Pembuatan Kakas Bantu untuk Mendeteksi Ketidaksesuaian Diagram Urutan (*Sequence Diagram*) dengan Diagram Kasus Penggunaan (*Use Case Diagram*)

Andrias Meisyal Yuwantoko, Daniel Oranova Siahaan, Adhatus Solichah Ahmadiyah
Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111 Indonesia
e-mail: daniel@if.its.ac.id

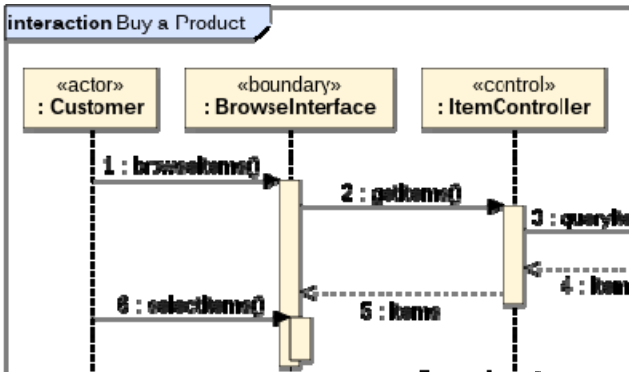
***Abstrak*— Sebuah diagram urutan dibuat berdasarkan alur yang ada pada deskripsi kasus penggunaan. Alur tersebut dire- presentasikan dalam bentuk interaksi antara aktor dan sistem. Pemeriksaan rancangan diagram urutan perlu dilakukan untuk mengetahui ketidaksesuaian urutan alur kasus penggunaan dengan urutan pesan yang dikirimkan oleh objek-objek pada diagram urutan. Rancangan diagram yang sesuai merupakan kunci ketepatan (*correctness*) implementasi perangkat lunak. Namun, pemeriksaan ketidaksesuaian masih dilakukan secara manual. Hal ini menjadi masalah apabila sebuah proyek perangkat lunak memiliki banyak rancangan diagram dan sumber daya manusia tidak mencukupi. Pemeriksaan membutuhkan waktu yang lama dan memiliki dampak pada waktu pengembangan perangkat lunak. Penelitian ini mengusulkan pembuatan kakas bantu untuk mendeteksi ketidaksesuaian diagram urutan dengan diagram kasus penggunaan. Ketidaksesuaian dilihat dari kemiripan semantik kalimat antara alur pada deskripsi kasus penggunaan dan *triplet*. Dari hasil pembuatan kakas bantu, kakas bantu yang dibuat dapat mendeteksi ketidaksesuaian diagram urutan dengan diagram kasus penggunaan. Kakas bantu ini diharapkan tidak hanya membantu pemeriksaan rancangan diagram akan tetapi mempercepat waktu pengembangan perangkat lunak.**

***Kata Kunci*—diagram urutan, diagram kasus penggunaan, ketidaksesuaian, pemrosesan bahasa alami, kemiripan semantik kalimat.**

I. PENDAHULUAN

SEORANG analis sistem memiliki peranan penting untuk menganalisis dan mengembangkan kebutuhanpengguna ke dalam sebuah dokumen spesifikasi kebutuhan. Dokumen spesifikasi kebutuhan yang dihasilkan menjadi panduan pe- ngembang untuk memahami dan menyelesaikan proyek per- angkat lunak dengan baik. Dalam pengembangan perangkat lunak berbasis objek, analisis dan perancangan sistem meng- acu kepada bahasa *unified modeling language* (UML) yang membantu kebutuhan perangkat lunak saling berkomunikasi [1]. Analis sistem menerjemahkan kebutuhan tersebut dengan merancang diagram UML. Salah satu diagram yang diran- cang adalah diagram urutan (*sequence diagram*). Diagram urutan dibuat berdasarkan alur yang ada pada deskripsi kasus penggunaan. Analis sistem menurunkan alur tersebut menjadi kelas-kelas (*class*). Kelas-kelas ini nantinya berinteraksi di dalam diagram urutan melalui serangkaian pesan (*message*). Analis sistem perlu memeriksa rancangan diagram urutan apabila ditemukan ketidaksesuaian. Sebagai contoh, ada kasus penggunaan "Buy a Product"[1] dengan potongan alur utama kasus penggunaan sebagai berikut:

1) Customer browses through catalog.
2) Customer selects items to buy.
3) ...



Gambar 1. Potongan diagram urutan kasus penggunaan "Buy a Product"

Diagram urutan yang sesuai dengan alur kasus penggunaan di atas ditunjukkan oleh Gambar 1. Alur nomor 1 sesuai dengan *triplet* {Customer, browseItems(), BrowseInterface}, sedangkan alur nomor 2 dengan *triplet* {Customer, select- Items(), BrowseInterface}.

Ketidaksesuaian didefinisikan apakah urutan alur yang ada pada deskripsi kasus penggunaan sudah sesuai dengan urut- an pesan yang dikirimkan oleh aktor ke kelas *boundary* atau sistem ke kelas *boundary* pada diagram urutan. Rancangan diagram yang sesuai berpengaruh kepada ketepatan (*correctness*) implementasi perangkat lunak. Namun, peme- riksaan ketidaksesuaian masih dilakukan secara manual. Hal

ini menjadi masalah apabila sebuah proyek perangkat lunak memiliki banyak rancangan diagram, tetapi sumber daya analis sistem tidak mencukupi. Analis sistem membutuhkan waktu yang lama untuk memeriksa rancangan diagram tersebut. Oleh sebab itu, penelitian ini mengusulkan kakas bantu yang dapat mendeteksi ketidaksesuaian diagram urutan dengan diagram kasus penggunaan.

Kakas bantu yang dibuat menerapkan pemrosesan bahasa alami (*natural language processing*). Pemrosesan bahasa alami yang digunakan adalah perhitungan kemiripan semantik kalimat. Pendeteksian ketidaksesuaian dilakukan dengan menghitung kemiripan semantik antara kalimat alur pada des- kripsi kasus penggunaan dengan triplet. Perhitungan kemirip- an semantik kalimat digunakan dalam pendeteksian karena adanya perbedaan bahasa pada kalimat alur pada deskripsi kasus penggunaan (menggunakan bahasa pengguna) dan *trip- let* (menggunakan bahasa pengembang). Perbedaan ini terjadi saat analis sistem menurunkan kalimat alur pada deskripsi kasus penggunaan ke dalam kelas-kelas selama tahap analisis. Contoh, kata "catalog" pada alur suatu kasus penggunaan men- jadi "BrowseInterface" pada *triplet*. Nilai kemiripan kalimat yang melewati ambang batas (*threshold*) yang telah ditentukan menjadi dasar penilaian ketidaksesuaian. Sehingga, pembuatan kakas bantu ini diharapkan tidak hanya membantu analis sistem menghasilkan rancangan diagram yang sesuai akan tetapi mempercepat waktu pengembangan perangkat lunak.

II. METODE DETEKSI KETIDAKSESUAIAN

Metode deteksi ketidaksesuaian yang diimplementasikan pada kakas bantu ditunjukkan oleh Gambar 2. Secara umum, ada tiga tahapan utama alur deteksi ketidaksesuaian, yaitu mengekstraksi elemen diagram dari rancangan, mendeteksi ketidaksesuaian, dan menampilkan hasil deteksi ketidaksesuaian. Penjelasan lebih detail dari masing-masing tahap dijelaskan sebagai berikut:

A. Tahap Memodelkan Dokumen Rancangan

Pemodelan dokumen rancangan dilakukan melalui aplikasi *StarUML* 2 [2]. Dokumen rancangan berisi diagram kasus penggunaan, deskripsi kasus penggunaan, dan diagram urutan dari satu sistem. Adapun aturan pembuatan elemen-elemen diagram tersebut. Aturan ini dijelaskan sebagai berikut:

1) Diagram Kasus Penggunaan

Diagram kasus penggunaan yang dibuat mengandung elemen aktor, kasus penggunaan, relasi, dan batas sistem.

2) Deskripsi Kasus Penggunaan

Deskripsi kasus penggunaan berisi alur utama atau dasar (*basic flow*) dari sebuah kasus penggunaan. Alur yang biasanya berupa urutan bernomor diubah ke dalam bentuk paragraf. Kemudian, alur menggunakan kalimat aktif dan sederhana. Kalimat sederhana adalah kalimat yang terdiri satu pelaku dan satu tindakan. Misal, kalimat *Customer browses through catalog.* terdiri satu pelaku "customer" dan satu tindakan "browse". Kata ganti pada alur deskripsi kasus penggunaan harus diganti dengan objek sebenarnya. Contoh, kalimat *The ATM returns the card to him.* menjadi *The ATM returns the card to customer*. Deskripsi ini dima- sukkan pada bagian *documentation* pada aplikasi *StarUML* 2.

3) Diagram Urutan

Diagram urutan yang dibuat mengandung elemen judul diagram yang sesuai dengan nama salah satu kasus penggunaan, *lifeline* yang berisi nama kelas dengan format *upper camel case*, *message* berupa *method* dengan format *lowercamel case*, dan *stereotype lifeline* yang terdiri dari empat jenis (*actor*, *boundary*, *control*, dan *entity*).

Ketiga elemen tersebut harus diletakkan pada satu model pada hierarki proyek aplikasi *StarUML* 2.

B. Tahap Mengekstraksi Dokumen Rancangan

Ekstraksi dokumen rancangan mengambil elemen diagram pada dokumen rancangan. Elemen diagram yang diambil antara lain:

1) Diagram kasus penggunaan

Elemen yang diambil pada diagram kasus penggunaan adalah nama kasus penggunaan.

2) Deskripsi kasus penggunaan

Elemen yang diambil adalah deskripsi kasus penggunaan yang berisi alur dasar kasus penggunaan.

3) Diagram urutan

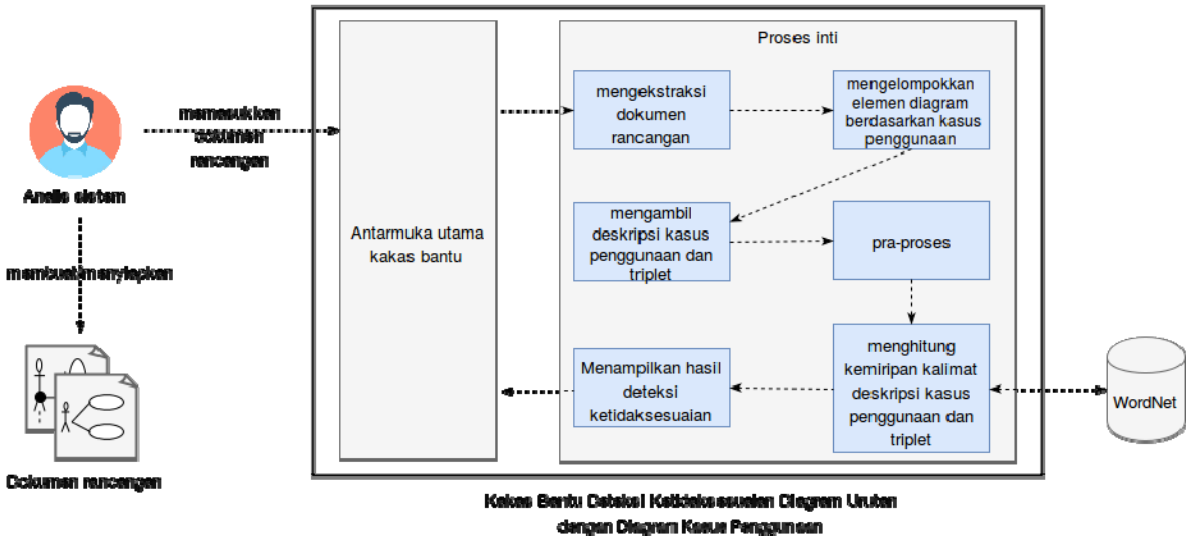
Elemen yang diambil pada diagram urutan, antara lain judul diagram urutan, nama kelas-kelas yang ada pada diagram urutan, *stereotype* kelas-kelas tersebut, nama *method* baik yang dikirim atau diterima oleh sebuah kelas, dan tipe *method* tersebut.

Elemen diagram yang telah diekstraksi kemudian disimpan untuk dikelompokkan diagram berdasarkan nama kasus penggunaan. Dari deskripsi kasus penggunaan yang telah didapatkan, deskripsi tersebut dikelompokkan berdasarkan kasus penggunaannya. Kelas-kelas yang berinteraksi melalui *method* pada diagram urutan diubah ke bentuk *triplet*. *Triplet* terdiri dari nama kelas pengirim (*sender class*), nama *method* yang dikirim, dan nama kelas penerima (*receiver class*). Saat membentuk *triplet*, hanya kelas dan *method* yang berasal dari aktor ke kelas *boundary* atau sistem ke kelas *boundary*. *Triplet-triplet* yang dihasilkan selanjutnya dikelompokkan berdasarkan judul diagram urutannya karena judul diagram urutan merealisasikan kasus penggunaan yang bersangkutan.

C. Tahap Mendeteksi Ketidaksesuaian

Deteksi ketidaksesuaian dilakukan per kasus penggunaan. Berdasarkan kasus penggunaan, deskripsi dan triplet-nya diambil. Triplet terlebih dahulu diubah menjadi kalimat dengan memisahkan penyusunnya dengan spasi. Kemudian, menggabungkan susunannya dengan lower-casing penyusun elemen triplet (kecuali singkatan atau akronim) dan menambah artikel "The" di awal kalimat. Kata kerja pada nama method bertipe synchronous ditambah akhiran "s" atau "es" dan artikel "the" setelah nama kata kerja tersebut. Pengubahan tidak dilakukan untuk method bertipe reply atau create. Sebagai contoh, triplet {Customer, browseItems, BrowseInterface} menjadi

kalimat The customer browses the items browse interface. Kalimat harus diawali dengan huruf kapital dan diakhiri tanda titik.



Gambar 2. Alur deteksi ketidaksesuaian pada kakas bant

Penambahan “s” atau “es” pada kata kerja nama method dilakukan karena tindakan dilakukan oleh orang ketiga tunggal. Baik aktor maupun sistem masuk ke dalam kategori tersebut. Selanjutnya, setiap kalimat pada alur deskripsi kasus penggunaan dihitung kemiripannya dengan triplet yang telah diubah menjadi kalimat. Terdapat alur saat menghitung kemiripan pasangan tersebut. Alur ini dijelaskan sebagai berikut:.

1) Jika nilai kemiripan kalimat memenuhi ambang batas yang telah ditentukan, pasangan diberikan label “Sesuai” dan perhitungan dilanjutkan oleh pasangan kalimat deskripsi kasus penggunaan dan triplet berikutnya.
2) Jika nilai kemiripan kalimat tidak memenuhi ambang batas yang telah ditentukan, menghitung nilai kemiripan kalimat deskripsi kasus penggunaan yang belum memenuhi tersebut dengan triplet berikutnya.
3) Jika hingga triplet terakhir nilai kemiripan kalimat tidak ada yang memenuhi, pasangan diberikan label “Tidak sesuai” dan perhitungan dilanjutkan oleh kalimat deskripsi kasus penggunaan berikutnya dan triplet yang belum memenuhi setelah triplet terakhir yang memenuhi.

A. Tahap Menghitung Kemiripan Semantik Kalimat

Penelitian ini menggunakan perhitungan kemiripan kalimat yang diajukan oleh Lee [3]. Tahap perhitungan kemiripan kalimat diawali dengan pra-proses kalimat. Pra-proses kalimat meliputi *tokenization*, *lower-casing*, *lemmatization*, dan *stop-word removing*. Pasangan kalimat dilakukan *tokenization* terlebih dahulu. *Tokenization* adalah proses memecah teks atau kalimat menjadi kata-kata yang disebut dengan *token*. *Token* yang didapatkan kemudian diubah ke dalam huruf kecil melalui *lower-casing*. Untuk mendapatkan bentuk dasar dari kata, *lemmatization* dilakukan. *Lemmatization* lebih dipilih dibandingkan dengan *stemming* karena *lemmatization* menghasilkan bentuk dasar suatu kata berdasarkan konteks kalimat (hubungan kata satu dengan kata lain dalam satu kalimat), sedangkan *stemming* tidak memperhatikan ini, hanya menghilangkan akhiran tertentu pada suatu kata. Contoh, kata “better” memiliki bentuk dasar “good” sebagai lemma.

Kemudian, melakukan *part-of-speech* (POS) *tagging* untuk mendapatkan kata benda dan kata kerja pada kalimat. *Stop-word removing* dilakukan dengan mengecek apakah suatu kata merupakan kata benda atau kata kerja pada *WordNet* [4]. Jika kata tidak ditemukan maka kata tidak dimasukkan. Setelah itu, perhitungan nilai akhir kemiripan kalimat dari hasil perhitungan nilai *cosine similarity* vektor kata benda dan vektor kata kerja. Vektor ini dihasilkan dari perhitungan kemiripan kata benda dan kata kerja. Perhitungan kemiripan kata dibantu dengan *WordNet* untuk mengetahui kedekatan makna (sinonim). Metrik yang digunakan dalam perhitungan kemiripan antar kata adalah metrik yang diusulkan oleh Lin [5].

B. Tahap Menampilkan Hasil Deteksi Ketidaksesuaian

Dari hasil perhitungan kemiripan semantik kalimat, pasangan kalimat pada alur deskripsi kasus penggunaan dan *triplet* yang memiliki label “Sesuai” dan “Tidak sesuai” disimpan. Data tersebut kemudian diambil dan ditampilkan pada antarmuka utama kakas bantu sebagai hasil deteksi ketidaksesuaian. Data yang ditampilkan berupa yang terdiri dari nama kasus penggunaan, daftar kalimat alur pada deskripsi kasus penggunaan, dan daftar *triplet*. Penanda kesesuaian dibedakan dengan memberikan warna pada kalimat alur pada deskripsi kasus penggunaan dan *triplet*. Warna hijau diberikan untuk penanda dengan label “Sesuai”, sedangkan warna merah untuk penanda dengan label “Tidak sesuai”.

III. KESIMPULAN/RINGKASAN

Penelitian ini mengusulkan kakas bantu untuk mendeteksi ketidaksesuaian diagram urutan dengan diagram kasus penggunaan. Kakas yang dibuat dapat mendeteksi ketidaksesuaian kedua diagram tersebut. Akan tetapi, kakas bantu memiliki kekurangan. Pembentukan *triplet* menjadi kalimat perlu dieksplorasi kembali, khususnya *triplet* yang mengandung *method* bertipe *reply*. Selain itu, kakas bantu belum dapat mendeteksi alur alternatif. Pada penelitian selanjutnya diharapkan kakas bantu dapat menangani kasus ini.

DAFTAR PUSTAKA

[1] D. Siahaan, “Analisa kebutuhan dalam rekayasa perangkat lunak,” Yogya- karta: Andi, 2012.
[2] MKLab, “Staruml 2,” http://staruml.io/, diakses terakhir 10 November 2015.
[3] M. C. Lee, “A novel sentence similarity measure for semantic-based expert systems,” Expert Systems with Applications, vol. 38, no. 5, pp. 6392–6399, 2011.
[4] G. Miller and C. Fellbaum, “Wordnet: An electronic lexical database,” 1998.
[5] D. Lin, “An information-theoretic definition of similarity.” in ICML, vol. 98. Citeseer, 1998, pp. 296–304