KINETIK, Vol. 2, No. 1, Februari 2017, Hal. 53-62
ISSN : 2503-2259,
E-ISSN : 2503-2267

Kakas Bantu Perhitungan Nilai Kopling Menggunakan Metrik Cognitive Weighted Coupling Between Object

Muhammad Ubaidillah*[1], Fajar Pradana[2], Bayu Priyambadha[3]
[1,2,3]Universitas Brawijaya
ubaidillah.m26@gmail.com*[1], fajar.p@ub.ac.id[2], bayu_priyambadha@ub.ac.id[3]

Abstrak

Konsep Object Oriented Programming (OOP) merupakan pemrograman yang dibangun dengan berpusat pada beberapa objek. Dengan konsep OOP dapat dilakukan pengukuran kualitas perangkat lunak dengan melalui kemungkinan hubungan antar beberapa objek atau kopling. Perangkat lunak yang baik adalah perangkat lunak yang memiliki desain yang baik, salah satu ciri-cirinya adalah memiliki nilai kopling yang rendah. Nilai kopling yang tinggi dapat menyebabkan desain perangkat lunak yang semakin kompleks dan buruk, sehingga akan mengakibatkan sulit untuk dipahami, terutama saat maintenance. Untuk meningkatkan kualitas perangkat lunak perlu mengontrol nilai kopling agar dapat meminimalkan kompleksitas perangkat lunak. Namun perhitungan nilai kopling secara manual pada jumlah perangkat lunak yang banyak akan membutuhkan waktu dan sumber daya yang besar. Oleh karena itu dibuatlah kakas bantu perhitungan nilai kopling menggunakan metrik Cognitive Weighted Coupling Between Object (CWCBO). Metrik CWCBO merupakan metrik kopling yang didasarkan pada bobot pemahaman untuk menghitung perbedaan jenis kopling dari berbagai peneliti. Kakas bantu tersebut dibangun dengan bahasa pemrograman utama Java, library Spoon untuk menganalisis source code, dan library JFreeChart untuk menampilkan grafik. Dari hasil pengujian akurasi sistem pada 5 program inputan berbahasa Java didapatkan akurasi sebesar 100%. Pengujian akurasi dilakukan dengan membandingkan perhitungan secara manual dan dengan sistem.

Kata kunci: *Object Oriented Programming, Kopling, Cognitive Weighted Coupling Between Object, Kode sumber*

Abstract

This concept could help to measure the software quality through a possible connection between multiple objects or coupling. A good software can be seen from its design; one of the characteristics is having a low value of coupling. High coupling value could affect the software design, make it even more complex and worse, which possibly lead to understanding difficulty, especially during maintenance time. To improve the software's quality, it is necessary to control coupling's value in order to minimize its complexity. However, the manual calculation of coupling's value on the sizeable amount of software will take an ample time and resources. Therefore, a tool to calculate coupling's value was made using Cognitive Weighted Coupling Between Object (CWCBO) metric. CWCBO metric is a coupling metric that is made based on the understanding weight to count the different types of coupling by various researchers. This tool was built by using primary component from Java programming language, Spoon library, to analyze the source code and JFreeChart library for displaying charts. From the experiment on system acuracy of fiva Java program inputs, we found that the accuracy is 100%. The accuracy testing is conducted by comparing manual calculation and system-based calculation.

Keywords: *Object Oriented Programming, Coupling, Cognitive Weighted Coupling Between Object, Source code*

1. Pendahuluan

Rekayasa Perangkat Lunak merupakan suatu cabang ilmu profesi tentang teknik-teknik dalam mengembangkan perangkat lunak, mulai dari perencanaan, pembuatan, pengujian hingga pemeliharaan. Rekayasa perangkat lunak merupakan tugas yang rumit dan [1] struktur perangkat lunak yang kompleks bisa berdampak pada kualitas perangkat lunak yang buruk, karena semakin sulit untuk dipahami saat proses *maintenance*. Sehingga banyak para pengembang berupaya

Pradana, F., Priyambadha, B., & Ubaidillah, M. (2017). Kakas Bantu Perhitungan Nilai Kopling Menggunakan Metrik Cognitive Weighted Coupling Between Object (CWCBO). KINETIK, 2(1), 53-62. doi:http://dx.doi.org/10.22219/kinetik.v2i1.153
Makalah dikirim 30 Desember 2016; Revisi 30 Januari 2017; Diterima 1 Februari 2017

menciptakan teknik-teknik untuk mempermudah dalam proses pengembangan dan *maintenance*. Salah satu upayanya adalah menerapkan model-model pengembangan perangkat lunak, seperti pendekatan berorientasi objek dan pendekatan terstruktur.

Lebih dari dua dekade terakhir, penggunaan teknik berbasis objek lebih dominan dibandingkan dengan penggunaan teknik terstruktur untuk mengembangkan perangkat lunak [2]. Perangkat lunak yang dibangun dengan menggunakan konsep *Object Oriented Programing* (OOP) akan dilakukan pemrograman yang berpusat pada beberapa objek. Pemanfaatan konsep OOP dapat dilakukan pengukuran kualitas suatu perangkat lunak berdasarkan kemungkinan hubungan antar *atribute* dan *method* yang terdapat pada masing-masing *class* [3]. Proses ini dapat dilakukan sebagai alat dalam memutuskan kualitas suatu perangkat lunak pada tahap implementasi berdasarkan nilai kopling antar objek.

Perangkat lunak yang baik adalah perangkat lunak yang memiliki desain baik. Salah satu ciri-ciri desain yang baik adalah memiliki nilai kopling yang rendah. Nilai kopling mempunyai dampak negatif pada kualitas perangkat lunak [4]. Apalagi kopling kelas yang tinggi dianggap sebagai desain yang buruk dan dapat menyebabkan kesulitan dalam memahami kelas [5]. Perangkat lunak yang sulit dipahami juga menyebabkan kesulitan developer untuk melakukan *maintenance* pada sistem. Namun, jika perangkat lunak tersebut memiliki desain yang baik, maka akan lebih mudah untuk dipahami, sehingga mudah dilakukan pengelolaan. Untuk meningkatkan kualitas perangkat lunak harus mempertimbangkan nilai kopling untuk meminimalkan kompleksitas perangkat lunak [6][7].

Nilai kopling dapat menentukan tingkat kualitas dari desain perangkat lunak, maka diperlukan suatu perhitungan nilai kopling yang dapat menentukan kualitas perangkat lunak berdasarkan relasi-relasi antar objek dan juga dapat mendefinisikan tingkat kesulitan dalam memahami hubungan antar kelas pada suatu program. Oleh karena itu, dipilih perhitungan kualitas desain berorientasi objek menggunakan metrik *Cognitive Weighted Coupling Between Object* (CWCBO). Metrik tersebut dipilih karena menurut [1], belum pernah ada metrik kopling yang didasarkan pada bobot pemahaman untuk menghitung perbedaan jenis kopling dari berbagai peneliti, sehingga metrik CWCBO yang dihitung berdasarkan bobot pemahaman diharapkan dapat mendefinisikan kopling pada berbagai tingkatan. Adanya perhitungan tersebut dirasa masih kurang efektif dan efisien jika dilakukan perhitungan secara manual dengan jumlah perangkat lunak yang banyak. Selain membutuhkan waktu yang lama, proses tersebut juga membutuhkan sumber daya yang besar. Terutama saat menentukan nilai kopling tersebut dibutuhkan sumber daya manusia yang benar-benar paham dengan model perhitungannya, sedangkan untuk mendapatkan sumber daya tersebut tidaklah mudah.

Oleh karena itu, berdasarkan permasalahan yang telah dipaparkan, dibuatlah suatu program yang dapat melakukan otomatisasi perhitungan nilai kopling, sehingga diharapkan dapat membantu para pengembang dalam menentukan tingkat kualitas perangkat lunak secara efektif dan efisien. Perhitungan kualitas desain *object-oriented* menggunakan metrik kopling *Cognitive Weighted Coupling Between Object* (CWCBO) dan dibangun menggunakan bahasa pemrograman Java. Dengan adanya sistem perhitungan kualitas ini nanti diharapkan dapat menjadi alternatif bagi *user* dalam mengembangkan perangkat lunak yang lebih baik. Terutama *source code* yang telah dibuat *user* akan menjadi lebih baik dan mudah untuk dipahami.

2. Metode Penelitian

2.1 Metrik Kopling

Metrik kopling menunjukkan hubungan dan interaksi elemen-elemen antar objek pada suatu perangkat lunak. Semakin tinggi nilai kopling, maka hubungan antar objek atau modul pada suatu perangkat lunak semakin kuat dan semakin kompleks. Nilai kopling yang tinggi dapat berakibat semakin buruknya kualitas desain perangkat lunak, karena semakin banyak pertukaran pesan antar objek [8]. Selain itu desain perangkat lunak yang buruk juga dapat menyebabkan kesulitan untuk memahami kelas [5].

2.2 *Coupling Between Object*

Coupling Between Object (CBO) merupakan metrik perhitungan nilai kopling yang pertama kali diperkenalkan oleh Chidamber dan Kamerer (CK) [1] berpendapat metrik CBO hanya menghitung kopling luar dari kelas lain dengan cara menggabungkan dua kelas. Nilai yang diberikan untuk kopling ditetapkan dengan nilai 1. Sehingga (CK) mengusulkan penelitian dengan berbagai nilai-nilai kopling lain juga. Seperti yang diusulkan oleh Edward Berard (Edward, 1993 dalam Aloysius & Arockiam, 2012), mengusulkan berbagai macam kopling yang didefinisikan sebagai berikut:

1. *Control Coupling* (CC)
   Melewati penanda kontrol antar modul, sehingga satu modul mengontrol pengurutan langkah-langkah dari modul lain. *Control coupling* terjadi antar modul dimana ketika data yang lewat dapat mempengaruhi logika internal pada salah satu modul, seperti penanda [9].
2. *Global Data Coupling* (GDC)
   Dua atau lebih modul yang menggunakan struktur data global yang sama. Kondisi ini terjadi ketika pada objek/kelas *parent* terdapat suatu variabel global dimana variabel tersebut juga digunakan pada kelas atau objek yang lain.
3. *Internal Data Coupling* (IDC)
   Suatu modul yang secara langsung memodifikasi data lokal dari modul yang lain. Pada sistem berorientasi objek, internal data *coupling* dapat terjadi ketika suatu objek memanggil suatu *method* yang sama pada objek yang berbeda, dimana *method* yang dipanggil memiliki modifikasi yang berbeda terhadap variabel global yang sama di setiap objek. Pernyataan yang berada dalam tanda kurung diberi tanda "titik" di "luar kurung" penutupnya (seperti ini). (Sebuah tanda titik). Hindari penggunaan singkatan, seperti contoh penulisan "yang", bukan "yg". Tanda koma serial lebih disarankan: "A, B, dan C" dan bukan "A, B and C."
4. *Data Coupling* (DC)
   Output dari satu modul merupakan suatu penambahan data pada modul yang lainnya menggunakan daftar parameter untuk melewatkan *item* antar *routine*.
5. *Control Coupling* (CC)
   Beberapa atau semua isi dari suatu modul yang termasuk dalam isi yang lainnya. *Lexical content coupling* pada sistem berorientasi objek dapat terjadi ketika dua atau lebih objek melakukan modifikasi yang berbeda pada variabel global yang sama.

2.3 Kalibrasi Penilaian Bobot

Dalam menentukan bobot pemahaman pada berbagai tipe kopling yang diusulkan oleh Edward Berard, Aloysius menyelenggarakan sebuah *test* pemahaman kepada 30 mahasiswa yang memiliki kemampuan cukup dalam menganalisis pemrograman berorientasi objek dan bahasa Java serta mendapatkan nilai 65% lebih pada setiap ujian semester. *Test* tersebut ditujukan untuk mencari waktu yang dibutuhkan dalam memahami kompleksitas dari program berorientasi objek dengan memperhatikan perbedaan macam kopling. Setiap kasus atau kategori disediakan dua program berorientasi objek yang berbeda, jadi total keseluruhan terdapat 10 macam kode program. Waktu yang diperoleh mahasiswa pada setiap kode program dirata-rata berdasarkan tiap-tiap kategori. Kategori waktu pemahaman dapat dilihat pada Tabel 1.

Tabel 1. Kategori Waktu Pemahaman

Program	Rata-rata Waktu Pemahaman	Kategori	Rata-rata waktu pemahaman
1	40.7	LCC	40.18333
2	39.66667		
3	30.76667	DC	30.88333
4	31		
5	21.43333	IDC	22.21667
6	23		
7	10.8	GDC	11.13333
8	11.46667		

2.4 *Cognitive Weighted Coupling Between Object*

Metrik *Cognitive Weighted Coupling Between Object* (CWCBO) yang diusulkan oleh Aloysius merupakan perpaduan antara metrik yang diusulkan oleh Edward Berard dan penggunaan teori bobot berdasarkan waktu pemahaman pada setiap kategori dalam penelitian yang dilakukan oleh Aloysius. Metrik CWCBO dapat dihitung menggunakan Persamaan 1.

$$cwcbo = ((CC * WFCC) + (GDC * WFGDC) + (IDC * WFIDC) + (DC * WFDC) + (LCC * WFLCC) \quad (1)$$

Keterangan:
CC : Jumlah dari modul yang berisi *Control Coupling.*
GDC : Jumlah Global Data *Coupling*
IDC : Jumlah Internal Data *Coupling*

DC : Jumlah Data *Coupling*
LCC : Jumlah *Lexical Content Coupling*

Penilaian faktor bobot pada setiap jenis kopling didasarkan pada hasil kalibrasi yang dilakukan Aloysius, yaitu pengukuran rata-rata waktu pemahaman mahasiswa pada setiap kode program dengan kategori-kategori yang telah ditentukan. Pemberian nilai bobot pada setiap jenis kopling dapat dilihat pada Tabel 2.

Tabel 2. Faktor Bobot Pada Setiap Kopling

No	Faktor Bobot	Bobot	Keterangan
1	WFCC	1	Faktor bobot *Control Coupling*
2	WFGDC	1	Faktor bobot *Global Data Coupling*
3	WFIDC	2	Faktor bobot *Internal Data Coupling*
4	WFDC	3	Faktor bobot *Data Coupling*
5	WFLCC	4	Faktor bobot *Lexical Data Coupling*

2.5 *Spoon*

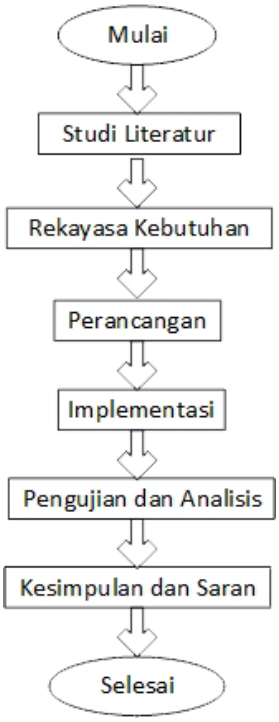
Spoon merupakan sebuah *library* yang dapat digunakan untuk menganalisis dan mengubah kode program dengan bahasa Java. *Spoon* memungkinkan pengembang untuk menulis dengan cakupan besar dalam menganalisis domain yang spesifik dan mengubah dengan mudah sesuai kebutuhan [10].

2.6 JFreeChart

JFreeChart merupakan sebuah *library* gratis yang digunakan oleh para *developer* untuk menampilkan diagram pada aplikasinya. JFreeChart pertama kali dibuat oleh David Gilbert pada Februari 2000. Hingga saat ini, JFreeChart menjadi *library* diagram untuk Java yang paling banyak digunakan oleh para *developer*. Hal itu dapat dilihat pada situs resmi JFreeChart, bahwa lebih dari 2,2 juta pengguna yang telah mengunduhnya [3].

3. Metodologi

Metodologi penelitian ini dilakukan dalam beberapa tahapan, yaitu studi literatur, rekayasa kebutuhan, perancangan, implementasi, pengujian dan analisis, serta kesimpulan dan saran. Gambar 1 menunjukkan tahapan-tahapan dalam penelitian ini.



Gambar 1. Metodologi Penelitian

3.1 Studi Literatur

Teori-teori mengenai metode sistematika pembangunan kakas bantu perhitungan kopling menggunakan metrik *Cognitive Weighted Coupling Between Object* (CWCBO) menjadi dasar penelitian yang diperoleh dari buku, jurnal, ebook, penelitian sebelumnya, situs internet serta literatur lain yang berkaitan. Teori pustaka yang berkaitan dengan penelitian ini, meliputi metrik kopling, *source code, spoon*, JfreeChart, dan konsep dasar berorientasi objek. Model pengembangan Waterfall, konsep dasar rekayasa perangkat lunak.

3.2 Rekayasa kebutuhan

Rekayasa kebutuhan dilakukan untuk mengetahui kebutuhan apa saja yang diperlukan dalam pembangunan kakas bantu perhitungan kopling menggunakan metrik *cognitive weight coupling between object* (CWCBO). Dalam melakukan rekayasa kebutuhan akan didapatkan kebutuhan fungsional dan kebutuhan non-fungsional. Salah satu contoh kebutuhan fungsional adalah sistem dapat menghitung nilai kopling dari *souce code* yang ditambahkan.

3.3 Perancangan sistem

Perancangan perangkat lunak terdiri dari beberapa tahapan diantaranya adalah perancangan arsitektur, perancangan *class* diagram, perancangan *sequence* diagram dan perancangan antarmuka. Perancangan arsitektur digunakan untuk mengetahui gambaran kinerja sistem secara keseluruhan. Perancangan *class* diagram digunakan untuk mengidentifikasi kelas-kelas yang berada dalam sistem. Perancangan *sequence* diagram digunakan untuk menjelaskan interaksi antar objek yang disusun dalam urutan waktu. Perancangan antarmuka digunakan sebagai media interaksi antara pengguna dan sistem.

3.4 Implementasi sistem

Implementasi sistem merupakan proses penerapan pembuatan aplikasi berdasarkan pada proses analisis dan perancangan yang telah dilakukan pada tahap sebelumnya. Implementasi perangkat lunak dilakukan dengan menggunakan bahasa pemrograman berorientasi objek, yaitu menggunakan bahasa pemrograman Java dengan *software* Netbeans IDE 8.0.

3.5 Pengujian dan Analisis

Strategi pengujian perangkat lunak yang digunakan, yaitu pengujian unit, pengujian integrasi, pengujian validasi, dan pengujian akurasi. Metode pengujian yang digunakan adalah White-Box Testing dan Black-Box Testing. Pada tahap pengujian unit digunakan metode White-Box Testing dengan teknik *basis path*. Sementara untuk pengujian integrasi, akurasi dan validasi digunakan metode Black-Box Testing. Kemudian dilakukan analisis dari hasil pengujian perangkat lunak sehingga dapat diperoleh kesimpulan dari pembuatan perangkat lunak yang telah dilakukan.

3.6 Pengambilan Kesimpulan

Pengambilan kesimpulan dilakukan setelah dilakukan proses pengujian dan analisis sistem sehingga dapat diketahui efektivitas kinerja dari sistem. Kesimpulan dibuat berdasarkan hasil pengujian dan analisis sehingga diperoleh inti dari keseluruhan proses penelitian. Pengambilan kesimpulan bertujuan untuk menjawab permasalahan yang ada dalam rumusan masalah. Saran digunakan untuk memberi masukan untuk menjadi bahan pertimbangan untuk pengembangan sistem selanjutnya. Tahap terakhir, yaitu penulisan laporan yang dapat membantu dalam pengembangan sistem selanjutnya.

4. Hasil dan Pembahasan

4.1 Hasil Perancangan

Perancangan arsitektur digunakan untuk menggambarkan struktur keseluruhan sistem sehingga sistem dapat terorganisir [5]. Perancangan arsitektur menggunakan notasi-notasi untuk menggambarkan langkah-langkah peneliti yang digunakan dalam pembangunan sistem.

Perancangan alur *parse source code* menjelaskan tentang alur bagaimana menganalisis data input yang berupa *source code* program. Perancangan ini bertujuan untuk mendapatkan nilai dari lima tipe kopling yang akan menjadi parameter dalam perhitungan nilai kopling CBO dan CWCBO. Kelima kopling tersebut adalah *Control Coupling* (CC), *Global Data Coupling* (GDC), *Internal Data Coupling* (IDC), *Data Coupling* (DC), dan *Lexical Content Coupling* (LCC). Alur dalam mendapatkan nilai kelima kopling tersebut adalah sebagai berikut:

1. Mencari semua *class* pada *file* Java yang diinputkan.
2. Mencari *parent class* beserta variabelnya.
3. Mencari *method* dari *class* yang merupakan *class* turunannya.
4. Menelusuri setiap *method* berdasarkan *statement*. Jika terdapat variabel yang sama dengan variabel yang ada pada parentclass maka variabel tersebut merupakan GDC.
5. Mencari *statement* yang merupakan percabangan *if else*.
6. Menelusuri setiap parameter kondisi yang ada. Jika parameter kondisinya terdapat variabel yang sama dengan GDC yang dicari sebelumnya, maka variabel tersebut termasuk ke dalam DC.
7. Mencari *statement else* (tidak terdapat parameter kondisi) pada percabangan *if else*. Jika parameter kondisi pada *statement if else* sebelumnya terdapat DC yang dicari sebelumnya, maka *statement else* tersebut merupakan CC.
8. Menelusuri setiap *method* yang lain berdasarkan *statement*. Jika terdapat variabel yang sama dengan GDC namun memiliki *class* yang berbeda, maka variabel tersebut merupakan LCC.
9. Mencari *statement* yang merupakan *statement* operasi. Jika *left assigned* (variable yang berada di sebelah kiri simbol operator, seperti "+=") sama dengan GDC, maka variabel tersebut disimpan sementara pada suatu variabel penyimpanan, diasumsikan variabel penyimpanannya tempIDC.10. Menelusuri setiap *method* berdasarkan *statement* pada *class* yang lainnya. Mencari *statement* yang merupakan *statement* operasi. Jika *left assigend* sama dengan tempIDC, *variable* tersebut disimpan sementara pada suatu variabel penyimpanan, diasumsikan variabel penyimpanannya dataIDC.
10. Menelusuri parentclass. Jika ada *method* pada parentclass yang berisi dataIDC, dimana dataIDC tersebut merupakan leftassigned dalam *statement* persamaan, maka *method* tersebut disimpan dalam variabel *method*WithIDC.
11. Mencari class yang mengandung main *method*.
12. Menelusuri main *method*. Jika pada main *method* terdapat lebih dari satu *instance* objek dari parentclass, dan setiap objek memanggil *method*WithIDC, maka variabel dalam dataIDC merupakan IDC.

Perancangan algoritma menjelaskan secara detail algoritma *method* terdapat pada suatu class yang telah dimodelkan pada class diagram. Salah satu perancangan algoritma yang dibuat adalah algoritma *method* check*Method*Parent dalam *class* SPOON_metamodel. Perancangan algoritma *method* check*Method*Parent dapat dilihat pada Tabel 3.

Tabel 3. Algoritma Method Checkmethodparent

Algoritma *method* check *Method* Parent()
check*Method*Parent(*method*Name, *method*Parent) { mengulang sejumlah *method*Parent variabel get = *method*Parent terpilih jika *method*Name = get return true return false

4.2 Hasil dan Analisis Pengujian Validasi

Proses pengujian validasi dilakukan dengan melihat kesesuaian antara fungsi hasil kerja sistem dengan daftar kebutuhan sistem. Hasil pengujian validasi dapat dilihat pada Tabel 4.

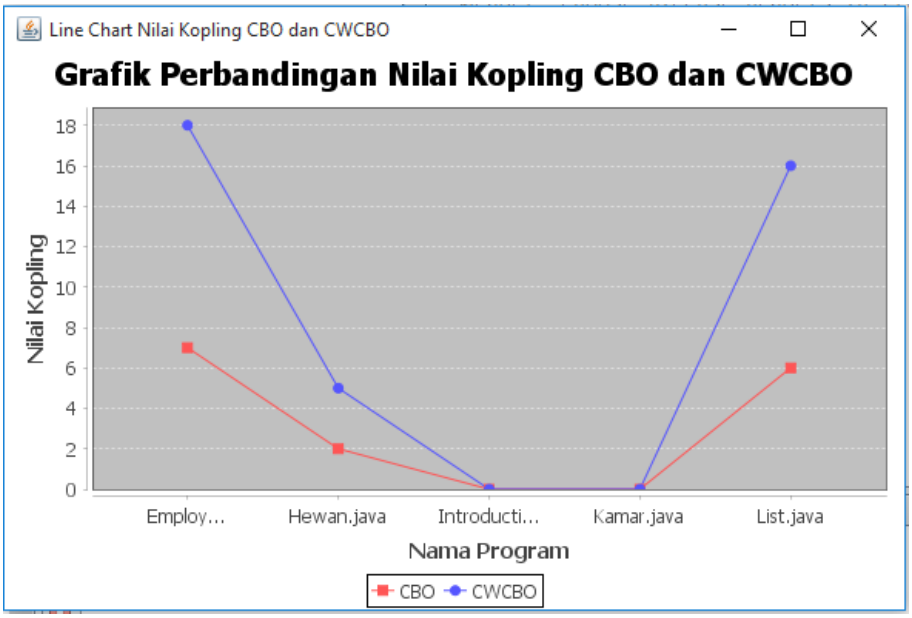
Berdasarkan hasil pengujian fungsional pada Tabel 4, maka dapat disimpulkan bahwa sistem telah memenuhi semua kebutuhan fungsional. membandingkan hasil perhitungan nilai kopling.

Tabel 4. Pengujian Validasi

No.	Kasus Uji	Hasil yang didapatkan	Status
1.	Uji Cari Berkas	Sistem dapat menampilkan Windows bagi *user* untuk memasukkan *file*. Sistem dapat menerima beberapa data yang berupa *file* berektensi Java	Valid
2.	Uji Cari Berkas Alternatif 1	Sistem dapat membatalkan penambahan data *file* ketika *user* menekan tombol *cancel*	Valid
3.	Uji Hitung Kopling	Sistem dapat menampilkan hasil perhitungan kopling dari beberapa data *input* kode program	Valid
4.	Uji Hitung Kopling Alternatif 1	Sistem dapat menampilkan pesan "*File* kosong, masukkan *file* yang akan diuji!" ketika *user* menekan tombol hitung kopling tanpa memasukkan *file* terlebih dahulu	Valid
5.	Uji Tampilkan Grafik	Sistem dapat menampilkan grafik perbandingan nilai kopling CBO dan CWCBO sesuai perhitungan kopling sebelumnya	Valid
6.	Uji Tampilkan Grafik *alternative* 1	Sistem dapat menampilkan pesan "*File* kosong, masukkan *file* yang akan diuji!" ketika *user* menekan tombol tampikan grafik tanpa memasukkan *file* atau melakukan perhitungan kopling terlebih dahulu	Valid

4.3 Hasil dan Analisis Pengujian Akurasi

Pengujian akurasi merupakan proses pengujian yang secara manual dan hasil perhitungan nilai kopling secara otomatis menggunakan sistem. Dalam proses pengujian akurasi menggunakan lima program dengan bahasa Java yang akan dijadikan sebagai inputan sistem. Program-program tersebut diambil dari sampel beberapa proyek mata kuliah. Hasil perhitungan nilai kopling dengan sistem dapat dilihat pada Gambar 2.



Gambar 2. Grafik Perbandingan CBO dan CWCBO

Dari grafik perbandingan nilai kopling tersebut menunjukkan bahwa metrik kopling CWCBO mempunyai nilai yang *relative* lebih tinggi dibandingkan dengan metrik kopling CBO, hal ini didasari karena metrik CWCBO telah ditambahkan bobot *cognitive* (pemahaman) pada setiap jenis kopling yang menjadi parameternya. Setiap jenis kopling yang menjadi parameter metrik tersebut mempunyai bobot yang berbeda sesuai dengan tingkat kesulitan untuk dipahami. Sehingga dua program yang berbeda dengan nilai CBO yang sama memungkinkan untuk menghasilkan nilai CWCBO yang berbeda, jika jumlah kopling pada masing-masing parameter berbeda. Setelah dilakukan perhitungan kopling secara manual dan secara otomatis menggunakan *system* pada Gambar 2, maka dapat diperoleh hasil perbandingan nilainya yang dapat dilihat pada Tabel 5.

Tabel 5. Hasil Pengujian Fungsional

Nama Program	Secara Manual		Dengan Sistem		Hasil
	CBO	CWCBO	CBO	CWCBO	
Employee.java	7	18	7	18	Valid
Hewan.java	2	5	2	5	Valid
Introduction.java	0	0	0	0	Valid
Kamar.java	0	0	0	0	Valid
List.java	6	16	6	16	Valid

Dari kasus pengujian yang telah dilaksanakan sesuai dengan prosedur pengujian akurasi, yaitu membandingkan hasil perhitungan secara menual dan dengan sistem pada lima program diperoleh hasil akurasi sebesar 100%. Hal ini dapat disimpulkan, bahwa pengujian akurasi yang dilakukan telah berhasil karena nilai akurasi lebih besar dari pada nilai batas minimalnya, yaitu 90%.

4.4 Pembahasan Hasil

Berdasarkan hasil perhitungan nilai kopling yang telah dilakukan pada lima program, seperti pada Gambar 2, diketahui bahwa dua program menghasilkan nilai kopling CBO dan CWCBO=0. Kedua program tersebut adalah Introduction.java dan Kamar.java. Hal tersebut dapat dikatakan bahwa kedua program dianggap mempunyai desain yang baik karena memiliki kopling rendah, sehingga mudah untuk dipahami. Sedangkan program dengan nama Employee.java memiliki nilai kopling tertinggi, yaitu CBO=8 dan CWCBO=18. Sehingga program Employee.java dapat dianggap mempunyai desain perangkat lunak yang buruk, sehingga akan lebih sulit untuk dipahami dibandingkan program Introduction.java dan Kamar.java.Nilai kopling CWCBO relatif lebih tinggi dibandingkan dengan nilai kopling CBO. Hal ini dikarenakan pada perhitungan kopling CWCBO ditambahkan bobot pemahaman pada setiap kategori dengan nilai bobot paling kecil 1 dan paling besar 4. Suatu program memungkinkan mempunyai nilai kopling CBO dan CWCBO yang sama, jika program tersebut hanya mengandung jenis-jenis kopling dengan bobot yang paling rendah, yaitu *control coupling* dan global data *coupling.* Keduanya memiliki bobot pemahaman dengan nilai 1.

5. Kesimpulan

Dalam penelitian pembangunan kakas bantu perhitungan nilai kopling menggunakan metrik *cognitive weighted coupling between object* (CWCBO) yang dilakukan oleh peneliti, terdapat beberapa hal yang dapat disimpulkan, yaitu:

1. Kakas bantu perhitungan nilai kopling mempunyai 3 fitur, yaitu cari berkas, hitung kopling, dan tampilkan grafik. Sistem ini dibangun dengan menggunakan 4 class, yakni Controller, Main_UI, ResultCoupling, dan SPOON_metamodel. Sistem ini diimplementasikan menggunakan bahasa pemrograman utama Java, library Spoon, dan libraryJFreeChart. Cara kerja sistem, pertama sistem akan menerima inputan berupa *source code* program berbahasa Java, kemudian sistem akan menganalisis *source code* dengan library Spoon untuk mendapatkan jumlah dari masing-masing tipekopling yang menjadi parameter dari metrik CWCBO. Setelah itu sistem akan menampilkan grafik perbandingan nilai kopling CBO dan CWCBO dengan pemanfaatan library JFreeChart.
2. Pengujian akurasi pada sistem ini dilakukan dengan menguji tingkat akurasi antara perhitungan yang dilakukan secara manual dan perhitungan yang dilakukan dengan sistem. Dari pengujian pada lima program yang menjadi data ujinya menghasilkan nilai kopling CBO dan CWCBO sama semua, sehingga nilai akurasinya adalah 100%. Dengan rincian dua program memiliki nilai kopling 0 dan nilai kopling tertinggi adalah 18.

Referensi

[1] A. Aloysius and L. Arockiam, “Coupling Complexity Metric: A Cognitive Approach,” International Journal of Information Technology and Computer Science (IJITCS), Vol. 4, No. 9, Pp. 29–35, 2012.

[2] S. Misra And I. Akman, “Weighted Class Complexity: A Measure Of Complexity For Object Oriented System,” Journal of Information Science and Engineering, Pp. 1689–1708, 2008.

[3] E. Lestari, Perhitungan Kualitas Desain Object-Oriented Menggunakan Matrix Similarity Based Class Cohesion (SCC) Pada Kelas Kohesi Berbasis Web. Program Teknologi Informasi Dan Ilmu Komputer, Universitas Brawijaya, 2015.

[4] A. Yadav and R. Khan, “Measuring Design Complexity: An Inherited Method Perspective,” ACM SIGSOFT Software Engineering Notes, Vol. 34, No. 4, Pp. 1–5, 2009.
[5] I. Sommerville, (2011). “Software Engineering.” Ninth Edition. Boston: Pearson Education, Inc.,.Stevens, W., Myers, G., & Constantine, L. (1974). “Structured Design.” IBM Systems Journal.
[6] A. Yadav And R. Khan, “Complexity: A Reliability Factor,” In Advance Computing Conference (IACC), Patiala, India, 2009, Pp. 2375–2378.
[7] B. V Edward, “Essays On Object-Oriented,” Prentice-Hall: Berard Software.Gilbert, David. Vol.1, 2005, 1993. [Online]. Available: Http://Www.Jfree.Org/Jfreechart/. [Accessed: 22-Jul-2016].
[8] P. Coad And E. Yourdon, Object Oriented Analysis, Vol. 2. Prentice-Hall,New Jersey, 1991.
[9] C. Borysowich, “Design Principles: Coupling (Data And Otherwise),” 2007. [Online]. Available: Http://It.Toolbox.Com/Blogs/Enterprisesolutions/Design-Principles-Coupling-Data-Andotherwise-16061. [Accessed: 22-Mar-2016].
[10] R. Pawlak, M. Monperrus, And N. Petitprez, “Spoon: A Library For Implementing Analyses And Transformations Of Java Source Code,” Software: Practice and Experience, 2015.