IDENTIFIKASI PELUANG REFAKTORISASI PADA KASUS GOD CLASS BERDASARKAN KESAMAAN KONSEP

Widhy Hayuhardhika Nugraha Putra[1], Siti Rochimah[2], Rizky Januar Akbar[3]

Abstrak

Sebuah kelas dalam Pemrograman Berbasis Objek (PBO) harus memiliki desain yang spesifik untuk menangani dan mengimplementasikan sebuah konsep. Kelas harus memiliki tanggungjawab spesifik terhadap sebuah konsep yang diimplementasikan dalam operasi. Pada pengembangannya, seringkali pengembang perangkat lunak tidak memperhatikan desain konsep kelas sehingga kelas menjadi berkembang dan memiliki implementasi yang kompleks atau biasa disebut dengan "God Class". Dalam hal ini diperlukan metode untuk mengidentifikasi peluang refaktorisasi God Class menjadi beberapa kelas baru yang nantinya akan direfaktorisasi menggunakan metode refaktorisasi kelas seperti ExtractClass. Pada penelitian sebelumnya, telah diusulkan metode identifikasi peluang refaktorisasi kelas menggunakan tiga pembobotan yaitu SSM (Structural Similariy between Methods), CDM (Call based Dependency between Methods) dan CSM (Conceptual Similarity between Methods).

Penelitian ini mengajukan metode dalam mengidentifikasi peluang refaktorisasi kelas dengan menghitung tingkat kedekatan konsep pada operasi. Operasi pada kelas diekstrak menjadi corpus dan dihitung nilai kedekatannya dengan operasi yang lain menggunakan SSM, CDM, dan CSM. Pada perhitungan CSM dilakukan penilaian kesamaan konsep secara semantik menggunakan kedekatan konsep term-term penyusun korpus menggunakan library Standford Dependency. Dimana untuk menghitung kedekatan konsep antar operasi diusulkan menggunakan pendekatan vector space models dengan memodifikasi pembobotan TF-IDF dengan word distance dan word dependency. Kemudian nilai kedekatan antar operasi tersebut direpresentasikan dalam bentuk graf dan dilakukan pemotongan graf menggunakan metode Max-Flow Min-

[1,2,3]*Jurusan Teknik Informtika, Fakulta Teknologi Informasi, Institut Teknologi Sepuluh Nopember.*

Cut untuk mendapatkan identifikasi operasi yang harus dipisahkan dari kelas God Class.
Dengan mekanisme yang diusulkan maka pemrogram mendapatkan rekomendasi peluang refaktorisasi kelas dengan mudah. Hasil penelitian ini mencapai hasil dengan tingkat precision 0,708% dan recall 0,936%.
Kata-kata kunci: Penyusunan kembali kelas, ExtractClass, God Class, Similaritas

Abstract

A class in Object-Oriented Programming (OOP) must have a specific design for handle and implement a concept. The class must have a specific responsibility to a concept which is implemented in the methods. In its development, software developers often do not pay attention to the design concept of the class so that the class be growing and has a complex implementation or commonly called the "God Class". In this case the necessary methods to identify refactoring opportunities to split God Class into several new classes that will be refactored using methods such ExtractClass Class Refactoring. In previous studies, it has been proposed methods for identification of refactoring opportunities using the three weighting which is SSM (Structural Similariy between Methods), CDM (Call-based Dependency between Methods) and CSM (Conceptual Similarity between Methods). CSM calculations have been done with the approach of LSI (Latent Semantic Indexing). But the concept of similarity assessment methods can not detect a similar concept to the sentences making up the name of the operation. The tendency of developers using choice words have meaning and use of the phrase work in shaping the operation name on this research creates the opportunity for the development of CSM calculation using semantic approach and word dependency as a concept on the location name of the operation.
This study proposed a method for identifying refactoring opportunities by identifying the concept location of methods. Methods of classes will be extracted into a corpus and calculated the value of its similarity to other operations using the SSM, CDM, and CSM. In the calculation of CSM assessment semantically similar concept of using concept similarity detection using StandfordDependency. Where to calculate the proposed operation to the similarities between the approach vector space models by modifying the TF-IDF weighting distance and word by word dependency. Then the similarity between these operations represented in the form of graphs and graph cuts made using Max-Flow

Min-Cut to get identification operations must be separated from God class Class.
With proposed method, software developer can easily get recomendation for class refactoring opportunities. The result of this research is precision 0,708% and recall 0,936%.
***Keywords**: Class Refactoring, Extract Class, God Class, Similarity*

1. PENDAHULUAN

Pemrograman berbasis objek merupakan konsep pemrograman yang berbasis tanggungjawab, dimana pengembang perangkat lunak akan menentukan tanggungjawab masing-masing kode yang dibuat dan direpresentasikan pada operasi dan kelas. Sebuah kelas umumnya menggambarkan objek dalam perangkat lunak yang dibangun dan kelas-kelas yang dibentuk memiliki tanggungjawab masing-masing yang direpresentasikan dalam bentuk operasi. (Coad & Yourdon, 1991).

Dalam siklus pengembangan perangkat lunak, seringkali terjadi evolusi terhadap kode sumber berupa penambahan operasi pada kelas-kelas yang dilakukan oleh pengembang perangkat lunak. Penambahan yang dilakukan oleh pengembang perangkat lunak ini kadang menjadi tidak seusai dengan konsep awal dibentuknya kelas, sehingga konsep dari kelas yang dibangun menjadi menyimpang. Penyimpangan terhadap desain kelas ini disebut dengan God Class. Untuk mengurangi God Class pada kode sumber, perlu dilakukan refaktorisasi atau penyusunan kembali kelas (Opdyke, 1992). Refaktorisasi merupakan proses menunjukkan peluang penyusunan kembali struktur kode sumber dengan tujuan untuk memperbaiki desain dan mempermudah dalam melakukan pengembangan selanjutnya.

Salah satu metode dalam refaktorisasi adalah dengan menggunakan metode Extract Class (Fowler, 1999) yaitu metode yang melakukan pemindahan operasi dalam kelas yang kompleks menjadi kelas lain sehingga kelas baru yang terbentuk tidak terlalu besar dan lebih spesifik. Dalam melakukan refaktorisasi, perlu diketahui terlebih dahulu operasi mana yang menyebabkan kelas menjadi God Class atau operasi mana yang tidak sesuai dengan konsep kelas dan perlu dipindahkan. Untuk mengetahui operasi mana yang perlu dipindahkan diperlukan sebuah metode

identifikasi peluang refaktorisasi. Penelitian ini berfokus pada identifikasi peluang refaktorisasi dengan mengukur kedekatan konsep operasi berdasarkan penghitungan struktural dan semantik.

Penelitian yang paling dekat dengan penelitian ini adalah yang dilakukan oleh Bavota dkk (Bavota, De Lucia, & Oliveto, 2011). Mereka menggabungkan metode struktural dan semantik untuk mengukur kohesi kelas dan mengidentifikasi peluang refaktorisasi pada sebuah God Class. Pada penelitian tersebut digunakan pendekatan SSM (Structural Similarity between Methods), CDM (Call-Based Dependency between Methods), dan CSM (Conceptual Similarity between Methods). Kemudian berdasarkan hasil perhitungan tersebut dilakukan pemotongan kelas God Class menjadi kelas baru menggunakan algoritma MaxFlow-MinCut, dengan cara merepresentasikan keterhubungan operasi dengan operasi lainnya dalam sebuah graf.

Penelitian ini mengusulkan perbaikan dari metode yang diusulkan oleh Bavota dkk (Bavota, De Lucia, & Oliveto, 2011). Pada penelitian sebelumnya, Bavota menggunakan pendekatan LSI pada perhitungan CSM dengan menghitung kemunculan kata yang sama sebagai faktor kedekatan semantik. Bavota tidak memperhitungkan kedekatan makna dari dua term yang dibandingkan, hanya menggunakan pendekatan kesamaan sintaksis. Misalkan terdapat operasi dengan nama getParentName dan getFatherName akan dianggap menjadi kata yang sama sekali berbeda. Jika menggunakan lokasi konsep, dalam fungsi getParentName mengandung dua konsep yaitu parent dan name. Sehingga penelitian ini menggunakan keterhubungan antar kata untuk mendapatkan perhitungan kedekatan operasi secara semantik yang lebih akurat.

Penelitian ini juga memperbaiki metode pencocokan term yang muncul dengan menggunakan word dependency berdasarkan kamus library Standford NLP. Sebagai contoh, term parent dan father akan dihitung sebagai term yang memiliki kedekatan walaupun tidak sama jika dilakukan pencocokan sintaksis.

2. KAJIAN PUSTAKA

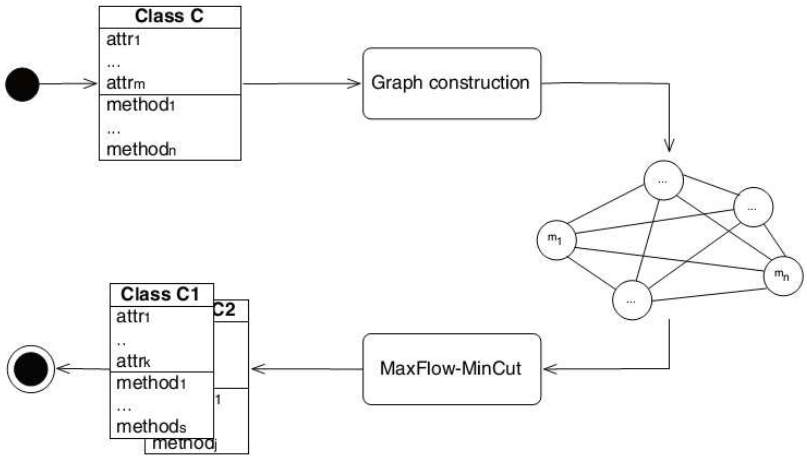
2.1 God Class

God Class merupakan desain kelas yang menggunakan gaya prosedural dengan banyak konsep didalamnya, dimana objek objek lainnya hanya menangani data atau menangani proses yang sederhana. (Brown, Malveau, & Mowbray, 1998).

Untuk menyelesaikan permasalahan God Class perlu dilakukan Refaktorisasi. Refaktorisasi adalah tindakan yang dilakukan untuk menyederhanakan sebuah God Class sehingga menjadi kelas yang memiliki kompleksitas rendah dengan cara mendistribusikan atribut atau operasi kedalam kelas yang lain. (Brown, Malveau, & Mowbray, 1998).

2.2 Identifikasi Peluang Refaktorisasi

Bavota, G, dkk (Bavota, De Lucia, & Oliveto, 2011) dalam penelitiannya mengusulkan metode identifikasi peluang refaktorisasi dengan menggunakan representasi Graf keterhubungan antar operasi. Metode yang diusulkan adalah dengan melakukan pembobotan kedekatan antar operasi dengan pendekatan bobot SSM, CDM, dan CSM, kemudian dari graf yang terbentuk digunakan algoritma Max-Flow-Min-Cut untuk membentuk memotong graf keterhubungan kelas baru. Gambaran metode yang diusulkan oleh Bavota, dkk dengan algoritma Max-Flow Min-Cut ditampilkan pada Gambar 1.



Gambar 1. Metode Identifikasi Peluang Extract Class

2.3 Bobot Kesamaan Antar Operasi

Menurut Bavota dkk, untuk membentuk graf keterhubungan antar operasi dalam algoritma Max-Flow Min-Cut digunakan metode pembobotan. Bobot diambil dari beberapa aspek yaitu:

a. *Structural Similarity between Methods* (SSM), merupakan pengukuran matrik kohesi kelas yang diusulkan oleh Gui dkk (Gui & Paul D., 2008). Metode ini menggunakan rasio jumlah variabel *instance* yang digunakan bersama antara dua operasi.

$$SSM(m_i, m_j) = \begin{cases} \frac{|I_i \cap I_j|}{|I_i \cup I_j|} \; if \; |I_i \cup I_j| \neq 0; \\ 0 \quad otherwise. \end{cases} \quad (1)$$

dimana:
I_i = Variabel *instance* yang dipanggil oleh operasi m_i
I_j = Variabel *instance* yang dipanggil oleh operasi m_j

b. Pemanggilan operasi juga dapat digunakan sebagai faktor dalam menghitung kohesi kelas (Bavota, De Lucia, & Oliveto, 2011) dengan menggunakan metode CDM (*Call-based Dependencies between Methods*)

$$CDM(m_i, m_j) = \begin{cases} \frac{calls(m_i, m_j)}{calls_{in}(m_j)} \; if \; calls_{in}(m_j) \neq 0; \\ 0 \quad otherwise. \end{cases} \quad (2)$$

dimana:
$calls(m_i, m_j)$ = jumlah pemanggilan yang dilakukan oleh operasi m_i terhadap operasi m_j
$calls_{in}(m_j)$ = jumlah total pemanggilan yang masuk ke operasi m_j

c. *Conceptual Similarity between Methods* (CSM), digunakan untuk pendekatan semantik melalui komentar dan atribut atau konsep operasi yang sama (Poshyvanyk, Guéhéneuc, Marcus, G. Anton, & Anton, 2007).

$$CSM(m_i, m_j) = \frac{\overrightarrow{m_i} \cdot \overrightarrow{m_j}}{\overrightarrow{\|m_i\|} \cdot \overrightarrow{\|m_j\|}} \quad (3)$$

dimana:
$CSM(m_i, m_j)$ = bobot kemiripan semantik antara m_i dan m_j

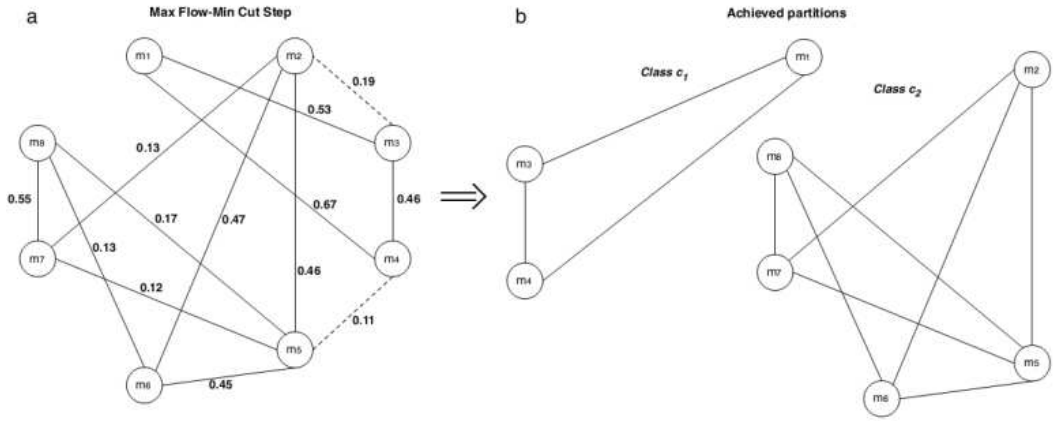
$\overrightarrow{m_x}$ = vektor kesesuaian operasi x

$\overrightarrow{\|m_i\|}$ = eucledian norm dari operasi x

Dari ketiga parameter bobot tersebut diatas (SSM, CDM, CSM) penentuan bobot final untuk mengukur keterhubungan antar-operasi ditentukan menggunakan rumus berikut:

$$w(m_i, m_j) = w_{SSM} \cdot SSM(m_i, m_j) + w_{CDM} \cdot CDM(m_i, m_j) + w_{CSM} \cdot CSM(m_i, m_j) \quad (4)$$

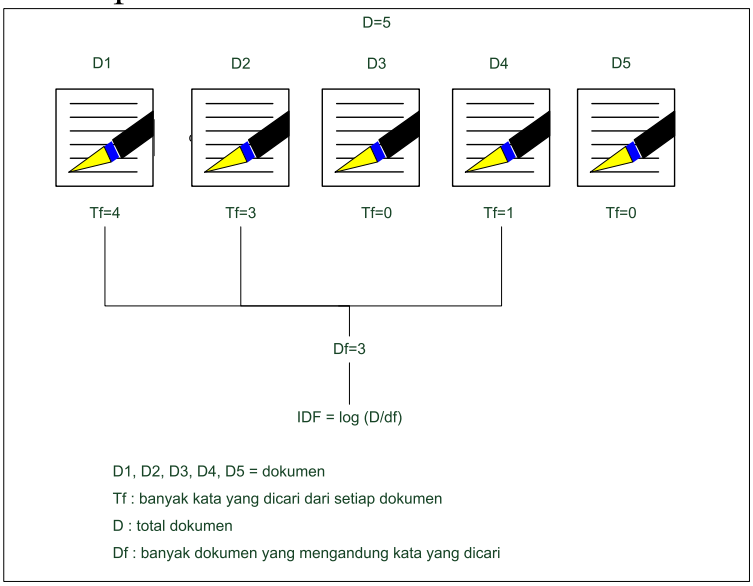
dimana $w_{SSM} + w_{CDM} + w_{CSM} = 1$ dan nilainya masing-masing adalah bobot untuk tiap pengukuran. Nilai ketiga parameter tersebut didapatkan nilai terbaik pada penelitian sebelumnya $\boldsymbol{w_{CSM} = 0,7, w_{SSM} = 0,2, w_{CDM} = 0,1.}$ (Bavota, De Lucia, & Oliveto, 2011). Kemudian dari hasil perhitungan graf keterhubungan antar operasi dilakukan penyaringan terhadap verteks yang membentuk graf. Dipilih sebuah ambang batas *minimum cohesion* = α. Verteks yang dibawah ambang batas nilai kohesifitas akan dihilangkan (dianggap 0). Operasi yang tidak saling terhubung akan membentuk kelas yang baru. Pemotongan graf menggunakan algoritma Max-Flow-Min-Cut digambarkan pada gambar 2 berikut:



Gambar 2(a). Graf berbobot yang menggambarkan keterhubungan antar operasi sebelum diklasterisasi. **(b)** Graf berbobot setelah mengalami proses klasterisasi dengan ambang batas.

2.4 TF-IDF

Salah satu algoritma *Text Mining* adalah algoritma TF/IDF. Algoritma TF/IDF digambarkan pada Gambar 3 berikut. Algoritma ini digunakan untuk mencari tingkat kemiripan sebuah dokumen terhadap sebuah kata kunci atau term.



Gambar 3. Algoritma TF-IDF

TF-IDF digunakan untuk menghitung bobot term *corpus* dokumen operasi pada kelas. Bobot ini digunakan untuk mengukur nilai kemiripan antar operasi dalam kelas menggunakan *vector space models*. Rumus TF-IDF adalah seperti pada persamaan 5.

$$\boldsymbol{w(m_i, m_j) = tf_{i,j} * log \frac{2}{df_t}} \qquad (5)$$

$tf_{i,j}$ = jumlah kemunculan term dalam operasi i dalam operasi j
t = indeks term/kata
m_i = operasi ke i
m_j = operasi ke j
df_t = jumlah dokumen yang mengandung kata kunci ke-t

Dan bobot (w) antar operasi dengan metode TF-IDF yang telah ternormalisasi dinyatakan dalam persamaan 6.

$$w(m_i, m_j) = \left\{\frac{tf_{i,j}}{\max tf_{i,j}}\right\} * log\left\{\frac{D}{df_t}\right\} \qquad (6)$$

Dengan *t* adalah indeks term, *tf* adalah term frequency, dan df_t adalah jumlah *method* yang mengandung term ke-*t*.

3. METODE PENELITIAN

Ada tiga tahapan yang dilakukan dalam mengidentifikasi peluang refaktorisasi God Class dalam penelitian ini, yaitu: praproses, perhitungan bobot kemiripan antar operasi, dan pemotongan graf keterhubungan antar operasi menggunakan Max Flow Min Cut.

3.1 Praproses

Praproses dilakukan pada kode sumber untuk mengambil kata-kata yang memiliki makna dari dalam suatu kode sumber. Praproses dilakukan terhadap nama kelas, atribut kelas dan informasi operasi. Informasi operasi terdiri dari nama operasi, tipe data kembalian dari operasi dan parameter dalam suatu operasi. Praproses dimulai dengan pembacaan kode sumber berbentuk dokumen teks dengan ekstensi .java. Dokumen kode sumber yang digunakan dalam penelitian ini merupakan dokumen kode sumber struktur kelas. Pembacaan dokumen kode sumber dilakukan dengan memanfaatkan *library* yang telah disediakan oleh Eclipse yaitu: **AST Parser** yang terdapat pada paket aplikasi **Java Development Tools (JDT)**. Struktur kelas yang diambil adalah: nama kelas, *instance variable, local variable, body of method* dan *return parameter*.

Kemudian untuk setiap *corpus* operasi ini dilanjutkan dengan proses tokenisasi, yaitu memisahkan berdasarkan karakter *non-alphanumeric* yaitu karakter yang bukan huruf ataupun angka dan melakukan pemisahan kata (*token splitting*) dalam *identifier* menjadi kata yang bermakna. Dalam kode sumber ada beberapa aturan yang sering digunakan untuk memisahkan *identifier*, yaitu:

- *Snake_case*, merupakan *identifier* yang dibentuk dari gabungan kata yang disambung menggunakan garis bawah, misalnya "*send_email*". Untuk memisahkan menjadi kata-kata bisa dilakukan dengan mengganti garis bawah mejadi spasi.
- *CamelCase*, merupakan *identifier* yang dibentuk dari gabungan kata dengan penanda yaitu pada awal setiap kata digunakan huruf kapital seperti contohnya *sendEmail*. Untuk memisahkan menjadi kata-kata dengan cara menyisipkan spasi sebelum huruf kapital.

- *MixedCase*, merupakan identifier yang dibentuk dari gabungan kata dengan huruf kapital dan kata dengan huruf kecil, seperti contoh pada identifier ASTVisitor menjadi AST Visitor.

Token *splitting* dilakukan dengan memeriksa setiap karakter apabila ditemukan kombinasi karakter garisbawah dengan huruf apapun, atau karakter huruf kecil bertemu huruf besar, atau huruf besar sebanyak lebih dari satu kali berturut-turut bertemu huruf kecil, maka akan diganti dengan karakter spasi.

Langkah berikutnya adalah mencari dan menghilangkan *stopword* dalam kode sumber. *Stopword* adalah frasa penyusun operasi atau identifier kelas yang tidak bermakna atau tidak memiliki arti. Frasa stop word yang digunakan adalah frasa yang tidak bermakna spesifik dalam kode sumber. Pada proses tokenisasi kode sumber, akan dicek apakah frase yang ditokenisasi terdapat dalam daftar *stopword* atau tidak, jika ya maka akan dihapus dari daftar kata unik.

3.2 Perhitungan Bobot Kemiripan Antar Operasi

Pada tahap ini, dilakukan perhitungan bobot kemiripan antar operasi dengan cara yang sama seperti yang dilakukan Bavota, dkk yaitu menggunakan 3 parameter: SSM (persamaan 1), CDM (persamaan 2), dan CSM. Khusus untuk bobot CSM, dilakukan modifikasi dengan menambahkan metode pembobotan konsep operasi pada perhitungan LSI. Bobot CSM yang diusulkan pada penelitian ini menggunakan metode TF-IDF termodifikasi. Metode ini berdasar pada analisa relasi semantik antara kata / token. Algoritma yang digunakan adalah algoritma TF/IDF dengan menambahkan bobot *word dependency*. Langkah-langkah perhitungan bobot TF-IDF *word dependency* antar operasi pada kode sumber adalah sebagai berikut:

1. Menggabungkan seluruh term penyusun operasi yang telah ditokenisasi menjadi daftar token untuk dihitung masing-masing bobotnya.
2. Dari masing-masing operasi yang telah terbentuk kata unik dihitung bobot TF-IDF-nya berdasarkan frekuensi kemunculan term dalam masing-masing kalimat nama operasi.

3. Dalam menghitung TF-IDF, dipertimbangkan keterhubungan antar kata dalam kalimat nama operasi untuk menentukan bobot vektor operasi. Dua kalimat nama operasi yang memiliki *root* kalimat yang sama, akan dianggap sebagai kalimat yang memiliki konsep yang sama, hal ini juga akan menambah bobot Vektor kalimat tersebut untuk metode *vector space models*. Untuk menentukan bobot keterhubungan antar operasi digunakan persamaan 11:

$$\boldsymbol{I(m_i, m_j)} = \begin{cases} \mathbf{1} & \boldsymbol{if\ t_i \epsilon m_j} \\ \mathbf{0} & \boldsymbol{otherwise} \end{cases} \quad (8)$$

$$V(m_i, m_j) = \partial \cdot tfidf(m_i, m_j) + \emptyset \cdot I(m_i, m_j)$$

dimana:

$tfidf(m_j, m_j)$ = bobot tf-idf yang dihitung dengan persamaan 5

$I(m_i, m_j)$ = bobot kemiripan konsep operasi

t_i = root kalimat m_i yang diidentifikasi dengan *Standford Dependency*

$V(m_i, m_j)$= vektor bobot m_i dan m_j

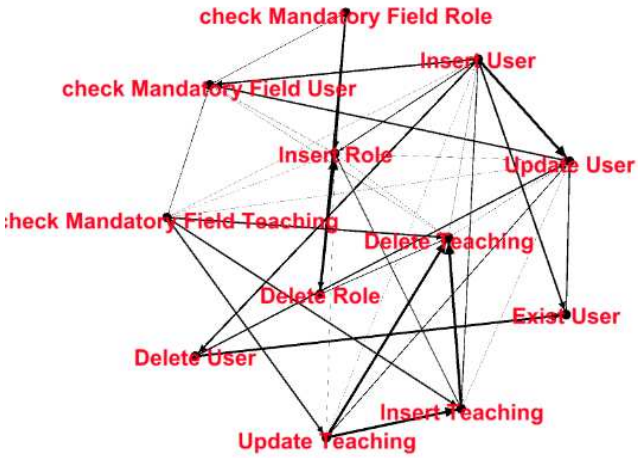
∂ = koefisien bobot *tf-idf*

$\emptyset$ = koefisien bobot kemiripan konsep kalimat

Bobot kesamaan konsep antar operasi diidentifikasi berdasarkan kesamaan konsep antar nama operasi. Jadi dalam penghitungan kesamaan konsep, hanya diambil nama dari operasi. Sebagai contoh, operasi InsertTeaching (Teaching pTeaching) diekstrak nama operasinya sehingga membentuk korpus kalimat "Insert Teaching" yang terdiri atas kata "Insert" dan "Teaching". Sedangkan operasi checkMandatoryFieldTeaching(Teaching pTeaching) diekstrak dan dihasilkan korpus kalimat "check mandatory field Teaching" yang terdiri atas kata "check", "mandatory", "field" dan "Teaching". dari dua korpus operasi itu diketahui informasi *Word Dependency* masing-masing nama operasi. Terdapat kesamaan root dari kalimat yaitu teaching, sehingga bobot kesamaan konsep dari dua operasi ini menjadi tinggi.

Hasil dari perhitungan bobot CSM dengan menggunakan TF-IDF word dependency ini kemudian dikombinasikan dengan bobot SSM dan CDM dengan faktor pengali masing-masing

$\boldsymbol{w_{CSM} = 0,7, w_{SSM} = 0,2, w_{CDM} = 0,1.}$ Sehingga didapatkan matriks bobot keterhubungan antar operasi (w) dengan menggunakan persamaan (4). Bobot keterhubungan antar operasi ini menggambarkan graf keterhubungan antar operasi yang bisa dibentuk seperti contoh pada Gambar 4.

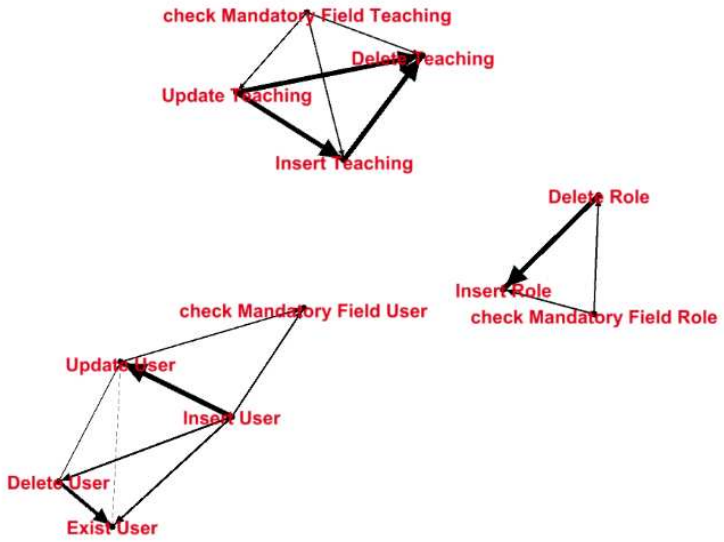


Gambar 4. Graf keterhubungan antar operasi (garis penghubung menyatakan derajat keterhubungan antar operasi, semakin tebal semakin terhubung)

Dari graf yang terbentuk, dapat dilakukan pemotongan graf menggunakan algoritma *MaxFlow MinCut* untuk mendapatkan potongan yang optimal dari graf tersebut. Pada penelitian ini dilakukan penyaringan bobot w yang dianggap memenuhi sebagai bobot keterhubungan antar operasi dengan menggunakan koefisien minimumCohesion (minCoh) yang bernilai antara 0.1 hingga 0.5 dengan kenaikan 0.1.

3.3 Pemotongan Graf

Algoritma *MaxFlow MinCut* digunakan untuk menghitung *Flow* maksimal dari graf keterhubungan antar operasi dan merekomendasikan pemutusan jalur yang optimal untuk menghasilkan 2 potongan atau lebih dari graf. Luaran dari proses ini adalah rekomendasi pemotongan jalur antara dua node atau lebih untuk menghasilkan potongan graf baru. Hasil dari algoritma *Max-Flow Min-Cut* adalah pemotongan graf seperti yang tampak pada Gambar 5.



Gambar 5. Hasil pemotongan graf bobot kombinasi. Garis tebal menunjukkan operasi tersebut memiliki keterhubungan lebih tinggi daripada garis yang lebih tipis

Seperti yang tampak pada Gambar 5, dihasilkan 2 sub-graf yang masing-masing menjadi rekomendasi penyusunan kelas sesuai dengan penilaian struktural dan semantik. Sub graf ini akan menjadi rekomendasi pemindahan kelas menjadi kelas baru. Dari hasil rekomendasi yang diusulkan sistem, dapat diketahui bahwa konsep kelas yang baru sudah sesuai secara semantik, dimana konsep kelas yang baru terdiri atas tiga konsep yaitu User, Teaching, dan Role.

3.4 Skenario Uji Coba

Dalam pengujian perangkat lunak ini akan dilakukan dua pengujian yaitu menguji metode LSI dengan TF-IDF normal untuk pembobotan CSM pada kelas dengan mengambil nama kelas. Sedangkan pada pengujian kedua dilakukan dengan metode LSI dan memodifikasi TF-IDF menggunakan *library* Standford Word Dependency untuk menentukan konsep lokasi dari operasi. Untuk masing-masing skenario digunakan koefisien *minCohesion* bervariasi mulai dari 0.1 hingga 0.5 dengan kenaikan 0.1.

4. HASIL DAN PEMBAHASAN

Berikut adalah hasil uji coba dari skenario uji coba yang telah digambarkan pada pembahasan sebelumnya:

1. Hasil Skenario I: Uji coba pada kelas sintetis
   Hasil uji coba skenario pertama ditunjukkan oleh Tabel 3

Tabel 3. Hasil Pengelompokan Kelas dengan metode LSI Normal, LSI Modifikasi dibandingkan dengan Rekomendasi Pakar

Nama Operasi	Pakar	LSI Norm	LSI Modifika
DeleteTeaching()	C1	C1	C1
InsertTeaching()	C1	C1	C1
checkMandatoryFieldTeaching()	C1	C4	C4
UpdateTeaching()	C1	C1	C1
DeleteRole()	C3	C3	C3
InsertRole()	C3	C3	C3
checkMandatoryFieldRole()	C3	C4	C3
UpdateUser()	C2	C2	C2
ExistUser()	C2	C2	C2
InsertUser()	C2	C2	C2
DeleteUser()	C2	C2	C2
checkMandatoryFieldUser()	C2	C4	C2

Dari Tabel 3 diatas dapat diketahui bahwa operasi UpdateUser, Exist User, DeleteUser, InsertUser sudah sesuai dengan rekomendasi pakar dikelompokkan menjadi satu kelas karena memiliki tanggungjawab terhadap pengelolaan objek User. Hal ini dikarenakan konsep operasi "User" berhasil terdeteksi oleh metode TF-IDF setelah token "Insert", Update" dan "Delete" dimasukkan sebagai daftar *Stop Word*, sehingga dalam penghitungan TF, yang diperhitungkan hanyalah term "User" dimana masing-masing operasi yang terdeteksi memiliki token "User".

Kasus lain terjadi dalam percobaan ini yaitu terpisahnya operasi checkMandatoryFieldUser dengan C1 yang secara konseptual memiliki kesamaan konsep dengan C1 yaitu "User". Hal ini bisa terjadi karena pada dasarnya metode TF-IDF mengidentifikasi kemunculan tiap token pada sebuah korpus dokumen dalam hal ini nama operasi, sehingga jika dihitung frekuensi ditemukannya term yang sama antara operasi checkMandatoryFieldUser dengan

checkMandatoryFieldTeaching lebih tinggi nilainya dibandingkan dengan operasi InsertUser atau UpdateUser.

Dari tabel 3 dapat diketahui bahwa operasi checkMandatoryFieldUser lebih memiliki kecondongan kesamaan konsep dengan term mandatory dan term field dibandingkan dengan term "user". Sehingga dengan perhitungan kesamaan *cosine*, akan didapatkan bahwa operasi checkMandatoryFieldUser memiliki nilai kemiripan lebih tinggi dengan operasi checkMandatoryFieldTeaching.

Sedangkan dengan menggunakan metode LSI termodifikasi diketahui bahwa operasi UpdateUser, Exist User, DeleteUser, InsertUser dan checkMandatoryFieldUser sudah sesuai dengan rekomendasi pakar dikelompokkan menjadi satu kelas karena memiliki tanggungjawab terhadap pengelolaan objek User. Hal ini dikarenakan konsep operasi "User" berhasil terdeteksi oleh metode TF-IDF termodifikasi *WordDependency* berhasil mengidentifikasi root kalimat checkMandatoryFieldUser, yaitu term User. Kemudian term user yang memiliki *dependency* sebagai *root* ini ditambahkan bobot TF-IDF-nya sehingga dalam perhitungan *cosine Similarity*, operasi checkMandatoryFieldUser akan diidentifikasi lebih mirip dengan insertUser, DeleteUser dan UpdateUser.

Tabel 1. Hasil Perhitungan F-Measure dengan metode LSI Normal dan Termodifikasi

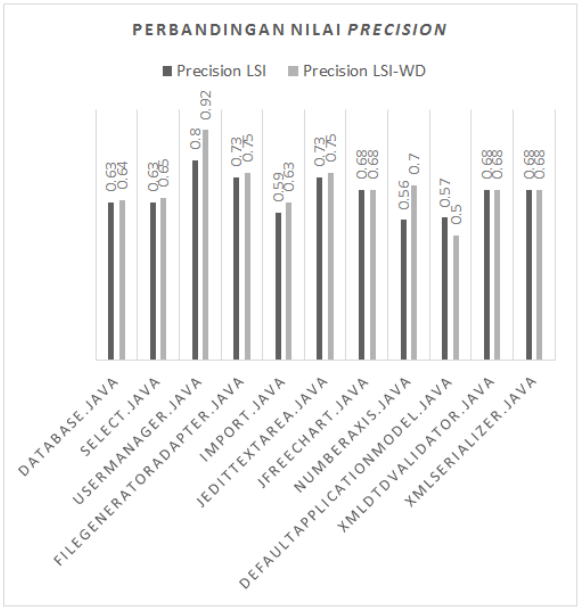
Nilai minCohesion	LSI-normal			LSI-modified		
	Prec	Rec	F-Meas	Prec	Rec	F-Meas
0.1	0.50	0.33	0.40	0.50	0.33	0.40
0.2	0.50	0.33	0.40	0.64	0.58	0.61
0.3	0.50	1.00	0.67	0.79	1.00	0.88
0.4	0.50	0.33	0.40	0.79	1.00	0.88
0.5	0.75	1.00	0.86	0.79	1.00	0.88

Dari percobaan terhadap kelas sintetis dengan melakukan variasi terhadap nilai minCohesion mulai 0.1 hingga 0.5 seperti yang tampak pada tabel 4 didapatkan hasil bahwa dengan menggunakan LSI-termodifikasi *Word Dependency* berhasil meningkatkan precision dari 0.75 menjadi 0.79 dengan nilai recall

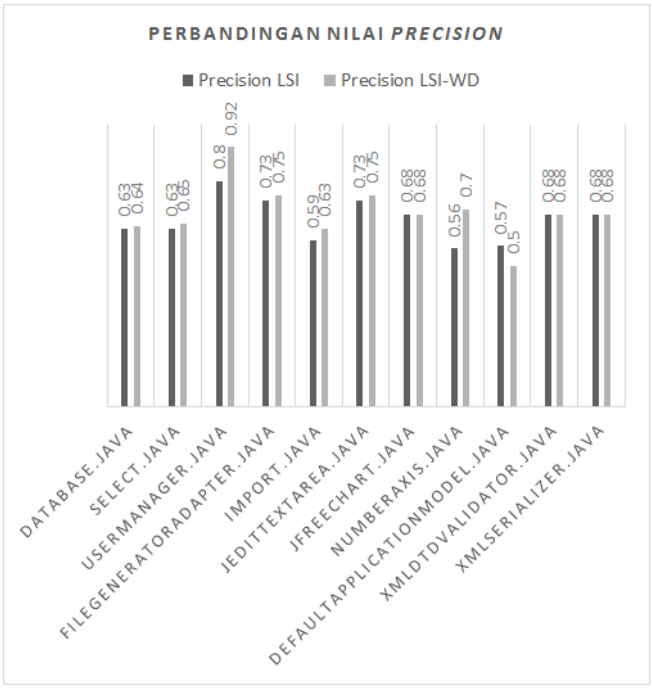
tetap 1.00 dan meningkatkan F-Measure dari 0.86 menjadi 0.88.

2. Hasil Skenario II: Uji coba pada kelas proyek terbuka

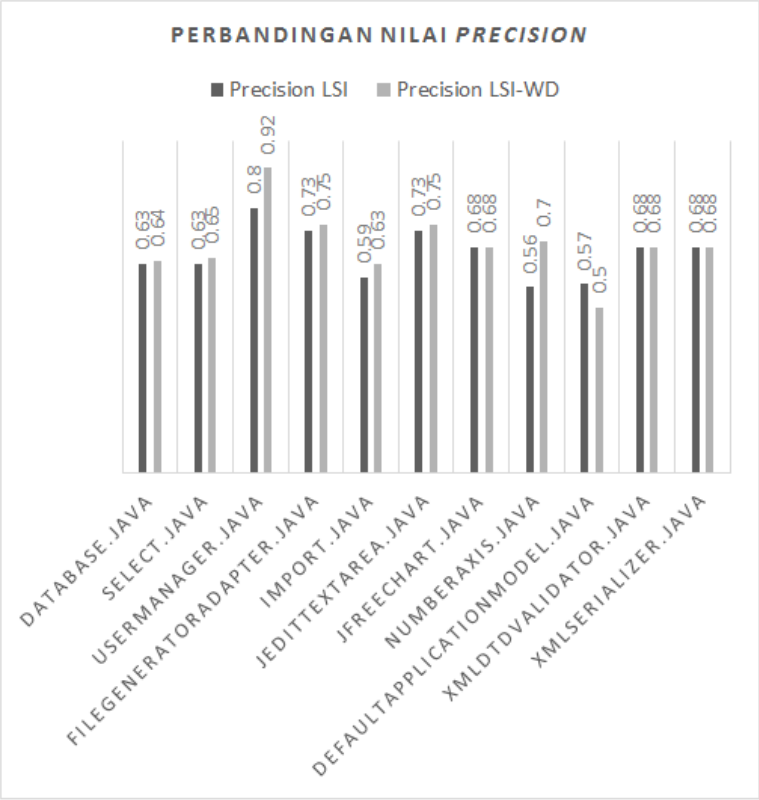
   Hasil skenario percobaan pada kelas proyek terbuka ditampilkan sebagai grafik batang pada gambar berikut:



Gambar 6. Perbandingan Nilai Precision



Gambar 7. Perbandingan Nilai Recall



Gambar 8. Perbandingan Nilai F-Measure

Dari hasil rata-rata nilai precision, recall dan F-Measure tiap kelas pada 11 kelas kode sumber terbuka tampak bahwa metode yang diusulkan cenderung lebih baik dalam mengidentifikasi peluang refaktorisasi sesuai dengan rekomendasi yang dihasilkan oleh pakar rekayasa perangkat lunak

Uji coba yang dilakukan pada kelas Database.java, DefaultApplicationModel, dan JfreeChart menunjukkan bahwa metode yang diusulkan, yaitu LSI dengan modifikasi pada TF-IDF menggunakan *word dependency* menghasilkan nilai *precision* dan *recall* yang cukup rendah dibandingkan dengan metode LSI. Hal ini dikarenakan pada kelas Database.java banyak digunakan operasi-operasi yang hanya menggunakan single term atau disertai dengan *stopword* dengan operasi yang lainnya sehingga nilai kemiripan operasi tersebut dengan operasi lainnya menjadi rendah. Contoh nama operasi yang menggunakan single term adalah: open, reopen, URI, State, close, finalize, path, type.

5. PENUTUP

Dari hasil pengamatan yang dilakukan selama proses perancangan, implementasi, serta pengujian terhadap perangkat lunak, dapat diambil kesimpulan diantaranya adalah sebagai berikut :

1. Pendekatan *WordDependency* dapat digunakan untuk menambah akurasi identifikasi kemiripan semantik menggunakan metode LSI dengan memodifikasi perhitungan TF-IDF.
2. Terdapat hasil peningkatan precision recal yang cukup signifikan. yaitu precision meningkat dari 0.68 menjadi 0.70 meningkat 0.02. dan recall dari 0.88 meningkat menjadi 0.93, terjadi peningkatan sebanyak 0.05. hasil pengujuan ini membuktikan bahwa pendekatan konsep lokasi operasi dapat digunakan untuk menambah akurasi sistem identifikasi peluang refaktorisasi.
3. Akurasi sistem menjadi rendah jika menemui nama operasi yang menggunakan single term karena nilai kemiripan kosinenya akan menjadi rendah.
4. Perlu meningkatkan penilaian kemiripan term dengan mempertimbangkan makna kata dengan menggunakan kamus WordNet.

Berikut merupakan beberapa saran yang dapat digunakan sebagai acuan untuk pengembangan penelitian lebih lanjut di masa yang akan datang:

1. Perlu tambahan parameter *semantic similarity* untuk menentukan bobot kemiripan sebuah operasi dengan kosep pada sebuah kelas (*class responsibility*), misalnya baris komentar dari *programmer* dan attribut-attribut dalam operasi *body*.
2. Perlu analisa lebih lanjut pada metode pengidentifikasian nama kelas dan operasi dimana terdapat nama kelas atau operasi yang merupakan singkatan atau istilah teknis yang tidak dikenali secara makna hanya berdasarkan kamus WordNet.

6. DAFTAR PUSTAKA

[1] Bavota, G., De Lucia, A., & Oliveto, R. (2011). Identifying extract class refactoring opportunities using structural and semantic cohesion measures. Journal of System and Software, 397–414.

[2] Brown, W., Malveau, R., & Mowbray, T. (1998). AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis. Canada: John Wiley & Sons, Inc.

[3] Coad, P., & Yourdon, E. (1991). Object-Oriented Design, 1st edition. Prentice Hall.

[4] Fowler, M. (1999). Refactoring: Improving the Design of Existing Code. Addison-Wesley.

[5] Gui, G., & Paul D., S. (2008). New Coupling and Cohesion Metrics for Evaluation of Software Component. Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for (hal. 1181-1186). Zhang Jia Jie, Hunan, China: IEEE.

[6] Marie-Catherine de Marneffe, B. M. (2006). Generating typed dependency parses from phrase structure parses. 5th International Conference on Language Resources and Evaluation (LREC 2006). Genoa, Italy.

[7] Opdyke, W. F. (1992). Refactoring object-Oriented Frameworks. Urbana, Illinois: University of Illinois at Urbana-Champaign.

[8] Poshyvanyk, D., Guéhéneuc, Y., Marcus, G. Anton, A., & Anton, G. (2007). Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval. IEEE Transactions on Software Engineering 33, 420–432.

[9] Thomas H. Cormen, C. E. (2009). Introduction To Algorithms: Third Edition. Massachusetts: Massachusetts Institute of Technology.