PENGUKURAN *OBJECT ORIENTED CONCEPTS* PADA APLIKASI MOVIE DB MENGGUNAKAN *CK METRICS SUITE*

(Skripsi)

Oleh

HARTSA HANIFAH

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2023

ABSTRAK

PENGUKURAN *OBJECT ORIENTED CONCEPTS* PADA APLIKASI MOVIE DB MENGGUNAKAN *CK METRICS SUITE*

Oleh

HARTSA HANIFAH

Pembuatan atau pengembangan perangkat lunak memiliki salah satu tujuan yaitu menciptakan perangkat lunak berkualitas. Saat ini, pengembang perangkat lunak lebih cenderung menggunakan konsep *Object Oriented Programming* (OOP) yang mempermudah pembangunan perangkat lunak. Perkembangan teknologi perangkat lunak yang terstruktur juga berpengaruh pada pengembangan penelitian di bidang metrik perangkat lunak berorientasi objek. Chidamber dan Kemerer mengusulkan enam metrik yang dapat digunakan dalam konteks berorientasi objek, seperti *Weight Methods 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 Methods* (LCOM).

Dalam penelitian ini, *source code* setiap *clas*s Java dalam Aplikasi Movie DB dianalisis, dan dilakukan perhitungan metrik berorientasi objek terhadap semua *class* dalam Aplikasi Movie DB menggunakan *CK-Metrics Suite*. Selanjutnya, nilai metrik Chidamber & Kemerer pada setiap *class* dibandingkan dengan indikator metrik untuk menentukan kualitas konsep berorientasi objek yang terdapat dalam kode perangkat lunak tersebut.

Hasil penelitian menunjukkan bahwa konsep berorientasi objek dalam Aplikasi Movie DB memiliki tingkat kaidah-kaidah yang baik dalam hal *maintainability, usability, reusability, understandability, modifiability,* dan *testability*.

Kata Kunci: *CK Metrics Suite*, Faktor Kualitas, Metrik Chidamber & Kemerer, *Object Oriented Metrics.*

ABSTRACT

MEASUREMENT OF OBJECT ORIENTED CONCEPTS IN MOVIE DB APPLICATION USING CK METRICS SUITE

By

HARTSA HANIFAH

The creation or development of software has one of its goals, which is to create high-quality software. Currently, software developers tend to use the Object-Oriented Programming (OOP) concept, which facilitates software development. The structured development of software technology also influences research in the field of object-oriented software metrics. Chidamber and Kemerer proposed six metrics that can be used in the context of object-oriented programming, such as Weight Methods per Class (WMC), Depth of Inheritance Tree (DIT), Number of Children (NOC), Coupling Between Objects (CBO), Response for a Class (RFC), and Lack of Cohesion in Methods (LCOM).

In this study, the source code of each Java class in Movie DB Application is analyzed, and object-oriented metrics are calculated for all classes in Movie DB Application using the CK-Metrics Suite. Subsequently, the Chidamber & Kemerer metrics values for each class are compared with metric indicators to determine the quality of the object-oriented concepts present in the software code.

The research results indicate that the object-oriented concepts in Movie DB Application adhere well to the principles of maintainability, usability, reusability, understandability, modifiability, and testability.

Keywords: *CK Metrics Suite, Quality Factor, Chidamber & Kemerer Metrics, Object Oriented Metrics.*

PENGUKURAN *OBJECT ORIENTED CONCEPTS* PADA APLIKASI MOVIE DB MENGGUNAKAN *CK METRICS SUITE*

Oleh

HARTSA HANIFAH

Skripsi

Sebagai Salah Satu Syarat untuk Mencapai Gelar
SARJANA KOMPUTER

Pada

Jurusan Ilmu Komputer
Fakultas Matematika dan Ilmu Pengetahuan Alam

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2023

Judul Skripsi	: **PENGUKURAN *OBJECT ORIENTED CONCEPTS* PADA APLIKASI MOVIE DB MENGGUNAKAN *CK METRICS SUITE***
Nama Mahasiswa	: **Hartsa Hanifah**
Nomor Pokok Mahasiswa	: 1617051029
Jurusan	: Ilmu Komputer
Fakultas	: Matematika dan Ilmu Pengetahuan Alam

MENYETUJUI

1. Komisi Pembimbing

Anie Rose Irawati, S.T., M.Cs.
NIP 19791031 200604 2 002

Yohana Tri Utami, S.Kom., M.Kom.
NIP 19900110 201903 2 010

2. Ketua Jurusan Ilmu Komputer

Didik Kurniawan, S.Si., M.T.
NIP 19800419 200501 1 004

MENGESAHKAN

1. Tim Penguji

Ketua : **Anie Rose Irawati, S.T., M.Cs.** .....

Sekretaris : **Yohana Tri Utami, S.Kom., M.Kom.** .....

Penguji : **Dr. rer. nat. Akmal Junaidi, M.Sc.** .....

2. Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

Dr. Eng. Heri Satria, S.Si., M.Si.
NIP 19711001 200501 1 002

Tanggal Lulus Ujian Skripsi : **19 Juni 2023**

PERNYATAAN

Saya yang bertanda tangan di bawah ini, menyatakan bahwa skripsi saya yang berjudul **"Pengukuran *Object Oriented Concepts* pada Aplikasi Movie DB Menggunakan *CK Metrics Suite*"** merupakan karya saya sendiri dan bukan karya orang lain. Semua tulisan yang tertuang dalam skripsi ini telah mengikuti kaidah penulisan karya ilmiah Universitas Lampung. Apabila di kemudian hari terbukti skripsi saya merupakan hasil salinan atau dibuat orang lain, maka saya bersedia menerima sanksi berupa pencabutan gelar yang telah saya terima.

Bandar Lampung, 19 Juni 2023

HARTSA HANIFAH
NPM. 1617051029

RIWAYAT HIDUP

Penulis dilahirkan di Bandar Lampung pada tanggal 26 November 1998, sebagai anak kedua dari tiga bersaudara, dari Bapak Drs. Mukhaidori dan Ibu Umi Ipadah, S.Pd.

Pendidikan Taman Kanak-kanak (TK) ditempuh di TK Citra Melati diselesaikan pada tahun 2004. Sekolah Dasar (SD) ditempuh di SD Negeri 3 Gedung Air, Bandar Lampung pada tahun 2004 – 2007, MIN 1 Bandar Lampung pada tahun 2007 – 2010. Sekolah Menengah Pertama (SMP) ditempuh di MTs Negeri 1 Bandar Lampung pada tahun 2010 – 2013. Dan Sekolah Menengah Atas (SMA) ditempuh di MAN 2 Bandar Lampung pada tahun 2013 – 2016.

Tahun 2016, penulis terdaftar sebagai mahasiswa Jurusan Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung melalui Seleksi Nasional Masuk Perguruan Tinggi Negeri (SNMPTN). Selama menjadi mahasiswa, penulis mengikuti beberapa kegiatan antara lain:

1. Menjadi Anggota Baru Computer Science (ABACUS) Jurusan Ilmu Komputer Universitas Lampung pada periode 2016/2017.
2. Menjadi anggota bidang Keilmuan dalam organisasi Himpunan Mahasiswa Jurusan Ilmu Komputer Universitas Lampung (HIMAKOM UNILA) pada periode 2017-2018.
3. Menjadi Asisten Dosen mata kuliah Sistem Operasi pada semester 4 tahun pelajaran 2017/2018 dan mata kuliah Basis Data pada semester 5 tahun pelajaran 2018/2019.

4. Melaksanakan Kerja Praktik di Dinas Komunikasi dan Informatika Kota Bandar Lampung pada bulan Desember 2018 – Februari 2019.
5. Melaksanakan Kuliah Kerja Nyata (KKN) di Desa Gunung Sugih Kecil, Kecamatan Jabung, Kabupaten Lampung Timur, Lampung selama 40 hari pada periode II, Juli – Agustus 2019.

PERSEMBAHAN

Alhamdulillahirobbil'alamin, puji dan syukur saya panjatkan kepada Allah SWT atas segala Rahmat dan Hidayah-Nya serta shalawat dan salam senantiasa tercurahkan kepada Nabi Muhammad SAW sehingga saya dapat menyelesaikan skripsi ini dengan baik.

Kupersembahkan karya tulis ini kepada:

Kedua orang tuaku, kakakku, dan adikku yang telah memberi dukungan moral maupun materi dan banyak berkorban untuk kelulusanku.
Buyut, kakek, dan nenek yang telah mendoakan dan memberi dukungan.
Semua saudara dan sahabatku yang telah memberi banyak motivasi dan selalu ada di saat suka maupun duka.

Bapak dan Ibu dosen pembimbing.
Keluarga Ilmu Komputer 2016.
Serta almamaterku tercinta, Universitas Lampung.

Terima kasih.

MOTTO

“Bisa jadi kamu membenci sesuatu, padahal ia amat baik bagimu, dan bisa jadi kamu menyukai sesuatu, padahal ia amat buruk bagimu. Allah mengetahui, sedang kamu tidak mengetahui.”
(QS. Al-Baqarah: 216)

“Siapa yang membaca Al-Qur’an, mempelajarinya dan mengamalkannya, maka dipakaikan kepada kedua orang tuanya pada hari kiamat mahkota dari cahaya yang sinarnya bagai sinar matahari, dan dikenakan kepada kedua orang tuanya perhiasan yang nilainya tak tertandingi oleh dunia.”
(HR. Al-Hakim)

“It's not always easy but that's life, be strong cause there are better days ahead.”
(Mark Lee)

SANWACANA

Segala puji syukur penulis panjatkan atas kehadirat Allah SWT karena atas berkah, rahmat, dan hidayah-Nya penulis dapat menyelesaikan skripsi dengan judul penelitian "Pengukuran *Object Oriented Concepts* pada Aplikasi Movie DB Menggunakan *CK Metrics Suite*" dengan baik. Shalawat dan salam senantiasa tercurahkan kepada Nabi Muhammad SAW.

Penulis menyadari selesainya skripsi ini tidak terlepas dari partisipasi bimbingan serta bantuan dari berbagai pihak baik secara langsung maupun tidak langsung. Dalam kesempatan ini penulis mengucapkan banyak terima kasih kepada:

1. Orang tua terkasih, Bapak Drs. Mukhaidori dan Ibu Umi Ipadah S.Pd., yang selalu memberikan doa serta kasih sayang dan sangat sabar dalam memberikan dukungan, nasihat, dan motivasi dalam menyelesaikan perkuliahan di Jurusan Ilmu Komputer;
2. Kakak dan adik tercinta, Mufidah Aulia Annisa S.T., dan Jihan Jazilah yang selalu memberikan semangat dalam menyelesaikan perkuliahan;
3. Buyut, nenek, dan kakek yang selalu mendoakan dan memberi semangat;
4. Ibu Anie Rose Irawati, S.T., M.Sc., selaku Dosen Pembimbing Utama yang telah membimbing penulis serta memberikan dorongan dan motivasi sehingga skripsi ini dapat diselesaikan. Penulis mengucapkan banyak terima kasih atas kesediaannya memberikan ide dan saran serta kritik dalam proses penyelesaian skripsi ini;
5. Ibu Yohana Tri Utami, S.Kom., M.Kom., selaku Dosen Pembimbing Kedua yang telah membimbing dan memberikan banyak arahan kepada penulis sehingga penelitian yang penulis lakukan dapat menjadi lebih baik;

6. Bapak Dr. rer. nat. Akmal Junaidi. M.Sc., selaku Dosen Penguji Utama yang telah memberikan kritik dan saran pemikiran dalam penyempurnaan skripsi;
7. Bapak Prof. Admi Syarif, Ph.D., selaku Pembimbing Akademik;
8. Bapak Didik Kurniawan, S.Si., M.T., selaku Ketua Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Lampung;
9. Bapak Dr. rer.nat. Akmal Junaidi, M.Sc., selaku Sekretaris Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Lampung;
10. Bapak Dr. Eng. Heri Satria, S.Si., M.Si., selaku Dekan FMIPA Universitas Lampung;
11. Bapak dan Ibu Dosen Jurusan Ilmu Komputer yang telah memberikan bekal ilmu pengetahuan kepada penulis selama menjadi mahasiswa di Jurusan Ilmu Komputer, Fakultas MIPA, Universitas Lampung;
12. Seluruh staf Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Lampung yang telah memberikan bantuan dan bimbingan selama penulis melakukan penelitian;
13. Teman terbaik seperjuangan yaitu Amelia Jasmine, Dian Pitaloka, Intan Dinasti, Cynthia Resti, dan Annisa Julia Dwisanti yang telah memberikan dukungan moril dan selalu ada di saat suka maupun duka;
14. Teman-teman Brikden yaitu Ayi, Aulia, Baim, Yunan, Sigit, Tantut, Abbi, Rifki, dan Arnaldo;
15. Teman-teman seperbimbingan skripsi yaitu Krisanti, Elva, dan Ilham;
16. Teman-teman KKN Desa Gunung Sugih Kecil yaitu Ara, Siska, Dhaniel, Rias, Eli, dan Andrian;
17. Mark Lee dan NCT yang selalu memberikan motivasi dan menjadi penyemangat penulis dalam menyelesaikan skripsi ini;
18. Saudara-saudara di Ilmu Komputer Universitas Lampung Angkatan 2016 yang berjuang bersama serta berbagi kenangan, pengalaman, dan membuat kesan yang tak terlupakan, terima kasih atas kebersamaan kalian;
19. Semua pihak yang telah membantu tanpa pamrih yang tidak dapat disebutkan secara keseluruhan satu per satu, semoga kita semua berhasil menggapai impian.

Penulis menyadari bahwa penulisan skripsi ini masih jauh dari kesempurnaan karena masih terbatasnya pengetahuan, pengalaman, dan kemampuan penulis. Akan tetapi, penulis berharap semoga skripsi ini dapat berguna dan bermanfaat bagi kita semua. Aamiin.

Bandar Lampung, 20 Juni 2023
Penulis,

Hartsa Hanifah
NPM. 1617051029

DAFTAR ISI

Halaman

DAFTAR TABEL

Tabel Halaman

DAFTAR GAMBAR

Gambar Halaman

DAFTAR KODE

Kode Halaman

I. PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi perangkat lunak secara terstruktur menuju perangkat lunak berorientasi objek ikut mempengaruhi perkembangan penelitian di bidang *software metrics*. Pemrograman secara prosedural umumnya diukur menggunakan *metrics* untuk mengetahui kompleksitas sebuah program. Tujuan dari pembuatan perangkat lunak adalah untuk menciptakan perangkat lunak yang berkualitas. Kualitas perangkat lunak akan mempengaruhi baik buruknya suatu kinerja dari perusahaan yang bersangkutan dalam menjalankan bisnisnya. Perangkat lunak yang berkualitas akan memudahkan perusahaan dalam menjalankan proses bisnisnya sehingga sumber daya yang dikeluarkan oleh perusahaan akan semakin efisien dan efektif (Ayubi, 2015).

Pemrograman berorientasi objek (*object oriented programming*) dilakukan berdasarkan objek dengan sekumpulan data yang ditetapkan dan sekumpulan operasi-operasi (metode) yang dapat dilaksanakan dalam data tersebut (Simarmata, 2010). Pengembang perangkat lunak saat ini lebih cenderung menggunakan konsep *Object Oriented Programming* (OOP) yang memudahkan untuk membangun perangkat lunak. Beragamnya pola implementasi berorientasi object menimbulkan perbedaan pendapat dalam melihat kualitas sebuah perangkat lunak (Khan, 2010 dalam Ayubi, 2015).

Penelitian yang mengajukan *object oriented metrics* salah satunya adalah penelitian yang dilakukan oleh Chidamber dan Kemerer pada tahun 1994. Chidamber dan

Kemerer mengajukan enam buah *metrics* yang dapat digunakan di dalam lingkup berorientasi objek, antara lain *Weight Methods 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 Methods* (LCOM) (Chidamber & Kemerer, 1994). *CK Metrics Suite* yaitu WMC, DIT, NOC, CBO, dan RFC cocok digunakan untuk mengevaluasi kualitas perangkat lunak dan mengukur kualitas perangkat lunak berorientasi objek pada level *class* (Sharma, dkk, 2012).

WMC digunakan untuk menghitung jumlah *method* yang diimplementasikan dalam kelas. *Class* yang dimiliki banyak *method* yang kompleks berarti *class* tersebut semakin spesifik. Hal ini dapat membatasi *reuse* (Chidamber and Kemerer, 1994; Systa and Yu, 1999; Lindroos, 2004). DIT mengukur kedalaman sebuah *class* dalam hierarki pewarisan. Semakin dalam sebuah *class* dalam hierarki pewarisan maka semakin banyak *method* yang dimilikinya, baik *method* pada *class* itu sendiri maupun *method* yang diwarisinya. Hal ini mengakibatkan *class* tersebut semakin kompleks (Systa and Yu, 1994 Lindroos, 2004). NOC adalah jumlah *subclasses* langsung yang mewarisi sebuah *class* pada hierarki *class* (Chidamber and Kemerer, 1994). Semakin besar NOC maka semakin tinggi *reusability.* Semakin besar NOC maka semakin besar upaya dan waktu untuk *testing* (Chidamber & Kemerer, 1994; Lindroos, 2004). CBO digunakan untuk menghitung jumlah kelas yang digabungkan dengan kelas tertentu (Gomathi & Linda, 2013). *Class* digabungkan dengan yang lain jika metode dari satu *class* menggunakan metode atau atribut dari kelas lain. Metrik CBO mengukur upaya yang diperlukan untuk menguji *class* (Harrison, dkk, 1998). RFC adalah sejumlah *method* yang dapat dipanggil sebagai *respons* terhadap sebuah pesan di *class* (Sarker, 2005). Semakin banyak *method* yang dapat dipanggil pada sebuah *class,* semakin besar kompleksitas *class* tersebut. Ini berarti semakin besar upaya pemeliharaan *class* tersebut (El-Ahmadi, 2006). LCOM mengukur jumlah keterpaduan yang ada, seberapa baik suatu sistem telah dirancang dan seberapa kompleks suatu kelas (Harrison, dkk, 1998). Definisi dari Chidamber dan Kemerer (1994) yaitu nilai LCOM bergantung pada jumlah

methods pada sebuah *class.* Semakin kecil nilai aktual LCOM dibanding nilai maksimumnya, semakin baik *cohesiveness class* tersebut (Rosenberg, 1998).

Penelitian ini merupakan implementasi perhitungan dari *Object Oriented Metrics* untuk mengukur dan mengevaluasi kualitas perangkat lunak berorientasi objek dengan menggunakan studi kasus Aplikasi Movie DB yang dikembangkan oleh Putra Pribowo (2019). Movie DB merupakan aplikasi yang menampilkan informasi mengenai film terkini yang sedang tayang dan film yang akan datang. Aplikasi ini memiliki kode program yang mempunyai konsep desain berorientasi *object* (OOP) dengan Bahasa Pemrograman Java. Pengukuran perangkat lunak dengan menggunakan studi kasus Aplikasi Movie DB akan menghasilkan nilai kuantitatif yang akan mempresentasikan kualitas Aplikasi Movie DB. Hasil tugas akhir ini diharapkan dapat menjadi referensi bagi pengembang Movie DB untuk memperbaiki desain kode program perangkat lunak tersebut.

1.2 Rumusan Masalah

Rumusan masalah pada penelitian ini adalah bagaimana mengimplementasikan pengukuran *object oriented metrics* pada Aplikasi Movie DB dengan parameter *CK Metrics Suite*.

1.3 Batasan Masalah

Batasan masalah pada penelitian ini sebagai berikut.

a. Penelitian ini membahas tentang implementasi pengukuran menggunakan *Object Oriented Metrics* dengan menerapkan enam buah *metrics* dari Chidamber dan Kemerer pada Aplikasi Movie DB yaitu *Weight Methods 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 Methods* (LCOM).

b. Dalam pengimplementasiannya, penelitian ini melakukan perhitungan secara manual.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah, tujuan dari penelitian ini adalah mengimplementasikan pengukuran *object oriented metrics* menggunakan *CK Metrics Suite* pada Aplikasi Movie DB.

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat digunakan sebagai referensi oleh pengembang perangkat lunak agar dapat mengetahui estimasi kualitas Aplikasi Movie DB yang berorientasi objek melalui faktor-faktor kualitas perangkat lunak secara kuantitatif. Berdasarkan penelitian Chidamber dan Kemerer, terdapat enam metrik yang akan digunakan dalam penelitian ini. Enam metrik tersebut memiliki penilaian yang berbeda. Metrik WMC memprediksikan waktu dan usaha yang diperlukan untuk membangun dan *maintenance* suatu *class.* Dalam peningkatan Metrik DIT dapat meningkatkan *bug* dan menurunkan kualitas perangkat lunak. Metrik NOC dapat menjadi indikator besarnya pengaruh sebuah *class* terhadap desain sistem secara keseluruhan dan nilai NOC akan menunjukkan tingkat *reusability* dan usaha *testing* yang diperlukan. Pada metrik CBO, nilai yang tinggi akan mengindikasikan *coupling* yang berlebihan sehingga merugikan desain modular dan membatasi *reuse,* menyulitkan *testing,* dan modifikasi. Metrik RFC digunakan untuk mengukur banyaknya komunikasi antar *object* dalam suatu *class.* Metrik LCOM digunakan untuk mengukur derajat kemiripan *method* oleh variabel *input* data atau atribut dalam *class.*

II. TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian terdahulu yang berkaitan dan menjadi acuan dalam penelitian ini diantaranya sebagai berikut:

1. Chidamber dan Kemerer mempublikasikan penelitian pertama tentang *object oriented metrics* pada tahun 1994. Dalam penelitiannya, Chidamber dan Kemerer mengembangkan dan mengimplementasikan kumpulan *software metrics* baru untuk *Object Oriented Design*. *Metrics* didasarkan dengan teori pengukuran dan mewakili sudut pandang pengalaman pengembang perangkat lunak berorientasi objek. Dalam melakukan evaluasi *metrics* berlawanan dengan kriteria standar yang kemudian menganjurkan beberapa jalan yang mana metode *Object Oriented* mungkin berbeda dengan metode tradisional. Sehingga, Chidamber dan Kemerer mengembangkan enam buah *metrics* yang kriteria evaluasinya sesuai dengan perangkat lunak berorientasi objek yaitu *Weight Methods per Class* (s), *Depth of Inheritance Tree* (DIT), *Number of Children* (NOC), *Coupling Between Object* (CBO), *Response for A Class* (RFC), dan *Lack of Cohesion in Methods* (LCOM) (Chidamber & Kemerer, 1994).
2. Penelitian pada tahun 2012 yang dilakukan oleh Kornelis Letelay dan Azhari SN mengenai evaluasi kualitas perangkat lunak dengan *metrics* berorientasi objek. Dalam penelitian ini, Kornelis dan Azhari menggunakan enam buah *object oriented metrics* yang diajukan oleh Chidamber dan Kemerer dimana keenam *metrics* tersebut memiliki sudut pandang terkait dengan faktor-faktor kualitas perangkat lunak sehingga *metrics* dapat digunakan untuk melakukan

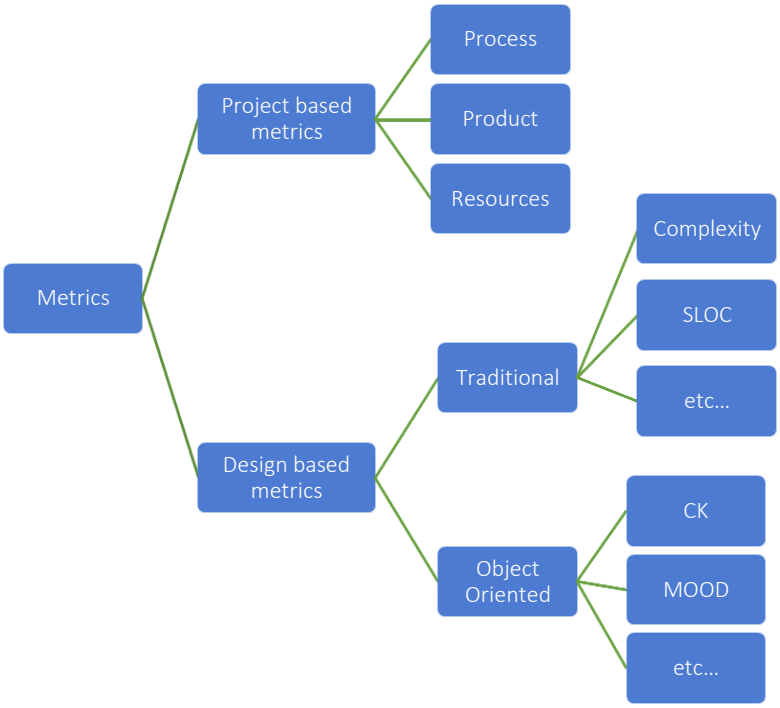
evaluasi kualitas perangkat lunak berorientasi objek dari beberapa faktor kualitas yang terkait. Penelitian ini menjelaskan bahwa penentuan kualitas perangkat lunak dilakukan dengan penentuan nilai yang dipakai untuk mengukur rendah atau tingginya kualitas perangkat lunak dengan menggunakan enam *metrics* berorientasi. Selain itu, *metrics* tersebut dapat digunakan untuk mengetahui *class* mana saja yang perlu diperiksa ulang atau direvisi (Letelay & Azhari, 2012).

3. Analisis dari penelitian lain mengenai *object oriented metrics* yaitu penelitian yang dilakukan oleh D.I George Amalarethinam dan P.H Maitheen Shahul Hameed. Goerge dan Shahul (2015) melakukan pengembangan dan implementasi dari deretan *metrics* untuk *object oriented design*. Dalam penelitian ini sebuah program java yang diambil sebagai model dengan konsep OOP seperti *inheritance*, *polymorphism*, dan *abstraction*. Metrik-metrik berorientasi objek diterapkan pada program java dan hasil setiap metrik ditabulasikan dengan jelas. *Object oriented metrics* yang digunakan merupakan *object oriented metrics* dari berbagai macam penelitian untuk memilih *object oriented metrics* yang tepat untuk model dan aplikasi bagi pengembang perangkat lunak (Amalarethinam and P.H. Maitheen, 2015).
4. Penelitian selanjutnya yang menjadi referensi adalah penelitian yang dilakukan oleh Aula Ayubi (2015) dari Fakultas Teknologi Informasi dan Komunikasi, Institut Teknologi Sepuluh Nopember. Penelitian ini melakukan pengukuran kualitas *software* berdasarkan aspek *object oriented* dengan menggunakan Metrik Chidamber dan Kemerer. Dalam penelitian ini juga menjabarkan karakteristik kualitas yang dapat diukur menggunakan metrik Chidamber dan Kemerer berdasarkan ISO/IEC 9126-1 akan menghasilkan nilai kuantitatif yang mempresentasikan tingkat *efficiency, understandability*, *reusability*, dan *maintainability/testability* sehingga pengembang *software* dapat mengetahui kualitas konsep *object oriented* yang terdapat pada kode perangkat lunak tersebut. Dalam penelitian menghasilkan rekomendasi perbaikan struktur kode program berupa *class diagram* berdasarkan *design pattern* untuk memperbaiki desain kode program perangkat lunak tersebut (Ayubi, 2015).

2.2 *Software* Metrics

Software metrics adalah entitas yang berharga dalam seluruh siklus hidup perangkat lunak. *Software metrics* menyediakan pengukuran untuk pengembangan perangkat lunak, termasuk dokumen persyaratan perangkat lunak, desain, program, dan pengujian (Rawat, dkk, 2012). *Software metrics* juga dapat diartikan sebagai sebuah ukuran dari perangkat lunak dan proses produksi perangkat lunak yang memberi nilai kuantitatif pada setiap atribut yang terdapat dalam produk atau proses (Saini, dkk, 2014). *Software metrics* dapat digunakan untuk memahami aplikasi, untuk mendapatkan gambaran umum tentang sistem besar dan mengidentifikasi masalah desain potensial (Lanza & Marinescu, 2007).

Metrics terbagi menjadi dua kategori, yaitu *project based metrics* dan *design based metrics* (Sarker, 2005). *Project based metrics* terdiri dari *process, product,* dan *resources* sedangkan *design based metrics* terdiri dari *traditional metrics* dan *object oriented metrics.* Hierarki *metrics* menurut Sarker (2005) dapat dilihat pada Gambar 1.



Gambar 1. Hierarki Metrics (Sarker, 2005).

Process metrics, disebut juga *management metrics,* berhubungan dengan proses dalam mengembangkan sebuah sistem (Syst & Yu, 1999). *Process metrics* biasanya digunakan untuk membantu memprediksi ukuran akhir sebuah sistem, memprediksi tingkat usaha yang dibutuhkan dalam sebuah proyek, dan menentukan apakah sebuah proyek sesuai dengan jadwal. *Product metrics,* disebut juga *quality metrics,* digunakan untuk mengontrol kualitas dari produk perangkat lunak (Syst & Yu, 1999). *Metrics* ini digunakan untuk perangkat lunak yang belum selesai dibuat agar dapat diprediksi properti dan kompleksitas hasil akhir perangkat lunak tersebut. *Resources* adalah entitas yang dibutuhkan dalam aktivitas proses pengembangan. *Resources* yang diukur adalah semua *input* dalam menghasilkan perangkat lunak (Sarker, 2005).

Pada sistem berorientasi objek, *traditional metrics* umumnya digunakan dalam ruang lingkup *methods* yang terdiri atas operasi-operasi sebuah *class* (Rhamdani, 2008). Beberapa contoh *traditional metrics* antara lain *cyclomatic complexity* (CC), *source line of code* (SLOC), dan *comment percentage* (CP). *Object oriented metrics* digunakan untuk merefleksikan perangkat lunak berorientasi objek. Beberapa contoh *object oriented metrics* yang telah diajukan antara lain Chidamber & Kemerer (CK) *metrics*, *Metrics for Object Oriented Design* (MOOD), dan lain-lain.

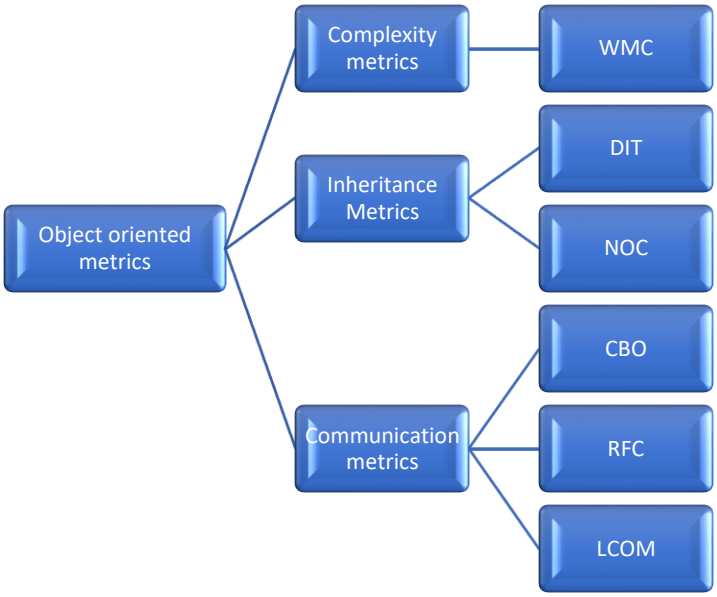
2.3 *Object Oriented Metrics*

Object Oriented Metrics adalah sebuah ukuran yang digunakan untuk mengevaluasi dan memprediksi kualitas dari perangkat lunak (Gomathi & Edith, 2013). *Object oriented metrics* diartikan sebagai sebuah ukuran yang didasarkan pada data dan model prosedur dari analisis terstruktur. *Object oriented metrics* juga didasarkan pada objek dan karakteristiknya (Amalarethinam & P.H. Maitheen, 2015).

Object oriented metrics terbagi ke dalam tiga kategori (Systa & Yu, 1999) dan enam metrik yang diajukan oleh Chidamber & Kemerer (1994), yaitu:

1. *Complexity metrics*, memperkirakan kompleksitas perangkat lunak terdiri dari *weighted methods per class* (WMC).
2. *Inheritance metrics*, mengukur hierarki pewarisan terdiri dari *depth of inheritance tree* (DIT) dan *number of children* (NOC).
3. *Communication metrics*, mengukur pasangan dan keterpaduan antar *class* terdiri dari *coupling between object* (CBO), *response for a class* (RFC), dan *lack of cohesion in method* (LCOM).

Kategori CK *metrics* menurut Systa & Yu (1999) dapat dilihat pada Gambar 2.



Gambar 2. Kategori CK *Metrics* (Syst & Yu, 1999).

2.4 Metrik Chidamber dan Kemerer

Metrik Chidamber dan Kemerer adalah salah satu metrik yang digunakan untuk mengukur kualitas desain sebuah perangkat lunak berdasarkan enam metrik dengan

melihat perspektif desain berorientasi objek (Septian, 2010). Metrik Chidamber dan Kemerer mempunyai enam metrik yaitu *Weighted Method per Class* (WMC), *Depth of Inheritance Tree* (DIT), *Number OF Children* (NOC), *Coupling Between Object Classes* (CBO), *Response For a Class* (RFC), dan *Lack of Cohesion Method* (LCOM).

Chidamber dan Kemerer mendefinisikan apa yang disebut CK *metrics suite*. CK *metrics* ini menawarkan wawasan informatif tentang apakah pengembang mengikuti prinsip berorientasi objek dalam desain mereka. Mereka mengklaim bahwa menggunakan beberapa metrik mereka secara kolektif membantu manajer dan desainer untuk membuat keputusan desain yang lebih baik. Metrik CK telah menghasilkan sejumlah besar minat dan saat ini merupakan rangkaian pengukuran yang paling terkenal untuk perangkat lunak berorientasi objek.

SATC *(Software Assurance Technology Center)* mendiskusikan pendekatannya untuk mengidentifikasi sekumpulan *object oriented metric* (Rosenberg, 1998). Dalam penelitian (Ayubi, 2015) menyatakan bahwa SATC telah menghasilkan pemetaan Metrik Chidamber dan Kemerer pada kode program perangkat lunak dengan parameter-parameter Metrik Chidamber dan Kemerer yang mempunyai objektivitas dari pengukuran kualitas perangkat lunak seperti yang dapat dilihat pada Tabel 1.

Tabel 1. Pemetaan Metrik Chidamber dan Kemerer dengan Kode Program

Source	***Metrics***	***Object Oriented Construct***
Object oriented (New)	*Weighted Method per Class* (WMC)	*Class/Method*
Object oriented (New)	*Depth of Inheritance Tree* (DIT)	*Inheritance*
Object oriented (New)	*Number of Children* (NOC)	*Inheritance*
Object oriented (New)	*Coupling Between Object Classes* (CBO)	*Coupling*

Tabel 1. (Lanjutan)

Source	***Metrics***	***Object Oriented Construct***
Object oriented (New)	*Response For a Class* (RFC)	*Class/Message*
Object oriented (New)	*Lack of Cohesion Method* (LCOM)	*Class/Cohesion*

Nilai metrik WMC, DIT, NOC, CBO, dan RFC yang dianjurkan oleh penelitian sebelumnya (Ayubi, 2015) telah menghasilkan kategori dari Metrik Chidamber dan Kemerer yang telah disajikan pada Tabel 2 dan nilai yang dianjurkan untuk metrik LCOM berdasarkan penelitian yang lain (Mitra, 2013).

Tabel 2. Nilai Metrik Chidamber dan Kemerer

Metrik	**Baik**	**Sedang**	**Buruk**
WMC	$1 \leq x < 50$	$50 \leq x \leq 150$	$150 < x$
DIT	$x < 5$	$5 \leq x < 8$	$8 \leq x$
NOC	$x < 4$	$4 \leq x < 8$	$8 \leq x$
CBO	$x < 14$	$14 \leq x \leq 150$	$150 < x$
RFC	$x < 100$	$100 \leq x \leq 1000$	$1000 < x$
LCOM	$0 \leq x \leq 1$		

Aplikasi Movie DB menggunakan Metrik Chidamber dan Kemerer sebagai tolak ukur dalam pengukuran. Metrik Chidamber dan Kemerer dapat dideskripsikan secara singkat pada Tabel 3.

Tabel 3. Batasan Metrik Chidamber dan Kemerer (Ayubi, 2015)

Metrik	**Hasil (Semakin Tinggi Nilai Berarti)**
Weight Method per Class	Membatasi potensi *reuse* Semakin kompleks Membutuhkan waktu dan usaha yang tinggi pada tahapan *developing and maintenance* Kemungkinan banyaknya *bug*

Tabel 3. (Lanjutan)

Metrik	Hasil (Semakin Tinggi Nilai Berarti)
Depth of Inheritance Tree	Kemungkinan *reuse* semakin tinggi Semakin kompleks
Number of Children	Kemungkinan *reuse* semakin tinggi Membutuhkan usaha *testing* yang lebih tinggi Semakin besar kemungkinan ketidakcocokan *subclass*
Coupling Between Object Classes	Kemungkinan banyaknya *bug* dalam perangkat lunak tersebut Membatasi potensi *reuse*
Response For a Class	*Testing* dan *debugging* akan semakin kompleks Kemungkinan banyaknya *bug* dalam perangkat lunak tersebut
Lack of Cohesion Method	Membatasi potensi *reuse* Mengindikasikan *class* tersebut sederhana tidak kompleks)

Beberapa jenis *metrics* yang digunakan untuk mengukur *object oriented programming* dalam suatu sistem antara lain:

2.4.1 *Weight Methods per Class* (WMC)

Weight Methods per Class (WMC) merupakan pengukuran jumlah *method* dalam setiap *class*nya (Chidamber & Kemerer, 1994). *Method* merupakan properti dan objek. Kompleksitas sebuah *object* ditentukan dengan menghitung propertinya (Ayubi, 2015). Dalam penelitian (Marpaung, 2015) mengatakan bahwa metrik WMC adalah jumlah *method* per kelas, dimana setiap *method* dibebani oleh sebuah kompleksitas yang didasarkan pada tipe dari *method*, jumlah dari propertis *method* yang memengaruhi dan jumlah dari layanan yang *method* sediakan untuk sistem. *Metrik* WMC memprediksikan waktu dan usaha yang diperlukan untuk membangun dan *maintenance* suatu *class*. Nilai WMC yang tinggi dapat mengindikasikan kesalahan lebih banyak. *Class* dengan WMC rendah seringkali mengindikasikan polimorfisme yang lebih besar. *Class* dengan *method* yang banyak akan cenderung menjadi aplikasi atau perangkat lunak yang spesifik

sehingga akan membatasi kemungkinan penggunaan kembali. Semakin besar nilai WMC maka akan membatasi potensi penggunaan kembali, oleh karena itu WMC harus dijaga serendah mungkin (Rosenberg, 1998).

Dalam penelitian (Mitra, 2013) menyebutkan bahwa WMC merupakan perhitungan jumlah *methods* yang diterapkan dalam suatu *class*. WMC yang tinggi telah mengakibatkan program rentan terhadap *bug* dan terlalu kompleks untuk dipahami. Nilai WMC yang baik adalah $1 \leq x < 5$ (Ayubi, 2015). Semakin rendah akan semakin bagus. Semakin tinggi nilai *methods* akan semakin sulit pengujiannya. Nilai WMC dilihat dari jumlah total *method* yang terdapat pada *class*. WMC dihitung dengan memakai jumlah dari *methods* dalam setiap *class* (Marpaung, 2015).

2.4.2 *Depth of Inheritance Tree* (DIT)

Depth of Inheritance Tree (DIT) didefinisikan sebagai pengukuran kedalaman penurunan suatu *class*. Metrik *Depth of Inheritance Tree* adalah panjang maksimum dari *root* ke *leaves* dalam sebuah pohon hierarki pada sebuah *class*. Kedalaman suatu *class* disebut hierarki, DIT akan mengukur kedalaman sebuah *class* dalam hierarki pewarisan, dimulai dari *class leaves* dan menurun pada *class root* (Chidamber & Kemerer, 1994).

Dalam penelitian (Rhamdani, 2008) mengatakan bahwa setiap *class* di Java mewarisi *class* Object dan *class* Object adalah *root* pada hierarki pewarisan *class* di Java, sehingga DIT setiap *class* pada pemrograman Java minimal 1. *Class* Object sendiri memiliki nilai DIT = 0. Semakin dalam sebuah kelas dalam hierarki, semakin besar jumlah metode, kemungkinan akan lebih mudah mewarisi dan membuatnya lebih kompleks untuk diprediksi. Semakin dalam pohon semakin besar kompleksitas desain dan begitu pula potensi untuk menggunakan kembali metode lebih besar. Hal ini dapat diprediksi menggunakan diagram pohon (Saini,

dkk, 2014). Nilai yang direkomendasikan untuk DIT adalah < 5 (Chidamber & Kemerer, 1994).

Inheritance melihat hierarki kelas dari bawah ke atas. *Subclass* mewarisi perilaku dan atribut dari *superclass*-nya. Di dalam Java, untuk mendeklarasikan suatu *class* sebagai *subclass* dilakukan dengan cara menambahkan kata kunci *extends* setelah deklarasi nama *class*, kemudian diikuti dengan nama *superclass*-nya (Andrea & Karim, 2021).

Cara menghitung DIT adalah dengan menampung data semua nama *class* yang ada pada *file* inputan, kemudian *class* tersebut terdapat *inheritance* dengan mencari *keywords "extends"*. Apabila kelas tersebut merupakan keturunan dari kelas lain, maka nilai DIT akan ditambah (Marpaung, 2015).

```
public class B extends A {
…
}
```

Kode 1. Contoh Deklarasi *Inheritance.*

Contoh Kode 1 adalah memberitahukan *compiler* Java bahwa *class* A di *extend* ke *class* B. *Class* B adalah *subclass* (*class* turunan) dari *class* A, sedangkan *class* A adalah *parent class* dari *class* B (Andrea & Karim, 2021).

2.4.3 *Number of Children* (NOC)

Number of Children (NOC) adalah jumlah *subclasses* langsung yang mewarisi sebuah *class* pada hierarki *class* (Chidamber & Kemerer, 1994). Metrik NOC dapat

menjadi indikator besarnya pengaruh sebuah class terhadap desain sistem secara keseluruhan. Semakin besar nilai NOC, semakin besar kemungkinan ketidakcocokan *subclass* dengan abstraksi yang ada pada *superclass* (Glasberg, dkk, 2000 dalam Ayubi, 2010).

Metrik NOC menghitung jumlah anak *(subclass)* dari keturunan langsung dari *superclass* pada sebuah *class*. Nilai NOC akan menunjukkan tingkat *reusability* dan usaha *testing* yang diperlukan. Semakin banyak jumlah *subclass* maka akan semakin besar tingkat penggunaan kembali karena *inheritance* merupakan bentuk dari *reuse*, semakin banyak *testing* yang harus dilakukan karena apabila terjadi perubahan di *superclass* dapat mempengaruhi *subclass* dari *class* tersebut. Sebuah *class* dengan NOC yang besar akan sulit dimodifikasi dan membutuhkan pengujian yang lebih karena berpengaruh pada perubahan *children.* Nilai metrik NOC yang baik adalah < 4 (Chidamber & Kemerer, 1994).

```
public class Parent {
    private String output = "hallo";
    public void print(){
        System.out.println(output);
    }
}
public class Child extends Parent {
    Private String output = "child";
}
public class Child2 extends Parent {
    Private String output = "child2";
}
```

Kode 2. Contoh Kode yang Memiliki Anak Kelas.

NOC dihitung dengan cara membaca data inputan dan menampungnya dalam *string,* kemudian dari data *string* tersebut akan dipindai kelas data dan ditampung dalam list. Setiap kelas akan diperiksa apakah merupakan anak dari kelas lain atau bukan dengan menggunakan *keyword 'extends'*. Apabila kelas tersebut merupakan kelas anak, maka cari kelas induknya pada daftar yang telah ditampung, kemudian tambahkan nilai 1 untuk setiap anak kelas yang memiliki induk. Apabila ditemukan

anak kelas lain yang memiliki induk yang sama maka nilainya juga akan ditambahkan (Marpaung, 2015). Pada Kode 2, *Class* Parent memiliki nilai Metrik NOC 2, *class* Parent mempunyai dua *subclass* yaitu *class* Child dan *class* Child2. *Class* Child memiliki kelas *parent* Parent dan *class* Child2 memiliki kelas *parent* Parent, sehingga *class* Parent dikatakan memiliki nilai Metrik NOC 2.

2.4.4 *Coupling Between Object Classes* (CBO)

Coupling Between Objects (CBO) adalah banyaknya kolaborasi untuk sebuah kelas atau jumlah hubungan kelas dengan kelas yang lain (Marpaung, 2015). CBO didefinisikan untuk sebuah *class*, yaitu banyaknya *class* yang terpasang oleh *class* yang diukur. Sebuah *class* A dikatakan terpasang dengan *class* B jika *class* A menggunakan *method* atau *instance variable* pada *class* B (Chidamber & Kemerer, 1994).

Nilai CBO yang tinggi akan mengindikasikan *coupling* yang berlebihan sehingga merugikan desain modular dan membatasi *reuse*, menyulitkan *testing* dan modifikasi. Kelas yang lebih independen lebih mudah digunakan kembali pada program lain. Untuk peningkatan modularitas dan mempromosikan enkapsulasi, pasangan kelas antar objek harus dilakukan seminimal mungkin (Saini, dkk, 2014). Nilai CBO yang baik adalah < 14 (Chidamber & Kemerer, 1994).

Cara menghitungnya adalah pada *string* yang merupakan isi dari inputan akan di list terlebih dahulu seluruh *class* yang ada, kemudian dilakukan pemeriksaan apakah *class* tersebut merupakan *child class* dari kelas lain, kemudian dilakukan pula pemeriksaan apakah pada kelas tersebut ada dilakukan pemanggilan kelas lain dengan *keyword "new"* dan memeriksa apakah yang dipanggil merupakan data dari *class* yang ada atau bukan (kelas yang mungkin merupakan bagian dari *library* yang ada) (Marpaung, 2015).

2.4.5 *Response For a Class* (RFC)

Response for Class (RFC) didefinisikan sebagai sekumpulan metode yang dapat dieksekusi pada respon dan pesan yang menerima sebuah pesan dari objek dari kelas (Gomathi & Edith, 2013). Metrik *Response For a Class* menghitung jumlah semua *method* yang dipanggil sebagai respon terhadap *object* luar dari sebuah *class* (Chidamber & Kemerer, 1994).

RFC mencangkup semua *method* yang diakses dalam hirarki class tersebut. RFC digunakan untuk mengukur banyaknya komunikasi antar *object* dalam suatu *class.* Sehingga semakin tinggi nilai RFC maka, akan semakin banyak *method* yang digunakan untuk merespon *object* dari luar dan semakin kompleks sehingga akan meningkatkan waktu untuk *testing*. Nilai RFC yang tinggi akan meningkatkan kemungkinan banyak kesalahan, karena class dengan RFC tinggi akan lebih kompleks dan sulit dimengerti. RFC yang tinggi akan mengidentifikasi pengujian dan *debugging* yang rumit. Nilai RFC yang baik adalah < 100 (Chidamber & Kemerer, 1994).

Cara menghitungnya adalah dengan menghitung dan menampung data *method* dari setiap kelas kemudian menampung data kelas dan *method* tersebut. Setelah itu dilakukan perhitungan untuk setiap pemanggilan kelas pada kelas lain serta penggunaan *method* yang terdapat pada kelas yang dipanggil (Marpaung, 2015).

2.4.6 *Lack of Cohesion Method* (LCOM)

Lack of Cohesion in Methods (LCOM) digunakan untuk menghitung jumlah dari pasangan metode terpisah dikurangi jumlah pasangan metode serupa (Gomathi & Edith, 2013). Metrik *Lack of Cohesion Method* mengukur kemiripan *method* dalam sebuah *class* dari *instance variabel* atau atribut. LCOM mengukur *method* yang

tidak terhubung dengan *class* lain (mewakili bagian independen dalam class). Tingginya kohesi mengidentifikasikan potensi yang baik pada class tersebut, mengindikasikan *class* tersebut sederhana dan memiliki sifat *reusability* yang tinggi. Sedangkan semakin rendah kohesi maka semakin kompleks *class* tersebut (Ayubi, 2015).

LCOM adalah hasil yang diperoleh dari memperkirakan jumlah pasangan metode di kelas yang tidak memiliki kesamaan atribut (Chidamber & Kemerer, 1994). LCOM sudah mengalami beberapa penyempurnaan dan sekarang yang digunakan adalah LCOM5 (Mitra, 2013). Dalam penelitian (Chandrika, dkk, 2011), LCOM5 didefinisikan oleh Henderson-Seller (1996). Nilai LCOM yang baik berada pada rentang 0 sampai 1. LCOM = 0 berarti *class* adalah sangat kohesif, sedangkan LCOM > 1 maka sebaiknya *class* dibagi lagi menjadi lebih dari satu *class*. Untuk LCOM5 memiliki nilai 0 dianggap kohesi sempurna. Rumus metrik LCOM5 (Henderson-Seller, 1996 dalam Chandrika, dkk, 2011) ada pada persamaan 1 berikut.

$$LCOM5 = \frac{\left(\frac{1}{a} x\ Mu\right) - m}{1 - m} \quad \ldots\ldots\ldots(1)$$

Keterangan:

a = banyak *variable* dalam satu *class*

Mu = jumlah pemanggilan atribut oleh seluruh *method* dalam satu *class*

m = jumlah *method* dalam class

2.5 Pemrograman Java

Java merupakan bahasa pemrograman yang tergolong pada *high level language* (mudah bagi manusia untuk memahami), mengingat kata-kata/statemennya menyerupai bahasa manusia (*english*). Namun demikian, dalam penulisannya memerlukan aturan (*syntax*) yang ketat (Rusli, dkk, 2016).

Java merupakan bahasa pemrograman yang berbasis objek, atau biasa disebut dengan istilah OOP (*Object Oriented Programming*). Beberapa konsep dasar *object oriented programming* dalam bahasa java antara lain sebagai berikut (Jubilee Enterprise, 2015):

1. Objek

   Objek merupakan bagian penting dalam pemrograman berbasis objek. Secara umum, objek menyatukan data dan fungsi menjadi suatu unit dalam sebuah aplikasi. Setiap objek menampung data dan kode untuk mengelola data-data tersebut. Objek dapat berinteraksi tanpa harus mengetahui informasi rinci mengenai data atau kode dalam objek lain.

2. Kelas

   Kelas adalah sekumpulan objek dengan properti (atribut), perilaku (operasi), dan hubungan yang sama dengan antar objek. Sebuah objek merupakan variabel dengan tipe kelas. Kelas merupakan tipe data yang dibuat oleh pengguna.

3. Abstraksi Data

   Abstraksi data dilakukan dengan menyediakan deskripsi penting tanpa menyertakan informasi detail atau penjelasan. Kelas-kelas menggunakan konsep abstraksi dan didefinisikan sebagai daftar atribut abstrak seperti ukuran, dan fungsi-fungsi yang mengelola atribut-atribut tersebut yang digunakan pada kelas.

4. Enkapsulasi Data

   Enkapsulasi data artinya membungkus data dan fungsi dalam sebuah unit tunggal. Dengan enkapsulasi, data tidak dapat diakses secara langsung dari luar. Untuk mengakses data digunakan fungsi-fungsi yang ada dalam kelas tersebut.

5. Pewarisan

   Pewarisan adalah proses yang memungkinkan objek dari suatu kelas dapat memperoleh properti dari objek dari kelas yang lain. Pewarisan juga dapat diartikan bahwa suatu kelas mewarisi data dan perilaku dari kelas yang lain. Dengan adanya pewarisan, dapat menambahkan karakteristik pada suatu kelas tanpa melakukan modifikasi.

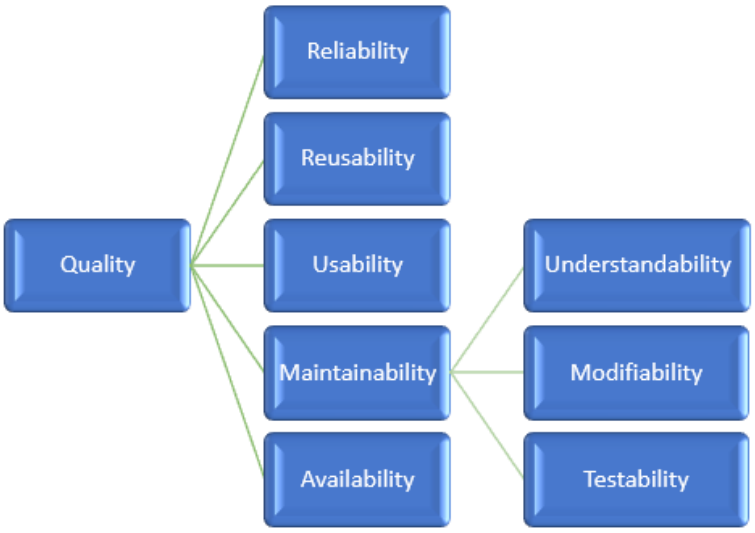
6. Polimorfisme

   Polimorfisme adalah kemampuan untuk memiliki lebih dari satu bentuk. Polimorfisme artinya setiap operasi yang sama dalam melakukan hal yang berbeda untuk tiap-tiap kelas.

2.6 Kualitas Perangkat Lunak

Kualitas perangkat lunak akan mempengaruhi baik buruknya suatu kinerja dari perusahaan yang bersangkutan dalam menjalankan bisnisnya. Perangkat lunak yang berkualitas akan memudahkan perusahaan dalam menjalankan proses bisnisnya sehingga sumber daya yang dikeluarkan oleh perusahaan akan semakin efisien dan efektif (Ayubi, 2015).

Kualitas perangkat lunak yang baik dari sudut pandang *customer* adalah perangkat lunak yang memuaskan kebutuhan *customer*. Kualitas perangkat lunak yang baik dari sudut pandang pengembang yaitu pengembang akan melihat produk perangkat lunak dari dalam perangkat lunak itu sendiri. Pengembang yang menggunakan pemikiran berorientasi objek memiliki tujuan pada terpenuhinya karakteristik tertentu. Terpenuhinya karakteristik-karakteristik tersebut merupakan kualitas dari sudut pandang pengembang (El-Ahmadi, 2006). Karakteristik tersebut berupa faktor-faktor kualitas. Faktor-faktor tersebut dapat dilihat pada Gambar 3.



Gambar 3. Faktor-faktor Kualitas Perangkat Lunak (El-Ahmadi, 2006)

Definisi faktor-faktor tersebut sebagai berikut:

1. *Reliability*

   Menurut Jawadekar (2004), *reliability* adalah tingkat ketepatan fungsi-fungsi perangkat lunak tanpa adanya kesalahan.

2. *Reusability*

   Reusability adalah tingkat kemampuan perangkat lunak atau bagian perangkat lunak untuk dapat digunakan ulang di lain tempat sebagai komponen yang *reusable* (Jawadekar 2004).

3. *Usability*

   Usability adalah tingkat usaha yang dibutuhkan untuk mempelajari, mengoperasikan, dan menggunakannya untuk tujuan yang diinginkan (Jawadekar, 2004).

4. *Maintainability*

   Maintainability adalah tingkat usaha untuk mengetahui letak kesalahan dan memperbaikinya. Faktor ini dapat dipecah menjadi tiga subfaktor, masing-masing *understandability, modifiability*, dan *testability* (El-Ahmadi 2006).

5. *Understandability*

   Understandability terkait dengan peningkatan kompleksitas psikologis (SATC 1995).

6. *Modifiability*

   Modifiability dari segi *flexibility* adalah tingkat kemudahan sebuah sistem atau komponen agar dapat dimodifikasi untuk penggunaan dalam suatu aplikasi atau lingkungan (IEEE Std. 610.12 diacu dalam Barbacci 2004).

7. *Testability*

   Menurut Jawadekar (2004), *testability* adalah tingkat usaha yang dibutuhkan untuk menguji perangkat lunak dalam proses *quality assurance*.

8. *Availability*

   Availability adalah tingkat kemudahan suatu sistem atau komponen agar dapat dioperasikan dan diakses ketika ingin digunakan (IEEE Std. 610.12, diacu dalam Barbacci 2004).

Semua faktor kualitas tersebut dapat diukur secara kuantitatif menggunakan *metrics.* Berdasarkan penelitian (Letelay & SN, 2012) kualitas perangkat lunak yang digunakan berdasarkan faktor-faktor kualitas yang terkait dengan enam metrik tersebut memiliki masing-masing fungsi seperti yang dapat dilihat pada Tabel 4.

Tabel 4. Pengaruh Metrik Chidamber dan Kemerer terhadap Faktor-faktor Kualitas (Letelay & SN, 2012 dan Rhamdani, 2008)

Faktor Kualitas	**Parameter *Metric***
Usability	WMC
Reusability	WMC, DIT, NOC, CBO
Understandability	DIT, CBO, RFC, LCOM
Modifiability	DIT, CBO, RFC, LCOM
Testability	WMC, DIT, NOC, CBO, RFC, LCOM

Berdasarkan faktor-faktor kualitas yang terkait dengan keenam *metrics* tersebut menentukan tinggi rendahnya nilai *metrics* yang mempengaruhi pada tinggi rendahnya faktor-faktor kualitas yang terkait pada masing-masing *metrics.* Sebagai

contoh, nilai WMC yang tinggi menyebabkan menurunnya tingkat *maintainability (testability)*, *usability*, dan *reusability* (Letelay & SN, 2012).

III. METODOLOGI PENELITIAN

3.1 Tempat dan Waktu Penelitian

Penelitian dilakukan di Gedung Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam (FMIPA) Universitas Lampung. Penelitian ini dilaksanakan pada Semester Genap 2021/2022 sampai dengan Semester Genap 2022/2023.

3.2 Data dan Alat

Data dan alat yang digunakan dalam penelitian ini adalah sebagai berikut.

3.2.1 Data

Data primer yang digunakan untuk penelitian yaitu *source code* Aplikasi Movie DB yang diperoleh langsung dari Putra Pribowo dan telah mendapat izin, ditunjukkan dalam Tabel 5.

Tabel 5. Daftar Nama *File Source Code* Aplikasi Movie DB

No.	*Class* pada Aplikasi Movie DB
1.	CariFragment.java
2.	DatangFragment.java

Tabel 5. (Lanjutan)

No.	*Class* pada Aplikasi Movie DB
3.	FavoritFragment.java
4.	TayangFragment.java
5.	Movie.java
6.	DatabaseContract.java
7.	DatabaseHelper.java
8.	MovieHelper.java
9.	MovieModel.java
10.	Filem.java
11.	DetailActivity.java
12.	MainActivity.java

3.2.2 Alat

Penelitian dilakukan dengan menggunakan beberapa alat untuk mendukung dan menunjang pelaksanaan penelitian. Alat yang digunakan berupa perangkat keras *(hardware)* dan perangkat lunak *(software).*

a. Perangkat Keras (*Hardware*)

Perangkat keras yang digunakan yaitu *laptop* dengan spesifikasi:

1. *Processor* Intel(R) Core(TM) i3-5005U,
2. *Installed memory* (RAM) 4GB,
3. *Hard drive* 500GB.

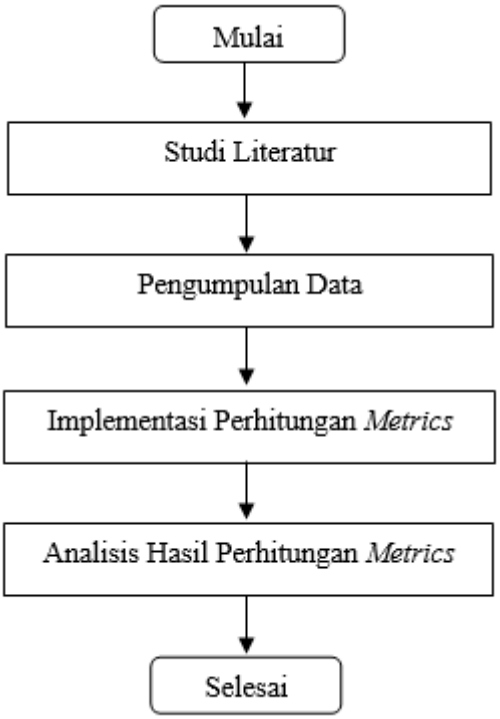
b. Perangkat Lunak (*Software*)

Perangkat lunak yang digunakan adalah:

1. Sistem Operasi *Windows* 10 64 bit;
2. Android Studio;
3. *Tool* simpleUMLCE_8205.jar

3.3 Metode Penelitian

Metode penelitian berisi tahapan-tahapan yang akan dilaksanakan dalam penelitian. Diagram alur yang menjelaskan tahapan penelitian ini ditunjukkan pada Gambar 4.



Gambar 4. Diagram Alur Metodologi Penelitian.

Penelitian diawali dengan studi literatur, pengumpulan data, kemudian implementasi perhitungan *metrics*, dan tahapan terakhir yaitu implementasi perhitungan *metrics*.

3.3.1 Studi Literatur

Studi literatur adalah mencari referensi teori yang relevan dengan kasus atau permasalahan yang ditemukan. Studi literatur dilakukan dengan cara mempelajari jurnal, buku, artikel, dan situs yang terkait dengan *metrics* perangkat lunak berorientasi objek. Pengukuran yang digunakan pada penelitian ini adalah *object oriented metrics* yaitu enam buah *metrics* berdasarkan penelitian yang dilakukan oleh Chidamber dan Kemerer (1994). Penggunaan enam *object oriented metrics* untuk mengukur Aplikasi Movie DB sangat sesuai dikarenakan aplikasi ini dikembangkan menggunakan bahasa pemrograman java yang merupakan *object oriented programming.* Enam *metrics* yang diajukan Chidamber dan Kemerer adalah pelopor penelitian *object oriented metrics* untuk mengevaluasi perangkat lunak yang merefleksikan desain berorientasi objek serta telah mencakup semua pengukuran dari berbagai komponen yang ada dalam *object oriented programming* seperti pewarisan, polimorfisme, dan abstraksi data.

3.3.2 Pengumpulan Data

Pengumpulan data dilakukan setelah Aplikasi Movie DB selesai dikembangkan. Data yang dikumpulkan adalah semua yang terkait dengan pengukuran *object oriented metrics* pada Aplikasi Movie DB. Dalam proses pengukuran *object oriented metrics* dijelaskan beberapa informasi yang dibutuhkan, ukuran yang digunakan, serta pemilihan *metrics.*

a. Detail Informasi yang Dibutuhkan

   Dalam proses pengukuran kualitas dengan enam *object oriented metrics* pada Aplikasi Movie DB diperlukan berbagai hal antara lain.

   1. *Source Code* lengkap dari Aplikasi Movie DB dengan Bahasa Pemrograman Java (diperoleh langsung oleh Putra Pribowo).
   2. *Class.*
   3. Setiap atribut dan operasi dalam *class.*

4. Nama atribut.
5. *Method* setiap *class.*
6. *Subclass.*

b. Ukuran yang Digunakan

Informasi yang diperlukan untuk pengukuran Aplikasi Movie DB dengan enam *object oriented metrics* sehingga didapatkan hasil pengukuran berupa data kuantitatif dapat diketahui faktor-faktor ukuran kualitas yang berpengaruh untuk setiap *object oriented metrics* yang diukur yaitu sebagai berikut:

1. *Weight Methods per Class* (WMC) berpengaruh pada *maintainability (testability, usability,* dan *reusability.*
2. *Depth of Inheritance Tree* (DIT) berpengaruh pada *reusability, understandability, modifiability,* dan *testability.*
3. *Number of Children* (NOC) berpengaruh pada *reusability* dan *testability.*
4. *Coupling Between Object* (CBO) berpengaruh pada *reusability, understandability, modifiability,* dan *testability.*
5. *Response for A Class* (RFC) berpengaruh pada *understandability, modifiability,* dan *testability.*
6. *Lack of Cohesion in Methods* (LCOM) berpengaruh pada *understandability, modifiability,* dan *testability.*

3.3.3 Implementasi Perhitungan *Metrics*

Implementasi perhitungan *metrics* meliputi perhitungan Metrik Chidamber dan Kemerer yang terdiri dari Metrik WMC, DIT, NOC, CBO, RFC, dan LCOM. Setiap *class* akan diukur nilai metriknya dan dihitung rata-ratanya *(average).* Patokan nilai yang dapat dijadikan parameter dalam pengukuran di tahap ini terdapat pada pembahasan pada Tabel 2 berdasarkan referensi dari Chidamber dan Kemerer (1994) dalam penelitian (Ayubi, 2015) dan (Mitra, 2013). Penelitian ini melakukan perhitungan pengukuran secara manual.

3.3.4 Analisis Hasil Pengukuran

Analisis pengukuran dilakukan dengan cara membandingkan nilai enam *metrics* Aplikasi Movie DB, yaitu metrik WMC, metrik DIT, metrik NOC, metrik CBO, metrik RFC, dan metrik LCOM dengan nilai standar *metrics* yang dianjurkan oleh Chidamber & Kemerer (1994). Berdasarkan penelitian (Ayubi, 2015) pada Tabel 2, nilai metrik WMC yang baik adalah kurang dari 50, untuk kategori sedang berada dalam range 50-150, dan untuk kategori buruk adalah lebih dari 150. WMC yaitu menghitung jumlah *method* yang diimplementasikan dalam suatu *class. Class* dengan *method* yang banyak akan cenderung menjadi aplikasi atau perangkat lunak yang spesifik sehingga akan membatasi kemungkinan penggunaan kembali. (Ayubi, 2015). Chidamber dan Kemerer (1994) menyatakan bahwa objek yang memiliki jumlah *method* yang banyak, maka akan membatasi kemungkinan penggunaan ulang suatu *class*. Nilai WMC pada suatu *class* yang berada pada kategori baik dan sedang, maka kelas tersebut tidak perlu dipecah menjadi beberapa *class.*

Berdasarkan Tabel 2, nilai metrik DIT yang baik adalah kurang dari 5, untuk kategori sedang berada pada range 5 sampai kurang dari 8, dan kategori buruk adalah lebih dari 8. DIT didefinisikan sebagai pengukuran kedalaman penurunan suatu *class.* Peningkatan DIT dapat meningkatkan bug dan menurunkan kualitas perangkat lunak (Ayubi, 2015). Semakin dalam hirarki suatu *class*, semakin banyak jumlah method yang diturunkan, sehingga *class* tersebut akan semakin kompleks (Chidamber & Kemerer, 1994). Untuk DIT pada setiap kelas yang memiliki nilai yang berada di dalam range, maka masing-masing kelas tersebut tidak perlu mengalami penggabungan dengan kelas lain.

Berdasarkan Tabel 2, nilai metrik NOC yang baik adalah kurang dari 4, untuk kategori sedang berada pada range 4 sampai kurang dari 8, dan kategori buruk adalah lebih dari 8. NOC yaitu perhitungan jumlah *subclass* yang diturunkan

langsung. Sebuah class dengan NOC yang besar akan sulit dimodifikasi dan membutuhkan pengujian yang lebih karena berpengaruh pada perubahan *children* dari suatu *class* (Mitra, 2013). Untuk NOC pada setiap kelas memiliki nilai yang berada di dalam range, sehingga jumlah kelas turunan yang dimiliki oleh masing-masing kelas tidak perlu dikurangi.

Berdasarkan Tabel 2, nilai metrik CBO yang baik adalah kurang dari 14, untuk kategori sedang berada pada range 14-150, dan kategori buruk adalah lebih dari 150. Dua *class* dikatakan berpasangan ketika salah satu *method* yang didefinisikan dalam sebuah *class* digunakan oleh *class* yang lain. Jumlah CBO yang dimiliki setiap kelas masih berada dalam range. Hal ini mengindikasikan bahwa setiap kelas tidak perlu mengalami penggabungan dengan kelas-kelas yang memiliki relasi dengan kelas tersebut.

Berdasarkan Tabel 2, nilai metrik RFC yang baik adalah kurang dari 100, untuk kategori sedang berada pada range 100-1000, dan kategori buruk adalah lebih dari 1000. Nilai RFC yang tinggi akan meningkatkan kemungkinan banyak kesalahan, karena class dengan RFC tinggi akan lebih kompleks dan sulit dimengerti. RFC yang tinggi akan mengidentifikasi pengujian dan *debugging* yang rumit (Ayubi, 2015). Untuk jumlah RFC pada masing-masing kelas juga masih berada di bawah batas ambang. Sehingga *method* yang dimiliki oleh setiap kelas tidak perlu digabung dengan *method* lain (Mitra, 2013).

Berdasarkan Tabel 2 pada penelitian (Mitra, 2013), nilai metrik LCOM yang baik berada pada rentang 0 sampai 1. Suatu kelas memiliki nilai LCOM yang melebihi batas ambang, maka kelas tersebut dapat dibagi menjadi beberapa kelas turunan. Untuk kelas yang memiliki nilai LCOM nol sampai satu, maka kelas tersebut tidak perlu dibagi menjadi beberapa kelas turunan.

Dari hasil perhitungan, dicari nilai minimal, maksimal, dan rata-rata secara keseluruhan dari masing-masing *metrics.* Dari hasil keenam *object oriented metrics*

dapat dianalisis lebih lanjut seberapa baik Aplikasi Movie DB secara keseluruhan berdasarkan faktor-faktor kualitas yang terkait pada masing-masing *metrics*. Tinggi rendahnya faktor-faktor kualitas yang terkait pada masing-masing *metrics* dipengaruhi oleh tinggi rendahnya nilai *metrics.* Dengan demikian dapat diketahui seberapa tinggi nilai *maintainability, usability, reusability, understandbility, modifiability,* dan *testability* Aplikasi Movie DB dari nilai-nilai yang dipakai untuk pengukuran keenam *object oriented metrics* tersebut dan untuk mengetahui nilai hasil dari pengukuran tersebut.

V. SIMPULAN DAN SARAN

5.1 Simpulan

Berdasarkan penelitian yang telah dilakukan, maka dapat disimpulkan bahwa pengukuran *object oriented metrics* dengan enam metrik Chidamber dan Kemerer, yaitu *Weight Methods 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 Methods* (LCOM) yang dapat digunakan untuk mengetahui kualitas perangkat lunak berorientasi objek dari beberapa faktor kualitas pada Aplikasi Movie DB memiliki nilai metrik yang baik. Aplikasi Movie DB sudah memiliki komponen kualitas *object-oriented* yang baik dari segi *usability, reusability, understandability, modifiability,* dan *testability*.

5.2 Saran

Penelitian selanjutnya disarankan untuk dapat menggunakan *tools* untuk memastikan hasil perhitungan komponen *object-oriented* dalam *source code* sehingga hasil perhitungan Metrik Chidamber dan Kemerer lebih *valid*. Selain itu, penelitian selanjutnya dapat dilakukan perhitungan menggunakan metrik lain untuk membandingkan hasil dari Metrik Chidamber dan Kemerer.

DAFTAR PUSTAKA

Andrea, Reza, & Karim, Syafei. 2021. *Pemrograman Berorientasi Objek dengan Java.* Tanesa. Samarinda.

Amalarethinam, D. I. George & Hameed, P.H. Maitheen Shahul. 2015. Analysis of Object Oriented Metrics on a Java Application. *International Journal of Computer Applications (0975–8887)*, *123*, 32–39.

Ayubi, Aula. 2015. *Quality Measurement of Object Oriented Code Using Chidamber and Kemerer Metric in The Perspective of Maintainability, Efficiency, Understandability, and Replaceability (Case Studies Software Accounting XYZ)*. Institut Teknologi Sepuluh Nopember. Surabaya.

Chandrika, S. Megha, Babu, E. Suresh, & Srikanth, N. 2011. Conceptual Cohesion of Classes in Object Oriented Systems. *International Journal of Computer Science and Telecommunications*, 38-44.

Chidamber, Shyam R. & Kemerer, Chris F. 1994. A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*, *20*.

El-Ahmadi, Abdellatif. 2006. Software Quality Metrics For Object Oriented Systems. Kongens Lyngby: Technical University of Denmark.

Enterprise, Jubilee. 2015. *Mengenal Java dan Database dengan Netbeans*. PT Elex Media Komputindo, Jakarta.

Gomathi, S. & P, Edith Linda. 2013. An overview of Object Oriented Metrics A complete Survey. *International Journal of Computer Science & Engineering Technology (IJCSET)*, *4*, 1243–1247.

Harrison, R., Counsell, S., & Nithi, R. 1998. Coupling Metrics for Object-Oriented Design. *Proceedings of the 5th International Software Metrics Symposium, Bethesda, MD*, 150–157.

Henderson-Sellers, B. 1996. *Software Metrics*. Prentice Hall.

Letelay, Kornelis, & SN, Azhari. 2012. Evaluasi Kualitas Perangkat Lunak dengan Metrics Berorientasi Objek. *Seminar Nasional Informatika 2012 (semnasIF 2012) UPN "Veteran" Yogyakarta*, 139–145.

Lindroos, Jaana. 2004. Code and Design Metrics for Object-Oriented Systems. Helsinski: Department of Computer Science, University of Helsinsky.

Marpaung, Wati Margaretha. 2015. *Pembuatan Kakas Bantu Pengukuran Kualitas Perangkat Lunak pada Kode Pemrograman Java*. Institut Teknologi Sepuluh Nopember. Surabaya.

Mitra, A. 2013. *Analisis Sistem Informasi Data Nilai Siswa Berbasis PHP di SMK YPKK 1 Sleman*. Universitas Negeri Yogyakarta. Yogyakarta.

Nurdin, Muhammad Syarif. 2013. *Analisis dan Perancangan Sistem Informasi Pelayanan Rawat Jalan Puskesmas Kota Surabaya dengan Metode Metrik Berorientasi Objek (Studi Kasus pada Puskesmas Klampis Ngasem Surabaya)*. Universitas Airlangga. Surabaya.

Rawat, Mrinal S., Mittal, Arpita, & Dubey, Sanjay Kumar. 2012. Survey on Impact of Software Metrics on Software Quality. (IJACSA) *International Journal of Advanced Computer Science and Applications, 3*, 137–141.

Rhamdani. 2008. Evaluasi Kualitas Perangkat Lunak Berorientasi Objek. Institut Pertanian Bogor. Bogor.

Rosenberg, Linda H. 1998. Applying and Interpreting Object Oriented Metrics. Utah: SATC NASA.

Rosenberg, Linda H. 2001. Software Quality Metrics for Object Oriented System Environments. NASA Technical, 11–58.

Rusli, M., Negara, I Komang Rinartha Yasa, & Atmojo, Yohanes Priyo. 2016. *Belajar Pemrograman Java dengan Netbeans Sebuah Pengantar*. Adi. Yogyakarta.

Saini, Neha, Kharwar, Sapna, & Agrawal, Anushree. 2014. A Study of Significant Software Metrics. *International Journal of Engineering Inventions*, vol. 3, Issue 12, 01–07.

Sarker, Muktamyee. 2005. An overview of Object Oriented Design Metrics. Department of Computer Science, Umea University, Swaden.

Septian, Wahyu Rifa'i Dwi. 2010. *Analisis Perbandingan Framework PHP Disain Menggunakan Metode Analytic Hierarchy Process (AHP)*. Universitas Islam Negeri Syarif Hidayatullah. Jakarta.

Sharma, Aman K., Kalia, Arvind, & Singh, Hardeep. 2012. Metrics Identification for Measuring Object Oriented Software Quality. *International Journal of Soft Computing and Engineering (IJSCE), 2*(5), 255–258.

Simarmata, Janner. 2010. *Rekayasa Perangkat Lunak*. Andi. Yogyakarta.

Systa, Tarja, & Yu, P. (1999). Using OO Metrics and Rigi to Evaluate Java Software. Tampere: Department of Computer Science, University of Tampere.