SYSTEMIC
Vol. 04, No. 01, Agustus 2018, 39-43

ANALISIS DAMPAK PERUBAHAN PERANGKAT LUNAK MENGGUNAKAN GRAF RELASI FUNGSI-ATRIBUT

Ahmad Yusuf[1)], Noor Wahyudi[2)], Nita Yalina[3)]

[1,2,3)] UIN Sunan Ampel Surabaya
e-mail: ahmadyusuf@uinsby.ac.id[1)] ,n.wahyudi@uinsby.ac.id[2)]
nitayalina@uinsby.ac.id[3)]

Abstrak

Analisis dampak perubahan perangkat lunak merupakan sebuah teknik untuk melakukan estimasi dampak adanya perubahan komponen pada artifak perangkat lunak. Salah satu perubahan komponen artifak yang mungkin terjadi adalah artifak kode sumber. Analisis dampak perubahan ini bertujuan supaya ketika terjadi perubahan kode sumber dapat diestimasi sejauh mana perubahan itu berpengaruh ketika perubahan itu benar-benar diterapkan pada perangkat lunak yang dibangun. Penelitian ini mengusulkan sebuah metode baru untuk melakukan analisis dampak perubahan perangkat lunak dengan menggunakan graf relasi fungsi-atribut. Graf relasi fungsi-atribut terbentuk atas konsep matrik kohesi yang mengalami perluasan hingga pengaksesan atribut yang dipanggil oleh tiap fungsi sehingga dapat meningkatkan performa hasil analisis dampak perubahan perangkat lunak. Hasil pengujian terhadap aplikasi Java open source memperlihatkan bahwa metode ini mampu mendapatkan dampak perubahan yang lebih banyak dibandingkan metode graf panggil. Metode graf relasi mampu meningkatkan recall 1.8 % dari dataset yang digunakan.

Kata Kunci: *Analisis dampak perubahan, graf relasi fungsi-atribut, perangkat lunak, ,matrik kohesi*

Abstract

Change Impact Analysis in software evolution is a technique for estimating the impact of changes in components on software artifacts. Changes in the artifact component that might occur from source code artifact. The analysis of the impact of this change is intended so that when there is a change in the source code, it can be estimated the extent to which the change has an effect when the change is actually applied to the software being built. This study proposes a new method for analyzing the impact of software changes by using graphs of function-attribute relations. Relationship graphs of functions are formed on the concept of cohesion matrix which undergoes expansion to access attributes called by each function so that it can improve the performance of the results of the analysis of the impact of software changes. The test results on open source Java applications show that this method is able to get more impact than the call graph method. The relation graph method is able to increase the recall 1.8% of the dataset used..

Keywords: *Change impact analysis, function-attribute graph relations, software evolution, cohesion matrix*

1. PENDAHULUAN

Analisis dampak perubahan perangkat lunak, yang selanjutnya disebut sebagai *change impact analysis* (CIA), merupakan sebuah teknik untuk melakukan estimasi dampak adanya perubahan komponen pada artifak perangkat lunak. Artifak perangkat lunak yang mungkin dapat mengalami perubahan antara lain model rancangan, kode sumber, dan lain-lain.

Perubahan ini dapat terjadi karena beberapa faktor antara lain perpindahan lingkungan perangkat lunak ke lingkungan baru, perubahan skenario sistem, perubahan kebutuhan, dan sebagainya.

Dengan adanya analisis dampak perubahan ini, para pengembang perangkat lunak akan terbantu dalam mengestimasi dampak yang terjadi ketika ada perubahan artifak. Tujuannya adalah sebelum perubahan tersebut diterapkan, perubahan tersebut dikaji terlebih dahulu untuk diketahui sejauh mana dampaknya ketika perubahan itu benar-benar diterapkan. Analisis dampak perubahan lebih

difokuskan untuk meminimalkan resiko-resiko yang berpotensi terjadi akibat adanya perubahan sehingga harapannya akan lebih menghemat cost dan efisiensi waktu.

Di dalam kode sumber, perubahan dapat terjadi pada struktur kelas, fungsi maupun atribut. Penelitian sebelumnya yang dilakukan terkait dengan analisis dampak perubahan dalam konteks perubahan kode sumber dilakukan oleh Li Bixin [1] dengan menggabungkan konsep *lattice* dan graf panggil. Metode yang dikembangkan ini mampu menampilkan indikator faktor dampak dari setiap dampak perubahan yang ada sehingga memudahkan dalam pemeriksaan dampak sesuai dengan tingkat resikonya.

Namun, penelitian ini hanya sampai pada level fungsi sedangkan dalam kenyataan yang ada terdapat kondisi dimana fungsi dapat terhubung karena pengaksesan atribut yang sama sehingga fungsi-fungsi yang terkait dengan atribut tersebut juga akan terkena dampak jika ada perubahan pada suatu fungsi.

Penelitian lain juga dilakukan oleh Imran Baig [2] dengan menggunakan matrik kohesi untuk melihat relasi antara fungsi dan attribut pada sistem perangkat lunak berorientasi objek. Terlihat bahwa metode yang dikembangkan ini mampu menangani relasi antara fungsi dan atribut dengan baik.

Kontribusi pada penelitian ini adalah dengan mengembangkan konsep gabungan antara graf panggil dan matrik kohesi yang menbentuk graf relasi fungsi. Penggunaan graf relasi fungsi memungkinkan untuk memperoleh dampak perubahan yang lebih banyak dibandingkan metode graf panggil sehingga nantinya dapat meningkatkan performa hasil analisis dampak perubahan perangkat lunak.

Sistematika penulisan artikel ini terdiri atas 5 bagian. Bab I menjelaskan pendahuluan yang menjelaskan permasalahan secara umum serta latar belakangnya. Bab II merupakan studi literatur yang memaparkan metode-metode yang digunakan pada penilitian ini. Bab III berisi metode-metode yang digunakan dalam menganalisi dampak perubahan perangkat lunak. Pada bab IV menjelaskan skenario pengujian dari metode yang digunakan, dan pada bab V menjelaskan hasil pengujian yang dilakukan berdasar skenario pengujian pada bab IV.

2. METODE

Metode dalam menganalisis dampak perubahan komponen artifak kode sumber pada perangkat lunak berfokus pada pembuatan graf relasi fungsi-atribut. Graf relasi fungsi-atribut terbentuk oleh konsep graf panggil dan matrik kohesi

2.1 Graf Relasi Fungsi-Atribut untuk Analisis Dampak Perubahan pada Tingkat Fungsi dan Atribut

Untuk menganalisis dampak perubahan ketika ada fungsi yang berubah maka pada teknik ini variabel yang terlibat adalah fungsi dan atribut pada masing-masing kelas. Hasil dampak perubahan tingkat kelas akan kembali ditelaah pada tahapan ini apakah benar-benar terjadi perubahan didalamnya jika dilihat dari isi kelaqs tersebut yaitu fungsi dan atribut.

Penggunaan atribut bertujuan untuk mengantisipasi pengaruh perubahan bukan hanya pada panggil memanggil suatu fungsi dengan fungsi lainnya, tetapi juga pada atribut kelas yang diakses dalam fungsi yang berubah tersebut. Atribut kelas merupakan atribut pada tingkatan kelas yang dapat diakses semua fungsi pada kelas tersebut. Jika ada suatu fungsi berubah dan mengandung pengaksesan atribut kelas didalamnya, maka ada kemungkinan fungsi lain yang menggunakan atribut kelas tersebut juga berubah. Meskipun fungsi-fungsi tersebut tidak memanggil fungsi satu sama lain.

Model graf yang dibuat bertujuan untuk mengombinasikan hubungan antara fungsi dan atribut dalam suatu kelas serta hubungan fungsi-fungsi dari satu kelas ke kelas yang lain. Tidak ada analisis hubungan antara atribut suatu kelas terhadap fungsi kelas lainnya karena diasumsikan atribut pada suatu sistem seharusnya bersifat *private*. Pengaksesan atribut private dalam suatu kelas pada umumnya menggunakan jembatan enkapsulasi dengan fungsi set atau get.

Pemodelan graf relasi fungsi-atribut menggunakan metode metrik kohesi serta graf panggil. Metrik kohesi berfungsi untuk membangun graf relasi antara fungsi dan atribut dalam satu kelas. Graf panggil digunakan dalam menghubungkan fungsi-fungsi dari satu kelas ke kelas lainnya yang mempunyai hubungan memanggil fungsi.

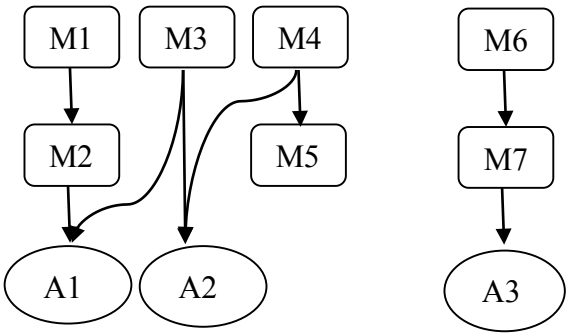
Langkah-langkah pemodelan graf relasi fungsi-atribut adalah sebagai berikut

1. Membuat metrik kohesi dari masing-masing kelas
2. Menggabungkan graf metrik kohesi dengan graf panggil
3. Visualisasi graf relasi

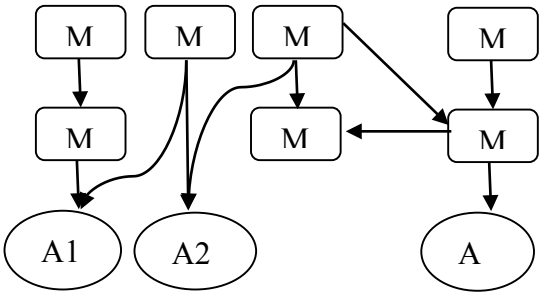
Metrik kohesi membangun graf relasi pada masing-masing kelas. Gambar 4 menunjukkan contoh hasil graf relasi dari metrik kohesi pada setiap kelas. Pada contoh terdapat dua kelas yaitu C1 dan C2. Pada kelas C1 terdapat fungsi M1, M2, M3, M4, M5 dan atribut A1 serta A2, Sedangkan pada C2 terdapat fungsi M6 dan M7 serta atribut A3. Berdasarkan teori metrik kohesi maka dibangun minimal n graf relasi yang tidak berhubungan dimana n merupakan jumlah kelas. Pada Gambar 3

terdapat 2 metrik relasi dimana sebelah kiri pada gambar merupakan representasi dari kelas C1, sedangkan representasi C2 berada disebelah kanan.

Graf-graf relasi fungsi-atribut yang belum berkaitan satu sama lain akan dihubungkan dengan menggunakan metode graf panggil.



Gambar 1. Graf relasi fungsi-atribut dengan metrik kohesi



Gambar 2. Graf relasi fungsi-atribut (metrik kohesi & graf relasi)

Suatu fungsi dimungkinkan untuk memanggil fungsi yang berada di kelas lainnya. Hubungan panggil-memanggil antara fungsi tersebut yang direpresentasikan selanjutnya pada graf relasi fungsi-atribut. Sebagai contoh fungsi M7 pada kelas C2 dipanggil oleh fungsi M4 dan memanggil fungsi M5 pada kelas C1. Gambar 4 menunjukkan representasi graf relasi fungsi-atribut hasil penggabungan dengan metode graf panggil.

Graf relasi fungsi-atribut seperti contoh pada gambar 5 yang akan digunakan dalam menganalisis dampak perubahan perangkat lunak. Dari daftar fungsi-fungsi yang berubah maka akan dilakukan analisis dampak perubahan. Analisis dampak dari fungsi-fungsi yang berubah dilakukan dengan mengecek sampai pada atribut apa saja yang terhubung. Dari atribut yang terlibat maka ditelusuri fungsi-fungsi apa saja yang terhubung. Fungsi-fungsi yang terhubung merupakan dampak perubahan pada tingkat fungsi dan atribut yang dianalisis dengan graf relasi fungsi-atribut.

3. SUBJEK EVALUASI

3.1 Dataset

Subyek penelitian dari studi ini adalah aplikasi-aplikasi open source berbasis Java yaitu ArgoUML, yakni aplikasi untuk UML Editor, dan JabRef, sebuah aplikasi pengatur referensi BibTeX[7]. Tabel 3 menunjukkan karakteristik dari dataset tersebut.

3.2 Parameter pengukuran performa sistem

Adapun untuk parameter pengukur performa sistem menggunakan nilai presisi dan *recall*. Presisi adalah perbandingan terhadap hasil prediksi yang terbukti sesuai dengan data aktual, dibanding dengan keseluran hasil prediksi himpunan dampak. Sedangkan *recall* adalah nilai perbandingan terhadap hasil prediksi yang terbukti sesuai dengan data aktual, dibanding dengan keseluran data himpunan dampak yang aktual. Tabel 2 menunjukkan parameter-parameter yang digunakan untuk mengukur presisi dan *recall*.

Semakin tinggi nilai presisi dan *recall*, menunjukkan bahwa hasil prediksi dampak telah sesuai dengan data aktual. Namun, biasanya nilai kedua parameter ini saling berbanding terbalik, untuk itu, jika perolehan kedua nilai ini tinggi dan perbedaan nilai antara presisi dan *recall* tidak terlalu jauh maka hasil analisis dapat dibilang cukup baik.

$$\Pr esisi = \frac{TP}{TP + FP} \times 100\% \quad (1)$$

$$\mathrm{Re}call = \frac{TP}{TP + FN} \times 100\%. \quad (2)$$

Tabel 1
Karakteristik dataset

Aplikasi	Versi	LoC	Jumlah File	Jumlah Fungsi
ArgoUML	0.22	148,892	1,439	11,000
JabRef	2.6	74,182	577	4,604

Tabel 2
Parameter pengukuran performa

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

3.3 Skenario Pengujian

Pengujian dilakukan dengan membaca kode sumber dari dataset open source yang digunakan. Pada masing-masing kode sumber akan didapatkan daftar kelas, fungsi, dan atribut. Dari masing-asing kode sumber tersebut kemudian dilakukan pengambilan secara acak sebesar *p*% dari perubahan fungsi kode sumber dari versi sebelumnya sebagai himpunan perubahan kode sumber, sedangkan sisanya sebagai dampak perubahan.

Nilai *p* pada pengambilan acak ini dilakukan pada 30%, 40%, 50%, dan 60%. Dari masing-masing nilai *p* didapatkan nilai *True Positive*, *False Positive*, *False Negative* sehingga didapatkan nilai presisi dan *recall*.

4. HASIL PENGUJIAN

4.1 Hasil Pengujian

4.1.1 JabRef v2.6

JabRef versi 2.6 terdiri atas 577 file, 4.604 fungsi dan 74.182 baris kode. Pada kode sumber JabRef versi 2.6 dilakukan pengujian berdasarkan skenario pengujian. Tabel 3 menunjukkan hasil pengujian dengan menggunakan data kode sumber JabRef v.2.6.

Terlihat bahwa jika nilai *p*% semakin besar maka nilai TP yang dihasilkan semakin tinggi. Pada Tabel 3 juga memperlihatkan perbandingan antara metode graf fungsi-attribut dengan graf panggil fungsi bahwa nilai TP dan FP pada graf relasi fungsi lebih tinggi dibandingkan dengan yang menggunakan graf panggil, sedangkan nilai FN pada metode yang diusulkan lebih kecil dibandingkan dengan graf panggil.

4.1.2 ArgoUML v0.22

ArgoUML versi 0.22 terdiri atas 1,439 file dan 11,000 fungsi. Pengujian dilakukan pada kode sumber ArgoUML sesuai dengan skenario pengujian yang dijelaskan sebelumnya. Hasil pengujian juga ditunjukkan pada Tabel 3.

Nilai TP akan semakin besar jika persentase himpunan perubahan juga meningkat baik pada metode graf relasi fungsi atau graf panggil. Nilai TP dan FP pada metode yang diusulkan juga lebih besar dibandingkan dengan metode grafpanggil. Nilai FN pada metode yang diusulkan memiliki nilai yang lebih kecil dibandingkan pada graf panggil.

Tabel 3
Hasil pengujian pada JabRef v2.6 dan ArgoUML v0.22

p% himpunan perubahan	*True Positive*		*False Positive*		*False Negative*		*Presisi*		*Recall*	
	Graf Panggil	Usulan Metode	Graf Panggil	Usulan Metode	Graf Panggil	Usulan Metode	Graf Panggil	Usulan Metode	Graf Panggil	Usulan Metode
Dataset : JabRef v2.6										
30	37	39	5	12	68	66	0.88	0.76	0.35	0.37
40	45	47	5	16	60	58	0.90	0.75	0.43	0.45
50	52	55	7	17	53	50	0.88	0.76	0.50	0.52
60	57	60	7	17	48	45	0.89	0.78	0.54	0.57
Dataset : ArgoUML v0.22										
30	24	25	1	1	52	51	0.96	0.96	0.32	0.33
40	29	30	1	1	47	46	0.97	0.97	0.38	0.39
50	33	34	1	1	43	42	0.97	0.97	0.43	0.45
60	37	38	1	1	39	38	0.97	0.97	0.49	0.50

4.2 Analisis Hasil Pengujian

Dari kedua pengujian diatas memiliki karakteristik hasil yang sama. Nilai TP sebanding dengan nilai persentase himpunan perubahan. Hal ini dapat terjadi dikarenakan himpunan perubahan yang menjadi objek analisis jumlahnya semakin

besar, sehingga himpunan dampak yang berhasil terdeteksi menjadi lebih banyak jumlahnya. Berbeda halnya dengan *p* yang bernillai kecil, himpunan perubahan yang menjadi objek analisi hanyalah berjumlah sedikit, untuk itu hasil himpunan perubahan pada data sebenarnya hanya sedikit yang dapat terdeteksi.

Nilai TP pada metode yang diusulkan lebih besar dibandingkan dengan metode graf panggil. Hal ini menunjukkan metode yang diusulkan mampu mendeteksi dampak perubahan lebih banyak dibandingkan metode graf panggil. Hanya saja FP yang dihasilkan juga berdampak lebih besar, sedangkan nilai FN pada metode yang diusulkan lebih kecil dibandingkan dengan graf panggil. Hal ini disebabkan prediksi dampak perubahan yang dihasilkan lebih banyak dibandingkan dengan metode graf panggil. Hal ini dapat terjadi dikarenakan graf fungsi-atribut turut memperhitungkan atribut yang diakses sehingga analisis yang dilakukan lebih menjangkau fungsi yang ikut berubah karena pengaksesan atribut.

Oleh karenanya, metode yang disulkan mampu menghasilkan nilai *recall* yang lebih baik dari metode sebelumnnya. Rata-rata selisih recall yang ditunjukkan pada Tabel 5 adalah sebesar 2.25 % untuk dataset JabRef v2.6, 1.25 % untuk ArgoUML v0.22 dan 1.8 % untuk keseluruhan dataset.

Pada kedua metode nilai FN lebih besar dibandingkan dengan nilai FP, sehingga nilai presisi yang dihasilkan lebih besar dibandingkan nilai *recall*

SIMPULAN

Analisis dampak perubahan pada artifak perangkat lunak sangat penting untuk dilakukan karena dapat mengestimasi sejauh mana dampak yang ditimbulkan jika perubahan tersebut terjadi.

Hasil pengujian menunjukkan bahwa metode graf relasi fungsi yang diusulkan mampu mendapatkan dampak perubahan yang lebih banyak dibandingkan metode graf panggil. Metode graf relasi mampu memperbaiki nilai *recall* metode graf panggil dengan rata-rata peningkatan sebesar 1.8 persen dari dataset yang digunakan.

DAFTAR PUSTAKA

[1] Li Bixin, Xiaobing Sun and Hareton Leung, “Combining concept lattice with call graph for impact analysis” *In Engineering Software Elsevier Advances 1-13*, 2012.

[2] Imran Baig, “Thesis report: Measuring Cohesion and Coupling of Object Oriented Systems”, *Blekinge Institute of Technology*, Sweden, 2004.

[3] Gethers, M., Dit, B., Kagdi, H., and Poshyvanyk, D., "Integrated Impact Analysis for Managing Software Changes" *In Proceedings of 34th IEEE/ACM International Conference on Software Engineering (ICSE'12), Zurich, Switzerland, pp. 430-440*, 2012.

[4] Ryder, B.G., "Constructing the Call Graph of a Program" *in Software Engineering, IEEE Transactions on, vol. SE-5, no.3pp. 216- 226*, 1979.

[5] Bodden E. “JAnalyzer, a visual static analyzer for Java. Tech. Rep.”, University of Kent, Computing Laboratory, 2003.

[6] Rai RV, Gagnon E, Hendren L, Lam P, Pominville P, Sundaresan V. Soot, “A Java Optimization Framework” *In: Proceedings of the conference of the centre for advanced studies on collaborative research*, 1999.

[7] Gethers, M., Dit, B., Kagdi, H., and Poshyvanyk, D., "Integrated Impact Analysis for Managing Software Changes" *In Proceedings of 34th IEEE/ACM International Conference on Software Engineering (ICSE'12), Zurich, Switzerland, pp. 430-440*, 2012.

[8] Hitz Martin, Montazeri, Behzad. “Measuring coupling and cohesion in object-oriented systems” *In Conference: Proc. Int. Symposium on Applied Corporate Computing*, 1995.