Alat Pengukur Keatomikan Kebutuhan Perangkat Lunak Berbasis Kemajemukan Kalimat

Compound Sentence-Based Software Requirement Atomicity Measuring Tool

Jati H. Husen*[1], Rosa Reska Riskiana[2]
[1]Program Studi Rekayasa Perangkat Lunak, Fakultas Informatika, Telkom University, Bandung
[2] Program Studi Informatika, Fakultas Informatika, Telkom University, Bandung
e-mail: *[1]jatihusen@telkomuniversity.ac.id, [2]rosareskaa@telkomuniversity.ac.id

Abstrak

Keatomikan kebutuhan perangkat lunak adalah bagian penting dalam pengembangan perangkat lunak. Kebutuhan yang bersifat tidak atomik dapat mengakibatkan masalah serius dalam proses pengembangan perangkat lunak. Kami membangun metode perhitungan keatomikan kebutuhan perangkat lunak berdasarkan kemajemukan kalimat spesifikasi kebutuhan perangkat lunak. Untuk menguji metode tersebut kami membuat alat pengukur keatomikan yang hasilnya dibandingkan dengan hasil perhitungan ahli. Berdasarkan pengujian didapatkan sistem yang dibangun memiliki akurasi sebesar 75%. Masalah-masalah seperti pendeteksian konteks dan ketergantungan terhadap kebenaran struktur kalimat yang harus diperbaiki agar metode yang diajukan dapat digunakan dengan baik.

Kata kunci: Rekayasa kebutuhan perangkat lunak, Keatomikan kebutuhan perangkat lunak, Kualitas kebutuhan perangkat lunak

Abstract

Requirement atomicity is an important aspect of software development. Nonatomic requirement may lead to serious problems during software development process. We developed an atomicity calculation method based on compound sentences of software requirement specification. In order to evaluate it, we implemented the method into a requirement atomicity measurement tool that the result is compared with expert judgement. Based on the test, our tool resulted 75% accuracy. On top of that, we also defined the problems that needs to be fixed such as context detection and correctness of sentence structure before our tool can be utilized properly in real cases.

Keywords: requirement engineering, requirement atomicity, requirement quality

1. PENDAHULUAN

Kebutuhan adalah faktor penting dalam pengembangan perangkat lunak [1]. Menggambarkan apa yang harus diimplementasikan dalam sebuah perangkat lunak, kebutuhan menjadi dasar untuk hampir seluruh kegiatan yang yang ada di dalam pengembangan perangkat lunak. Selain itu kebutuhan juga menjadi dasar untuk melakukan estimasi biaya, waktu, dan tenaga yang dibutuhkan untuk menjalankan proyek pengembangan perangkat lunak [2]. Hal ini menjadikan kualitas kebutuhan sebagai salah satu hal penting yang harus diperhatikan untuk mengembangkan perangkat lunak dengan sukses.

Kualitas kebutuhan dijabarkan menjadi beberapa kriteria, dimana salah satunya adalah keatomikan kebutuhan perangkat lunak [3]. Kriteria kualitas tersebut menyatakan bahwa satu kebutuhan perangkat lunak hanya boleh mengandung satu hal saja. Kegagalan dalam

memastikan keatomikan kebutuhan dapat menyebabkan pengaturan keterlacakan antara kebutuhan dengan kode program maupun artefak-artefak lainnya menjadi sulit. Jika dibiarkan, hal tersebut akan berujung kepada analisis dampak yang tidak akurat [4] dan menyebabkan dokumen spesifikasi kebutuhan perangkat lunak yang dibuat sulit dimodifikasi [5].

Pengukuran tingkat keatomikan sebuah kebutuhan perangkat lunak digolongkan kedalam kategori yang disebut dengan *requirements decomposition metrics* (metrik dekomposisi kebutuhan) yang ada dalam *requirements completeness metrics* (metrik kekomplitan kebutuhan) [6]. Metrik dekomposisi kebutuhan menggambarkan apakah sebuah kebutuhan perangkat lunak sudah dikomposisi ke tingkat yang cukup [7]. Kebutuhan perangkat lunak dengan metrik dekomposisi yang buruk mencangkup dua atau lebih fungsionalitas didalamnya [8].

Salah satu cara untuk memastikan keatomikan kebutuhan perangkat lunak adalah menggunakan struktur kalimat spesifikasi kebutuhan. Sebuah kebutuhan perangkat lunak yang atomik tidak memiliki konjugasi, baik dalam bentuk kata hubung maupun tanda baca dalam spesifikasinya [9]. Dengan kata lain, spesifikasi kebutuhan perangkat lunak yang menggunakan kalimat majemuk memiliki keatomikan yang buruk.

Berdasarkan metode tersebut, kami membuat sebuah metode pengukuran keatomikan perangkat lunak berdasarkan dekomposisi kemajemukan kalimat spesifikasi kebutuhan perangkat lunak. Metode pengukuran kami menggunakan jumlah kalimat tunggal yang dapat dihasilkan dari kalimat spesifikasi kebutuhan perangkat lunak sebagai dasar pengukuran. Semakin banyak kalimat tunggal yang dapat dibuat maka semakin buruk keatomikan kebutuhan perangkat lunak yang diukur.

Kami juga mengaplikasikan metode tersebut menjadi sebuah alat yang dapat digunakan. Sistem yang dibuat mengolah spesifikasi kebutuhan perangkat lunak berbasis bahasa natural dalam Bahasa Indonesia dan berdasarkan tanda baca dan kata penghubung yang ada didalamnya. Metrik yang dihasilkan berupa seberapa banyak kebutuhan perangkat lunak atomik yang dapat dibuat dengan memecahkan kebutuhan perangkat lunak tersebut.

Untuk menguji penggunaan alat tersebut kami mendefinisikan pertanyaan-pertanyaan penelitian (PP) sebagai berikut:

- PP1. Berapa tingkat akurasi alat yang dibangun dalam mengukur jumlah kebutuhan atomic yang dapat dihasilkan?
- PP2. Apa kekurangan dari alat yang telah dibuat?

Penelitian ini memiliki kontribusi terhadap badan ilmu rekayasa perangkat lunak terutama dalam area rekayasa kebutuhan dalam bentuk sebuah alat pendeteksian keatomikan kebutuhan perangkat lunak. Selain itu terdapat juga kontribusi berupa pengetahuan mengenai pengaruh struktur kalimat terhadap keatomikan kebutuhan perangkat lunak.

Kelanjutan dari tulisan ini disusun dengan sistematika sebagai berikut: Bab 2 menjelaskan rancangan sistem dan metode perhitungan serta metode pengujian terhadap sistem yang dibangun. Bab 3 membahas hasil pengujian yang sudah dilakukan beserta pembahasan. Bab 4 menyimpulkan tulisan ini dengan jawaban dari kedua pertanyaan penelitian yang sudah ditentukan. Bab 5 menutup tulisan ini dengan saran untuk penelitian selanjutnya.

2. METODE PENELITIAN

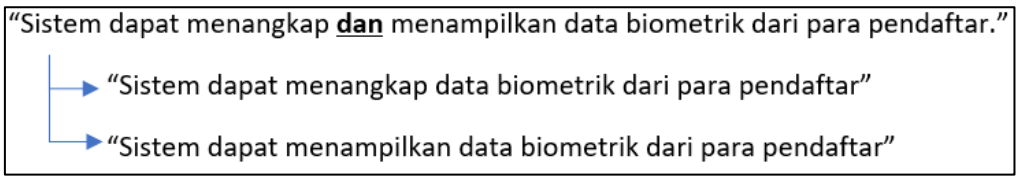
2.1. Metode Perhitungan Metrik

Metode perhitungan yang dibuat berfokus kepada tanda baca dan kata penghubung yang terdapat dalam spesifikasi kebutuhan perangkat lunak. Untuk menentukan tanda baca dan kata penghubung yang menghasilkan kata majemuk kami mengacu kepada kaidah tata tulis dan tata bahasa Bahasa Indonesia [10]. Berdasarkan hasil studi literatur yang dilakukan, ditemukan dua tanda baca dan dua kata penghubung yang berpengaruh kedalam keatomikan kebutuhan

perangkat lunak. Selain itu terdapat juga satu tanda baca khusus yang membutuhkan perhatian khusus.

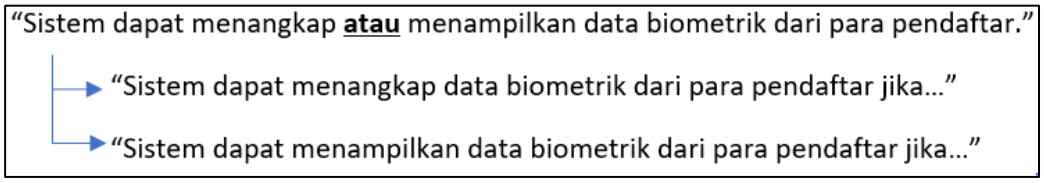
2.1.1. Kata Penghubung

Kata penghubung pertama yang ditemukan adalah kata ‘dan’. Kata penghubung ini menggambarkan penambahan dalam sebuah kalimat majemuk. Kebutuhan perangkat lunak yang menggunakan dan memiliki dua hal yang harus dipenuhi. Sebuah kebutuhan perangkat lunak yang menggunakan kata ‘dan’ dapat dipisahkan menjadi dua kebutuhan yang lebih atomik. Contoh dari hal ini tergambar di gambar 1. Selain itu, ditemukan juga bahwa kata penghubung ‘serta’ juga memiliki sifat yang sama dengan kata penghubung ‘dan’.



Gambar 1 Pemisahan kebutuhan dengan kata penghubung ‘dan’

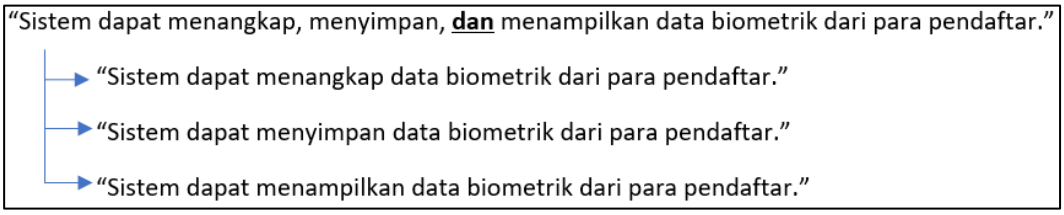
Kata penghubung lainnya yang mempengaruhi keatomikan kebutuhan perangkat lunak adalah kata ‘atau’. Kata penghubung ini berkebalikan dengan kata penghubung ‘dan’ dimana kebutuhan perangkat lunak yang mengandung kata penghubung ‘atau’ harus memenuhi salah satu dari dua hal yang ada didalamnya. Kebutuhan yang memiliki kata penghubung ‘atau’ dapat dipisahkan menggunakan cara yang sama dengan kebutuhan yang memiliki kata ‘dan’ namun kondisi dimana masing-masing hal akan dipenuhi harus ditambahkan. Gambar 2 menggambarkan pemisahan kebutuhan dengan kata atau.



Gambar 2 Pemisahan kebutuhan dengan kata penghubung ‘atau’

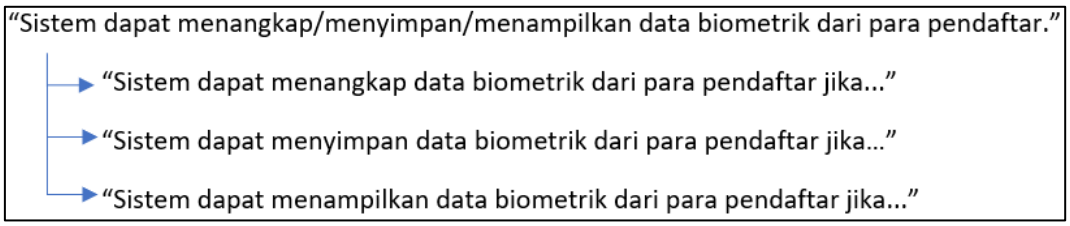
2.1.2. Tanda Baca

Tanda baca yang kami temukan mempengaruhi keatomikan kebutuhan perangkat lunak adalah tanda baca koma (,) dan garis miring (/). Tanda baca koma digunakan untuk menjabarkan hal-hal yang setingkat dalam sebuah kalimat majemuk. Sifat pemisahan dari tanda koma sendiri ditentukan oleh kata penghubung yang menyertai penjabarannya. Kebutuhan yang memiliki tanda baca koma yang disertai kata penghubung ‘dan’ menggambarkan tiga atau lebih hal dimana semua hal tersebut harus dipenuhi. Sebaliknya jika disertai kata penghubung ‘atau’ maka salah satu hal harus dipenuhi. Pemecahan dengan kebutuhan dengan tanda baca koma dapat dilihat di gambar 3.



Gambar 3 Pemisahan kebutuhan dengan tanda baca koma

Tanda baca garis miring digunakan sebagai pengganti kata hubung ‘dan’ maupun ‘atau’ dalam sebuah kalimat. Garis miring dapat digunakan lebih dari sekali untuk menjabarkan lebih dari dua kondisi. Pemisahan kebutuhan perangkat lunak dengan garis miring memerlukan informasi mengenai kata hubung apa yang digantikan oleh garis miring. Jika hal tersebut sudah diketahui maka pemisahan dapat mengikuti aturan kata hubung yang terkait. Contoh pemisahan kebutuhan perangkat lunak dengan garis miring dalam konteks ‘atau’ dapat dilihat di gambar 4.



Gambar 4 Pemisahan kebutuhan dengan tanda baca garis miring

Tanda baca lain yang tidak secara langsung mempengaruhi kemajemukan kalimat namun perlu diperhatikan dalam kebutuhan perangkat lunak adalah tanda baca titik (.). Kebutuhan yang memiliki tanda baca titik dapat mengindikasikan bahwa kebutuhan tersebut sebenarnya terdiri dari dua atau lebih kebutuhan. Namun perlu diperhatikan juga penggunaan titik untuk hal lainnya seperti pemberian nomor versi (contoh: .NET 3.5).

2.1.3. Rumus Perhitungan

Berdasarkan kata penghubung dan tanda baca yang sudah dijabarkan sebelumnya kami menggunakan rumus berikut sebagai berikut:

$$Atomicity = 2^a * 2^b * f(c) * f(d) \quad (1)$$

Dimana a adalah jumlah kata penghubung ‘dan’ juga sinonimnya dalam suatu kebutuhan perangkat lunak. b adalah jumlah kata penghubung ‘atau’ dan juga sinonimnya dalam suatu kebutuhan perangkat lunak.

$f(c)$ adalah fungsi perhitungan dekomposisi kebutuhan dengan tanda baca koma. Dekomposisi penggunaan koma dalam satu kebutuhan perangkat lunak dihitung sebagai berikut:

$$f(c) = \begin{cases} 1, & n = 0 \\ (x_1 + 1) * (x_2 + 1) * \dots * (x_n + 1), & otherwise \end{cases} \quad (2)$$

Dimana x_i adalah jumlah koma yang terdapat dalam perincian i dan n adalah banyaknya perincian dengan garis miring dalam kebutuhan perangkat lunak. Sebagai contoh, kebutuhan “sistem dapat membaca, merubah, dan menyimpan file .jpg, .png, dan .bmp.” terdiri dari 2 buah perincian dengan masing-masing perincian berisi 2 koma.

$f(d)$ adalah fungsi perhitungan dekomposisi kebutuhan dengan garis miring. Dekomposisi kebutuhan perangkat lunak yang memiliki garis miring dihitung dengan cara yang serupa dengan $f(c)$ sebagai berikut:

$$f(d) = \begin{cases} 1, & m = 0 \\ (y + 1) * (y + 1) * \dots * (y_m + 1), & otherwise \end{cases} \quad (3)$$

Dimana y_i adalah jumlah koma yang terdapat dalam perincian i dan m adalah banyaknya perincian menggunakan garis miring dalam kebutuhan perangkat lunak.

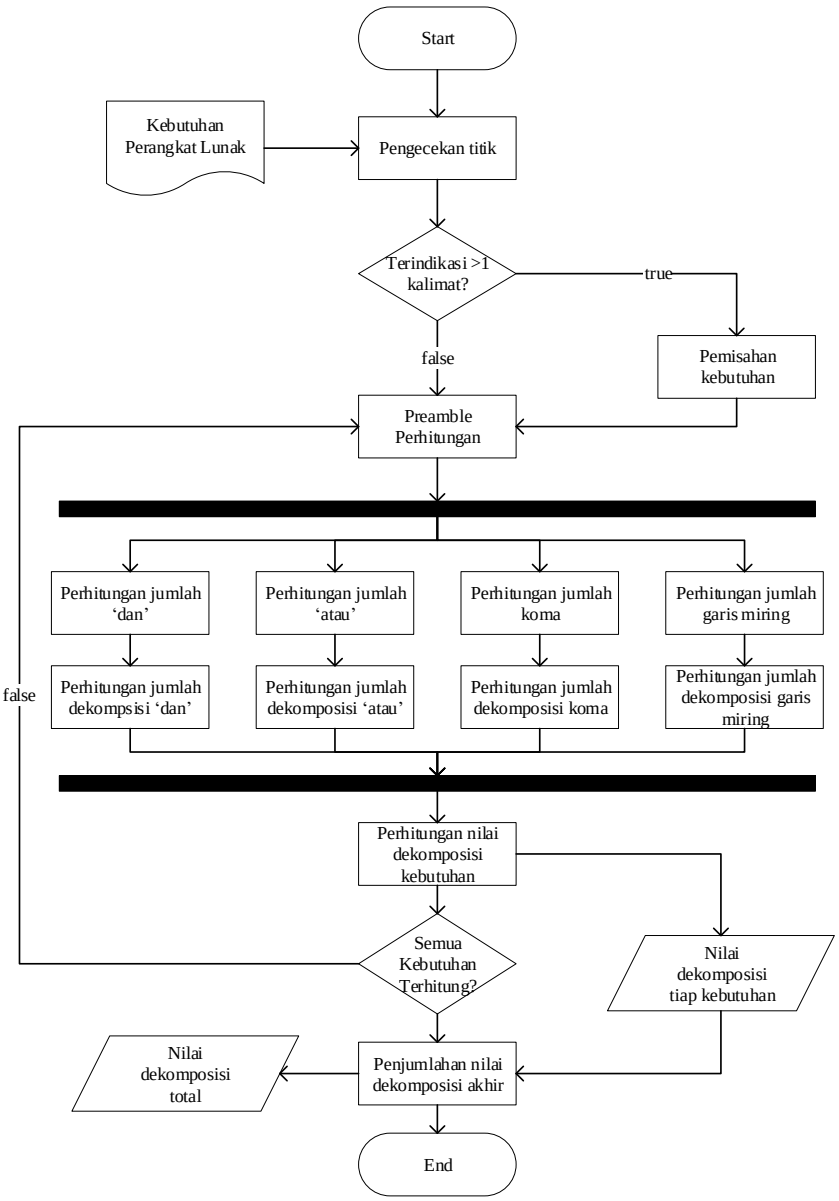
Dalam kondisi dimana kebutuhan memiliki dua buah kalimat atau lebih dengan tanda titik maka keatomikan dihitung menggunakan jumlah keatomikan untuk tiap kalimat sebagai berikut:

$$Atomicity_{total} = \sum_{i=1}^{j} Atomicity_i \quad (4)$$

2.2. Rancangan Sistem

Keseluruhan alur dalam sistem yang dibangun tergambar dalam gambar 5. Sistem dimulai dengan membaca masukan berupa kebutuhan perangkat lunak yang ditulis dalam Bahasa Indonesia. Sistem kemudian mendeteksi apakah terdapat tanda baca titik yang mengindikasikan terdapat dua atau lebih kalimat dalam kebutuhan tersebut. Jika ditemukan, maka sistem akan membagi masukan menjadi kebutuhan-kebutuhan dengan jumlah yang sesuai sebelum semua kebutuhan tersebut diproses. Sebaliknya jika tidak ditemukan, maka sistem akan meneruskan input sebagai kebutuhan yang akan diproses.

Setiap kebutuhan yang dimasukan ke pemrosesan akan melalui proses perhitungan jumlah kata hubung ‘dan’, kata hubung ‘atau’, tanda baca titik (.), dan tanda baca garis miring (/). Sistem kemudian akan melakukan perhitungan nilai dekomposisi untuk setiap kata hubung dengan formula yang sudah dijelaskan sebelumnya. Sistem kemudian akan menghitung nilai keatomikan tiap kebutuhan sebelum dijumlahkan dengan kebutuhan lainnya untuk menghasilkan nilai keatomikan akhir.



Gambar 5 Alur kerja sistem

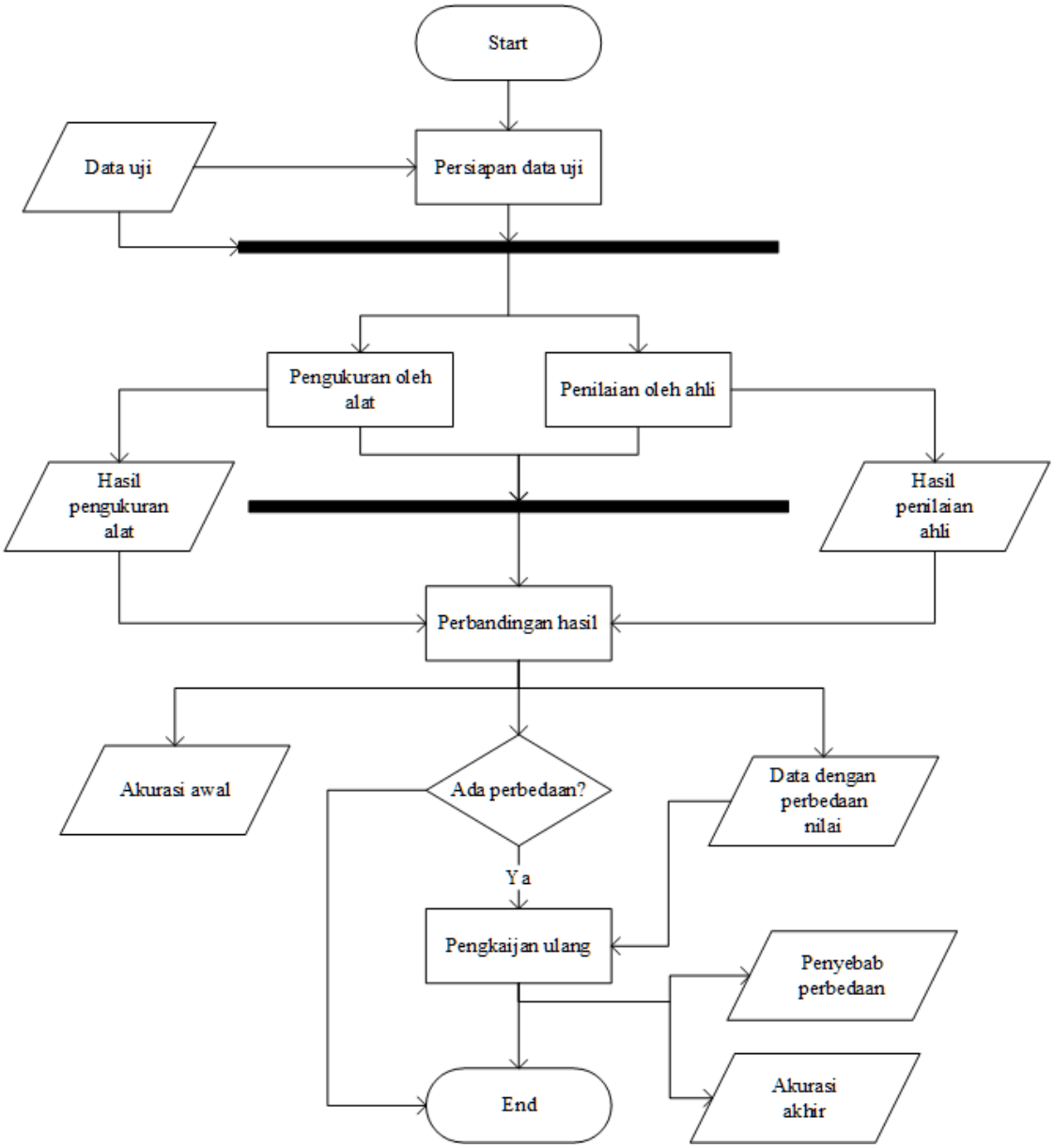
2.3. Pengujian

Pengujian dari sistem penghitung keatomikan yang dibangun dilakukan dengan membandingkan hasil dari sistem tersebut dengan hasil perhitungan oleh ahli dimana keduanya akan melakukan perhitungan keatomikan dengan data yang sama. Ahli yang menjadi sumber perbandingan merupakan seorang *business analyst* yang terlibat dalam 18 proyek pengembangan perangkat lunak selama empat tahun pengalaman kerjanya. Kebutuhan perangkat lunak yang digunakan sebagai data uji diambil dari PURE (Public Requirement Dataset) [11]. PURE berisi 79 dokumen spesifikasi perangkat lunak yang secara total memiliki 34.268 kata. Pengujian menggunakan 40 kebutuhan yang dipilih dan secara acak dari 4 dokumen yang berbeda. Jumlah tersebut dipilih berdasarkan ketersediaan waktu ahli. Kebutuhan tersebut diterjemahkan kedalam Bahasa Indonesia sebelum digunakan untuk pengujian. Contoh dari data uji dapat dilihat dalam tabel 1.

Hasil dari perbandingan akan digunakan untuk melihat tingkat akurasi dari sistem yang dibangun sebagai jawaban terhadap PP1. Data uji yang memiliki perbedaan nilai keatomikan antara sistem dan ahli akan dikaji ulang untuk mengetahui apakah kesalahan ada di sisi sistem atau ahli. Apabila kesalahan ada di sisi ahli, maka perbedaan nilai keatomikan akan diabaikan dan dianggap sama dan nilai akurasi akan diperbaharui. Sebaliknya jika kesalahan ada di sisi sistem, maka penyebab terjadinya kesalahan akan dievaluasi untuk menentukan jawaban dari PP2. Alur dari metode pengujian yang sudah dilakukan tergambar di gambar 6.

Tabel 1 Contoh data uji

No	Bahasa Inggris	Hasil Penerjemahan
1	*The number of digits that can be used for the unique [FILENUMBER] prefix must be 10 or less.*	Jumlah digit yang bisa digunakan sebagai kode unik untuk *prefix* harus kurang dari 10 digit.
2	*Pages number in the page selection must be comprehended.*	Nomor halaman di pemilihan halaman harus dapat dipahami.
3	*Torrent search will share the same search bar with the streaming search.*	Pencarian *torrent* akan berbagi *search bar* yang sama dengan pencarian *streaming*.
4	*Management Processes shall operate on a Linux or Solaris server.*	Proses manajemen akan beroperasi di *server* Linux atau Solaris.
5	*The system counts all types of check-in individually and cumulatively: book-drop, by terminal, by user or self-service.*	Sistem menghitung semua tipe *check-in* secara individual atau kumulatif berdasarkan: *book-drop*, *terminal*, pengguna, atau *self service*.
6	*These requirements shall not constrain functionality or features of the Online Public Access Catalog (OPAC) module.*	Requirement yang ada tidak akan membatasi fungsionalitas dan fitur modul *Online Public Access Catalogue*.
7	*Results will be arranged in size/date/alphabetical order by clicking on the column headers.*	Hasil akan diurutkan berdasarkan ukuran/tanggal/alfabetikal dengan mengklik header kolom.
8	*The compression of the output files requires pdf version 1.5 or above.*	Kompresi dari file output membutuhkan pdf minimal versi 1.5.



Gambar 6 Alur pengujian

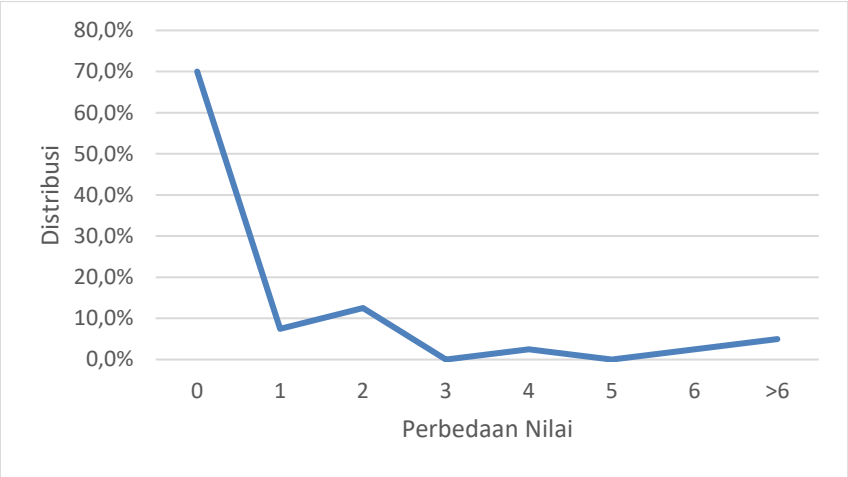
3. HASIL DAN PEMBAHASAN

3.1. Hasil Pengujian Awal

Setelah melakukan pengujian seperti yang digambarkan sebelumnya didapatkan data seperti yang tergambar di gambar 7. Dari 40 data uji yang digunakan didapatkan sebanyak 28 data dengan nilai keatomikan yang sama dari hasil penilaian alat yang dibuat dan penilaian ahli. Jumlah ini menunjukan akurasi awal dari alat yang dibuat sebesar 70%. 30% atau sebanyak 12 data lainnya memiliki perbedaan nilai keatomikan antara pengukuran dengan alat yang dibangun dan penilaian ahli.

Dari data yang memiliki perbedaan nilai, terdapat delapan data uji atau 20% dari total data uji berbeda satu atau dua nilai antara penilaian ahli dan alat yang dibuat. Dua data uji atau 5% dari total data uji memiliki perbedaan sebesar antara tiga hingga enam. Dua data uji lainnya

sebanyak 5% dari total jumlah data uji memiliki perbedaan yang tak hingga karena ahli memberi nilai tidak hingga terhadap keatomikan kedua data uji tersebut.

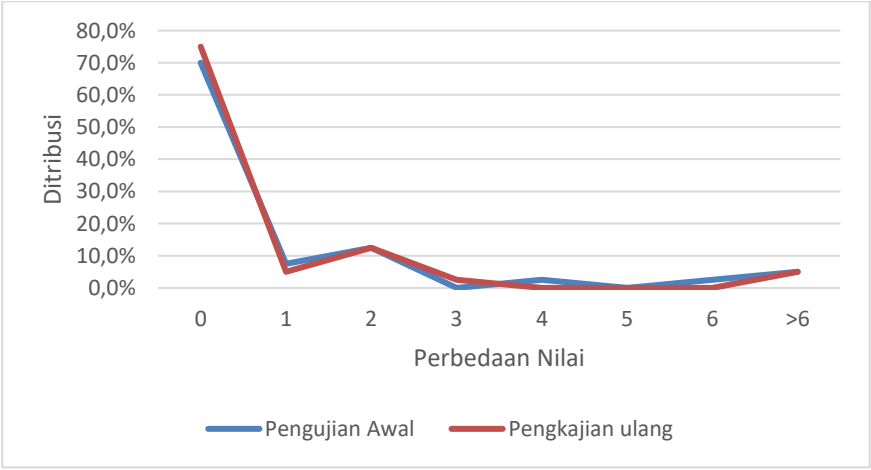


Gambar 7 Distribusi jumlah data uji terhadap perbedaan nilai

3.2. Hasil Pengkajian Ulang

Pengkajian ulang dilakukan kepada 12 data dengan nilai keatomikan yang berbeda bersama dengan ahli. Hal ini dilakukan untuk memutuskan nilai mana yang dapat dianggap benar dan mencari penyebab perbedaan nilai keatomikan. Berdasarkan hasil pengkajian ulang ditemukan bahwa pada sembilan data nilai keatomikan dari ahli lebih tepat, di dua data lainnya alat yang dibuat, dan satu data dimana baik penilaian ahli dan pengukuran alat tidak tepat. Berdasarkan hasil pengkajian ulang tersebut alat yang dibangun memiliki akurasi sebesar 75% dimana 17,5% data uji memiliki perbedaan nilai antara satu dan dua, 2,5% data uji memiliki perbedaan nilai sebesar tiga, dan 5% dengan nilai tak hingga. Persebaran data berdasarkan perbedaan nilai keatomikan antara pengukuran alat dan penilaian ahli dapat dilihat di gambar 8.

Dari sepuluh data dimana pengukuran alat tidak tepat, ditemukan empat penyebab kesalahan pengukuran dalam alat yang dibuat. Penyebab tersebut antara lain: konteks dalam penggunaan kata, penggunaan kata hubung ‘dan’ dalam penjabaran jarak, struktur kalimat yang tidak sesuai kaidah penulisan, dan penggunaan ungkapan “dan lain-lain” dalam perincian. Persebaran data berdasarkan penyebab kesalahan pengukuran dapat dilihat di gambar 9.

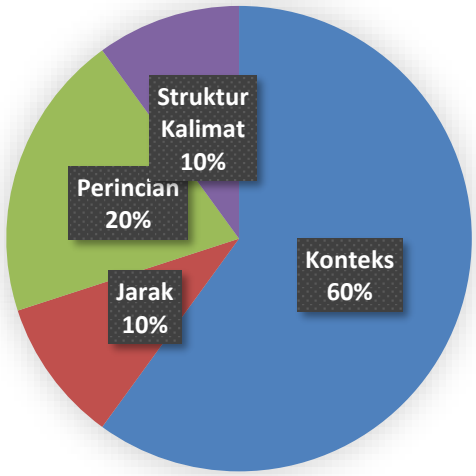


Gambar 8 Perubahan distribusi jumlah data uji terhadap perbedaan nilai pasca pengkajian ulang

Konteks merupakan penyebab kesalahan terbesar dalam alat yang dibangun. Kami menemukan bahwa penggunaan kata hubung ‘atau’ sering digunakan dalam dua kata yang sebenarnya bersifat sinonim. Hal ini menyebabkan alat mendeteksi adanya dua kebutuhan yang berbeda ketika pada kenyataannya hanya ada satu kebutuhan yang diperlukan. Dalam data uji yang digunakan salah satu contoh yang muncul adalah “Pengguna dengan hak akses admin akan dapat menambah/membuat sebuah pengguna baru dan mengasoisasikan pengguna tersebut dengan role khusus dalam sistem,” dimana kata menambah dan membuat dalam kalimat tersebut memiliki makna yang sama. Hal yang sama juga terjadi ketika kata dan digunakan dalam menghubungkan awal dan akhir dari sebuah jarak. Kebutuhan yang berbunyi “Tim proyek akan menjadwalkan *update* terhadap lingkungan produksi hanya pada hari selasa antara jam 7 malam dan 7 pagi,” akan menghasilkan dua buah kebutuhan ketika kata hubung ‘dan’ dalam kalimat tersebut menggambarkan ujung dari sebuah jarak.

Struktur kalimat juga menjadi masalah dalam perhitungan keatomikan perangkat lunak berdasarkan kemajemukan kalimat. Masalah ini muncul dalam kebutuhan yang berbunyi “Query akan mengambil jumlah *seed* dan *peer*, ukuran *file*, tanggal *post*, dan *link* ke *website* terkait.” Kebutuhan tersebut menjabarkan lima hal dengan struktur dimana *seed* dan *peer* disatukan sedangkan tiga hal lainnya dipisahkan menggunakan koma. Hal tersebut menyebabkan pengukuran yang tidak tepat baik dari alat maupun penilaian ahli.

Terakhir, penggunaan ekspresi “dan lain-lain” menjadi masalah yang muncul dalam metode pengukuran yang digunakan oleh alat yang dibangun. Penggunaan ekspresi tersebut menyebabkan perincian tak hingga, dimana jumlah kebutuhan atomik yang ada di dalamnya berjumlah tak hingga. Alat yang dibuat tidak memperhitungkan hal ini sehingga muncul kesalahan dalam pengukuran keatomikan kebutuhan.



Gambar 9 Grafik distribusi jumlah data uji berdasarkan penyebab kesalahan

3.3. Ancaman Terhadap Validitas

Ancaman terbesar dalam penelitian ini adalah minimnya jumlah ahli yang menjadi sumber perbandingan. Hal ini terjadi akibat keterbatasan dalam sarana dan prasarana yang digunakan dalam penelitian ini untuk memfasilitasi keterlibatan lebih banyak ahli dalam waktu yang cukup. Selain itu terdapat ancaman yang muncul ketika data uji yang digunakan merupakan hasil terjemahan dari Bahasa Inggris. Ancaman ini diatasi dengan melakukan pengkajian yang intensif terhadap hasil terjemahan yang sudah dilakukan.

4. KESIMPULAN

Dari penelitian yang sudah dilakukan diteemukan bahwa alat pengukuran keatomikan kebutuhan perangkat lunak yang dibuat dapat digunakan dengan perhatian khusus dari penggunanya untuk mengatasi deviasi nilai yang ada. Alat pengukuran keatomikan kebutuhan perangkat lunak yang dibangun memiliki tingkat akurasi sebesar 75%. 20% dari pengukuran akan memiliki deviasi nilai sebesar satu hingga tiga. 5% dari pengukuran dapat dianggap menghasilkan nilai yang tidak tepat. Alat yang sudah dibuat memiliki kekurangan dalam mendeteksi penggunaan ekspresi "dan-lain" dalam perincian dan penggunaan kata sambung sebagai penggambaran jarak. Selain itu metode pengukuran yang menjadi dasar alat yang dibuat tidak dapat membedakan konteks antara dua kata yang mengapit kata sambung. Metode tersebut juga memiliki ketergantungan terhadap kebenaran struktur kalimat yang digunakan untuk menspesifikasikan kebutuhan perangkat lunak.

5. SARAN

Langkah berikutnya dari penelitian kami dalam mengembangkan alat pengukur keatomikan kebutuhan perangkat lunak adalah dengan cara menambahkan metode dan teknik lain kedalam alat yang sudah dibuat. Salah satunya adalah penggunaan corpus untuk mengatasi kelemahan alat dalam mendeteksi konteks kata yang mengapit kata sambung atau tanda baca. Selain itu metode untuk memperbaiki struktur kalimat menjadi struktur yang baku dapat ditambahkan untuk meningkatkan keandalan alat. Begitu pula perbaikan alat dalam mendeteksi ekspreksi "dan lain-lain" dalam perincian.

Selain peningkatkan kehandalan alat yang sudah dibangun, pembangunan alat yang dapat menghasilkan kebutuhan perangkat lunak yang atomik dari kebutuhan yang memiliki kalimat majemuk juga menjadi penelitian yang dapat dilakukan. Pengembangan alat untuk bahasa lain terutama Bahasa Inggris juga menjadi perhatian kami. Terakhir, pengujian terhadap alat dengan data yang lebih beragam perlu dilakukan untuk meningkatkan keyakinan terhadap hasil penelitian.

DAFTAR PUSTAKA

[1] Wiegers K. and Beatty J., 2013, *Software Requirements*, Ed.3, Microsoft Press, Washington.

[2] Sharma A. and Kushwaha S. D., 2011, Estimation of Software Development Effort from Requirement Based Complexity, *Procedia Technology*, vol 4, hal 716-722.

[3] *Systems and Software Engineering – Life Cycle Processes – Requirement Engineering*, ISO/IEC/IEEE 29148:2018, 2018

[4] Figuerido C. M. and de Souza B. R. C., WOLF: Supporting Impact Analysis in Distributed Software Development, *Proceedings of the 5th International Workshop on Co-operative and Human Aspects of Software Engineering*, Zurich, June 2012.

[5] Thitisathienkul P. and Prompoon N., 2015, Quality Assesment Method for Software Requirement Specifications based on Document Characteristics and its Structure, *Proceeding of 2015 2nd International Conference of Trustworthy Systems and Their Application*, Hualien, July 2015.

[6] Iqbal S. and Khan A. N. M., 2012, Yet another Set of Requirement Metrics for Software Projects, International Journal of Software Engineering and Its Applications, vol 6, no. 1, hal 19-28.

[7] Bokhari U. M. and Siddiqui T. S., Metrics for Requirements Engineering and Automated Requirements Tools, *Proceedings of the 5th National Conference; INDIACom-2011*, March 2011.

[8] Haleem M., Beg R., and Ahmad F. S., 2013, Overview of Impact of Requirement Metrics in Software Development Environment, *International Journal of Advanced Research in Computer Engineering & Technology*, vol 2, no 5, hal 1811-1815.
[9] Zielzynski, P., 2007, *Requirement Management Using IBM Rational RequisitePro*, IBM Press, Boston.
[10] Waridah, E., 2018, *Pedoman Umum Ejaan Bahasa Indonesia dan Seputar Kebahasa-Indonesiaan*, Ed.4, RuangKata, Bandung.
[11] Ferrari A., Spagnolo O. G., and Gnesi S., 2017, PURE: A Dataset of Public Requirements Documents, *Proceeding of 2017 IEEE 25th International Requirement Engineering Conference*, Lisbon, September 2017.