JACIS : Journal Automation Computer Information System Volume: **5**, Nomor **2**, **2025**, *Hlm*: 125-137

Evaluasi Kompleksitas, Kualitas, dan Efisiensi dalam Desain Perangkat Lunak Berbasis Obyek

Siti Munawaroh Yusminnu Zirah[1], Nadila Shifa Auria*[2], Muhammad Ainul Yaqin[3]

[1] Universitas Islam Negeri Maulana Malik Ibrahim Malang; 230605110198@student.uin-malang.ac.id
[*2] Universitas Islam Negeri Maulana Malik Ibrahim Malang; 230605110072@student.uin-malang.ac.id
[3] Universitas Islam Negeri Maulana Malik Ibrahim Malang; yaqinov@ti.uin-malang.ac.id

Abstrak: Perkembangan teknologi menuntut desain perangkat lunak yang lebih terstruktur, namun penerapan Desain Berorientasi Objek (OOD) menghadapi tantangan kompleksitas yang memengaruhi kualitas dan efisiensi, terutama pada proyek skala kecil dengan keterbatasan sumber daya. Penelitian ini mengusulkan model evaluasi kuantitatif untuk desain OOD dengan mengkaji hubungan antara kompleksitas, kualitas, dan efisiensi menggunakan metrik *Chidamber & Kemerer* (CK): *Weighted Methods per Class* (WMC), *Depth of Inheritance Tree* (DIT), *Number of Children* (NOC), dan *Coupling Between Objects* (CBO). Empat aplikasi Java skala kecil dianalisis secara manual. Hasil menunjukkan bahwa desain berkualitas tinggi dapat mengkompensasi kompleksitas tinggi dan meningkatkan efisiensi, sementara kualitas rendah menurunkan efisiensi meskipun kompleksitasnya sedang. Penelitian ini memberikan kontribusi dengan menawarkan model evaluasi yang dapat digunakan sebagai panduan dalam pengelolaan desain perangkat lunak di tahap awal, khususnya dalam pengembangan sistem skala kecil.

Keywords: desain; kompleksitas; kualitas; efisiensi;

DOI: 10.47134/jacis.v5i2.117
*Correspondensi: Nadila Shifa Auria
Email: 230605110072@student.uin-malang.ac.id

Receive: 28 Mei 2025
Accepted: 16 Juni 2025
Published: 15 Juli 2025

Abstrak: The rapid growth of technology requires more structured software designs, but the implementation of Object-Oriented Design (OOD) often faces challenges with complexity, which can affect both quality and efficiency, especially in small-scale projects with limited resources. This study proposes a quantitative evaluation model for OOD design by examining the relationship between complexity, quality, and efficiency using four key metrics from Chidamber & Kemerer (CK): Weighted Methods per Class (WMC), Depth of Inheritance Tree (DIT), Number of Children (NOC), and Coupling Between Objects (CBO). Four small-scale Java applications were analyzed manually. The results show that high-quality design can help manage high complexity and improve efficiency, while low-quality design tends to reduce efficiency, even with moderate complexity. This study provides a valuable evaluation model that can guide the management of software design from the early stages, particularly in small-scale system development

Keywords: design; complexity; quality; efficiency; software design.

PENDAHULUAN

Perkembangan teknologi informasi yang pesat dan meningkatnya kompleksitas sistem telah mendorong perlunya pendekatan desain perangkat lunak yang lebih adaptif, modular, dan terstruktur. Desain berorientasi objek (*Object-Oriented Design*/OOD) telah diakui secara luas sebagai pendekatan utama dalam pengembangan perangkat lunak modern, karena mendukung prinsip-prinsip modularitas, fleksibilitas, dan kemudahan dalam maintainability sistem[1]. Pendekatan ini memungkinkan pengembangan perangkat lunak yang dapat digunakan kembali (*reusable*), dapat ditingkatkan skalanya (*scalable*), dan lebih mudah dikelola dalam jangka panjang. Namun, penerapan OOD juga menghadapi tantangan tersendiri, terutama dalam pengelolaan struktur desain yang kompleks. Struktur sistem dengan banyak kelas, pewarisan bertingkat, dan hubungan objek yang kompleks dapat meningkatkan kesulitan dalam maintainability dan testability. Elemen-elemen desain seperti kedalaman hirarki pewarisan (*Depth of Inheritance Tree*/DIT), jumlah anak kelas (*Number of Children*/NOC), dan tingkat keterkaitan antar kelas (*Coupling Between Objects*/CBO) berkontribusi secara signifikan terhadap meningkatnya beban *maintainability*, waktu *testability* yang lebih lama, serta potensi penurunan stabilitas sistem secara keseluruhan[2][3]. Kompleksitas ini berdampak langsung pada kualitas desain perangkat lunak serta efisiensi proses pengembangan dalam hal waktu, tenaga, dan biaya.

Diperlukan pendekatan evaluatif yang komprehensif untuk menilai desain perangkat lunak secara kuantitatif dan objektif, terutama pada tahap awal pengembangan. Evaluasi semacam ini bertujuan untuk menjaga keseimbangan antara kompleksitas struktur, kualitas desain, dan efisiensi proses implementasi. Salah satu kerangka evaluasi yang telah banyak digunakan adalah Chidamber & Kemerer Metrics Suite, yang secara umum terdiri dari enam metrik utama. Namun, dalam konteks penelitian ini, evaluasi akan difokuskan pada empat metrik yang dianggap paling representatif dalam mencerminkan keterkaitan antara kompleksitas, kualitas, dan efisiensi, yaitu: *Weighted Methods per Class* (WMC), *Depth of Inheritance Tree* (DIT), *Number of Children* (NOC), dan *Coupling Between Objects* (CBO)[4][5]. Metrik WMC mengukur jumlah dan kompleksitas metode dalam sebuah kelas, yang berpengaruh pada tingkat beban fungsional. DIT dan NOC mengevaluasi tingkat hirarki dan sebaran pewarisan, yang berkaitan dengan tingkat generalisasi dan potensi *reuse*. Sementara itu, CBO digunakan untuk menilai tingkat keterkaitan antar kelas, yang berpengaruh pada derajat coupling serta usaha yang diperlukan untuk *testability* dan *maintainability*. Keempat metrik ini diyakini mampu memberikan representasi yang cukup untuk mengevaluasi desain perangkat lunak dari aspek kompleksitas struktural, kualitas desain, hingga efisiensi pengembangan sistem.

Beberapa penelitian sebelumnya telah memanfaatkan metrik-metrik ini dalam berbagai pendekatan. Paradis *et al.*[6] menerapkan metrik *Cognitive Weighted Inherited Class Complexity* (CWICC) untuk mengukur kompleksitas diagram kelas. Aunillah *et al.*[7] memvalidasi penerapan metrik CK dalam konteks desain berbasis objek, sedangkan Deter *et al.*[8] menggabungkan pendekatan heuristik untuk meningkatkan akurasi evaluasi kualitas desain. Hanifah *et al.*[5], Almogahed *et al.* [9] serta Poornima dan Suma[10] mengeksplorasi dampak refactoring terhadap nilai metrik dan kualitas sistem, serta menekankan pentingnya modularitas dan coupling dalam proses *testability* dan modifikasi sistem. Namun, sebagian besar studi tersebut masih menitikberatkan pada satu atau dua aspek metrik CK, tanpa mengintegrasikan ketiga aspek utama yaitu kompleksitas, kualitas, dan efisiensi secara

komprehensif. Selain itu, penelitian terdahulu umumnya belum mengarahkan penerapan metrik ini pada konteks pengembangan sistem skala kecil, yang notabene memiliki karakteristik keterbatasan sumber daya dan tuntutan efisiensi tinggi[3][8][5]. Dengan demikian, masih terdapat celah penelitian dalam pengembangan model evaluasi kuantitatif yang secara terpadu mengkaji kompleksitas, kualitas, dan efisiensi dalam konteks proyek perangkat lunak skala kecil.

Dengan latar belakang tersebut, penelitian ini bertujuan untuk mengevaluasi desain perangkat lunak berorientasi objek secara kuantitatif berdasarkan hubungan antara kompleksitas, kualitas, dan efisiensi menggunakan empat metrik utama dalam CK Metrics WMC, DIT, NOC, dan CBO. Penelitian ini diharapkan dapat membentuk sebuah model evaluasi desain yang menyeluruh, yang tidak hanya memberikan penilaian terhadap struktur desain saat ini, tetapi juga menjadi landasan prediktif dan sistematis untuk perbaikan desain sejak tahap awal pengembangan, khususnya dalam proyek berskala kecil.

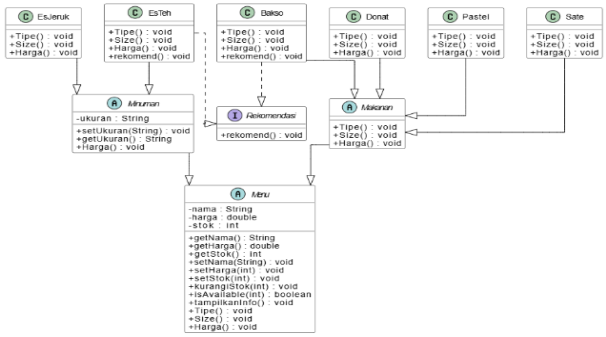
METODE

Data Yang Digunakan

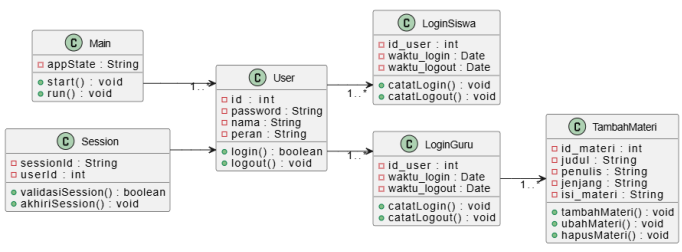
Data yang digunakan dalam penelitian ini diperoleh melalui pengumpulan proyek akhir mahasiswa yang dikembangkan dengan menggunakan bahasa pemrograman Java dan pendekatan berorientasi obyek. Aplikasi-aplikasi ini dipilih karena mewakili proyek skala kecil dalam konteks pendidikan tinggi. Beberapa aplikasi yang dianalisis diantaranya:

1. Aplikasi Tiket Konser
   Terpilih karena mewakili aplikasi berbasis transaksi dalam industri hiburan, dengan kompleksitas manajerial yang terkelola dengan baik.
2. Aplikasi Pemesanan Restoran
   Terpilih karena mewakili aplikasi layanan dengan interaksi pengguna yang melibatkan pengelolaan data dan transaksi.
3. Aplikasi Belajar Online
   Terpilih karena mewakili sektor pendidikan dengan desain yang mendukung aksesibilitas dan interaksi pengguna.
4. Aplikasi Pemesanan Tiket Kereta Api
   Terpilih dipilih karena mengelola pemesanan dalam sektor transportasi, dengan fokus pada efisiensi sistem pemesanan

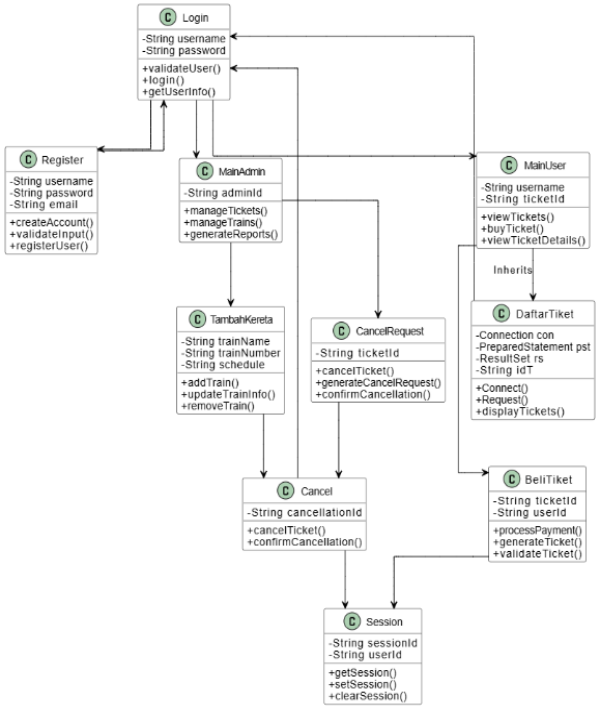
Gambar 1 merupakan class diagram pad amasing-masing aplikasi yang akan dianalisis.



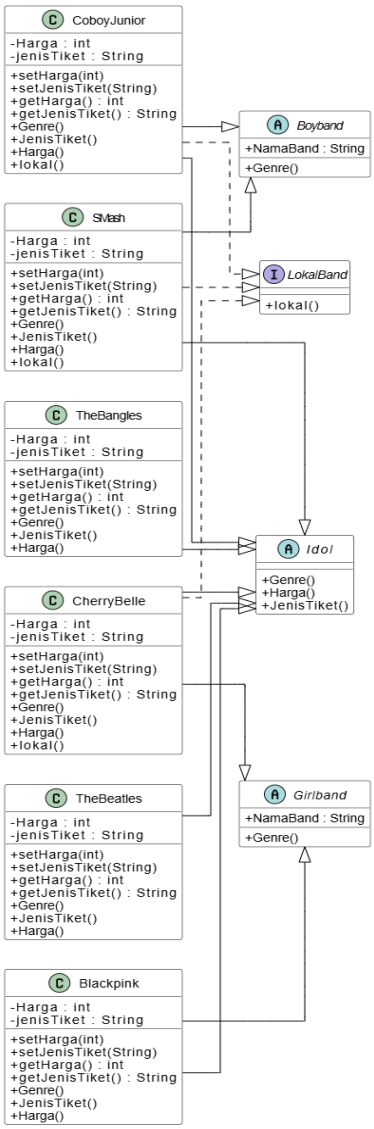
Gambar 1. Diagram *Class* Aplikasi Pemesanan Restoran



Gambar 2. Diagram *Class* Aplikasi Pemesanan Restoran



Gambar 3. Diagram *Class* Aplikasi Kereta Api



Gambar 4. Diagram *Class* Aplikasi Tiket Konser

Desain Penelitian

Dalam penelitian ini, pendekatan manual digunakan untuk menghitung kompleksitas desain perangkat lunak yang berorientasi objek dengan fokus pada empat metrik utama dari *Chidamber & Kemerer Metrics*: *Weighted Methods per Class* (WMC), *Depth of Inheritance Tree* (DIT), *Number of Children* (NOC), dan *Coupling Between Objects* (CBO). Pengukuran dilakukan terhadap sejumlah proyek mahasiswa tanpa menggunakan alat otomatis seperti CKJM atau SonarQube. Tujuan dari pendekatan ini adalah untuk mengevaluasi akurasi nilai metrik serta memahami secara langsung hubungan antara elemen desain dan tingkat kompleksitas. Selain mengisi kekosongan dalam studi di bidang pendidikan, metode ini juga memungkinkan identifikasi akar penyebab kompleksitas dengan lebih tepat, sehingga mendukung pengambilan keputusan desain yang efisien dan efektif dalam pengembangan perangkat lunak berskala kecil.

Tabel 1. Hubungan Metrik *Chidamber & Kemerer* (CK) dengan 3 Faktor Kompleksitas, kualitas dan efisien pada Perangkat Lunak

Metrik	Maintainability	Reusability	Understandability	Testability	Hasil Gabungn Aspek
WMC	√	√	√	√	Menurunkan efisiensi karena meningkatkan kompleksitas kelas dan beban *maintainability*.
DIT	√	√	√	√	Meningkatkan efisiensi melalui modularitas dan penggunaan ulang kode.
NOC	X	√	X	√	Meningkatkan efisiensi dalam penggunaan ulang, namun bisa mempersulit pengelolaan jika berlebihan.
CBO	√	√	√	√	Menurunkan efisiensi akibat tingginya ketergantungan antar kelas.

Identifikasi Metrik

Perhitungan menggunakan WMC (*Weighted Methods per Class*)

Perhitungan menggunakan WMC (*Weighted Methods per Class*) bertujuan untuk mengukur jumlah metode dalam sebuah *class*. Metrik ini pertama kali diperkenalkan oleh Chidamber & Kemerer[4]. Semakin banyak metode yang dimiliki suatu *class*, semakin tinggi nilai kompleksitasnya. Persamaan (1) digunakan untuk menghitung WMC.

$$WMC = \sum_{i=1}^{n} c_i \quad (1)$$

Dimana :

n adalah jumlah metode dalam class.

c_i adalah kompleksitas dari metode ke-i (sering kali dianggap 1 jika tidak dihitung secara terpisah).

Nilai WMC yang tinggi menunjukkan tingkat kompleksitas yang tinggi, yang dapat berdampak negatif pada aspek maintainability, testability, dan reusability[11]. Kategori penilaian untuk WMC adalah sebagai berikut[12]:

Baik : $1 \leq x < 50$
Sedang : $50 \leq x \leq 150$
Buruk : $x > 150$

Perhitungan menggunakan DIT (*Depth of Inheritance Tree)*

Mengukur kedalaman hierarki pewarisan dari suatu *class* hingga mencapai akar (*root*) dapat dilakukan dengan menggunakan DIT (*Depth of Inheritance Tree*)[4]. Persamaan (2) adalah persamaan yang digunakan untuk menghitung DIT

DIT = jumlah level *inheritance* dari *class* sampai ke *root* (2)

Nilai DIT yang dalam (tinggi) menunjukkan tingkat abstraksi dan potensi reusability yang tinggi. Namun, nilai yang berlebihan dapat membuat sistem menjadi sangat kompleks, sulit dipahami (*understandability*), dan sulit dimodifikasi (*modifiability*), karena perubahan pada kelas induk di level atas dapat memengaruhi banyak sekali kelas turunan di bawahnya[4]. Kategori penilaian yang sering diterapkan untuk NOC adalah sebagai berikut[7][11]:

Baik : $x < 5$
Sedang : $5 \leq x < 8$
Buruk : $x \geq 8$

Perhitungan menggunakan NOC (*Number of Children*)

Perhitungan menggunakan NOC (*Number of Children*) bertujuan untuk mengukur jumlah *class* anak (*subclass*) yang langsung berasal dari suatu *class* induk[4]. Persamaan (3) digunakan untuk menghitung NOC.

NOC = jumlah *subclass* langsung dari suatu *class* (3)

Nilai NOC yang tinggi dapat meningkatkan *reusability*, namun juga dapat menambah kompleksitas dalam *testability* dan *maintainability.* Hal ini disebabkan oleh fakta bahwa perubahan yang dilakukan pada *class* induk dapat berdampak pada *subclass* yang ada. Kategori penilaian untuk NOC adalah sebagai berikut[7][11]

Baik : $x < 4$
Sedang : $4 \leq x < 8$
Buruk : $x \geq 8$

Perhitungan menggunakan CBO (*Coupling Between Objects*)

Perhitungan menggunakan CBO (*Coupling Between Objects*) bertujuan untuk mengukur jumlah *class* lain yang berhubungan dengan suatu *class*, baik melalui pemanggilan metode maupun penggunaan atribut[4]. Persamaan (4) merupakan rumus untuk menghitung CBO.

CBO = jumlah *class* lain yang digunakan oleh suatu *class* (4)

Nilai CBO yang tinggi menunjukkan adanya tingkat *coupling* yang tinggi, yang dapat menurunkan *modularity* dan menyulitkan *testing* serta *reusability*. Kategori penilaian untuk CBO adalah sebagai berikut[4][12]

Baik : $x < 14$
Sedang : $14 \leq x \leq 150$
Buruk : $x > 150$

Berdasarkan Gambar 1 sampai dengan Gambar 4, serta Persamaan (1) sampai dengan (4) diketahui bahwa nilai masing-masing *Weighted Methods per Class* (WMC), *Depth of Inheritance Tree* (DIT), *Number of Children* (NOC), dan *Coupling Between Object Classes* (CBO) untuk keempat aplikasi tersebut terdapat pada Tabel 2.

Tabel 2. Hasil Perhitungan Metrik WMC, DIT, NOC, dan CBO.

Nama Aplikasi	n	ci	WMC	DIT	NOC	CBO
APLIKASI PEMESANAN RESTORAN	39	10	39	14	8	14
APLIKASI TIKET KONSER	41	10	41	9	8	9
APLIKASI BELAJAR ONLINE	79	16	79	16	12	16

Nama Aplikasi	n	ci	WMC	DIT	NOC	CBO
APLIKASI PEMESANAN KERETA API	36	10	36	15	5	15

Dari empat aplikasi berbeda yang dibandingkan dari sisi kompleksitas desain berorientasi objek, maka dapat diketahui bahwa:

1. Aplikasi Belajar Online memiliki jumlah kelas dan nilai metrik paling tinggi di hampir semua aspek, hal ini menunjukkan aplikasi ini memiliki desain yang paling kompleks.
2. Sebaliknya, Aplikasi Pemesanan Kereta Api memiliki jumlah kelas dan kompleksitas yang relatif lebih rendah.

HASIL DAN PEMBAHASAN

Hasil Pengukuran Kompleksitas Desain Perangkat Lunak

Untuk menilai kompleksitas desain perangkat lunak berbasis objek, dilakukan pengukuran terhadap empat aplikasi menggunakan metrik CK yang meliputi *Weighted Methods per Class* (WMC), *Depth of Inheritance Tree* (DIT), *Number of Children* (NOC), dan *Coupling Between Object Classes* (CBO). Nilai-nilai dari masing-masing metrik ini selanjutnya digunakan sebagai indikator kompleksitas struktural dalam arsitektur berorientasi objek. Kategori kompleksitas ditentukan berdasarkan akumulasi nilai dari empat metrik utama. Aplikasi dikategorikan Tinggi apabila minimal tiga dari empat metrik memiliki nilai signifikan (WMC > 60, DIT > 15, atau CBO > 20), sedangkan nilai-nilai yang berada pada rentang sedang dikelompokkan dalam kategori Sedang. Tabel 3 merupakan hasil evaluasi kompleksitas pada aplikasi.

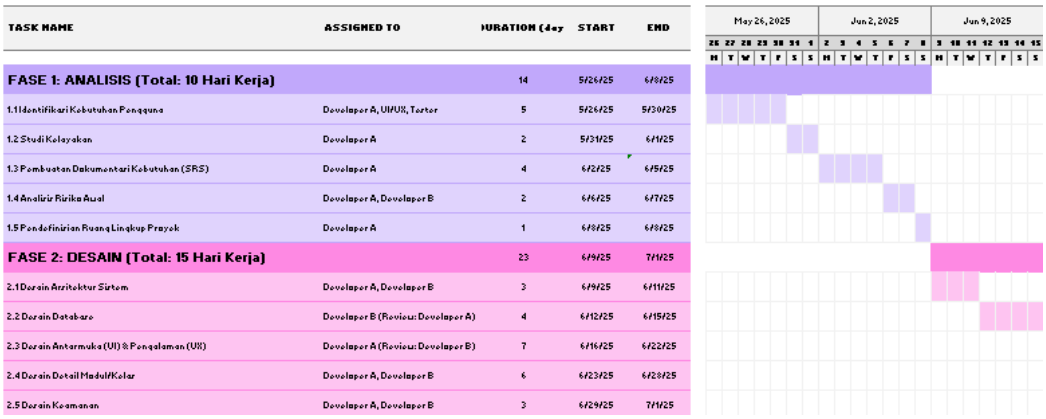
Tabel 3. Hasil Evaluasi Kompleksitas pada Aplikasi

Nama Aplikasi	WMC	DIT	NOC	CBO	Kategori
APLIKASI PEMESANAN RESTORAN	39	14	8	16	Sedang
APLIKASI TIKET KONSER	41	9	8	16	Sedang
APLIKASI BELAJAR ONLINE	79	16	12	24	Tinggi
APLIKASI PEMESANAN KERETA API	36	15	5	14	Sedang

Berdasarkan Tabel 3 diketahui bahwa Aplikasi Belajar *Online* memiliki skor tertinggi pada metrik WMC, DIT, NOC, dan CBO, yang mengindikasikan tingkat kompleksitas yang cukup tinggi. Hal ini sejalan dengan desain yang lebih besar dan lebih rumit, yang dapat menghambat *maintainability* dan *understandability code*, tetapi dapat memberikan keuntungan dalam hal *reusabilitas* dan penggunaan kembali jika diterapkan dengan baik. Sebaliknya, aplikasi lainnya, seperti Aplikasi Pemesanan Kereta Api, menunjukkan kompleksitas yang lebih moderat dengan skor WMC dan CBO yang lebih rendah, namun masih cukup efektif dalam pengelolaan kode. Skor Aplikasi Pemesanan Restoran dan Aplikasi Tiket Konser menunjukkan tingkat kompleksitas yang sedang, yang mengindikasikan keseimbangan yang lebih baik antara modularitas dan pengelolaan kode yang lebih sederhana.

Hasil evaluasi kualitas desain pada empat aplikasi berdasarkan nilai metrik WMC dan CBO terdapat pada Tabel 4. Dimana nilai kualitas dihitung dengan mempertimbangkan tingkat kompleksitas metode dalam kelas serta tingkat keterkaitan antar kelas. Dari segi kualitas, Aplikasi Pemesanan Kereta Api menunjukkan nilai kualitas yang paling rendah, meskipun memiliki skor WMC dan CBO yang masih dalam batas wajar. Hal ini mungkin

menunjukkan adanya masalah lain dalam desain yang mempengaruhi kohesi dan integrasi antar komponen sistem. Sebaliknya, Aplikasi Tiket Konser dan Aplikasi Belajar *Online* menunjukkan kualitas yang lebih baik, meskipun keduanya memiliki WMC dan CBO yang lebih tinggi. Kualitas ini berkaitan dengan desain yang lebih modular dan efisiensi dalam pengelolaan kode yang mengurangi potensi masalah pada *testability* dan *maintainability*. Aplikasi Pemesanan Restoran menunjukkan kualitas yang sedang, yang menandakan ada keseimbangan antara kemudahan pengelolaan kode dan penggunaan kembali yang terbatas.



Gambar 5. Contoh Jadwal Pengembangan Aplikasi

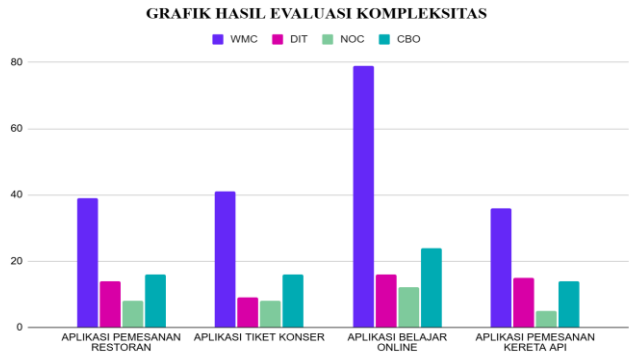
Gambar 5 menyajikan contoh jadwal pengembangan aplikasi yang menjadi acuan dalam mengukur efisiensi pengembangan pada Tabel 5, mencakup alokasi waktu dan sumber daya personel. Hal ini sejalan dengan pandangan rekayasa perangkat lunak yang menekankan bahwa estimasi dan alokasi sumber daya merupakan komponen kunci dalam keberhasilan proyek[13]. Efisiensi yang diukur kemudian menjadi indikator bagaimana sumber daya yang dialokasikan mampu mengatasi kompleksitas desain selama proses pengembangan.

Tabel 5. Hasil Evaluasi Efisien pada Aplikasi

Nama Aplikasi	Durasi Pengembangan (Bulan)	Jumlah Personel (Orang)	Jumlah Durasi per orang (Bulan)	Kategori
APLIKASI PEMESANAN RESTORAN	5	4	1,25	Tinggi
APLIKASI TIKET KONSER	5	3	1,67	Sedang
APLIKASI BELAJAR ONLINE	5	4	1,25	Tinggi
APLIKASI PEMESANAN KERETA API	7	5	1,40	Sedang

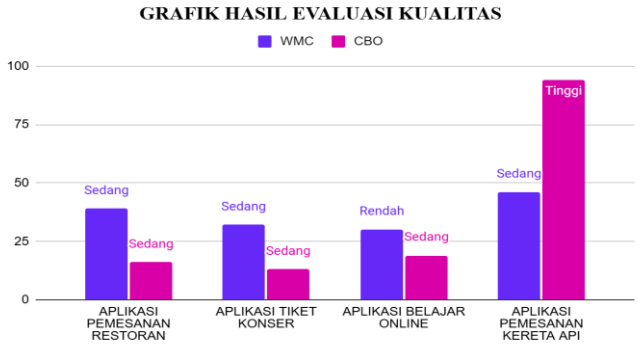
Dalam evaluasi efisiensi, Aplikasi Pemesanan Restoran dan Aplikasi Belajar *Online* menunjukkan efisiensi yang tinggi, dengan durasi pengembangan yang lebih pendek dan distribusi durasi per personel yang lebih efisien. Kedua aplikasi ini memiliki tim pengembang yang lebih kecil, tetapi mampu mengoptimalkan waktu pengembangan. Sebaliknya, Aplikasi Tiket Konser dan Aplikasi Pemesanan Kereta Api memiliki durasi pengembangan yang sedikit lebih lama, dengan Aplikasi Pemesanan Kereta Api memiliki nilai yang lebih rendah terkait efisiensi pengelolaan personel. Ini mengindikasikan adanya peningkatan upaya dalam pengelolaan kompleksitas atau masalah yang dihadapi selama pengembangan.

Berdasarkan hasil evaluasi kompleksitas, kualitas, dan efisiensi, Aplikasi Belajar Online menunjukkan kompleksitas tertinggi berdasarkan metrik WMC, DIT, NOC, dan CBO yang melampaui batas wajar[1], yang berpotensi menghambat *understandability* dan *maintainability* kode[12]. Namun, kompleksitas ini dikompensasi oleh efisiensi tinggi dan kualitas desain yang baik, sebagaimana tercermin dari nilai kualitas dan durasi pengembangan per personel yang rendah. Temuan ini diperkuat oleh[14] yang menunjukkan bahwa penggunaan perangkat lunak berbasis AST parser dan metrik *Cyclomatic Complexity* mampu membantu pengembang mengidentifikasi kompleksitas kritis sejak tahap awal desain. Hal ini sejalan dengan literatur yang menyatakan bahwa kualitas desain tidak hanya ditentukan oleh banyaknya metode, tetapi juga oleh tingkat coupling antar kelas[3]. Sebaliknya, meskipun metrik WMC dan CBO pada Aplikasi Pemesanan Kereta Api masih dalam batas wajar, nilai kualitasnya terendah, menunjukkan kemungkinan adanya masalah pada aspek desain lain yang memengaruhi kohesi sistem[8][9], serta berkontribusi pada efisiensi yang hanya tergolong sedang, meskipun memiliki sumber daya lebih besar. Sementara itu, aplikasi lainnya menunjukkan kompleksitas lebih terkendali, dengan nilai CBO di bawah 20 yang masih dalam batas modularitas yang baik[10][14], serta pewarisan aktif yang tetap fungsional dan dapat dikelola. Temuan ini mempertegas bahwa kualitas desain berperan signifikan dalam menentukan efisiensi pengembangan, terutama pada tahap konstruksi dan *testability*[5][10]. Hal ini sejalan dengan prinsip dalam rekayasa perangkat lunak, yang menekankan bahwa metrik kompleksitas struktural seperti CBO dan DIT sangat berperan dalam memprediksi tingkat kesulitan *maintainability* dan *understandability* sistem[15].



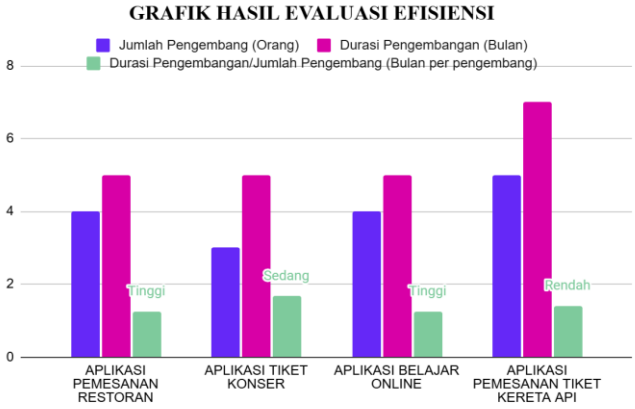
Gambar 5. Grafik Hasil Evaluasi Kompleksitas pada Aplikasi

Berdasarkan hasil evaluasi menyeluruh terhadap dimensi efisiensi, kompleksitas, dan kualitas perangkat lunak, Aplikasi Belajar Online menunjukkan kompleksitas tertinggi. Meskipun demikian, aplikasi ini mencapai kualitas "Tinggi" dan efisiensi "Tinggi".Sebaliknya, Aplikasi Pemesanan Kereta Api, dengan kompleksitas "Sedang", memiliki kualitas "Rendah" dan efisiensi "Sedang", meskipun dengan alokasi sumber daya yang lebih besar. Hal ini mengindikasikan bahwa efisiensi merupakan hasil dari keseimbangan antara kompleksitas dan kualitas desain, di mana desain berkualitas tinggi dapat mengkompensasi kompleksitas struktural, sementara kualitas desain yang rendah akan selalu berdampak negatif pada efisiensi pengembangan. Penelitian ini sejalan dengan studi sebelumnya yang mengukuhkan bahwa metrik desain seperti metrik CK (WMC, DIT, NOC, CBO) tidak hanya berfungsi sebagai indikator kualitas retrospektif, tetapi juga sebagai alat prediktif strategis untuk mendorong desain yang efisien, adaptif, dan berkelanjutan.



Gambar 6. Grafik Hasil Evaluasi Kualitas pada Aplikasi

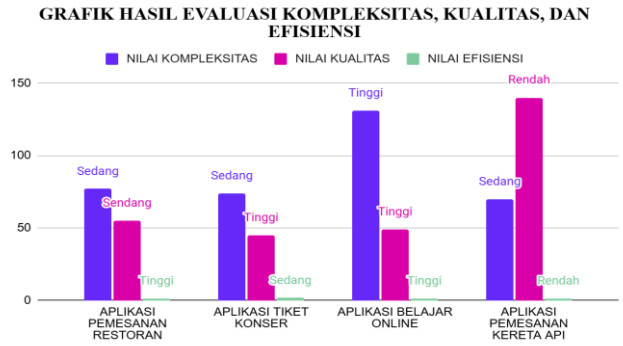
Berdasarkan hasil evaluasi kualitas aplikasi, terlihat bahwa kualitas perangkat lunak sangat bergantung pada keseimbangan antara kompleksitas dan efisiensi desain. Meskipun Aplikasi Belajar Online memiliki kompleksitas desain yang tinggi, aplikasi ini berhasil mencapai kategori kualitas "Tinggi".Hal ini disebabkan oleh desain yang matang dan penerapan prinsip *Object-Oriented Programming* (OOP) yang optimal, menghasilkan reusabilitas dan modularitas yang tinggi. Sebaliknya, Aplikasi Pemesanan Kereta Api, dengan kompleksitas "Sedang", menunjukkan kualitas "Rendah". Data ini menetapkan bahwa kompleksitas tinggi bukanlah penghalang efisiensi jika didukung oleh desain berkualitas, sementara kualitas rendah akan selalu berdampak negatif pada efisiensi pengembangan. Temuan ini konsisten dengan studi sebelumnya yang menunjukkan bahwa metrik desain seperti metrik CK bukan hanya indikator kualitas retrospektif, tetapi juga alat prediktif strategis untuk mendorong desain yang efisien dan adaptif. Ini menegaskan bahwa kualitas perangkat lunak adalah hasil dari keseimbangan antara kompleksitas dan kualitas desain, di mana desain berkualitas tinggi dapat mengkompensasi kompleksitas struktural, sementara kualitas desain yang rendah akan selalu berdampak negatif. Penelitian ini sejalan dengan studi sebelumnya yang mengukuhkan bahwa metrik desain (seperti metrik CK) bukan hanya indikator kualitas retrospektif, tetapi juga alat prediktif strategis untuk mendorong desain yang efisien, adaptif, dan berkelanjutan.



Gambar 7. Grafik Hasil Evaluasi Efisiensi pada Aplikasi

Berdasarkan hasil evaluasi efisiensi aplikasi secara komprehensif mengilustrasikan bahwa efisiensi perangkat lunak adalah konsekuensi langsung dari interaksi antara kompleksitas dan kualitas desain yang dievaluasi. Efisiensi "Tinggi" pada Aplikasi Belajar *Online* (1.25 bulan/orang) disebabkan oleh kualitasnya yang superior, meskipun dengan kompleksitas

tinggi, berkat desain yang optimal dan penerapan prinsip OOP; sebaliknya, Aplikasi Pemesanan Kereta Api yang berkualitas rendah namun berkompleksitas sedang, menunjukkan efisiensi "Sedang" (1.40 bulan/orang) karena memerlukan sumber daya lebih untuk *maintainability* dan *testability*. Ketidakseimbangan ini memperkuat pandangan bahwa efisiensi bersumber dari keseimbangan optimal antara kompleksitas dan kualitas desain, sejalan dengan literatur yang menekankan peran modularitas dan *low coupling* dalam sistem kompleks dan dinamis[16], serta menegaskan bahwa metrik desain merupakan prediktor strategis untuk pengembangan yang efisien dan adaptif[3][8][12].



Gambar 8. Grafik Hasil Evaluasi Kompelsitas, Kualitas dan Efisiensi pada Aplikasi

Berdasarkan hasil evaluasi gabungan kompleksitas, kualitas dan efisiensi merepresentasikan hasil evaluasi komprehensif yang menunjukkan bahwa ketiganya saling terkait erat dan tidak dapat dinilai secara terpisah. Berdasarkan data yang disajikan, Aplikasi Belajar Online meskipun memiliki kompleksitas tertinggi, mencapai kualitas dan efisiensi "Tinggi" disebabkan oleh desain yang matang dan optimalisasi prinsip OOP, seperti reusabilitas dan modularitas. Sebaliknya, Aplikasi Pemesanan Kereta Api dengan kompleksitas sedang, menunjukkan kualitas dan efisiensi "Sedang", bahkan cenderung "Rendah" dalam kualitas yang mengindikasikan perlunya alokasi sumber daya lebih besar untuk *maintainability* dan *testability*. Ketidakseimbangan ini menegaskan bahwa efisiensi merupakan hasil dari keseimbangan optimal antara kompleksitas dan kualitas desain, di mana desain berkualitas tinggi dapat mengkompensasi kompleksitas, sedangkan kualitas yang rendah akan selalu berdampak negatif pada efisiensi pengembangan. Temuan ini didukung oleh literatur yang ada, yang menggarisbawahi bahwa metrik desain berfungsi sebagai alat prediktif strategis untuk mendorong desain yang efisien, adaptif, dan berkelanjutan, serta menekankan peran modularitas dan *low coupling* dalam sistem yang kompleks dan dinamis. Untuk mendukung transparansi dan replikasi analisis, hasil evaluasi lengkap yang mencakup metrik kompleksitas, kualitas, dan efisiensi dari masing-masing aplikasi yang diuji dapat diakses melalui tautan berikut: [bit.ly/HasilEksperimenPengembangan]

SIMPULAN

Penelitian ini menegaskan pentingnya penerapan metrik CK yang mencakup WMC, DIT, NOC, dan CBO dalam mengevaluasi desain perangkat lunak berorientasi objek secara komprehensif, khususnya dalam konteks kompleksitas, kualitas, dan efisiensi. Temuan menunjukkan bahwa kompleksitas tinggi tidak selalu menghambat efisiensi pengembangan apabila struktur pewarisan dan reusabilitas kode dikelola dengan coupling yang rendah. Sebaliknya, tingginya coupling antar kelas dapat menurunkan kualitas desain dan memperlambat proses pengembangan, meskipun tingkat kompleksitasnya tidak tergolong

ekstrem. Hal ini memperkuat peran indikator CBO sebagai elemen krusial dalam maintainability sistem. Dengan demikian, studi ini memberikan kontribusi signifikan dalam mendorong praktik evaluasi desain sejak tahap awal pengembangan, terutama untuk proyek berskala kecil, guna memastikan sistem yang modular dan efisien. Di masa mendatang, disarankan adanya perluasan eksperimen dengan melibatkan lebih banyak jenis proyek perangkat lunak dan pendekatan perbandingan dengan alat bantu otomatis guna mengkaji akurasi serta efektivitas evaluasi desain secara lebih luas dan terintegrasi.

DAFTAR PUSTAKA

[1] T. G. S. Filó, M. A. S. Bigonha, and K. A. M. Ferreira, "Evaluating Thresholds for Object-Oriented Software Metrics," *J. Brazilian Comput. Soc.*, vol. 30, no. 1, pp. 313–346, 2024, doi: 10.5753/jbcs.2024.3373.

[2] E. Purwawijaya, "Kompleksitas Fungsional Perangkat Lunak Menggunakan Serangkaian Kriteria Baru dalam Unified Modeling Language (UML)," *J. Minfo Polgan*, vol. 13, no. 1, pp. 271–277, 2024, doi: 10.33395/jmp.v13i1.13623.

[3] A. A. Mir, M. Raees, and A. Ahmed, "Object Oriented-Based Metrics to Predict Fault Proneness in Software Design," *arXiv Is Hiring a DevOps Eng.*, 2025, [Online]. Available: http://arxiv.org/abs/2504.08230

[4] S. R. Chidamber and C. F. Kemerer, "Towards a metrics suite for object oriented design," *ACM SIGPLAN Not.*, vol. 26, no. 11, pp. 197–211, 1991, doi: 10.1145/118014.117970.

[5] H. Hanifah, A. R. Irawati, and Y. T. Utami, "Jurnal Pepadun Implementasi Pengukuran Object Oriented Metrics (Studi Kasus Aplikasi Movie DB)," *J. Pepadun*, vol. 5, no. 3, pp. 238–248, 2024, doi: 10.23960/pepadun.v5i3.238.

[6] C. N. Paradis, M. R. Yusuf, M. Farhanudin, and M. A. Yaqin, "Analisis dan Perancangan Software Pengukuran Metrik Skala dan Kompleksitas Diagram Class," *J. Autom. Comput. Inf. Syst.*, vol. 2, no. 1, pp. 58–65, 2022, doi: 10.47134/jacis.v2i1.40.

[7] K. N. Aunillah, A. Roihan, H. D. Ribilanam, and M. A. Yaqin, "Metrik Kompleksitas Software Berorientasi Objek," *Ilk. J. Comput. Sci. Appl. Informatics*, vol. 3, no. 1, pp. 103–110, 2021, doi: 10.28926/ilkomnika.v3i1.128.

[8] H. Deters, J. Droste, and K. Schneider, "A Means to what End? Evaluating the Explainability of Software Systems using Goal-Oriented Heuristics," *ACM Int. Conf. Proceeding Ser.*, pp. 329–338, 2023, doi: 10.1145/3593434.3593444.

[9] 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, no. July, pp. 99143–99158, 2023, doi: 10.1109/ACCESS.2023.3313186.

[10] U. S. Poornima, V. Suma, and H. Vasanth Kumar, "Design Patterns as Quality Influencing Factor in Object Oriented Design Approach," *Dayananda Sagar Institutions*, no. February, pp. 3–6, 2014, doi: https://doi.org/10.48550/arXiv.1402.2372.

[11] J. Al Dallal and B. Alkhazi, "Exploring the Impact of Alternatives of Object-

Oriented Cohesion Measures on Machine Learning-Based Predictions of Inheritance Reusability," *IEEE Access*, vol. 12, no. October, pp. 159252–159266, 2024, doi: 10.1109/ACCESS.2024.3484007.

[12] M. K. Sharma, S. Ranjan, and A. Gupta, "Object-Oriented Metrics for Quality Improvement of Object-Oriented Software," *Int. J. Innov. Technol. Explor. Eng.*, vol. 9, no. 3, pp. 26–29, 2020, doi: 10.35940/ijitee.b7394.019320.

[13] O. Cico, L. Jaccheri, A. Nguyen-Duc, and H. Zhang, "Exploring the intersection between software industry and Software Engineering education - A systematic mapping of Software Engineering Trends," *J. Syst. Softw.*, vol. 172, p. 110736, 2021, doi: 10.1016/j.jss.2020.110736.

[14] R. B. Mulkan Ghifari, S. Fitri, A. Ardhyandoko, and M. Ainul Yaqin, "Analisis dan Perancangan Software Pengukuran Matriks Skala dan Kompleksitas Kode Program," *J. Autom. Comput. Inf. Syst.*, vol. 4, no. 1, pp. 42–49, 2024, doi: 10.47134/jacis.v4i1.72.

[15] X. Hou *et al.*, "Large Language Models for Software Engineering: A Systematic Literature Review," *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 8, 2024, doi: 10.1145/3695988.

[16] A. F. Lukmana, P. A. Wiratama, R. Ibrahim, and M. A. Yaqin, "Survey Metrik Skala dan Kompleksitas Sistem Berorientasi Service Metrics Scale Survey and Complexity of Oriented System Service," *JACIS J. Autom. Comput. Inf. Syst.*, vol. 3, no. 1, pp. 37–43, 2023, doi: 10.47134/jacis.v3i1.41.