak dengan Refactoring diuji kembali perangkat lunak.
6. Kesimpulan
   Memeberikan kesimpulan terhadap peningkatan kualitas perangkat lunak Dropship setelah melakukan peningkatan dengan Metode Refactoring.

2.2 McCall Model

Salah satu model pengujian kualitas perangkat lunak yang paling awal adalah McCall, yang dikembangkan pada tahun 1976. Tujuan dari model ini yaitu untuk menghubungkan sisi pengguna dan pengembang. Model ini berasumsi bahwa akar penyebab kinerja yang buruk sebuah perangkat lunak terletak pada kekurangan persyaratan yang jelas untuk mencakup aspek dari fungsional yang krusial dari sebuah perangkat lunak. Untuk mencapai kinerja perangkat lunak yang optimal, langkah pertama adalah memahami kebutuhan pengguna secara menyeluruh. Pada tahun 1977, McCall dan timnya mengusulkan klasifikasi faktor atau kriteria yang mempengaruhi kualitas perangkat lunak, yaitu *Product Revision*, *Product Transition*, dan *Product Operation* [10].



Gambar 1. McCall Model

McCall membagi elemen-elemen tersebut menjadi tiga komponen utama yang saling berpengaruh dan terkait, yaitu Operasi Produk, Revisi Produk, dan Transisi Produk. Operasi Produk mencakup: *Correctness*, *Reliability*, *Usability*, *Integrity*, dan *Efficiency*. Revisi Produk meliputi: *Maintainability*, *Flexibility*, dan *Testability*. Transisi Produk mencakup: *Portability*, *Reusability*, dan *Interoperability* [11].

Aspek	Faktor	Penjelasan
Product Revision	*Maintainability*	Mudahnya pemeliharaan *software*
	Flexibility	Mudahnya pengembangan *software* dalam memenuhi kebutuhan.
	Testability	Kemampuan dalam pengujian *software*
Product Transition	*Portability*	pemindahan menggunakan biaya lebih kecil dibandingkan mengembangan *software* dari awal.
	Reusability	Seberapa mungkin properti dan modul dari *software* dapat digunakan lagi dalam sistem lain
	Interoperability	Seberapa bisa kerja sistem untuk berkolaborasi dengan *software* lainnya.
Product Operations	*Correctness*	Sistem memenuhi spesifikasi dan kebutuhan *user*.
	Reliability	Sistem tidak adanya kesalahan atau memiliki kegagalan yang kecil.
	Efficiency	Hubungan antara *software* dan sumber daya.
	Integrity	Keamanan yang ada pada *software*, *developer* harus tahu kebutuhan akses pada software yang dirancang.
	Usability	*Software* dijalankan dan dipahami dengan mudah.

Gambar 2. Aspek, Faktor, dan Penjelasan metode McCall

2.3 Refactoring

William F. Opdyke merupakan yang merumuskan pertama teori refaktoring dalam tesisnya dan mulai dipraktikkan dan digunakan segera setelah diterbitkannya buku *"Refactoring: Improving the Design of Existing Code"* Martin Fowler menuliskan buku tentang refactoring pada tahun 1999. Sejak saat itu, teknik ini telah tersebar luas dalam penerapannya. Umumnya digunakan untuk memperbaiki perangkat lunak dan meningkatkan mutunya [12].

Proses mengubah sistem perangkat lunak dengan cara yang memungkinkan melalui perbaikan struktur internalnya tanpa mengubah perilaku kode di luarnya dikenal sebagai refactoring. Ini adalah cara untuk membersihkan kode Anda dan mengurangi kemungkinan kesalahan [5].

Dalam penelitian yang memanfaatkan metode McCall untuk menguji kualitas Sistem Informasi Akademik (SIAKAD), aspek operasional seperti kehandalan (reliabilitas), efisiensi, dan integritas disajikan dengan cermat [13]. Metode ini memungkinkan pengukuran kualitas berdasarkan faktor-faktor yang terkait dengan revisi produk, seperti kemudahan maintability, flexibility, dan testability [14]. Metode McCall memberikan keuntungan dalam pengujian kualitas perangkat lunak dan membantu mengidentifikasi area yang memerlukan perbaikan, terutama ketika melibatkan proses refactoring.

2.3 Code Smell

Code Smell adalah kode yang keberadaannya dapat menyebabkan masalah, terutama dalam konteks program yang memerlukan pemeliharaan [15].

Beberapa literatur mengidentifikasi berbagai jenis *bad smell* (termasuk anti-pattern), yang meliputi :

1. Duplicated Source Code
   Duplicated Source Code adalah salah satu bentuk *bad smell* yang ditandai dengan adanya struktur kode sumber yang identik di berbagai lokasi. Contoh sederhana adalah ketika terdapat dua metode dengan struktur yang sama dalam satu kelas [16].
2. Long Method
   Bad smell yang disebut *Long Method* terjadi ketika sebuah metode memiliki terlalu banyak baris kode untuk dieksekusi. Ini membuat pemahaman terhadap metode tersebut menjadi lebih sulit. Fowler menekankan bahwa hal ini juga dapat meningkatkan jumlah fungsionalitas dan kompleksitas operasional [16].
3. Feature Envy
   Feature Envy terjadi ketika sebuah method atau anggota class lebih tertarik pada *class* lain daripada *class* tempat *method* atau anggota *class* tersebut didefinisikan [16]. Biasanya, situasi ini terjadi karena kesalahan dalam menempatkan anggota *class*.
4. Data Clumps
   Ketika dua atau lebih data atau primitif selalu dideklarasikan bersama dan saling bergantung, ini menunjukkan adanya *Data Clumps* dalam *source code*. Contohnya adalah keberadaan pasangan

variabel *start* dan *end* di setiap *class* atau parameter *method*. *Data Clumps* biasanya terdiri dari pasangan nilai primitif yang sebenarnya bisa diubah strukturnya dengan menggabungkannya menjadi sebuah obyek [16].

2.4 Perangkat Lunak

Perangkat lunak merupakan suatu entitas yang tidak dapat dilihat atau disentuh secara langsung, tetapi pengguna dapat menggunakannya. Perangkat lunak komputer adalah sekumpulan informasi elektronik yang disimpan dan diatur oleh sistem komputer. Data elektronik yang tersimpan dalam komputer dapat berupa program atau serangkaian instruksi yang memungkinkan komputer untuk menjalankan perintah tertentu. Perangkat lunak juga dikenal sebagai interpreter atau penyalur dari perintah-perintah yang diberikan oleh pengguna komputer, yang bertujuan untuk mengarahkan atau memprosesnya melalui perangkat keras. Dengan bantuan program ini, komputer dapat melaksanakan perintah. Secara umum, perangkat lunak dapat diklasifikasikan ke dalam tiga kategori utama: bahasa pemrograman, sistem operasi, dan aplikasi sistem [17].

2.5 Kualitas Maintainability

Pada penelitian yang di lakukan oleh [18] dengan menggunakan metode *clean code* peneliti dapat meningkatkan kualitas *maintainability* pada *software* sistem informasi akademik sekolah. *Maintainability* adalah kemampuan perangkat lunak untuk dimodifikasi guna memperbaiki kesalahan, menyesuaikan dengan lingkungan baru, atau meningkatkan kinerja [19]. Dalam konteks pengujian perangkat lunak, pengujian *maintainability* dilakukan untuk mengevaluasi sejauh mana sistem atau produk perangkat lunak dapat dipelihara dengan mudah dan efisien [20].

2.6 Object Oriented Metrics

Object-oriented metrics adalah metrik perangkat lunak yang dirancang khusus untuk perangkat lunak berorientasi objek. Dalam konteks ini, konsep utama yang digunakan adalah objek, yang memiliki properti dan operasi yang merepresentasikan sebuah kelas. Kelas merupakan spesifikasi dari objek, dan properti serta operasi objek didefinisikan oleh kelas dalam bentuk atribut dan metode [9]. CK-Metric adalah sekumpulan metrik kualitas yang digunakan untuk mengevaluasi kualitas perangkat lunak yang berbasis objek. CK-Metric memiliki 6 metric yaitu :

1. Weighted Method per Class (WMC)
   WMC adalah metrik yang digunakan untuk menilai kompleksitas sebuah kelas. Terdapat dua pendekatan untuk menghitung metrik ini. Pendekatan pertama adalah dengan menjumlahkan nilai kompleksitas dari setiap metode dalam kelas menggunakan *Cyclomatic Complexity*. Pendekatan kedua adalah memberikan nilai kompleksitas 1 untuk setiap metode dalam kelas dan kemudian menjumlahkannya. Semakin tinggi nilai WMC, semakin rendah kualitas aplikasi tersebut [21].
2. Depth of Inheritance Tree (DIT)
   DIT adalah kedalaman maksimum dari simpul akar pohon ke simpul tertentu. Kelas direpresentasikan sebagai simpul atau node. Semakin dalam node berada di pohon, semakin banyak metode yang dapat dimilikinya karena mewarisi metode dari kelas-kelas di atasnya, yang membuat kelas semakin kompleks. Semakin tinggi nilai DIT berarti semakin besar potensi penggunaan kembali metode yang diwarisinya [21].
3. Number Of Children (NOC)
   NOC adalah jumlah subclass langsung dari sebuah kelas dalam suatu hierarki kelas. NOC mengukur seberapa banyak *subclass* yang akan mewarisi metode dari kelas induknya. Nilai NOC berbanding lurus dengan besar upaya dan waktu untuk testing [9].
4. Coupling Between Objects (CBO)
   Coupling dalam perangkat lunak berbasis objek adalah kondisi di mana sebuah kelas menggunakan metode atau atribut dari kelas lain. Dua kelas dianggap terhubung (*coupled*) ketika metode yang dideklarasikan dalam satu kelas menggunakan metode atau variabel yang didefinisikan oleh kelas lain. Semakin tinggi nilai CBO, semakin rendah kualitas aplikasi tersebut [9].
5. Response For a Class (RFC)
   RFC adalah metrik yang mengukur jumlah metode yang dapat dieksekusi sebagai tanggapan terhadap pesan yang diterima oleh objek dari suatu kelas. Semakin banyak metode yang dapat dipanggil dari suatu kelas, semakin kompleks kelas tersebut [21].
6. Lack of Cohesion in Methods (LCOM)
   LCOM adalah metrik yang menghitung perbedaan antara jumlah pasangan metode dalam sebuah kelas yang tidak memiliki hubungan atau kesamaan, dengan jumlah pasangan metode yang memiliki hubungan atau kesamaan. Metrik ini berkaitan dengan variabel instan dan metode, serta mengukur

atribut dari kelas objek. Semakin banyak metode yang memiliki kesamaan, semakin kohesif kelas tersebut [21].

Dengan mempertimbangkan pengaruh kualitas dan nilai metrik, kita dapat menetapkan nilai ambang batasnya. Mago dan Kaur telah mengusulkan *threshold* untuk CK Metric [22].

Tabel 1. *Threshold values* CK Metrics

Metrics	*Threshold values*
WMC	*Low*, 0 - 15
DIT	*Low*, 0 - 6
NOC	*Low*, 0 - 6
CBO	*Low*, 0 - 8
RFC	*Low*, 0 - 35
LCOM	*Low*, 0 - 1

Berdasarkan tabel yang telah disajikan mengenai threshold values CK-Metrics dapat diartikan bahwa nilai WMC mengukur kompleksitas kelas berdasarkan jumlah metode yang dimilikinya. Nilai WMC antara 0 hingga 15 dianggap rendah, menunjukkan kelas tersebut sederhana dan mudah dipahami. DIT mengukur kedalaman kelas dalam hierarki pewarisan. Nilai DIT 0 hingga 6 dianggap rendah, menunjukkan hierarki tidak terlalu dalam dan lebih mudah dikelola. NOC mengukur jumlah subclass yang mewarisi suatu kelas. Nilai NOC 0 hingga 6 menunjukkan kelas tersebut memiliki sedikit subclass, membuatnya lebih mudah dipahami. CBO mengukur ketergantungan suatu kelas pada kelas lain. Nilai CBO 0 hingga 8 menunjukkan ketergantungan rendah, membuat kelas lebih modular dan mudah diubah tanpa mempengaruhi sistem lain. RFC mengukur jumlah metode yang dipanggil oleh sebuah kelas. Nilai RFC 0 hingga 35 menunjukkan kelas yang lebih sederhana dengan interaksi yang lebih sedikit dan kurang kompleks. LCOM mengukur kohesi antara metode dalam sebuah kelas. Nilai LCOM 0 hingga 1 menunjukkan kohesi yang baik, artinya metode saling terkait dan berkolaborasi untuk fungsi yang sama, sehingga kelas lebih mudah dipahami.

Secara keseluruhan, nilai ambang batas yang rendah pada metrik-metrik ini mengindikasikan bahwa kelas-kelas dalam perangkat lunak tersebut relatif sederhana, memiliki kompleksitas yang rendah, dan lebih mudah untuk dikelola serta dipelihara.

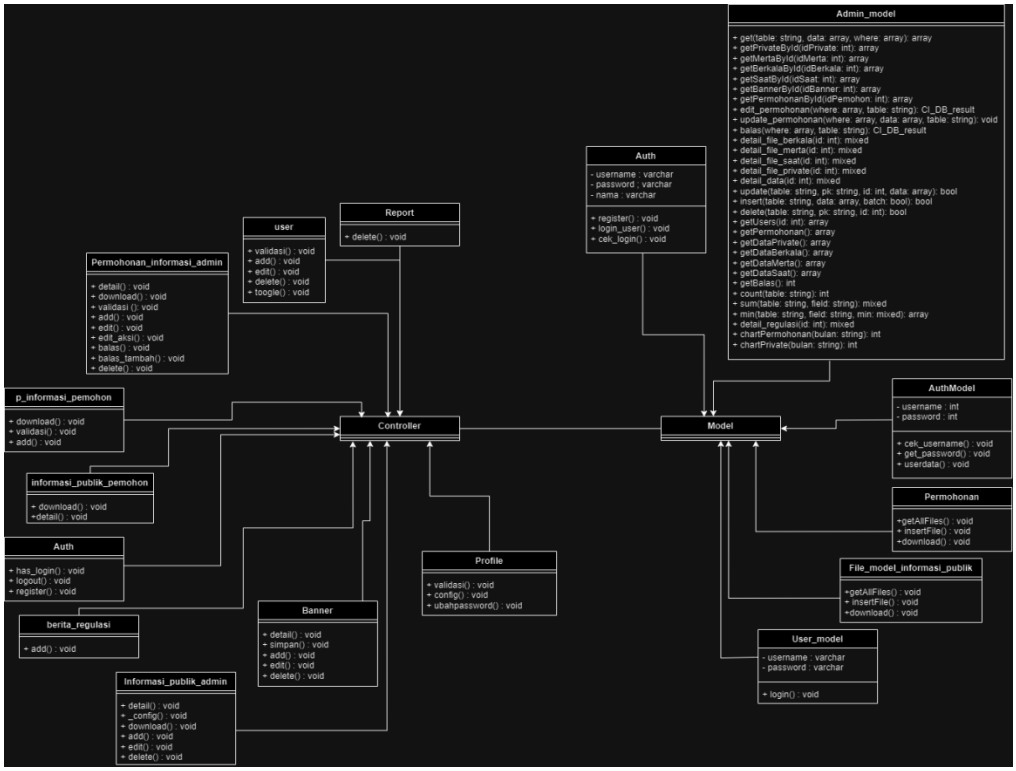
3. HASIL DAN PEMBAHASAN

3.1 Analisis Perangkat Lunak

Pada tahapan ini peneliti menampilkan *Class Diagram* pada sistem yang berjalan saat ini untuk dijadikan sebagai acuan pengukuran *object oriented metrics*.

3.1.1 Class Diagram

Pada tahapan ini peneliti membuat sebuah *class diagram* dari sistem PPID Dinkes Cimahi yang sedang berjalan, dapat dilihat dari gambar dibawah.



Gambar 3. *Class Diagram*

3.2 Identifikasi Masalah

Pada tahapan ini melakukan pengukuran object oriented metrics pada sistem. Setelah dilakukan pengukuran akan dilakukan analisis smell code dan analisis object oriented metrics untuk mengetahui permasalahan pada class yang di analisa.

3.2.1 Pengukuran menggunakan OO Metrics

Pada tahapan ini peneliti melakukan pengukuran sistem PPID Dinkes Cimahi dengan menggunakan OO Metrics yaitu CK Metrics, Ada enam kriteria yang akan diukur, yaitu *Weighted Method per-Class* (WMC), *Depth of Inheritance Tree* (DIT), *Number Of Children* (NOC), *Coupling Between Object* (CBO), *Response For a Class* (RFC), dan *Lack of Cohesion in Method* (LCOM). Metrics tersebut akan digunakan untuk menghitung pada tiap class-class yang ada di sub bab class diagram.

Tabel 2. Pengukuran OO Metrics

Class	WMC	DIT	NOC	CBO	RFC	LCOM
Permohonan_informasi_admin	9	2	0	8	22	0
Banner	8	1	0	9	13	0
User	5	2	0	8	35	1
P_informasi_pemohon	3	1	0	9	16	5
Informasi_publik_pemohon	2	1	0	7	13	3
Auth	3	1	0	9	20	4
Berita_regulasi	3	1	0	8	7	2
Profile	6	1	0	6	24	5
Dashboard	1	1	0	4	11	0
Report	3	1	0	8	10	0
Admin_model	31	1	0	15	40	1
Auth_model	3	1	0	5	6	1
Auth	3	1	0	6	13	1
UserModel	1	1	0	3	6	0
File_model_informasi_publik	3	1	0	2	7	0

Permohonan	4	1	0	2	7	0

Pada hasil pengukuran CK-Metrics yang telah dilakukan dapat dilihat terdapat di beberapa modul mendapatkan nilai yang masih tinggi di beberapa kriterianya. Seperti pada modul kelas 'Banner' pada kriteria CBO mendapatkan nilai yang tinggi yakni 9 (nilai *Threshold* CBO = 0-8), pada modul kelas 'P_informasi_pemohon' pada kriteria CBO dan LCOM yang tinggi melebihi batas nilai threshold yang telah ditentukan seperti pada tabel 2, dan masih terdapat beberapa modul kelas lain yang memiliki nilai kriteria yang tinggi dengan ditandai warna merah.

Pengukuran *object oriented metrics* ini tidak dapat dijadikan sebagai acuan untuk menemukan *code smell*, karena walaupun nilai *object oriented metrics* ini bagus tidak menutup kemungkinan bahwa di modul tersebut bersih dari *code smell*. Namun, peneliti melakukan pengujian ini untuk menentukan prioritas dari modul kelas mana saja yang harus diperbaiki dan ditingkatkan kualitasnya berdasarkan nilai *object oriented metrics*.

3.2.2 Analisis Smell Code

Pada tahapan ini akan dilakukan analisis Smell Code yang ditemukan pada modul 'Admin_model', 'Banner', 'Auth' dan 'Profile'. Berikut ini merupakan smell code yang ditemukan :

3.2.2.1 Long Method

Pada kelas 'Admin_model' terindikasi adanya *code smell long method* dimana metode 'get' dan Sekumpulan metode untuk mendapatkan data dari tabel-tabel tertentu seperti 'getPrivateById', 'getMertaById', 'getBerkalaById', dst. Hal tersebut meyebabkan metode melakukan terlalu banyak hal atau memiliki tanggung jawab yang terlalu besar, sehingga sulit untuk dipelihara, diuji, atau dimodifikasi secara efisien.

3.2.2.2 Duplicated Code

Pada kelas 'Banner' terindikasi adanya *smell code duplicated code* yaitu pada 'Pengaturan Konfigurasi Upload' dan 'Duplikasi pada Validasi dan Proses Upload'. Pengaturan konfigurasi *upload* diulang beberapa kali di beberapa metode.

3.2.2.3 Feature Envy

Pada kelas 'Auth' Metode index menggunakan metode dari model 'Auth_model' seperti 'cek_username', 'get_password', dan 'userdata;. Ini menunjukkan bahwa logika autentikasi sebenarnya berinteraksi lebih dekat dengan model 'Auth_model' daripada dengan 'Auth' itu sendiri.

3.2.2.4 Data Clumps

Di dalam kelas 'Profile', *data clumps* dapat terindikasi dari grup data yang selalu muncul bersama-sama atau diolah secara bersamaan. contoh *data clumps* yang terlihat adalah penggunaan session data dan manipulasi data input dari form.

Data clumps dapat diamati pada bagian berikut:

- Manipulasi Data Input : Penggunaan '$input = $this->input->post(null, true);' untuk mengambil semua data input dari *form*. Data ini kemudian digunakan untuk melakukan operasi validasi, penyimpanan, dan pengolahan lainnya.

- Pengelolaan Foto : Pengelolaan foto yang melibatkan '$input['foto']', serta penghapusan foto lama jika ada, berdasarkan data session seperti 'userdata('foto')'.

Pada kelas 'Admin_model' juga terindikasi adanya *Code Smell Data Clumps* terlihat pada *method-method* yang menggunakan parameter yang sama berulang kali untuk operasi yang serupa.

3.2.3 Analisis OO Metrics

a. Admin_model
   WMC : 33
   CBO : 15
   RFC : 43
   Masalah : Kelas ini memiliki nilai WMC, CBO, dan RFC yang sangat tinggi. Hal ini menunjukkan bahwa kelas ini sangat kompleks, memiliki banyak ketergantungan, dan merespon terhadap banyak pesan. Hal ini mengindikasikan bahwa 'Admin_model' mungkin melakukan terlalu banyak pekerjaan dan memiliki terlalu banyak tanggung jawab.
b. Banner

CBO : 9
Masalah : Kelas ini memiliki nilai CBO yang tinggi, menunjukkan bahwa ia memiliki terlalu banyak ketergantungan terhadap kelas atau objek lain. Ini mungkin menunjukkan bahwa kelas tersebut memiliki terlalu banyak tanggung jawab atau terlalu banyak interaksi dengan kelas lain, yang membuatnya sulit untuk dikelola, diuji, dan dipelihara.

c. Auth
LCOM : 5
CBO : 9
Masalah : Nilai LCOM yang sangat tinggi menunjukkan bahwa kelas ini memiliki metode-metode yang tidak saling terkait. Nilai CBO yang tinggi menunjukkan bahwa kelas tersebut memiliki banyak ketergantungan terhadap kelas atau objek lain.

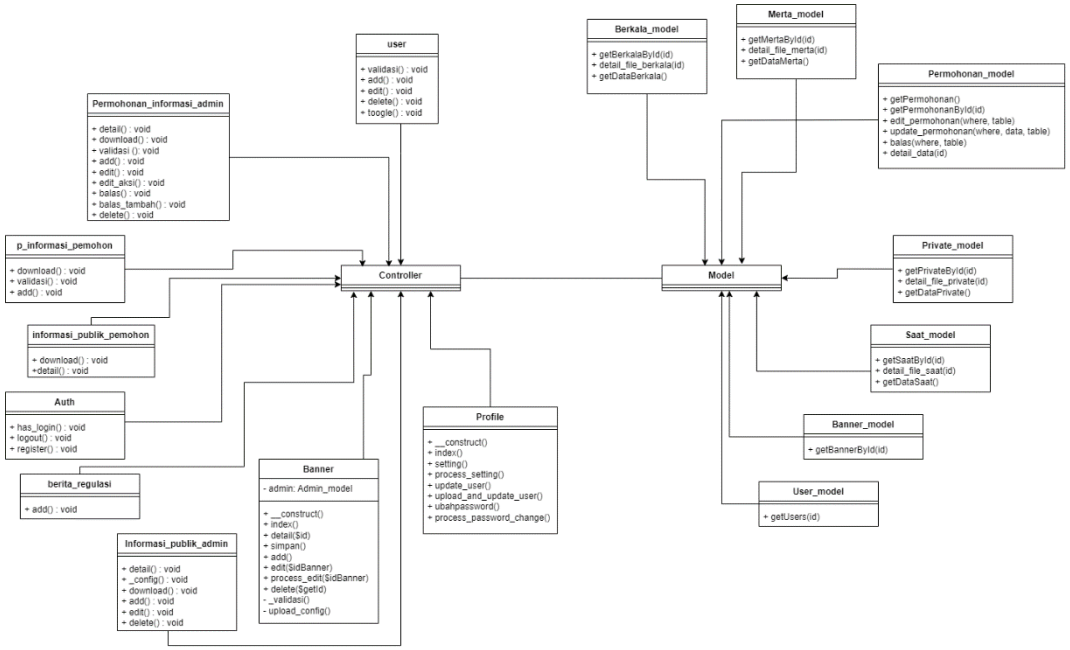
d. Profile
LCOM : 5
Masalah : Nilai LCOM yang sangat tinggi menunjukkan bahwa kelas ini memiliki metode-metode yang tidak saling terkait.

3.3 Perencanaan Peningkatan Kualitas Perangkat Lunak

Pada tahapan ini yang dilakukan peneliti yaitu membuat *class diagram* yang baru untuk refactoring sistem.

3.3.1 Class Diagram yang baru

Pada tahapan ini dilakukan perancangan *Class Diagram* yang baru pada sistem PPID Dinkes Cimahi untuk dijadikan sebagai acuan saat pelaksanaan refactoring.



Gambar 4. *Class Diagram* Baru

Pada *class diagram* yang baru ini memecahkan beberapa kelas seperti pada kelas ‘Admin_model’ yang sebelumnya bertanggung jawab banyak sekali metode, sehingga di pecah menjadi beberapa kelas yang bertanggung jawab pada metodenya masing-masing seperti ‘Merta_model’, ‘Permohonan_model’, ‘Private_model’, dan sebagainya. Agar mempermudah pengembang memahami struktur dari *source code* dan lebih modular.

3.4 Implementasi Refactoring

Setelah melakukan analisa dan pengukuran sistem, peneliti akan menerapkan implementasi refactoring untuk meningkatkan kualitasnya.

3.4.1 Refactoring Code

Pada tahapan ini peneliti melakukan refactoring *code* pada class yang telah diukur pada sub-bab sebelumnya pada Sistem PPID Dinkes Cimahi :

a. Admin_model

Untuk menormalkan nilai WMC, CBO dan RFC pada class 'Admin_model' , perlu dilakukan refaktorisasi untuk mengurangi kompleksitas, meningkatkan pemisahan tanggung jawab dan mengurangi ketergantungan antar metode. Berikut ini langkah yang yang dilakukan :

1. Memisahkan fungsi berdasarkan kategori
   Memisahkan fungsi-fungsi berdasarkan kategori atau modul dapat membantu mengurangi kompleksitas juga menghilangkan *long method* pada modul.
2. Mengurangi ketergantungan antar metode
   Kurangi ketergantungan antar metode dalam satu model dengan menghindari penggunaan metode satu sama lain dalam model yang sama. Setiap metode harus melakukan satu tugas tertentu.

b. Banner

Pada kelas Banner memiliki nilai kriteria CBO yang tinggi yakni 9 sehingga melebihi dari nilai *threshold* yaitu (0-8) maka dari itu akan dilakukan Refactoring pada kelas Banner agar nilai mendapatkan nilai yang sesuai.

Pada penelitian [23] peneliti menyimpulkan bahwa *Extract Method* adalah metode refaktorisasi yang efektif dalam meningkatkan kualitas perangkat lunak dengan mengurangi kompleksitas kode, meningkatkan keterbacaan, dan memudahkan pemeliharaan. Temuan ini memberikan wawasan berharga bagi para pengembang perangkat lunak dan peneliti tentang pentingnya menggunakan teknik refaktorisasi dalam praktik pengembangan perangkat lunak.

Berikut ini langkah-langkah yang akan dilakukan :

1. Mengurangi ketergantungan pada kelas lain
2. Memecah fungsi besar menjadi fungsi kecil
3. Memisahkan konfigurasi upload

c. Auth

Pada kelas Auth memiliki nilai CBO dan LCOM yang tinggi yakni 9 dan 4 yang menyebabkan nilai tersebut melebihi nilai threshold (CBO = 0-8) dan (LCOM = 0-1). Untuk mengatasi hal tersebut dapat dilakukan refactoring dengan teknik *Move Method* dimana logika autentikasi dipindahkan ke model 'auth'. Penelitian yang dieksplorasi dalam [24] membahas metode "*move method*" untuk mengidentifikasi peluang pemindahan metode antar kelas guna meningkatkan desain perangkat lunak. Pendekatan ini bertujuan untuk meningkatkan pemeliharaan kode dan mengurangi masalah desain seperti kopling tinggi dan kohesi rendah. Dengan demikian, kualitas perangkat lunak secara keseluruhan dapat ditingkatkan.

Berikut ini tahapan refactoring yang dilakukan :

1. Memisahkan Tanggung Jawab
   Metode 'index()' dibagi menjadi beberapa metode yang lebih kecil ('validate_login_input()', 'authenticate_user()', 'login_user()') untuk meningkatkan kohesi dan mengurangi kompleksitas.
2. Validasi Registrasi
   Metode 'validate_registration_input()' dibuat untuk memisahkan logika validasi input registrasi.
3. Pemindahan Metode
   Pindahkan logika yang terlalu banyak mengakses data dari objek lain ke dalam objek yang lebih sesuai. Metode authenticate() dan get_userdata() dipindahkan ke Auth_model untuk menangani otentikasi dan pengambilan data pengguna.

d. Profile

Pada class 'Profile' ini memiliki nilai LCOM yang cukup tinggi yaitu 5, sedangkan untuk ambang batas LCOM yang sudah di tentukan yakni (0 – 1). Hal ini disebabkan karena kelas memiliki metode-metode yang kurang terkait antara satu dengan yang lainnya, sehingga hal tersebut menjadikan kelas memiliki tanggung jawab yang terlalu banyak atau tidak terstruktur dengan baik. Untuk mengurangi nilai LCOM yang tinggi tersebut maka dilakukan Refactoring dengan menggunakan teknik *Extract Class*. *Extract Class* adalah teknik refactoring di mana tanggung jawab dari sebuah kelas yang terlalu besar atau kompleks dipisahkan menjadi beberapa kelas yang lebih kecil dan lebih spesifik. Refactoring *Extract Class* bertujuan untuk merestrukturisasi sebuah kelas dengan memindahkan beberapa fungsionalitasnya ke kelas baru. Langkah ini dilakukan untuk mengurangi kompleksitas dan meningkatkan kohesi kelas tersebut [25].

Berikut ini langkah yang diambil untuk memperbaiki nilai LCOM pada kelas Profile :

1. Memisahkan Logika Validasi dan *Upload*

Metode '_validasi' dapat dipindahkan ke kelas atau pustaka validasi yang terpisah. Hal ini akan mengurangi tanggung jawab kelas 'Profile' dan membuat kode lebih modular dan Metode '_config' yang mengatur konfigurasi upload dapat dipindahkan ke pustaka atau kelas konfigurasi yang terpisah.

2. Refactor Fungsi *Setting*
   Fungsi 'setting' memiliki logika yang cukup kompleks yang dapat dipisahkan menjadi metode-metode yang lebih kecil. Misalnya, pemrosesan unggahan file dan pembaruan data pengguna bisa dijadikan metode terpisah.
3. Memisahkan Logika Ubah *Password*
   Fungsi 'ubahpassword' dapat dipindahkan ke kontroler yang terpisah atau dipecah menjadi metode-metode yang lebih kecil dan lebih modular.

3.5 Pengujian setelah refactoring

Setelah dilakukan refactoring akan dilakukan pengujian blackbox untuk mengetahui apakah sistem masih berjalan dengan semestinya atau tidak. Selain itu, akan dilakukan pengukuran kembali object oriented metrics pada kelas-kelas yang di refactoring.

3.5.1 Pengujian Blackbox

Berdasarkan hasil pengujian menggunakan metode *black box* pada sistem PPID Dinkes Cimahi berbasis *website* yang telah direfaktorisasi, semua fungsi telah terpenuhi dan berjalan dengan baik serta sesuai dengan yang diharapkan. Semua skenario pengujian yang dilaksanakan akan dihitung dan diukur menggunakan persentase kesesuaian, yang dihitung berdasarkan hasil pengujian sebagai berikut :

Jumlah Item Uji

= 19 Item

Item uji dengan hasil SESUAI = 19

Item uji dengan hasil TIDAK SESUAI = 0

Persentase kesesuaian = $\left(\frac{jumlah\ item\ uji - item\ uji\ tidak\ sesuai}{jumlah\ item\ uji}\right) x100\%$

= $\left(\frac{19-0}{19}\right) x100\% = 100\%$

Perhitungan kesesuaian atas semua fungsi yang ada pada sistem yang diuji akan menunjukkan apakah sistem ini SESUAI atau TIDAK SESUAI dengan tujuan uji yang sudah dilaksanakan sebelumnya. Berikut merupakan perhitungan persentase kesesuaian sistem dengan fungsi kesesuaian :

Persentase Hasil = $\frac{jumlah\ informasi\ yang\ sesuai}{jumlah\ item\ uji}\ x\ 100\%$

= $\frac{19}{19}\ x\ 100\% = 100\%$

Berdasarkan perhitungan yang telah dilakukan sebelumnya, dapat disimpulkan bahwa semua kasus uji dalam pengujian sistem sesuai dengan rancangan dan harapan yang telah ditetapkan, dengan tingkat kesesuaian mencapai 100%.

3.5.2 Pengukuran OO Metrics yang baru

Pada tahapan ini peneliti melakukan pengujian kembali *Object Oriented Metric* (CK-Metrics) pada sistem PPID Dinkes Cimahi setelah dilkukan proses refactoring. Pengujian dilakukan hanya terhadap modul yang di refactoring. Dapat dilihat dari tabel dibawah ini.

Tabel 3. Pengukuran OO Metrics (Baru)

Class	WMC	DIT	NOC	CBO	RFC	LCOM
Banner	10	1	0	8	20	1

Auth	3	1	0	5	20	2
Profile	6	1	0	7	20	0
Admin_model	15	1	0	4	23	0
Permohonan_model	6	1	0	2	6	0
Private_model	3	1	0	2	3	0
Merta_model	3	1	0	2	3	0
Berkala_model	3	1	0	2	3	0
Saat_model	3	1	0	2	3	0
Banner_model	1	1	0	2	1	0
User_model	1	1	0	2	1	0

3.5.3 Analisis Pengukuran/Hasil Perubahan

Berdasarkan hasil pengujian *Object Oriented Metrics* yang telah dilakukan dapat dilihat perbandingan pada tabel 2 dan tabel 3 setelah dilakukan Pemecahan kelas pada 'Admin_model' saat di hitung kembali nilai Kriteria WMC, CBO dan RFC menurun sehingga mendapatkan nilai di bawah nilai threshold pada kriterianya masing-masing. Kelas Admin_model saat di hitung kembali mendapatkan nilai WMC, CBO dan RFC yang baik (berada di nilai *Threshold*).

Begitu juga dengan kelas 'Profile' yang memiliki nilai kriteria LCOM yang tinggi, setelah dilakukan refactoring dengan melakukan langkah-langkah yang sudah dijabarkan untuk mendapatkan nilai yang sesuai dan juga menghilangkan duplikasi kode. Dapat dilihat pada saat diuji kembali nilai LCOM nya berada di nilai *threshold* yang seharusnya yakni 0.

Pada kelas 'Banner' juga setelah dilakukan Refactoring mendapatkan nilai kriteria CBO = 8 (sudah berada di nilai *treshold* yakni (0-8)), yang sebelumnya mendapatkan nilai kriteria CBO = 9. Pada Kelas 'Auth' juga pada saat sebelum refactoring dilakukan memiliki nilai CBO dan LCOM yang tinggi, yakni CBO = 9 dan LCOM = 10, namun pada saat dilakukan refactoring nilai CBO menurun menjadi 5 dan LCOM menurun menjadi 2, Meskipun nilai LCOM masih belum mencapai target nilai *threshold* (0 – 1) tetapi angka tersebut berhasil mendekati nilai *threshold*.

Setelah dilakukan refactoring pada sistem PPID Dinkes Cimahi, *smell code* yang ditemukan pada beberapa modul yang di perbaiki oleh peneliti, berhasil dihilangkan dengan melakukan beberapa teknik refactoring.

4. KESIMPULAN

Berdasarkan penelitian yang telah dilakukan pada sistem PPID Dinkes Cimahi setelah dilakukan proses refactoring dapat dilihat hasil dari kriteria Object Oriented Metrics yang dinilai menunjukan peningkatan kualitas. Seperti pada kelas 'Admin_model' sebelum dilakukan refactoring nilai dari kriteria WMC, CBO dan RFC terbilang tinggi karena melebihi nilai threshold yang telah ditentukan, setelah dilakukan refactoring dengan memecah kelas-kelas yang berkaitan, nilai-nilai kriteria tersebut menurun hingga mencapai nilai threshold masing-masing kriteria, untuk nilai WMC sebelum dilakukan refactoring (31) setelah refactoring (15), nilai CBO sebelum refactoring (15) setelah refactoring (4), nilai RFC sebelum refactoring (40) setelah refactoring (23).

Begitu juga dengan kelas yang lain seperti kelas 'Profile' dengan melakukan refactoring nilai kriteria LCOM yang sebelumnya tinggi (5), menjadi rendah (0). Ada juga kelas 'Banner' yang memiliki nilai kriteria CBO tinggi (9) menjadi rendah (8) setelah dilakukan refactoring. Begitu juga dengan kelas 'Auth' yang memiliki nilai CBO dan LCOM yang sebelumnya tinggi melebihi threshold yakni CBO (9) dan LCOM (4), berhasil dikurangi nilainya sehingga berada dan mendekati nilai threshold CBO (5) dan LCOM (2).

Oleh karena itu dengan menggunakan metode refactoring nilai-nilai Object Oriented Metrics dapat di optimalisasi dan mengurangi smell code untuk meningkatkan kualitas maintainability pada sistem PPID Dinkes Cimahi, juga mempermudah keterbacaan pada source code-nya.

5. UCAPAN TERIMAKASIH

Pada kesempatan ini peneliti ingin mengucapkan terima kasih kepada seluruh pihak yang terlibat pada penelitian ini serta terima kasih kepada para Dosen Informatika Universitas Jenderal Achmad Yani yang telah memberikan bekal ilmu sehingga berhasil menuntaskan penelitian ini.

REFERENCES

[1] S. S. HILABI, "Analisis Kualitas Perangkat Lunak Terhadap Sistem Informasi Stt Wastukancana Purwakarta," *J. Inform.*, vol. 1, pp. 27–32, 2018.

[2] Andria, Kusrini, and A. Armadyah, "Evaluasi Kualitas Web Portal STT Dharma Iswara Madiun Menggunakan Metode McCall," *Politek. Sawunggalih Aji - Purworejo*, vol. 4, no. 1, pp. 341–348, 2020.

[3] A. Suhari Camara M, K. Aelani, and F. Dwi Juniar S, "Pengujian Kualitas Website menggunakan Metode McCall Software Quality," *J. Inf. Technol.*, vol. 3, no. 1, pp. 25–32, 2021, doi: 10.47292/joint.v3i1.43.

[4] R. Nindyasari, "Metode Non- Heuristic Untuk Deteksi Refactoring Non-Source Code (Systematic Literature Review)," *Simetris J. Tek. Mesin, Elektro dan Ilmu Komput.*, vol. 6, no. 2, p. 361, 2015, doi: 10.24176/simet.v6i2.473.

[5] A. Kaur and M. Kaur, "Analysis of Code Refactoring Impact on Software Quality," *MATEC Web Conf.*, vol. 57, 2016, doi: 10.1051/matecconf/20165702012.

[6] H. Fredianto, "Optimalisasi Perangkat Lunak Menggunakan Metode Refactoring," *J. Syntax Admiration*, vol. 2, no. 10, pp. 1885–1902, 2021, doi: 10.46799/jsa.v2i10.326.

[7] B. R. Reddy and A. Ojha, "How effective are maintainability metrics in estimating maintenance efforts? An empirical study," *Int. J. Syst. Assur. Eng. Manag.*, vol. 10, no. 5, pp. 984–1001, 2019, doi: 10.1007/s13198-019-00828-3.

[8] M. I. Susanto, E. Darwiyanto, and G. A. A. Wisudawan, "Pengukuran Software Metric Terhadap Implementasi Framework Laravel Pada Pembangunan Aplikasi Berbasis Web Studi Kasus : Jurnal Logic," *e-Proceeding Eng.*, vol. 2, no. 3, pp. 7731–7738, 2015.

[9] L. Cheikhi, R. E. Al-Qutaish, A. Idri, and A. Sellami, "Chidamber and Kemerer Object-Oriented Measures: Analysis of their Design from the Metrology Perspective," *Int. J. Softw. Eng. Its Appl.*, vol. VIII, pp. 359–374, 2014, [Online]. Available: https://www.researchgate.net/publication/260835125_Chidamber_and_Kemerer_Object-Oriented_Measures_Analysis_of_their_Design_from_the_Metrology_Perspective.

[10] A. N. Ikhsan, "Pengujian Sistem Informasi Akademik Universitas X dengan Menggunakan Teori Kualitas Mccall," *Citisee*, pp. 43–47, 2019, [Online]. Available: https://citisee.amikompurwokerto.ac.id/assets/proceedings/paper/9_Ali_Nur_Ikhsan_CITISEE_2019.pdf.

[11] S. A. Saputera, D. Sunardi, A. Syafrizal, and P. Samsidi, "Evaluasi Sistem Informasi Akademik Menggunakan Metode Mccall," *J. Technopreneursh. Inf. Syst.*, vol. 3, no. 2, pp. 9–16, 2020, doi: 10.36085/jtis.v3i2.878.

[12] R. Nindyasari and S. Rochimah, "Proses Refactoring Paket Menggunakan Teknik Clustering," *J. Ilm. NERO*, vol. 3, no. 1, pp. 21–29, 2017, [Online]. Available: https://nero.trunojoyo.ac.id/index.php/nero/article/view/69.

[13] A. Hidayati, E. Oktariza, F. Rosmaningsih, and S. A. Lathifah, "Analisa Kualitas Perangkat Lunak Sistem Informasi Akademik Menggunakan McCall," *Multinetics*, vol. 3, no. 1, p. 48, 2017, doi: 10.32722/multinetics.vol3.no.1.2017.pp.48-53.

[14] A. Farisi and H. Saputra, "Analisis Kualitas Sistem Informasi Menggunakan Metode McCall: Studi Kasus SPON MDP," *Techno.Com*, vol. 21, no. 2, pp. 237–248, 2022, doi: 10.33633/tc.v21i2.5970.

[15] F. Arcelli Fontana, M. V. Mäntylä, M. Zanoni, and A. Marino, "Comparing and experimenting machine learning techniques for code smell detection," *Empir. Softw. Eng.*, vol. 21, no. 3, pp. 1143–1191, 2016, doi: 10.1007/s10664-015-9378-4.

[16] M. Fowler, "Refactoring: Improving the Design of Existing Programs.," p. 337, 1999.

[17] A. R. Lubis, "Perangkat Lunak Komputer," *Program*, pp. 1–9, 2020, [Online]. Available: http://ptiasugeng.blogspot.com/2015/01/jurnal.

[18] D. Firmansyah, "Optimasi Maintainability Menggunakan Metode Clean Code Pada Sistem Informasi Akademik Sekolah," *Univ. Komput. Indones.*

[19] R. Malhotra and A. Chug, "Software Maintainability: Systematic Literature Review and Current Trends," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 26, no. 8, pp. 1221–1253, 2016, doi: 10.1142/S0218194016500431.

[20] L. Mustari S, A. Sa'ban Miru, and R. Amalia, "Pengujian Aplikasi Sistem Monitoring Perkuliahan Menggunakan Standar ISO 25010," *J. Mediat.*, vol. 3, no. 3, pp. 1–7, 2024.

[21] P. K. Bhatia and R. Mann, "An Approach to Measure Software Reusability of OO Design," *Technology*, pp. 26–30, 2008.

[22] P. Mago, J. & Kaur, "Analysis of quality of the design of the object oriented software using fuzzy logic," *Int. Conf. Recent Adv. Futur. Trends Inf. Technol. Proc. Publ. Int. J. Comput. Appl.*, pp. 21–25, 2012, [Online]. Available: https://pdfs.semanticscholar.org/043f/ddd25c3af905a24d762ee832221170f4bcb7.pdf.

[23] A. M. Taqui, J. P. S. Alcocer, G. Hecht, and A. Bergel, "Quality Histories of Past Extract Method Refactorings," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12955 LNCS, pp. 336–352, 2021, doi: 10.1007/978-3-030-87007-2_24.

[24] J. Al Dallal, "Predicting move method refactoring opportunities in object-oriented code," *Inf. Softw. Technol.*, vol. 92, pp. 105–120, 2017, doi: 10.1016/j.infsof.2017.07.013.

[25] G. Bavota, A. De Lucia, A. Marcus, R. Oliveto, and F. Palomba, "Supporting extract class refactoring in eclipse: The ARIES project," *Proc. - Int. Conf. Softw. Eng.*, pp. 1419–1422, 2012, doi: 10.1109/ICSE.2012.6227233.