Sebuah Studi Dalam Mengintegrasikan Teknik Analisis Dampak Perubahan Untuk Mengelola Perubahan Kode Pada Perangkat Lunak

Agung wahyudi[1], Febri Fernanda[2]
Program Studi Teknik Informatika
Fakultas Teknik - Universitas 45 Surabaya, Indonesia
Program Studi Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember Surabaya, Indonesia
agungwyudi@gmail.com[1], febri.fernanda11@mhs.if.its.ac.id[2]

Abstrak

Kode merupakan artefak perangkat lunak yang paling banyak mengalami perubahan. Perubahan kode disebabkan hal-hal seperti laporan kesalahan, penamabahan fitur dan sebagainya. Pada perangkat lunak sumber terbuka pengelolaan penyesuaian dampak perubahan kode sangat diperlukan, mengingat perubahan kode bisa dilakukan oleh siapa saja yang terdaftar sebagai kontributor.

Analisis dampak perubahan digunakan untuk mengetahui kode yang harus diubah ketika ada permintaan perubahan atau ada perubahan dari suatu kode. Hasil analisis berupa daftar kode yang mungkin juga perlu dirubah agar fungsionalitas bisa berjalan dengan baik. Ada banyak metode analisis dampak, diantaraya Information Retrieval (IR), dekomposisi nilai singular (Singular Value Decomposition, SVD), dan graf panggil. Pada penelitian ini dilakukan pendekatan baru analisis dampak perubahan berupa kombinasi dari ketiga metode tersebut untuk mengetahui apakah kombinasi dari metode menghasilkan nilai presisi dan recall yang lebih baik dari pendekatan secara individu.

Hasil pengujian didapatkan bahwa kombinasi dari ketiga metode secara bersama, IR, SVD, dan graf panggil menghasilkan prediksi fungsi terdampak yang paling optimal dibanding kombinasi antara dua metode ataupun metode IR individu.

Kata kunci: analisis dampak perubahan, evolusi perangkat lunak, integrasi teknik, graf panggil, SVD, IR

1. Pendahuluan

Perubahan kode pada suatu perangkat lunak, apalagi perangkat lunak berbasis sumber terbuka tidak bisa terhindari. Perubahan kode didasari oleh banyak hal, di antaranya karena laporan kesalahan (*bug)* baik dari pengembang maupun dari pengguna, penambahan fitur-fitur baru, atau untuk optimalisasi waktu eksekusi. Dalam evolusi perangkat lunak, perubahan kode pada suatu perangkat lunak menyebabkan perlunya penyesuaian kode-kode pada kelas atau fungsi lain yang berhubungan dengannya. Penentuan bagian kode mana yang juga harus diubah, terlebih lagi pada perangkat lunak sumber terbuka sangat sulit dilakukan karena pengembang atau pemrogram tidak mungkin mengetahui segala pengetahuan tentang perangkat lunak sepanjang waktu, belum lagi masalah kekompleksan

sistem. Pendeteksian analisis dampak perubahan merupakan hal yang sensitif, karena jika hasil analisis kurang tepat maka dapat menimbulkan kerusakan pada perangkat lunak, misalnya beberapa fungsionalitas yang tidak bisa berjalan sesuai kebutuhan. Oleh sebab itu diperlukan teknik analisis yang mempunyai nilai ketepatan yang tinggi.

Secara umum, ada beberapa jenis teknik analisis dampak perubahan, yaitu analisis statis, analisis dinamis, atau kombinasi dari kedua teknik tersebut. Teknis analisis statis merupakan teknis analisis tanpa memerlukan eksekusi kode, sebaliknya dengan teknik analisis dinamis. Teknik analisis statis yang telah diusulkan misalnya graf panggil (*call graph*) [1][2], penggalian data sejarah menggunakan dekomposisi nilai singular (Singular Value Decomposition, SVD) [3] dan analisis menggunakan temu kembali informasi (Information Retrieval, IR), sedangkan contoh analisis dinamis antara lain penggunaan Execution Profile Tracing, CoverageImpact dan PathImpact [4].

Penelitian ini memfokuskan masalah pada perubahan kode tanpa melakukan eksekusi sehingga hanya menggunakan teknis analisis statis. Penelitian ini melakukan implementasi ketiga teknik analisis statis yang disebutkan sebelumya yaitu graf panggil, SVD dan IR, dengan menambahkan kontribusi pengembangan yang disesuaikan dengan studi kasus. Hal yang diusulkan dalam penelitian ini adalah melakukan penggabungan hasil dari kombinasi ketiga teknik tersebut kemudian melakukan analisis dan perbandingan terhadap nilai akhir yang dihasilkan oleh tiap jenis kombinasi. Hipotesis yang diusulkan yaitu bahwa semakin banyak kombinasi metode teknik analisis dampak perubahan bisa menghasilkan nilai presisi dan *recall* yang lebih baik daripada pendekatan IR individu. Nilai presisi dan *recall* yang tinggi menyimpulkan bahwa hasil analisis dampak perubahan semakin tepat sesuai dengan kebutuhan. Level kode yang dilakukan analisis dampak perubahannya pada penelitian ini adalah level fungsi, dengan informasi untuk prosesnya didapat dari permintaan perubahan, baik dari pengguna maupun pengembang. Permintaan perubahan biasanya berisi kesalahan atau penambahan fitur yang direpresentasikan dalam bahasa alami sehingga metode IR dijadikan sebagai pusat skenario penggabungan. Dengan menjadikan IR sebagai pembanding utama, maka IR selalu terlibat pada setiap skenario.

2. Teknik Analisis Dampak Perubahan

Berikut dijelaskan mengenai definisi, proses, dan langkah-langkah implementasi dari jenis-jenis teknik analisis dampak perubahan yang bersifat statis. Teknik-teknik tersebut menjadi dasar dalam penelitian ini.

2.1. Graf Panggil [1]

Graf panggil secara umum digunakan untuk membantu manusia dalam memahami program, mengetahui perancangan desain program, dan juga dapat digunakan dalam aktivitas pemeliharaan perangkat lunak. Salah satu aktivitas pemeliharaan perangkat lunak adalah analisis dampak perubahan perangkat lunak.

Langkah-langkah untuk melakukan analisis dampak perubahan dengan graf panggil, yaitu:

a. Membangun graf panggil dari kode program yang ada. Tingkatan graf panggil yaitu berkas, kelas, atau fungsi.

b. Memetakan perubahan kode pada tingkat tertentu. Simpul-simpul yang berhubungan dengan simpul yang mengalami perubahan mewakili fungsi-fungsi yang terdampak.
c. Simpul yang mengalami perubahan dengan simpul tetangga dihubungkan oleh graf dengan bobot tertinggi. Semakin jauh simpul dari simpul yang mengalami perubahan maka semakin kecil bobot grafnya. Sebaliknya, semakin dekat simpul dengan simpul yang mengalami perubahan maka semakin besar bobotnya.
d. Keluaran dari analisis dampak ini berupa himpunan simpul yang terdampak akibat adanya perubahan kode program dan nilai masing-masing yang dihasilkan dari penjumlahan bobot graf. Hasil analisis graf panggil dapat menunjukkan tingkat dampak perubahan masing-masing fungsi.

2.2. Dekomposisi Nilai Singular (SVD)[3]

SVD adalah sebuah metode aljabar linier untuk memfaktorisasi sebuah matriks. Metode ini juga dapat digunakan untuk mengubah sekumpulan variabel yang bervariasi ataupun berdimensi tinggi ke dalam kumpulan variabel yang lebih sederhana sehingga hubungan atau pola di antara data-data tersebut akan terlihat dengan lebih jelas [6]. Salah satu penggunaan SVD yaitu dalam analisis dampak perubahan perangkat lunak oleh Sheriff dan Williams [3].

SVD didefinisikan dengan persamaan sebagai berikut.

$$M = U \sum V^*,$$

dengan *M* adalah matriks berukuran *m*x*n*, *U* adalah matriks orthonormal dari $M.M^t$ berukuran *m*x*m*, $\sum$ adalah matriks diagonal persegi berukuran *m*x*n* dengan diagonalnya bilangan real positif, dan *V** adalah matriks transpose orthonormal dari $M.M^t$ berukuran *n*x*n*. Masukan matriks $\sum_{i,i}$ adalah nilai singular dari M. $\sum$ adalah akar kuadrat dari nilai eigen matriks M. M^t. Nilai dari matriks ini menggambarkan seberapa pentingnya kelompok fungsi tersebut dan juga berarti semakin tinggi resiko yang akan dialami jika hal tersebut diubah.

Kaitannya dengan analisis dampak perubahan, matriks M dibentuk dari penggalian data repositori perubahan perangkat lunak dalam periode tertentu. Baris dan kolom matriks adalah indeks fungsi yang mengalami perubahan. Nilai-nilai pada matriks M tersebut adalah jumlah berapa kali fungsi mengalami perubahan atau diakses untuk diubah, baik perubahan dalam fungsi tersebut secara langsung ataupun perubahan yang disebabkan oleh fungsi lain yang berubah. Setelah matriks M terbentuk, maka dilakukan prosedur SVD untuk mendapatkan U, $\sum$, dan V*. Karena matriks M merupakan matriks simetris maka nilai U dan V* selalu sama.

Berdasarkan Sheriff dan Williams [3], langkah-langkah analisis dampak perubahan yaitu:

a. Membuat matriks M berdasarkan data repositori perangkat lunak.
   Misalnya terdapat lima fungsi di mana $fungsi_1$ berubah 12 kali akibat $fungsi_2$ dan mengalami perubahan sendiri sebanyak 18 kali. Fungsi2 mengalami perubahan 12 kali kaitannya dengan $fungsi_1$ dan 24 kali kaitannya dengan $fungsi_3$. $Fungsi_3$ mengalami perubahan kode sebanyak 4 kali. $Fungsi_4$

mengalami perubahan kode sebanyak 14 kali kaitannya dengan fungsi$_5$ dan 6 kali terhadap dirinya sendiri. Fungsi$_5$ mengalami perubahan kode individu sebanyak 4 kali.

$$M = \begin{pmatrix} 30 & 12 & 0 & 0 & 0 \\ 12 & 36 & 24 & 0 & 0 \\ 0 & 24 & 28 & 0 & 0 \\ 0 & 0 & 0 & 20 & 14 \\ 0 & 0 & 0 & 14 & 18 \end{pmatrix}$$

b. Menghitung matriks U, $\sum$, dan V*.
Menghitung ketiga matriks tersebut dapat dilakukan dengan alat bantuan perhitungan matematika sehingga didapatkan matriks orthonormal U dan V* serta nilai eigen $\sum$ dari perkalian matriks M dengan matriks transposenya.

$$U = V^* = \begin{pmatrix} -0.31 & 0 & 0.90 & 0.32 & 0 \\ -0.76 & 0 & -0.03 & -0.66 & 0 \\ -0.58 & 0 & -0.44 & 0.69 & 0 \\ 0 & -0.73 & 0 & 0 & -0.68 \\ 0 & -0.68 & 0 & 0 & 0.73 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 59.3 & 0 & 0. & 0 & 0 \\ 0 & 33.0 & 0 & 0 & 0 \\ 0 & 0 & 29.6 & 0 & 0 \\ 0 & 0 & 0 & 5.1 & 0 \\ 0 & 0 & 0 & 0 & 4.9 \end{pmatrix}$$

c. Interpretasi hasil faktorisasi. Berdasarkan matriks $\sum$ yang terbentuk dapat diinterpretasikan bahwa kelompok dengan nilai eigen tertinggi memiliki risiko paling tinggi karena kelompok yang terbentuk memiliki persentase kepentingan terbesar. Persentase tingkat kepentingan kelompok dapat dihitung dengan membagi nilainya dengan jumlah nilai pada seluruh matriks. Misalnya untuk kelompok pertama sebesar 59.3/(59.3+33.0+29.6+5.1+4.9) yaitu 45%. Semakin tinggi persentase tingkat kepentingan kelompok tersebut maka semakin tinggi pula risiko yang dimiliki.

d. Analisis dampak pada perubahan fungsi.
Dampak perubahan fungsi dapat dianalisis dengan menghubungkan kolom pada matriks U dengan nilai singular pada matriks $\sum$. Tanda positif atau negatif yang ada pada matriks U menggambarkan arah perubahan fungsi. Misalnya pada kelompok pertama diketahui bahwa arah perubahan fungsi$_1$, fungsi$_2$, dan fungsi$_3$ adalah sama. Kelompok pertama ini juga yang memiliki kekuatan terbesar sehingga jenis perubahannya yang paling mempengaruhi program. Kelompok kedua adalah perubahan fungsi$_4$ dan fungsi$_5$ yang bersama-sama berada pada rangking kedua. Kemudian kelompok ketiga menunjukkan bahwa fungsi$_1$ dapat mengalami perubahan yang besar tanpa berdampak pada fungsi$_2$ dan fungsi$_3$.

2.3. Penggunaan Temu Kembali Informasi (Information Retrieval, IR)

Permintaan perubahan kode seperti pelaporan kesalahan direfleksikan dalam bentuk kalimat bahasa alami berupa pengidentifikasi dan komentar. Informasi itu dapat digunakan untuk menghitung analisis dampak perubahan menggunakan temu kembali informasi. Temu kembali informasi digunakan untuk memeriksa kemiripan permintaan perubahan dengan artefak-artefak dari perangkat lunak semisal dokumen fungsi atau fungsi itu sendiri. Tingkat kemiripan kemudian akan diurutkan berdasarkan nilai kemiripan yang paling tinggi. Semakin tinggi nilai kemiripan diasumsikan bahwa fungsi tersebut berhubungan erat dengan permintaan perubahan. Menurut Gethers [7], ada lima langkah dalam penggunaan temu kembali informasi untuk menangani analisis dampak perubahan:

a. Membangun korpus
   Pada tahap ini dilakukan pembangunan corpus berdasarkan dokumen dan kode yang dimiliki tiap-tiap fungsi.
b. Pra-pemrosesan teks
   Operator dan variabel yang ada pada kode perangkat lunak dikenai pra-pemrosesan teks. Operator-operator dihilangkan karena bukan merupakan fitur yang penting, sedangkan untuk variabel yang biasanya merupakan gabungan dari beberapa kata dilakukan pemisahan, misalnya variabel ”nomorUrut”, dilakukan proses pemecahan yang hasilnya akan menjadi dua kata yakni ‘nomor’ dan ‘urut’. Pada tahap ini juga dilakukan perubahan menjadi kata dasar atau yang biasa disebut *stemming.*
c. Membuat indeks korpus
   Fitur-fitur yang terdata dari langkah 2 akan diberi indeks dan dihitung bobotnya. Penghitungan bobot didasarkan pada metode *term-frequency inverse document frequency* (TF-IDF) dan direpresentasikan dalam bentuk matriks. Beberapa metode dapat dikenakan pada langkah ini dengan tujuan untuk mereduksi jumlah dimensi dari matriks, di antaranya menggunakan SVD.
d. Menjalankan *query*
   Query yang dimaksud adalah permintaan perubahan dari pengguna yang sudah dikenakan pra-pemrosesan teks sehingga telah direpresentasikan ke dalam sebuah matriks. Matriks tersebut kemudian dikenakan pengindeksan dan pembobotan yang langkahnya sama seperti pada langkah 3.
e. Mengestimasi himpunan dampak perubahan
   Pada langkah ini dilakukan pengecekan kemiripan menggunakan perhitungan heuristik semisal koefisien *jaccard* atau persamaan *cosine*. Data yang dihasilkan berupa urutan nilai persamaan dari *query* ke masing-masing fungsi. Dari hasil tersebut dapat dilakukan pengurutan fungsi yang paling dekat dengan mengurutkan nilai yang kesamaannya tinggi.

3. Diskusi Kritis

Berdasarkan kajian teori yang telah dibahas di bab sebelumnya, dapat diketahui bahwa metode analisis dampak perubahan perangkat lunak memiliki karakteristik masing-masing. Karakteristik tersebut dijelaskan pada paragraf selanjutnya.

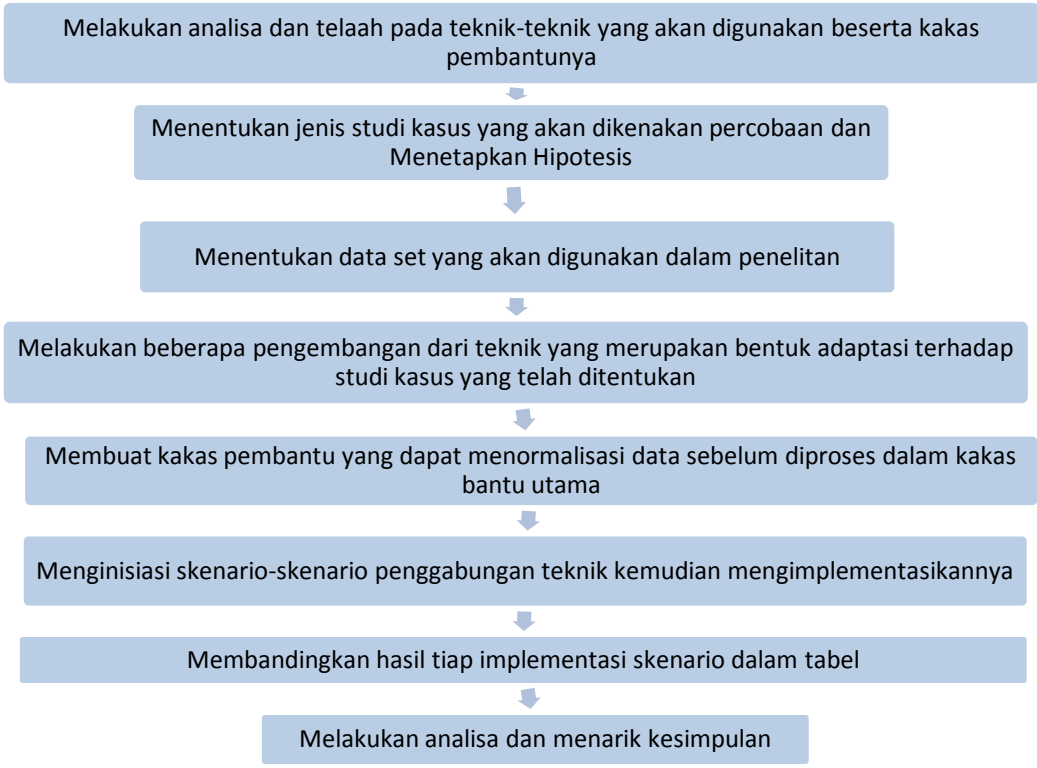
Graf panggil memiliki masukan berupa himpunan fungsi-fungsi dan fungsi yang terdampak. Sedangkan SVD memerlukan data masukan berupa data repositori

perangkat lunak. IR menggunakan permintaan perubahan kode yang tercatat sebagai masukan metode analisis dampak perubahan perangkat lunak.

Keluaran graf panggil adalah himpunan fungsi yang terdampak beserta bobot seberapa tinggi tingkatan masing-masing fungsi yang terdampak. Selain menghasilkan rangking fungsi yang terdampak, SVD dapat pula digunakan untuk mengetahui arah perubahan antar fungsi. Sedangkan IR menghasilkan himpunan nilai kesamaan fungsi-fungsi yang dapat menggambarkan tingkatan fungsi yang paling banyak mengalami perubahan. Tabel 1 menjelaskan kelebihan dan kelemahan utama dari masing-masing teknik. Penelitian ini bertujuan untuk menutupi masing-masing kelemahan dan
menggabungkan masing-masing kelebihan dari setiap teknik

Tabel 1. Kelebihan dan Kelemahan Teknik Analisis Dampak Perubahan

Parameter	Graf Panggil	SVD	Penggunaan IR
Kelebihan	Prediksi fungsi yang terkena dampak mudah didapat karena telah terbentuk graf hubungan antar fungsi	Mempelajari perubahan-perubahan di masa lalu dan menggunakannya sebagai acuan pendeteksi dampak	Mempertimbangkan korelasi antara permintaan perubahan tekstual dengan fungsi yang mengalami perubahan
Kelemahan	Pendeteksian urutan prediksi dampak pada fungsi yang mempunyai hubungan dengan banyak fungsi sulit karena graf yang terhubung memiliki bobot yang sama	Fungsi yang belum tercatat dalam daftar sejarah dapat mengakibatkan prediksi yang tidak tepat	Tidak maksimal untuk menangani kode yang berisi variabel-variabel yang kurang berarti (*meaningless)* atau tidak merepresentasikan fungsi.



Gambar 2. Metodologi Penelitian

4. Metodologi Penelitian

Metodologi yang dilakukan dalam penelitian ini digambarkan pada Gambar 2. Metodologi tersebut terdiri dari delapan langkah utama. Langkah pertama adalah melakukan analisis dan telaah pada teknik graf panggil, IR, dan SVD. Selain itu dilakukan eksplorasi terhadap kakas pembantu yang akan digunakan pada implementasi ini, diantaranya adalah MPLab, Matlab, dan IDE Netbeans. Langkah kedua adalah menentukan jenis studi kasus yang akan diuji coba. Studi kasus berupa perubahan pada perangkat lunak dengan kode sumber terbuka. Pada langkah kedua ini juga dilakukan penetapan hipotesis terhadap studi kasus. Langkah ketiga yaitu menentukan kumpulan data yang digunakan pada uji coba. Kumpulan data harus dapat digunakan sebagai masukan pada ketiga teknik yang akan diujicoba. Langkah keempat yaitu melakukan penyesuaian ketiga teknik yang digunakan dengan jenis studi kasus dan kumpulan data yang telah ditentukan. Penyesuaian ini dituliskan dalam draft implementasi.

Setelah proses pemahaman dan penyusunan konsep serta penentuan data, tahap selanjutnya adalah tahap implementasi. Langkah implementasi dimulai dari membuat kakas pembantu untuk menormalisasi data sehingga mempermudah proses selanjutnya. Setelah langkah tersebut, maka dilakukan implementasi kakas pembantu utama terhadap konsep yang sudah dibuat.

Pengujian dilakukan sesuai dengan skenario penggabungan. Langkah terakhir yaitu melakukan analisis terhadap hasil dan menarik kesimpulan. Analisis didasarkan pada hasil perangkingan teknik yang berdiri sendiri-sendiri dan juga hasil dari skenario penggabungan. Kesimpulan yang didapatkan membuktikan kebenaran hipotesis yang diajukan.

5. Studi Kasus dan Penyesuaian Teknik

Studi kasus dari penelitian ini adalah menganalisis dampak perubahan pada beberapa perangkat lunak kode terbuka yang ada pada beberapa repositori. Proses-proses dari ketiga teknik yang telah ditelaah ditambah atau diubah untuk disesuaikan dengan studi kasus. Penyesuaian dari masing-masing teknik yang akan digunakan terhadap studi kasus dijelaskan secara detail sebagai berikut.

a. Graf panggil
   - Mendapatkan kode sumber program yang akan dianalisis
   - Menyusun graf panggil dengan bantuan perangkat lunak (dengan MPLab)
   - Mendapatkan data perubahan program dari laporan kesalahan yang tercatat dan permintaan perubahan
   - Melakukan perangkingan terhadap fungsi yang terasosiasi dengan fungsi yang diubah

b. Penggalian sejarah dengan SVD
   - Mengumpulkan data histori perubahan program yaitu berupa laporan kesalahan yang tercatat dan permintaan perubahan
   - Menyusun graf panggil program agar dapat mengetahui fungsi-fungsi yang terhubung dengan bagian fungsi yang berubah
   - Menyusun tabel yang menyimpan rincian tiap perubahan pada masing-masing fungsi
   - Menyusun matriks M dari tabel rincian perubahan tersebut
   - Melakukan proses SVD pada matriks M dengan bantuan library perhitungan matriks pada Java yaitu JAMA.
   - Nilai batas ambang perubahan pada matriks S adalah 1 dan nilai batas ambang perubahan tiap fungsi adalah $1x10^{-15.}$
   - Menganalisis matriks hasil

c. Temu Kembali Informasi (IR)
   - Menganalogikan bahwa fungsi (kode+javadocs) adalah sebuah dokumen, sedangkan yang diartikan sebagai *query* adalah permintaan perubahan dan fungsi dimana kode telah diubah
   - Mendapatkan data dokumen dari penulisan dokumentasinya (javadocs biasanya tertulis di atas fungsi) sampai kode sumber dari fungsi tersebut
   - Melakukan pra-pemrosesan terhadap dokumen artefak dan *query.* Pra pemrosesan dokumen artefak terdiri dari fungi pemisahan camelCase, penghilangan *keyword* Java dan sebagainya. Detail algoritma untuk pra pemrosesan kode diadopsi dari Dit [9]. Secara umum ada tiga proses pada tahap ini

- *Tokenizer*
- *Stopper*
- *Stemmer*

- Melakukan pembentukan korpus dan pembobotan tiap term yang ada pada korpus tersebut
- Mencari kesamaan antar dokumen (fungsi) terhadap query
- Melakukan perangkingan nilai kesamaan

Kumpulan data yang digunakan dalam penelitian ini adalah tiga jenis perangkat lunak kode terbuka, diantaranya sebagai berikut.

1. JabRef v2.6
2. jEdit v4.3

Artifak dari tiap perangkat lunak di atas dan berbagai kebutuhan lainnya seperti permintaan perubahan dan isu kesalahan kode diambil dari http://www.cs.wm.edu/semeru/data/benchmarks/, yang telah diolah dan pernah digunakan dalam penelitian sejenis sebelumnya oleh Gethers [7].

6. Skenario Penggabungan

Tujuan utama dari penelitian ini adalah melakukan analisis skenario penggabungan hasil dari ketiga teknik yang ditelaah sehingga didapat skenario yang memiliki hasil analisis dampak yang lebih akurat. Hipotesis dalam penelitian adalah dengan melakukan penggabungan teknik maka akan dapat meningkatkan nilai presisi dan *recall* dan meningkatkan akurasi rata-rata. Penilaian skenario penggabungan teknik menggunakan metode N/2 rangking tertinggi [8], dimana N adalah ukuran yang diinginkan untuk mendapatkan nilai set dampak akhir. Ide pada N/2 adalah masing-masing teknik sama-sama memberikan kontribusi ke set akhir yang dihasilkan. Jika fungsi yang sama disarankan oleh kedua teknik, akan muncul hanya sekali pada set dampak akhir. Fungsi-fungsi akan terus dipilih, bergantian pada tiap sumber daftar peringkat, sampai dampak set sebanyak N diperoleh.

Dalam pengaturan yang realistis, beberapa permintaan perubahan biasanya ditentukan dalam bahasa alami (misalnya, bahasa Inggris). Hal ini masuk akal untuk mengasumsikan bahwa permintaan perubahan, dalam beberapa kasus, merupakan satu-satunya sumber informasi yang tersedia untuk melakukan pemeliharaan yang diperlukan. Oleh karena itu, metode IR yang dapat mengakomodasi masukan berupa bahasa alami diperlakukan sebagai pusat yang selalu ada pada tiap skenario penggabungan. Skenario penggabungan yang dilakukan antara lain.

- Kombinasi IR dengan Graf Panggil (IRgp)
- Kombinasi IR dengan SVD (IRsvd)
- Kombinasi IR, Graf Panggil, SVD (IRgpsvd)

7. Hasil dan Analisis Percobaan

Terdapat dua buah dataset yang digunakan dalam ujicoba penelitian ini yaitu JabRef v2.6 dan JEdit v.4.2. Dataset JabRef terdiri dari 4,604 fungsi dan 40 file goldset sedangkan JEdit terdiri dari 6.413 fungsi dan 272 file goldset.

Fungsi perubahan yang sama diujicoba dengan ketiga metode, yaitu IR, SVD, dan graf panggil. Pengambilan perubahan fungsi dilakukan secara acak

Uji coba dengan IR menggunakan query gabungan dari deskripsi singkat dan deskripsi lengkap permintaan perubahan. Uji coba SVD dilakukan dengan menjadikan 80% perubahan fungsi sebagai data training goldset dan 20% sisanya sebagai data testing testset. Uji coba dengan graf panggil dilakukan dengan membangun graf panggil dari kode dataset dan kemudian memasukkan fungsi yang berubah untuk diujicoba.

Performa sistem diukur menggunakan nilai presisi dan *recall*. Presisi adalah perbandingan hasil prediksi yang terbukti sesuai dengan data actual terhadap keseluruhan hasil prediksi himpunan dampak. Sedangkan *recall* adalah perbandingan hasil prediksi yang terbukti sesuai dengan data actual terhadap keseluruhan data himpunan dampak yang aktual. Tabel 2 menunjukkan matriks konfusi yang digunakan sebagai dasar perhitungan presisi dan *recall*.

Nilai presisi dan *recall* uji coba yang dilakukan dirangkum pada Tabel 3. Untuk dataset JabRef, nilai presisi dan *recall* terbaik terdapat pada kombinasi tiga buah metode yaitu sebesar 60% untuk presisi dan 54% untuk *recall*. Sedangkan untuk dataset jEdit, prediksi optimal yang dinilai dengan presisi didapatkan dengan kombinasi metode IR dan graf panggil, IRgp. Prediksi yang dinilai dengan *recall* menunjukkan hasil optimal dari kombinasi tiga buah meode. Nilai presisi dataset jEdit yang didapatkan dari kombinasi metode IR dengan graf panggil tidak menunjukkan perbedaan yang signifikan dengan kombinasi ketiga metode IRgpsvd.

Dari hasil tersebut dapat diketahui bahwa skenario penggabungan metode IR dengan metode lain menghasilkan nilai presisi dan *recall* yang lebih baik dibanding dengan metode IR yang berdiri sendiri. Penggabungan IR dengan metode lain menjadikan prediksi fungsi yang terdampak lebih mendekati kebenaran. Penggabungan metode IR dengan SVD menunjukkan hasil yang lebih optimal dibanding IR dengan graf panggil. Kombinasi ketiga metode secara bersama menghasilkan prediksi fungsi terdampak yang paling optimal. Hal ini dikarenakan adanya kemungkinan fungsi terdampak tidak ada pada artifak data latih pada metode SVD namun dikenali pada graf panggil. Dan juga sebaliknya, adanya fungsi terdampak yang tidak dikenali pada graf panggil namun dapat diidentifikasi dengan metode SVD.

$$\text{Pr}esisi = \frac{TP}{TP + FP} \times 100\% \quad (1) \qquad \text{Re}call = \frac{TP}{TP + FN} \times 100\% . \quad (2)$$

Tabel 2. Matriks Konfusi

		Impact Set Actual	
		(+)	**(-)**
Impact Set Prediction	(+)	TP	FP
	(-)	FN	TN

Tabel 3. Nilai Presisi dan *Recall* Ujicoba

Dataset	**JabRef4.6**		**jEdit2.3**	
	Presisi	***Recall***	**Presisi**	***Recall***
IR	0,0625	0,1453	0,0851	0,2946
IRgp	0,3125	0,2792	0,1009	0,3461
IRsvd	0,3575	0,4121	0,0908	0,3966
IRgpsvd	0,6075	0,5454	0,1063	0,4479

8. Ancaman Terhadap Validitas

Beberapa hal yang merupakan ancaman terhadap validitas dari penelitian ini diidentifikasi. Ancaman tersebut sangat mempengaruhi hasil dari penelitian ini.

Pertama, pra pemrosesan pada kode dan javadocs belum optimal karena beberapa sebab. Diantaranya antara lain ketidakkonsistenan penamaan fungsi dan variabel yang digunakan serta tidak lengkapnya dokumen javadocs untuk tiap fungsi. Dari dataset diketahui hanya beberapa yang mempunyai dokumen javadocs. Efek dari sebab-sebab tersebut sangat mempengaruhi pencarian nilai kesamaan antara *query* dan dokumen fungsi (proses IR).

Kedua, matriks SVD yang dibentuk pada penelitian ini masih belum optimal karena hanya mengandalkan sedikit data sejarah perubahan dengan jarak beberapa sub versi dari perangkat lunak yang diuji coba. Agar matriks yang dibangun lebih *reliable* sebaiknya data sejarah perubahan kode yang diambil mulai dari versi pertama sampai versi terakhir dari perangkat lunak yang diuji coba. Pendataan data sejarah perubahan kode secara lengkap cukup sulit untuk level fungsi karena tidak tertangani oleh perangkat lunak sub version seperti TortoiseSVN yang digunakan dalam penelitian ini, sehingga perlu pendataan manual yang membutuhkan banyak waktu dan tenaga.

Penelitian lanjutan yang bisa dilakukan adalah untuk menambal atau mengurangi dampak dari ancaman-ancaman yang telah teridentifikasi sehingga hasil dari penelitian bisa lebih baik dari penelitian saat ini.

9. Simpulan

Penelitian ini menyajikan pendekatan baru untuk mengelola analisis dampak perubahan kode dengan melakukan integrasi dari beberapa metode sehingga saling melengkapi kelebihan satu sama lain. Pendekatan yang dilakukan berupa skenario penggabungan dari tiga metode, diantaranya IR, graf panggil, dan penggalian sejarah menggunakan SVD.

Hasil percobaan secara empiris menunjukkan bahwa kombinasi dari ketiga metode yang dibahas menghasilkan rata-rata nilai presisi dan *recall* yang paling

optimal daripada IR individu atau kombinasi dua metode saja. Kombinasi dua metode juga menyajikan presisi dan *recall* yang baik daripada pendekatan individu. Dengan keadaan tersebut, maka hipotesis bahwa kombinasi dari metode analisis dampak menghasilkan nilai presisi dan *recall* yang lebih baik daripada pendekatan individu seperti yang dirumuskan pada awal penelitian telah terbukti dengan benar.

10. Penelitian Lanjutan

Penelitian lanjutan yang dimungkinkan adalah melakukan percobaan empiris untuk penggabungan beberapa metode analisis dampak yang lain, dan saling membandingkannya. Tujuannya agar ditemukan kombinasi yang menghasilkan nilai presisi dan *recall* yang paling baik sehingga dapat dikenalkan dan dipraktekkan ke industri perangkat lunak.

REFERENCES

[1] Grove, D., Defouw, G., and Dean, J. “Call Graph Construction in Object-Oriented Language”. 2008

[2] Li,Bixin , Sun,Xiaobing , Leung,Hareton. “Combining concept lattice with call graph for impact analysis”. 2012.

[3] Sheriff, M., and Williams, L. “Empirical Software Change Impact Analysis using Singular Value Decomposition”. 2008. International Conference on Software Testing, Verification, and Validation.

[4] Huang, L. and Song, Y.-T., "Dynamic Impact Analysis Using Execution Profile Tracing," Proceedings of the International Conference on Software Engineering Research, Management, and Applications, Aug 9-11, 2006, pp. 237-244.

[5] Wilcoxon, Frank. “Individual Comparisons by Ranking Method”. Biometrics Buletin, Vol.1, pp. 80-83. 1945

[6] Baker,Kirk. “Singular Value Decomposition Tutorial”. 2005.

[7] Gethers, M., Dit, B., Kagdi, H., Poshyvanyk, D., “Integrated Impact Analysis for Managing Software Changes”. 2012.

[8] Kagdi, H., Gethers, M., Poshyvanyk, D., and Collard, M., "Blending Conceptual and Evolutionary Couplings to Support Change Impact Analysis in Source Code", in Proc. of 17th IEEE Working Conference on Reverse Engineering (WCRE'10), Beverly, Massachusetts, USA, October 13-16 2010, pp. 119-128.

[9] Dit, B., Guerrouj, L., Poshyvanyk, D., and Antoniol, G., "Can Better Identifier Splitting Techniques Help Feature Location?" in Proc. of 19th IEEE International Conference on Program Comprehension (ICPC'11), Kingston, Ontario, Canada, June 22-24 2011, pp. 11-20.