Terakreditasi SINTA Peringkat 2
Surat Keputusan Dirjen Penguatan Riset dan Pengembangan Ristek Dikti No. 10/E/KPT/2019

Terbit online pada laman web jurnal: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 5 No. 2 (2021) 721 - 728 ISSN Media Elektronik: 2580-0760

Analisis Dampak Perubahan Artefak Kebutuhan Berdasarkan Kedekatan Semantik Pada Pengembangan XP

Arrijal Nagara Yanottama[1], Siti Rochimah[2]
[1,2]Teknik Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember
[1]arrijal.yanottama@um.ac.id, [2]siti@if.its.ac.id

Abstract

The Extreme Programming (XP) development method is popular because of the flexibility of the development process, it can accommodate changes quickly. But this method has a weakness in terms of documentation. It is expected that the speed of discovering which parts of the source code need to be changed will be greatly improved by analyzing the impact of changes on the requirements document. In this study, a method of analyzing the impact of changes is proposed by tracing changes in the artifact of the need to find out the source code that occurs. Early language methods and semantic approaches are used. Based on the proximity of the semantics, it will be analyzed to find out the elements in the source code that use the Spearman Correlation Coefficient. The test dataset in this study consisted of the source code in the PHP programming language as well as the functional requirements of the software. Requirements change list is generated by analysis of the latest 2 (two) expert versions of the source code. The changing needs are described in a user story document. Based on the test results in this study, the average precision was 0.1725 and the average recall value was 0.6041.

Keywords: XP, traceability link, TF-IDF, semantic proximity, natural language processing

Abstrak

Metode pengembangan Extreme Programming (XP) termasuk populer, dikarenakan adanya fleksibilitas proses pengembangan sehingga dapat menampung perubahan dengan cepat. Tetapi metode ini memiliki kelemahan dalam hal dokumentasi. Tim pengembang kesulitan menemukan bagian kode sumber yang perlu diubah jika kebutuhan diubah. Seberapa besar dampak atas perubahan kebutuhan terhadap kode sumber tersebut juga tidak diketahui. Diharapkan kecepatan menemukan bagian mana dari kode sumber yang perlu diubah akan sangat meningkat dengan analisis dampak perubahan pada dokumen kebutuhan. Dalam penelitian ini diusulkan metode analisis dampak perubahan dengan melakukan penelusuran perubahan yang terjadi pada artefak kebutuhan untuk mengetahui kode sumber yang terdampak untuk diterapkan dalam lingkungan pengembangan XP. Metode pemrosesan bahasa alami dan kedekatan semantik digunakan. Berdasarkan nilai kedekatan semantiknya akan dianalis untuk mengetahui elemen pada kode sumber yang terdampak menggunakan Koefisien Korelasi Spearman. Dataset pengujian dalam penelitian ini terdiri dari kode sumber dalam bahasa pemrograman PHP serta kebutuhan fungsional dari perangkat lunak. Dari kode sumber tersebut akan dilakukan pemrosesan populasi kode sumber. Daftar perubahan kebutuhan dibuat oleh pakar dengan menganalisa 2 (dua) versi terbaru dari kode sumber. Perubahan kebutuhan tersebut di deskripsikan dalam sebuah dokumen user story. Berdasar hasil ujicoba pada penelitian ini diperoleh rata-rata nilai precision 0.1725 serta rata-rata nilai recall 0.6041.

Kata kunci: XP, link penelusuran, TF-IDF, kedekatan semantik, pemrosesan bahasa alam.

1. Pendahuluan

Seiring dengan cepatnya perkembangan Teknologi Informasi, para pembuat perangkat lunak juga dituntut dapat membuat sebuah perangkat lunak yang dapat memenuhi kebutuhan dari pengguna secara tepat, efisien dan efektif. Untuk itulah dikembangkan sebuah metodologi pengembangan perangkat lunak yang dinamakan Xtreme Programming (XP)[1]. XP menekankan pada proses komunikasi yang intens antara tim pengembang dan pengguna sehingga dapat mempersingkat alur proses pembuatan perangkat lunak sehingga waktu pembuatan perangkat lunak dapat dilakukan secara cepat. Dalam prosesnya, XP memiliki banyak kelemahan karena minimnya dokumentasi pengembangan. Sering terjadinya perubahan kebutuhan akan membuat tim pengembang kesulitan untuk mengetahui elemen yang terdampak pada kode sumber sehingga dapat menyebabkan terlambatnya proses pembuatan perangkat serta meningkatkan resiko

Diterima Redaksi: 12-07-2021 | Selesai Revisi: 09-08-2021 | Diterbitkan Online: 20-08-2021

terjadinya penurunan kualitas perangkat lunak karena seberapa besar dampak atas perubahan tersebut belum dapat diketahui.

Selama dekade terakhir, para peneliti telah fokus pada bidang-bidang tertentu dari masalah kebertelusuran, mengembangkan alat kebertelusuran yang lebih canggih, menerapkan teknik pencarian informasi yang mampu mengotomatiskan proses pembuatan dan pemeliharaan jejak, mengembangkan bahasa permintaan jejak baru dan teknik visualisasi yang menggunakan tautan penelusuran, dan menerapkan kebertelusuran di domain tertentu seperti Pengembangan Berbasis Model, sistem lini produk, dan lingkungan proyek yang cerdas [2].

Kebertelusuran perangkat lunak telah lama diakui sebagai kualitas penting dari sistem perangkat lunak yang dirancang dengan baik[3]. Kebertelusuran didefinisikan oleh Center of Excellence untuk Perangkat Lunak dan System Traceability (CoEST) sebagai kemampuan untuk menghubungkan setiap artefak rekayasa perangkat lunak yang dapat diidentifikasi secara unik dengan yang lain, memelihara tautan yang diperlukan dari waktu ke waktu, dan menggunakan jaringan yang dihasilkan untuk menjawab pertanyaan dari kedua produk perangkat lunak dan proses pengembangannya. Meskipun penting, ketertelusuran mungkin merupakan salah satu kualitas yang paling sulit dipahami dari proses pengembangan perangkat lunak. Biaya dan usaha yang diperlukan untuk membuat dan memelihara jejak tautan dalam sistem perangkat lunak yang berkembang pesat bisa sangat tinggi. Selain itu, manfaatnya sering tidak direalisasi dalam praktik, baik karena proses penelusuran yang tidak lancar dan ad-hoc, pelatihan pengguna yang buruk, atau kurangnya perangkat yang efektif[2].

Analisis dampak perubahan (*Change Impact Analysis*/ CIA) merupakan upaya untuk mengidentifikasi konsekuensi potensial dari suatu perubahan, atau bisa juga memperkirakan apa yang perlu dimodifikasi untuk mencapai perubahan ini[4]. Keberhasilan teknik CIA tergantung pada ditemukannya elemen yang terkena dampak yang ada pada semua artefak perangkat lunak terkait. Bahkan, manajemen perubahan menyiratkan kebutuhan akan dukungan penelusuran yang bisa membantu pengembang memahami bagaimana perubahan yang diusulkan berdampak pada artefak dalam proses pengembangan perangkat lunak.

Ketertelusuran didefinisikan sebagai potensi untuk menghubungkan data yang disimpan dalam beberapa jenis artefak, bersama dengan kemampuan untuk memeriksa hubungan ini[5]. Pentingnya hubungan kebertelusuran disorot dalam banyak bidang. Misalnya [6],menyatakan bahwa hubungan kebertelusuran membantu orang memahami sistem perangkat lunak dan mendukung penggunaan kembali, pemeliharaan, evolusi, dan kontrol kualitas. Selain itu, menggunakan tautan kebertelusuran mengarah ke jumlah elemen yang hilang dan salah dalam jumlah yang jauh lebih rendah dalam memahami kegiatan, dan yang paling penting, kualitas CIA yang lebih tinggi untuk evolusi. Dari penelitian[7] mengenai alasan penerapan kebertelusuran dapat diketahui bahwa penggunaan kebertelusuran untuk analisis dampak diperlukan. Hasil penelitian ini menarik, karena mengasumsikan bahwa kebertelusuran sangat membantu ketika terjadi perubahan dan untuk mengevaluasi kembali keputusan yang dibuat di masa lalu.

Perubahan perangkat lunak tidak hanya memengaruhi kode sumber tetapi juga artefak lain seperti desain, artefak uji, dan kebutuhan dasarnya. Untuk tujuan ini, tantangan utama dalam kebertelusuran adalah menciptakan dukungan penelusuran untuk analisis dampak antara artefak di berbagai tingkat abstraksi[2]. Sebagai contoh, adanya ketertelusuran antara kebutuhan dan artefak kode[8], kebutuhan dan artefak uji coba[9], kebutuhan dan artefak desain[10] serta artefak dan desain[11]. Pembahasan CIA antara artefak kebutuhan dan kode sumber sangat menarik karena fase awal dalam siklus hidup perangkat lunak adalah artefak kebutuhan sehingga setiap terjadi perubahan dalam kebutuhan juga harus menyesuaikan kode sumber yang dihasilkan serta untuk keperluan dokumentasi perubahan desain.

Beberapa teknik digunakan untuk mengelola analisis dampak perubahan kebutuhan seperti pengambilan informasi (*Information Rerieval*/IR)[12], graph[13] dan aturan berdasarkan meta-model[14]. Meskipun beberapa hasil yang bermanfaat dapat diidentifikasi oleh penelitian-penelitian ini dalam konteks CIA, sifat ambigu kebutuhan yang diperoleh dari kasus penggunaan dan deskripsi tekstualnya (yang ditulis dalam bahasa alami) dapat menghasilkan kebertelusuran yang salah link dengan artefak yang sesuai, terutama ketika berargumen dengan artefak yang didokumentasikan secara tekstual heterogen. Selain itu, tidak ada pendekatan untuk menangani *use case* dan deskripsi tekstual yang ditulis dalam bahasa yang berbeda. Selain itu, pendekatan yang ada menghitung kesamaan antara berbagai jenis elemen dan tidak satu pun dari pendekatan ini menggunakan teknik pemrosesan alami yang menganalisis berbagai elemen kalimat untuk menghitung kesamaan antara elemen yang memiliki peran yang sama.

Penelitian ini fokus pada pengembangan metode analisis dampak perubahan pada artefak kebutuhan yang didekripsikan dalam *user story* dengan dokumentasi tekstualnya terhadap kode sumber yang diterapkan pada lingkungan pengembangan XP. Metode untuk analisis dampak pada proses pengembangan XP ini menggunakan pendekatan semantik yang disajikan pada penelitian[15] dengan modifikasi penambahan metode

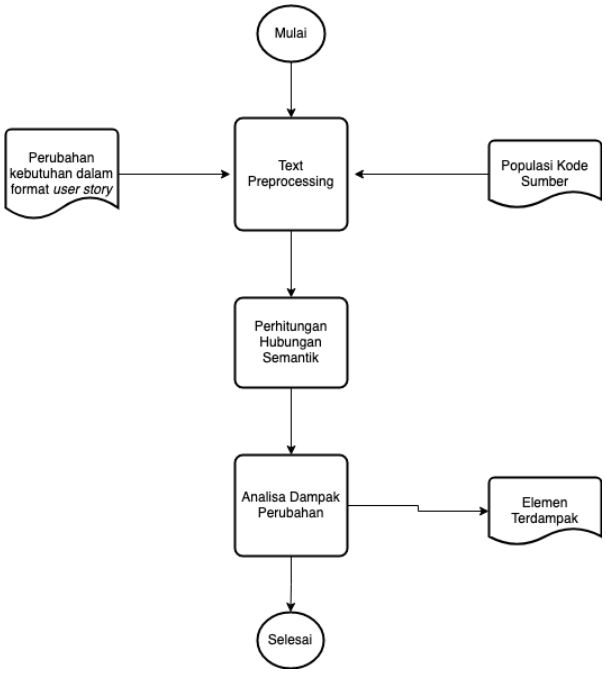
DOI: https://doi.org/10.29207/resti.v5i4.3281

untuk mengetahui dampak perubahan terhadap kode sumber. Model semantik ini memungkinkan untuk menghitung kesamaan antara kata-kata yang memiliki peran yang sama dalam kalimat, misalnya, membandingkan kesamaan antara kata kerja dalam aksi skenario penggunaan menggunakan kata kerja dalam sebuah fungsi atau variabel dalam kode sumber. Hal ini membuat perhitungan kesamaan menjadi fokus. Pemodelan semantik ini dilakukan dengan cara mengambil bagian kalimat yang diekstrak dari deskripsi tekstual kemudian diidentifikasi peran setiap kata dalam kalimat sehingga dalam membuat tautan kebertelusuran dan dibandingkan dengan kata-kata yang memiliki peran yang sama.

2. Metode Penelitian

2.1. Perancangan dan Implementasi Sistem

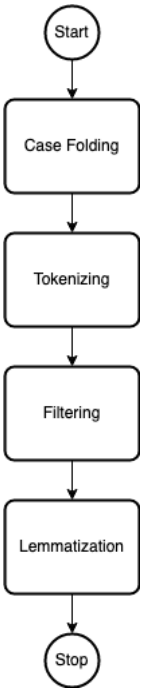
Pada penelitian ini terdapat beberapa proses yaitu yang pertama adalah pembentukan stuktur tekstual *user story* kemudian dilakukan *text preprocessing* dan perhitungan hubungan semantik serta yang terakhir merupakan hasil analisis dampak perubahan. Berikut merupakan penjelasan tiap bagian dalam *flowchart* yang ditunjukkan pada Gambar 1.



Gambar 1. Rancangan proses pendekatan

Dalam penelitian ini, identifikasi kebutuhan disajikan dengan *user story* yang terstruktur dengan baik dalam sebuah template. *User story* ini dibentuk dari spesifikasi kebutuhan yang dijabarkan dalam format kalimat "*As a <type of user>, i want <some goal> so that <some resaon>*".

Praproses teks dilakukan untuk mendapatkan kata-kata yang memiliki makna dari dokumen struktural *user story* dan populasi kode sumber. Praproses untuk dokumen struktural *user story* dilakukan terhadap kalimat-kalimat yang terdapat pada dokumen *user story* sedangkan praproses pada kode sumber dilakukan terhadap nama *package*, *class* serta *method signature* yang terdiri dari nama, parameter dan tipe data yang digunakan oleh sebuah *method*. Diagram alur untuk praproses teks ini diperlihatkan pada Gambar 2.



Gambar 2. Alur praproses teks

Langkah pertama praproses teks ini adalah mengubah semua huruf yang terdapat dalam dokumen menjadi huruf kecil. Langkah ini diperlukan karena tidak semua dokumen teks konsisten menggunakan huruf besar atau kecil semua sehingga keseluruhan teks dalam dokumen perlu dikonversi menjadi bentuk yang standar. Langkah kedua adalah melakukan proses pemisahan sekumpulan karakter dalam suatu teks ke dalam satuan kata (*tokenizing*) sehingga mendapatkan kata yang bermakna. Beberapa *identifier* yang dapat digunakan untuk memisahkan kata antara lain karakter *Whitespaces* seperti spasi serta Garis Bawah (*underscore*). Langkah ketiga praproses adalah melakukan pengambilan kata-kata penting dari hasil *tokenizing* dengan menggunakan algoritma *stopwords*. Kata penghubung seperti "from","that" dan "to" tidak memiliki arti penting dalam kalimat. Untuk menghilangkan *stopwords*, setiap kata dibandingkan dengan data *stopwords* yang tersedia. Dengan mengurangi kata-kata *stopwords* ini, indeks dan waktu pemrosesan dapat lebih cepat dan *noise* juga akan berkurang. Langkah terakhir praproses ini adalah mendapatkan akar kata dari sebuah kata untuk sehingga

DOI: https://doi.org/10.29207/resti.v5i4.3281

dapat diperoleh makna dari kata tersebut (*lemmatization*). *Lemmatization* menggunakan analisis morfologi dan kosakata yang sesuai dengan konteks kalimat yang disebut *lemma*.

Tahap selanjutnya adalah melakukan penghitungan hubungan semantik dari model semantik struktur tekstual *user story* dan populasi kode sumber yang telah diperoleh. Dari sumber kode yang telah diesktrak, akan dianalisis untuk mengetahui kemiripan peran yang sama dengan perubahan yang terdapat dalam dokumen tekstual *user story*. Dalam penelitian ini menggunakan konsep penghitungan kemiripan hubungan semantik TF-IDF.

Tahapan terakhir dalam penelitian ini adalah melakukan analisis dampak perubahan. Dalam tahapan ini dilakukan analisis korelasi dari hasil penghitungan hubungan semantik menggunakan algoritma pada sub-bab 2.4 untuk mengetahui apakah perubahan kebutuhan yang terjadi memiliki korelasi dengan suatu elemen pada kode sumber sehingga dapat diperoleh sebuah daftar rekomendasi elemen kode sumber yang terdampak atas perubahan tersebut. Dampak perubahan dalam penelitian ini menggunakan definisi [4] yang terdiri dari 3 kriteria yaitu: *Starting Impact Set (SIS)* yaitu bagian dari kode sumber yang pada awalnya dianggap dipengaruhi oleh perubahan kebutuhan yang terjadi. *Estimated Impact Set (EIS)* yaitu bagian dari kode sumber yang diperkirakan dipengaruhi oleh perubahan kebutuhan yang terjadi melalui penghitungan analisis dampak perubahan. *Actual Impact Set (AIS)* yaitu bagian dari kode sumber yang benar-benar dimodifikasi dari hasil perubahan kebutuhan.

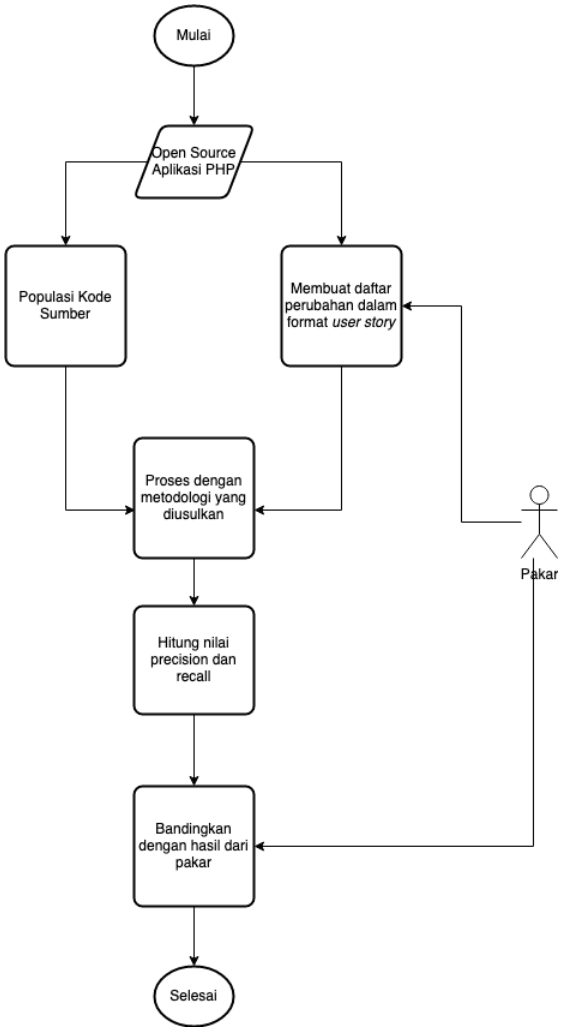
2.2. Pengujian

Pengujian atau eksperimen merupakan suatu proses yang penting untuk menilai dan mengevaluasi pendekatan yang telah dibuat. Dataset dalam penelitian ini terdiri dari kode sumber dalam bahasa pemrograman PHP serta perubahan kebutuhan fungsional dari perangkat lunak tersebut. Kode sumber diambil dari repositori kode sumber di internet dan bersifat terbuka (*open source*). Dari kode sumber tersebut dilakukan pemrosesan populasi kode sumber untuk mengetahui hubungan antar elemen pada kode sumber. Sedangkan dataset perubahan kebutuhan dibuat oleh pakar dengan menganalisa 2 (dua) versi terbaru dari kode sumber untuk membuat daftar perubahan kebutuhan yang terjadi kemudian di deskripsikan dalam sebuah dokumen *user story*. Dokumen *user story* yang digunakan dalam penelitian ini menggunakan bahasa inggris. Untuk pengukuran performa metode yang diusulkan nanti akan dilihat nilai akurasi dalam *precision* dan *recall* serta dibandingkan dengan pendekatan yang dilakukan oleh pakar. Dalam tahap pengujian ini, pakar akan melakukan analisis berdasarkan dataset kode sumber yang telah ditentukan untuk memperoleh data kode sumber yang terdampak oleh perubahan kebutuhan. Nilai precision didapatkan dengan menggunakan rumus 1 sedangkan nilai recall didapatkan dengan menggunakan rumus 2.

$$\text{recall} = \frac{\text{EIS} \cap \text{AIS}}{\text{AIS}} \quad (1)$$

$$\text{precision} = \frac{\text{EIS} \cap \text{AIS}}{\text{EIS}} \quad (2)$$

Ilustrasi proses pengujian pada penelitian ini dapat dilihat pada Gambar 3.



Gambar 3 Ilustrasi pengujian

3. Hasil dan Pembahasan

Ujicoba sistem dilakukan dengan memberikan beberapa data input berupa data *User Story* serta hasil populasi kode sumber. Berdasarkan data uji coba yang telah didapatkan, selanjutnya diberikan pembahasan dan analisis untuk mendapatkan hasil dari penelitian.

3.1. Implementasi Sistem

Untuk mempermudah proses pengujian, dibuat suatu alat bantu untuk melakukan parsing terhadap kode

DOI: https://doi.org/10.29207/resti.v5i4.3281

sumber untuk memperoleh data populasi kode sumber serta melakukan perhitungan kedekatan semantik dan analisis dampak perubahan. Alat bantu tersebut dikembangkan dalam lingkungan pengembangan sebagai berikut:

Sistem Operasi : macOS Big Sur 11.2.3
Prosessor : 1,6 Ghz Dual-Core Intel Core i5
RAM : 8 GB 2133 Mhz LPDDR3
IDE : PhpStorm 2021.1.2
Framework : Laravel 8.12 dengan PHP versi 7.3.28

Untuk melakukan proses populasi kode sumber, penelitian ini menggunakan library PHP-Parser. Library ini menyediakan fasilitas untuk mengurai kode sumber PHP ke dalam format *Abstract Syntax Tree* (AST). AST akan menghasilkan detil dari kode sumber yang direpresentasikan dalam bentuk *tree*. Fitur ini sangat penting karena dapat berguna menemukan deklarasi kelas serta fungsi apa saja yang terdapat dalam kelas tersebut dalam sebuah file PHP. Selain mendapatkan nama kelas dan fungsi, PHP-Parser dapat memperoleh komentar yang terdapat pada sebuah kelas atau fungsi.

Populasi kode sumber diimplementasikan oleh kelas ExtractCode yang merupakan turunan dari kelas NodeVisitorAbtract. Proses untuk melakukan penelusuran AST menggunakan pola *visitor*. Kelas NodeTraverser menjalankan proses penelusuran ini dengan melintasi simpul-simpul yang terdapat pada AST.

Karena dalam penelitian ini yang diperlukan hanya deklarasi kelas dan fungsi serta komentar yang terdapat dalam sebuah fungsi, maka dari hasil penelusuran AST hanya akan diambil bagian-bagian yang diperlukan saja. Untuk mendapatkan deklarasi kelas dilakukan dengan cara melakukan pengecekan jika variabel node yang ditelusuri merupakan *instance object* dari kelas Node/Stmt/Class_. Deklarasi method atau fungsi diperoleh dengan melakukan pengecekan variabel node apakah merupakan instance object dari kelas Node/Stmt/ClassMethod. Sedangkan komentar diperoleh dengan menggunakan fungsi getDocComment().

Implementasi sistem berikutnya dari metode yang diusulkan adalah proses *text preprocessing*. Hasil dari proses populasi kode sumber akan diproses dan data *user story* akan diproses *text preprocessing* untuk mendapatkan kata yang akan digunakan sebagai indeks. Indeks ini yang akan mewakili sebuah dokumen yang nantinya akan digunakan untuk pemodelan pada tahap berikutnya. *Text preprocessing* ini diimplementasikan dalam kelas PreProcessing. Dalam kelas tersebut terdapat fungsi mengubah semua kata dalam dokumen menjadi *lowercase* yaitu CaseFolding(). Setelah itu kumpulan kata akan diubah menjadi vektor jumlah token dengan fungsi Tokenizing() dan dilakukan pencarian kata dasar dengan proses lemmatizer yang diimplementasikan dalam fungsi lemmatizing(). Langkah terakhir dalam proses ini adalah menghilangkan kata yang tidak memiliki makna dengan fungsi StopWords().

Untuk melakukan perhitungan hubungan semantik diimplementasikan dalam kelas Similarity(). Dalam kelas tersebut terdapat beberapa fungsi seperti dotProduct() yang digunakan untuk menghitung perkalian dua titik vektor kemudian untuk menghitung akar kuadrat dari sebuah vektor digunakan fungsi absVector(). Kedua fungsi tersebut digunakan dalam fungsi Similarity() untuk mempoleh nilai hubungan semantik.

Untuk melakukan proses analisis dampak perubahan diimplementasikan dalam kelas SpearmanCorrelation(). Dalam kelas tersebut terdapat fungsi test() yang berfungsi untuk menghasilkan kandidat kelas pada kode sumber yang terdampak perubahan kebutuhan. Input data untuk kelas ini didapatkan dari proses pada sub bab 4.1.3 dengan mengambil nilai kedekatan yang memiliki nilai > 0.

3.2. Rencana Uji Coba

Berdasarkan metode yang diusulkan, penelitian ini memiliki beberapa tahapan proses dalam mendapatkan dampak pada kode sumber jika terjadi perubahan kebutuhan. Proses tersebut sudah dijelaskan pada Gambar 1. Hasil pengujian data *user story* tersebut adalah nilai penghitungan hubungan semantik antara *user story* dengan setiap data populasi kode sumber yang dihasilkan dari proses populasi dataset kode sumber. Kemudian dilakukan proses untuk analisa dampak perubahan berdasarkan data penghitungan hubungan semantik yang memiliki nilai > 0. Terdapat 2 (dua) *project open source* yang digunakan dalam uji coba dalam penelitian ini seperti yang tertera pada tabel 1.

Tabel 1. Aplikasi yang digunakan dalam uji coba

Nama Aplikasi	URL
Monica	https://github.com/monicahq/monica
Akaunting	https://github.com/akaunting/akaunting

3.3. Pengujian Aplikasi Monica

Aplikasi Monica merupakan aplikasi Personal Customer Relationship Manager (CRM) yang ditulis dengan bahasa pemrograman PHP dan bersifat terbuka. Fitur utama aplikasi ini adalah mengelola segala informasi interaksi dengan seorang teman seperti nama anggota keluarga, informasi tempat kerja dan lain sebagainya. Penelitian ini menggunakan aplikasi Monica versi 2.17.0 sebagai dataset perubahan kebutuhan yang dibuat dalam format *user story* oleh pakar. Sedangkan untuk dataset kode sumber, penelitian ini menggunakan kode sumber dari aplikasi Monica versi 2.16.0 yang diperoleh dari *repository*. Data *user story* yang digunakan dalam

DOI: https://doi.org/10.29207/resti.v5i4.3281

pengujian pada aplikasi Monica seperti yang tertera pada tabel 2.

Tabel 2. Dataset *user story* untuk pengujian aplikasi Monica

No	User Story
1.	As a user, I can associate photos with gifts so that I can send my photos as gifts to my friends
2.	As a user, I can enable activity editing so that I can make changes to my activity
3.	As a user, I can change my middle name so that my middle name can be updated
4.	As a user, I can manage my contact list so that my contact list can be updated
5.	As a user, I can add labels to contacts so that I can easily search for contacts

Dari pengujian *user story* yang tertera pada tabel 2 terhadap dataset kode sumber diperoleh nilai dampak perubahan. Sebagai contoh pengujian user story "As a user, I can associate photos with gifts so that I can send my photos as gifts to my friends" terhadap kelas "SendReminder" nilai dampak perubahannya sebesar 0.3000, terhadap kelas "UploadPhoto" dan "ExportAccount" memiliki nilai 0.5571. Nilai kemiripan *user story* tersebut dengan kelas "SendReminder" adalah 0.3967, nilai kemiripan dengan kelas "UploadPhoto" adalah 0.4210 dan nilai kemiripan kelas "ExportAccount" adalah 0.0290. Hasil pengujian *user story* kedua "As a user, I can enable activity editing so that I can make changes to my activity" sebagai contoh memiliki dampak perubahan sebesar 0.5250 terhadap kelas "ActivityStatisticService", "DestroyActivityType", "CreateActivityType" dan "UpdateActityType". Nilai kemiripan user story kedua ini dengan kelas-kelas tersebut adalah 0.4550. Kemudian hasil pengujian terhadap *user story* "As a user, I can change my middle name so that my middle name can be updated" memiliki dampak perubahan sebesar -0.4000 terhadap kelas "GendersController" dan "Contact" sedangkan terhadap kelas "Account" nilai analisa dampak perubahannya sebesar 0.8000. Pengujian terhadap *user story* "As a user, I can add manage my contact list so that my contact list can be updated" memiliki dampak perubahan sebesar 0.5250 terhadap kelas "ImportCSV" dan kelas "Update" sebesar 0.1500. Sedangkan pengujian terhadap *user story* "As a user, I can add labels to contacts so that I can easily search for contacts" memiliki dampak perubahan terhadap kelas "ImportCSV" dan "ActivityStatisticService" sebesar 0.0429.

3.3. Pengujian Aplikasi Akaunting

Aplikasi Akaunting merupakan aplikasi yang dirancang untuk usaha kecil dalam mengelola data keuangan dan bersifat *open source*. Fitur utama aplikasi ini adalah pengelolaan tagihan, pembayaran online serta pelacakan pengeluaran. Dalam penelitian ini, versi aplikasi Akaunting yang digunakan sebagai dataset perubahan kebutuhan oleh pakar adalah versi 2.1.8. Sedangkan dataset kode sumber yang digunakan dari repository adalah versi 2.1.7. Data user story yang digunakan pengujian pada aplikasi Akaunting seperti yang tertera pada Tabel 3.

Tabel 3. Dataset *user story* untuk pengujian aplikasi Akaunting

No	User Story
1.	As a user, I can add "From/To Account Rate" fields in transfer create/edit page so that i can get info "From/To Account Rate"
2.	As a user, I can enable currencies widget so that I can easily find out the currency data

Hasil pengujian user story "As a user, i can add 'From/ToAccount Rate' fields in transfer create/edit page so that I can get info 'From/ToAccount Rate'" memiliki nilai dampak perubahan terhadap kelas "Transfers" dan "Accounts" sebesar 0.4697 kemudian terhadap kelas "Reports" sebesar 0.2273 serta terhadap kelas "Reconciliations" sebesar 0.1061. Sedangkan hasil pengujian terhadap user story "As a user, I can enable currencies widget so that I can easily find out the currency data" memiliki nilai dampak perubahan terhadap kelas "Currency" sebesar 0.8036 serta terhadap kelas "Widget" sebesar 0.2560.

3.4. Analisis Hasil Pengujian

Untuk menganalisis hasil pengujian metode yang diusulkan dalam penelitian ini dilakukan perbandingan hasil pengujian dari metode yang diusulkan dengan hasil yang diperoleh dari pakar. Hasil analisis yang dilakukan oleh pakar akan diukur menggunakan metode Gwet's AC1 untuk memperoleh nilai kesepakatan pakar. Nilai Kesepakatan pakar ini digunakan untuk mengetahui tingkat kesepakatan hasil dari ujicoba melalui metode yang diusulkan dengan hasil pengamatan oleh pakar. Adapun hasil kesepakatan dari kedua pakar untuk pengujian Aplikasi Monica dapat dilihat pada Tabel 4.

Tabel 4. Hasil Gwet's AC1 Aplikasi Monica

	Pakar 1	**Pakar 2**	**Metode**
Pakar 1	-	0.2550	0.0480
Pakar 2	0.2550	-	0.2640
Metode	0.0480	0.2640	-

Tabel 4 menunjukkan hasil kesepakatan terendah terdapat pada kesepakatan pakar 1 dengan metode yang diusulkan, yaitu 0.0480 dimana dalam skala statistik kappa termasuk dalam keeratan kesepakatan "Rendah". Untuk hasil kesepakatan tertinggi terdapat pada kesepakatan pakar 2 dengan metode yang diusulkan, yaitu 0.2640 yang termasuk dalam keeratan kesepakatan "Lumayan".

Tabel 5. Hasil Gwet's AC1 Aplikasi Akaunting

	Pakar 1	**Pakar 2**	**Metode**
Pakar 1	-	0.2510	0.2000
Pakar 2	0.2510	-	0.5235
Metode	0.2000	0.5235	-

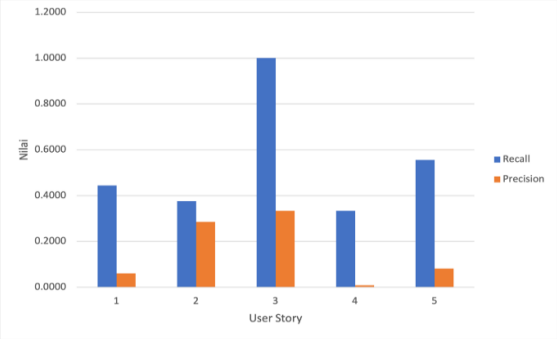
DOI: https://doi.org/10.29207/resti.v5i4.3281

Pada Tabel 5 ditunjukkan hasil kesepakatan dari kedua pakar untuk pengujian Aplikasi Akaunting. Pada tabel tersebut, hasil kesepakatan terendah terdapat pada kesepakatan pakar 1 dengan metode yang diusulkan, yaitu 0.2000 dimana dalam skala statistik kappa termasuk dalam keeratan kesepakatan "Lumayan". Sedangkan hasil kesepakatan tertinggi terdapat pada kesepakatan pakar 2 dengan metode yang diusulkan, yaitu 0.5235 yang termasuk dalam keeratan kesepakatan "Cukup".

Untuk memperoleh nilai precision dan recall, nilai dampak perubahan hasil pengujian yang memiliki nilai > 0 akan menjadi data EIS serta data AIS diambil dari data perubahan *commit* yang terdapat pada repository kode sumber. Hasil dari *precision* dan *recall* pada pengujian aplikasi Monica dapat dilihat pada Tabel 6 dan Gambar 4

Tabel 6. Dataset *user story* untuk pengujian aplikasi Monica

No User Story	*Recall*	*Precision*
1.	0.4444	0.0606
2.	0.3750	0.2857
3.	1.0000	0.3333
4.	0.3333	0.0084
5.	0.5556	0.0820
Rata-rata	**0.5147**	**0.1540**

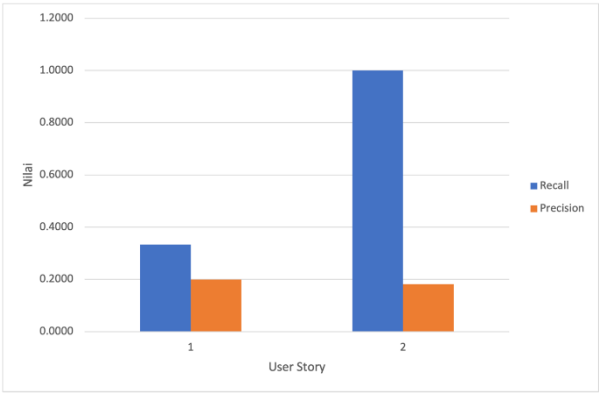


Gambar 4. Diagram Hasil Analisis Dampak Perubahan Aplikasi Monica

Pada Tabel 6 dan Gambar 5 menunjukkan hasil analisis dampak perubahan *user story* ketiga terhadap kode sumber aplikasi Monica memiliki nilai *precision* tertinggi yaitu 0.3333 dengan nilai *recall* 1.0000. Hal ini bisa diakibatkan karena hasil pengujian aplikasi Monika pada user story ketiga menghasilkan 3 data kandidat kelas yang terdampak dan 1 data kelas yang memang terdampak atas perubahan terhadap kode sumber sehingga menghasilkan nilai *precision* tertinggi. Untuk hasil pengukuran *precesion* dan *recall* pengujian pada aplikasi Akaunting dapat dilihat pada Tabel 7 dan Gambar 5

Tabel 7. Dataset *user story* untuk pengujian aplikasi Akaunting

No User Story	*Recall*	*Precision*
1.	0.3333	0.2000
2.	1.0000	0.1819
Rata-rata	**0.6667**	**0.1909**



Gambar 5. Diagram Hasil Analisis Dampak Perubahan Aplikasi Akaunting.

Sedangkan pada Tabel 7 dan Gambar 5 menunjukkan hasil analisis dampak perubahan *user story* terhadap kode sumber pada aplikasi Akaunting memiliki nilai *precision* tertinggi yaitu 0.2000 dengan nilai *recall* 0.3333 pada hasil pengujian *user story* kesatu. Pengujian pada *user story* kesatu menghasilkan 5 kandidat kelas yang terdampak dan 1 kelas pada kode sumber yang memang terdampak perubahan kebutuhan.

Hasil pengujian yang dilakukan oleh pakar untuk aplikasi Monica ditunjukkan pada Tabel 8 serta Tabel 9 untuk hasil pengujian aplikasi Akaunting.

Tabel 8. Hasil Pengukuran Recall dan Precision Aplikasi Monica oleh Pakar

No User Story	*Recall*	*Precision*
1.	0.2776	0.0516
2.	0.3125	0.2860
3.	1.0000	0.1834
4.	0.5000	0.0215
5.	0.3889	0.0778
Rata-rata	**0.4958**	**0.1240**

Tabel 9. Hasil Pengukuran *Recall* dan *Precision* Aplikasi Akaunting oleh Pakar

No User Story	*Recall*	*Precision*
1.	0.3333	0.1548
2.	0.7500	0.0250
Rata-rata	**0.5417**	**0.0899**

Dari Tabel 9 dapat dilihat hasil pengukuran oleh pakar pada aplikasi Monica memiliki rata-rata nilai *recall* 0.4958 serta nilai precision 0.1240. Sedangkan pada pengujian aplikasi Akaunting diperoleh rata-rata nilai precision 0.5417 serta nilai recall 0.0899. Dari hasil perbandingan hasil ujicoba tersebut dapat diketahui metode yang diusulkan pada aplikasi Monica memiliki nilai *recall* sedikit lebih baik daripada ujicoba yang dilakukan oleh pakar yaitu sebesar 0.0458 serta *precision* juga memiliki nilai yang lebih baik sebesar 0.0300. Kemudian pada pengujian aplikasi Akaunting, hasil ujicoba dari metode yang diusulkan menghasilkan nilai recall yang lebih baik sebesar 0.1250 serta nilai *precision* lebih baik sebesar 0.1010.

DOI: https://doi.org/10.29207/resti.v5i4.3281

Metode yang diusulkan dengan menggunakan kesamaan model semantik dalam menganalisis dampak perubahan bisa menghasilkan nilai *recall* dan *precision* yang lebih baik karena metode ini memanfaatkan kesamaan antara nama kelas beserta fungsi dan komentar yang ada didalamnya dengan kalimat deskripsi kebutuhan yang dituliskan dalam format *user story*.

4. Kesimpulan

Analisis dampak perubahan pada lingkungan pengembangan XP dapat dilakukan dengan melakukan pengukuran kedekatan semantik perubahan kebutuhan yang diwujudkan dalam dokumen *user story* terhadap kode sumber. Proses pengukuran kedekatan semantik *user story* terhadap kode sumber dilakukan melalui proses populasi kode sumber kemudian praproses teks yang meliputi *case folding*, *tokenizing*, *filtering* dan *lemmatizing*.

Dari hasil pengujian aplikasi Monica diperoleh rata-rata nilai *precision* 0.1540 dengan rata-rata nilai *recall* 0.5147. Nilai *precision* tertinggi terdapat pada pengujian *user story* ketiga dengan nilai *precision* 0.3333 dan nilai *recall* 1.000. Dari hasil pengujian aplikasi Akaunting diperoleh rata-rata nilai *precision* sebesar 0.1909 dengan rata-rata nilai *recall* 0.6667. Nilai precision tertinggi terdapat pada pengujian user story kesatu yaitu 0.2000 dan nilai recall 0.3333. Hasil pengujian menunjukkan nilai *recall* dan *precision* yang diperoleh dari pengujian metode yang diusulkan lebih baik daripada hasil pengujian yang dilakukan oleh pakar. Dari hasil pengujian tersebut dapat disimpulkan semakin detil deskripsi kebutuhan akan dapat menghasilkan nilai kedekatan semantik yang tinggi sehingga memberikan perkiraan dampak perubahan yang relevan.

User story yang digunakan dalam penelitian ini adalah bahasa inggris. Penelitian selanjutnya dapat mengembangkan untuk *user story* dalam bahasa Indonesia. Adapun identifikasi perubahan pada penelitian ini masih sebatas nama kelas. Penelitian selanjutnya dapat mengembangkan untuk melakukan identifikasi terhadap fungsi dan atribut yang terdapat dalam suatu kelas.

Daftar Rujukan

[1] M. Yasvi, K. Yadav, and Sahendrasingh. S., "Review On Extreme Programming-XP," *Int. Conf. Robot. Smart Technol. Electron. Eng. Delhi*, no. April, pp. 1–8, 2019, [Online]. Available: https://www.researchgate.net/publication/332465869.

[2] J. Cleland-Huang, O. C. Z. Gotel, J. H. Hayes, P. Mäder, and A. Zisman, "Software traceability: Trends and future directions," *Futur. Softw. Eng. FOSE 2014 - Proc.*, pp. 55–69, 2014, doi: 10.1145/2593882.2593891.

[3] B. Ramesh and M. Jarke, "Toward Reference Models for Requirements Traceability," *Softw. Eng. IEEE Trans.*, vol. 27, pp. 58–93, Feb. 2001, doi: 10.1109/32.895989.

[4] R. S. Arnold, *Software Change Impact Analysis*. Washington, DC, USA: IEEE Computer Society Press, 1996.

[5] O. Gotel *et al.*, "The quest for Ubiquity: A roadmap for software and systems traceability research," in *2012 20th IEEE International Requirements Engineering Conference (RE)*, 2012, pp. 71–80, doi: 10.1109/RE.2012.6345841.

[6] M. A. Javed and U. Zdun, "A systematic literature review of traceability approaches between software architecture and source code," *ACM Int. Conf. Proceeding Ser.*, 2014, doi: 10.1145/2601248.2601278.

[7] E. Bouillon, P. Mäder, and I. Philippow, "A Survey on Usage Scenarios for Requirements Traceability in Practice BT - Requirements Engineering: Foundation for Software Quality," 2013, pp. 158–173, doi: 10.1007/978-3-642-37422-7_12

[8] M. Grechanik, K. S. McKinley, and D. E. Perry, "Recovering and using use-case-diagram-to-source-code traceability links," *6th Jt. Meet. Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng. ESEC/FSE 2007*, pp. 95–104, 2007, doi: 10.1145/1287624.1287640.

[9] M. Shahid and S. Ibrahim, "Change impact analysis with a software traceability approach to support software maintenance," in *2016 13th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, 2016, pp. 391–396, doi: 10.1109/IBCAST.2016.7429908.

[10] A. von Knethen and M. Grund, "QuaTrace: a tool environment for (semi-) automatic impact analysis based on traces," in *International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings.*, 2003, pp. 246–255, doi: 10.1109/ICSM.2003.1235427.

[11] M. Hammad, M. L. Collard, and J. I. Maletic, "Automatically identifying changes that impact code-to-design traceability during evolution," *Softw. Qual. J.*, vol. 19, no. 1, pp. 35–64, 2011, doi: 10.1007/s11219-010-9103-x.

[12] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora, "Recovering traceability links in software artifact management systems using information retrieval methods," *ACM Trans. Softw. Eng. Methodol.*, vol. 16, no. 4, 2007, doi: 10.1145/1276933.1276934.

[13] H. Schwarz, J. Ebert, and A. Winter, "Graph-based traceability: A comprehensive approach," *Softw. Syst. Model.*, vol. 9, no. 4, pp. 473–492, 2010, doi: 10.1007/s10270-009-0141-4.

[14] A. Keller and S. Demeyer, "Change impact analysis for UML model maintenance," *Emerg. Technol. Evol. Maint. Softw. Model.*, pp. 32–56, Jan. 2011, doi: 10.4018/978-1-61350-438-3.ch002.

[15] D. Kchaou, N. Bouassida, M. Mefteh, and H. Ben-Abdallah, "Recovering semantic traceability between requirements and design for change impact analysis," *Innov. Syst. Softw. Eng.*, vol. 15, no. 2, pp. 101–115, 2019, doi: 10.1007/s11334-019-00330-w.

DOI: https://doi.org/10.29207/resti.v5i4.3281