Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.202295732
Vol.9, No. 3, Juni 2022, hlm. 623-630 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 36/E/KPT/2019 e-ISSN: 2528-6579

TEKNIK IDENTIFIKASI FITUR BERDASARKAN KALIMAT PERNYATAAN KEBUTUHAN DALAM KONTEKS PENGEMBANGAN SOFTWARE PRODUCT LINE

M. Syauqi Haris[1], Tri Astoto Kurniawan[*2]

[1] Institut Teknologi, Sains, dan Kesehatan RS dr Soepraoen Kesdam V/Brw, Malang
[2] Universitas Brawijaya, Malang
Email: [1] haris@itsk-soepraoen.ac.id, [2] triak@ub.ac.id
[*]Penulis Korespondensi

(Naskah masuk: 27 Oktober 2021, diterima untuk diterbitkan: 02 Juni 2022)

Abstrak

Software product line (SPL) adalah konsep *software reuse* di bidang industri perangkat lunak yang memiliki fase awal berupa *domain engineering* untuk mengidentifikasi dan memetakan fitur-fitur dari sekumpulan produk perangkat lunak yang akan dikembangkan. Fitur perangkat lunak sering kali diekspresikan secara eksplisit dalam kalimat pernyataan kebutuhan yang ada pada dokumen spesifikasi kebutuhan perangkat lunak (SRS). Saat ini, penelitian tentang otomatisasi identifikasi fitur perangkat lunak berdasarkan dokumen spesifikasi kebutuhan telah banyak diusulkan dengan berbagai metode, namun hasil yang diperoleh kebanyakan adalah kata benda yang dianggap sebagai kandidat fitur. Representasi fitur dengan kata benda dianggap masih terlalu abstrak dan tidak mewakili konsep fitur sebagai kemampuan atau fungsionalitas suatu perangkat lunak. Dalam penelitian ini, identifikasi fitur yang direpresentasikan dengan frasa gabungan kata kerja dan kata benda diusulkan karena dianggap lebih menjelaskan kemampuan dan fungsionalitas dari suatu perangkat lunak. Pola penulisan kalimat pernyataan kebutuhan dengan *requirement boilerplate* dimanfaatkan sebagai dasar identifikasi fitur perangkat lunak secara otomatis dengan menggunakan alat bantu pemrosesan bahasa natural atau NLP (*natural language processing*). Dalam penelitian ini diusulkan 4 (empat) aturan *dependency parser*, yang merupakan salah satu *pipeline* dalam NLP. Tingkat keberhasilan metode pada penelitian ini adalah antara 65% sampai dengan 88% untuk 5 kelompok kalimat pernyataan kebutuhan yang diujikan. Hasil tersebut menunjukkan bahwa metode yang diusulkan pada penelitian ini bisa mengautomasi proses identifikasi fitur pada tahapan *domain engineering* dalam pengembangan *software product line* khususnya yang menggunakan metode ekstraktif.

Kata kunci: *fitur perangkat lunak, definisi fitur, identifikasi fitur, software product line, requirement boilerplate, pemrosesan bahasa natura.*

FEATURE IDENTIFICATION FROM REQUIREMENT SENTENCES IN THE CONTEXT OF SOFTWARE PRODUCT LINE ENGINEERING

Abstract

Software product line (SPL) is a software reuse concept in the software industry that has an initial phase of domain engineering to identify and map the features of a set of software products to be developed. Software features are often expressed explicitly in the requirement sentences contained in the software requirements specification (SRS) document. Currently, research on the automation of software feature identification based on requirements specification documents has been proposed by various methods, but the results obtained are mostly nouns that are considered feature candidates. Representation of features with nouns is considered too abstract and does not represent the concept of features as capabilities or functionality of the software. In this study, the identification of features represented by combined phrases of verbs and nouns is proposed because it is considered to better explain the capabilities and functionality of software. The pattern of writing a requirement sentence with boilerplate requirements is used as the basis for automatically identifying software features using natural language processing (NLP) tools. In this research, 4 (four) dependency parser rules are proposed, which is one of the pipelines in NLP. The success rate of the method in this study is between 65% to 88% for the 5 groups of sentences that were tested. These results indicate that the method proposed in this study can automate the feature identification process at the domain engineering stage in product line software development, especially those using extractive methods.

Keywords: *software feature, feature definition, feature identification, software product line, requirement boilerplate, natural language processing*

1. PENDAHULUAN

Software Product Line (SPL) merupakan metode baru dalam konsep *software reuse* (Sommerville, 2015). SPL telah diakui sebagai strategi *software reuse* yang mampu mengurangi biaya pengembangan perangkat lunak sebesar 61%, mempersingkat *time-to-market*, dan meningkatkan kualitas perangkat lunak hasil pengembangan (Busaidi & Kraiem, 2017). Sedangkan SPL sendiri didefinisikan sebagai sekumpulan perangkat lunak yang memiliki fitur dasar yang sama serta fitur-fitur lain yang terkelola dan dikembangkan untuk memenuhi kebutuhan spesifik segmen pasar atau misi tertentu (Northrop et al., 2016).

Dalam pengembangan SPL atau *Software Product Line Engineering* (SPLE), *Domain Enginering* adalah tahap rekayasa kebutuhan (*requirements engineering*) terhadap keseluruhan produk perangkat lunak yang akan dibuat (Apel et al., 2013). Pada tahap ini, fitur-fitur yang direncanakan untuk keseluruhan produk perangkat lunak diinventarisasi dan dibedakan berdasarkan keberadaannya, apakah wajib (*mandatory*) atau pilihan (*optional*). Adapun salah satu definisi fitur itu adalah suatu unit logis atau perilaku perangkat lunak yang ditentukan oleh serangkaian kebutuhan fungsional (Pohl and Rupp, 2015).

Proses identifikasi fitur perangkat lunak dalam pengembangan SPL umumnya dilakukan oleh seorang ahli secara manual berdasarkan deskripsi produk perangkat lunak yang ada, baik deskripsi secara terstruktur maupun tidak. Hal tersebut menyebabkan rawan terjadi kesalahan (*error-prone*) dan tentunya memerlukan banyak waktu (Acher et al., 2012). Oleh karena itu, diperlukan suatu teknik otomatisasi identifikasi fitur untuk meningkatkan efektivitas dan efisiensi.

Penelitian sebelumnya perihal otomatisasi identifikasi fitur perangkat lunak bisa dikelompokkan menjadi 3 (tiga) berdasarkan sumbernya. Pertama yaitu proses identifikasi fitur dari *source code* program (Laguna et al., 2013) (Stankovv, 2013) (Kastnerv, 2014) (Fenske et al., 2017). Kedua yaitu identifikasi fitur berdasarkan artefak model hasil perancangan perangkat lunak (Martinez et al., 2016) (Martinez et al., 2017). Ketiga yaitu identifikasi fitur melalui proses *requirement engineering* (RE) berdasarkan dokumen kebutuhan dalam bahasa natural (Arora et al., 2017) (Sree-Kumar, Planas and Clarisó, 2018) (Li, Schulze and Saake, 2018)

Pada penelitian ini, kumpulan kalimat kebutuhan yang diekstraksi dari dokumen spesifikasi kebutuhan perangkat lunak dipilih sebagai sumber identifikasi fitur. Hal ini dilakukan karena pada penelitian sebelumnya, di mana telah diusulkan teknik otomatisasi untuk ekstraksi kalimat pernyataan kebutuhan dari suatu dokumen spesifikasi perangkat lunak (Haris and Kurniawan, 2020). Selain itu, dalam prinsip dalam rekayasa perangkat lunak disebutkan bahwa dokumen spesifikasi perangkat lunak yang berisi pernyataan kebutuhan adalah acuan untuk validasi dan verifikasi model hasil rancangan dan kode program hasil implementasi suatu perangkat lunak (Mili and Tchier, 2015).

Berdasarkan penelitian-penelitian terkait yang telah dikaji (Arora et al., 2017)(Sree-Kumar, Planas and Clarisó, 2018)(Li, Schulze and Saake, 2018) (Quirchmayr et al., 2018), fitur hasil identifikasi adalah berupa kata benda tertentu yang diekstraksi dari kalimat pernyataan kebutuhan sebagai kandidat fitur. Fitur perangkat lunak yang direpresentasikan dengan kata benda dianggap kurang mewakili konsep fitur sebagai representasi kemampuan atau fungsionalitas suatu perangkat lunak. Oleh karena itu, definisi fitur sebagai frasa gabungan kata kerja utama dan kata benda objek dari suatu kalimat pernyataan kebutuhan dijadikan dasar identifikasi fitur (Haris, Kurniawan and Ramdani, 2020).

Dalam penulisan kalimat pernyataan kebutuhan perangkat lunak, dikenal pola terstruktur yang dinamakan *requirement boilerplate* (Pohl and Rupp, 2015), sehingga membuka peluang identifikasi fitur dengan memanfaatkan pola tersebut. Dengan menggunakan alat bantu NLP, keterkaitan kata dalam suatu kalimat bisa diidentifikasi menggunakan *dependency parser pipeline*[1]. Oleh karena itu, dalam penelitian ini dilakukan analisis pola keterkaitan kata dalam kalimat pernyataan kebutuhan yang dituliskan menggunakan *requirement boilerplate* sehingga bisa diperoleh bagian kalimat yang merupakan representasi fitur dari perangkat lunak.

2. PENELITIAN TERKAIT

Identifikasi fitur perangkat lunak sebelumnya pernah dilakukan dengan cara melakukan ekstraksi daftar istilah-istilah yang ada pada dokumen spesifikasi kebutuhan (Arora et al., 2017). Daftar istilah tersebut kemudian dijadikan kandidat nama fitur dengan cara melakukan analisis hubungan antar istilah menggunakan pendekatan ontologi. Hasil penelitian tersebut menunjukkan bahwa 90% dari daftar istilah yang diperoleh dapat dijadikan kandidat fitur. Adapun himpunan yang terbentuk berdasarkan hubungan antar istilah tersebut hanya 36% yang layak untuk dijadikan dasar pembentukan pemodelan fitur.

Identifikasi fitur untuk *software product line* dari dokumen spesifikasi berbahasa natural juga pernah dilakukan penelitian sebelumnya dengan

[1] https://spacy.io/api/dependencyparser

metode *lexical analysis* (Sree-Kumar, Planas and Clarisó, 2018). Metode tersebut memanfaatkan *part-of-speech (POS) tagger* NLP untuk mengidentifikasi keberadaan kata benda sebagai subjek dan objek dalam suatu kalimat pernyataan kebutuhan. Setelah itu, daftar kata benda tersebut diproses secara statistik menggunakan *term frequency (TF)* dan *inverse document frequency (IDF)* sehingga diperoleh kata-kata yang bisa dijadikan sebagai kandidat fitur. Adapun akurasi dari penelitian ini untuk identifikasi fitur berkisar antara 41% sampai dengan 60%.

Penelitian tentang identifikasi fitur dari kalimat pernyataan kebutuhan juga pernah dilakukan dengan metode *unsupervised machine learning* menggunakan *convolutional neural network (CNN)* yang bersifat dinamis (Li, Schulze and Saake, 2018). Pada penelitian ini, kalimat pernyataan kebutuhan diproses lebih dahulu menggunakan *laplacian eigenmaps* untuk memperoleh representasi kalimat pada level rendah yang selanjutnya dikonversi menjadi kode biner. Selain itu, kalimat kebutuhan juga diproses menggunakan *word embedding model* untuk memperoleh nilai vektor dari tiap kata yang ada pada kalimat pernyataan kebutuhan sehingga bisa dipadukan untuk menghasilkan representasi kalimat dalam bentuk matriks yang bisa diproses menggunakan CNN. Hasil dari penelitian ini mampu mengidentifikasi 95 kalimat dari 117 kalimat pernyataan kebutuhan yang diujikan.

Selain berdasarkan dokumen spesifikasi kebutuhan, penelitian tentang identifikasi fitur juga pernah dilakukan pada dokumen manual bagi pengguna sistem (Quirchmayr et al., 2018). Tahapan pertama pada penelitian ini adalah melakukan ekstraksi informasi yang relevan terhadap fitur secara semi-manual dari dokumen manual. Tahapan kedua adalah melakukan pemrosesan bahasa manual menggunakan *part-of-speech (POS) tagger* sehingga bisa diperoleh data yang dinamakan *atomic feature-relevant information.* Metode yang dinamakan *Linguistic Information Model (LIM)* ini mampu menghasilkan nilai F sebesar 92% untuk presisi dan akurasinya.

Berdasarkan penelitian-penelitian terkait tersebut, semuanya menghasilkan kandidat fitur yang berupa kata benda dan belum ditemukan penelitian tentang identifikasi fitur yang merupakan frasa gabungan kata kerja utama dan kata benda objek dari kalimat pernyataan kebutuhan. Oleh karena itu, penelitian ini mengusulkan metode identifikasi fitur dari kalimat pernyataan kebutuhan yang menggunakan *requirement boilerplate* dengan memanfaatkan *dependency parser pipeline* NLP yang belum pernah dilakukan sebelumnya.

3. TEORI

3.1 Fitur

Dalam *software product line engineering* (SPLE), fitur merupakan fokus utama karena menjadi dasar pengembangan *core asset base* dan produk perangkat lunak yang akan dihasilkan (Reinhartz-Berger and Kemelman, 2020). Istilah fitur sering kali tidak dapat didefinisikan secara tepat dan pasti karena harus menyesuaikan sudut pandang berbagai pemangku kepentingan, seperti pengguna akhir, manajer proyek pengembangan perangkat lunak, dan *programmer* (Apel et al., 2013). Oleh karena itu, ditemukan berbagai definisi fitur dari beberapa literatur yang telah disebutkan pada Tabel 1 maupun definisi lainnya yang tidak disebutkan.

Tabel 1. Definisi Fitur dari Berbagai Literatur

No.	Definisi Fitur
1.	*Software product properties from existing specifications.* (Boutkova and Houdek, 2011)
2.	*One or more logically related system capabilities that provide value to a user and are described by a set of functional requirements.* (Wiegers and Beatty, 2013)
3.	*Unit of functionality of a software system that satisfies a requirement, represents a design decision, and provides a potential configuration option.* (Apel et al., 2013)
4.	*The description about functional and non-functional requirement characteristics of a system.* (Berger et al., 2015)
5.	Representasi fungsionalitas perangkat lunak yang direpresentasikan oleh frasa gabungan kata kerja utama dan kata benda objek dari kalimat pernyataan kebutuhan yang dituliskan menggunakan *requirement boilerplate.* (Haris, Kurniawan and Ramdani, 2020)

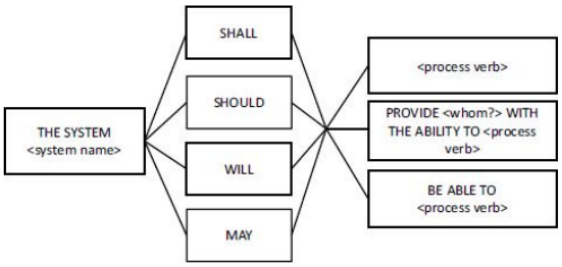
3.2 Requirement Boilerplate

Dalam penulisan kalimat pernyataan kebutuhan, penulisan kalimat harus bisa dipahami oleh semua pemangku kepentingan sistem yang kemungkinan tidak memiliki dasar pengetahuan teknis. Oleh karena itu, penggunaan istilah khusus teknis atau jargon dan notasi bahasa formal haruslah dihindari, sebaliknya yang harus dilakukan adalah penggunaan bahasa natural disertai gambar penjelasan, tabel sederhana, dan diagram yang mudah dipahami (Sommerville, 2015). Penggunaan bahasa yang konsisten akan membuat suatu pernyataan kebutuhan mudah untuk dipahami dan diidentifikasi. Contoh yang sederhana adalah penggunaan kata "*shall*" yang menunjukkan keberadaan suatu pernyataan kebutuhan dalam kalimat. Pola kalimat pernyataan kebutuhan dengan pola susunan kata tertentu disebut dengan *requirement boilerplate* sebagaimana diilustrasikan pada Gambar 1.

3.3 *Dependency Parser* NLP

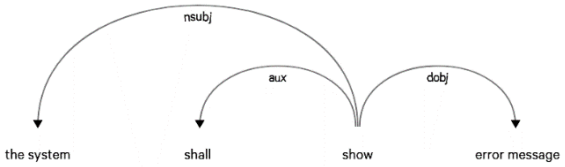
Dependency Parser adalah salah satu pipeline dalam NLP selain sentencizer, tokenizer, lemmatizer, part-of-speech, dan entity recognizer, yang digunakan untuk melakukan segmentasi pada kalimat berdasarkan hubungan antar kata yang ada dalam

kalimat tersebut. Pada alat bantu NLP Spacy2, dependency parser sudah tersedia untuk digunakan menggunakan pre-trained model.



Gambar 1. Pola *Requirement Boilerplate* (Pohl and Rupp, 2015).

Segmentasi kalimat berupa kata-kata dengan relasi berlabel sesuai dengan hubungan dengan kata utama yang disebut sebagai *root.* Sistem kerja *dependency parser* diilustrasikan pada Gambar 2 dan beberapa label *dependency parser* yang sering digunakan bisa dilihat pada Tabel 2.



Gambar 2. Ilustrasi *Dependency Parser* pada NLP menggunakan *Spacy*

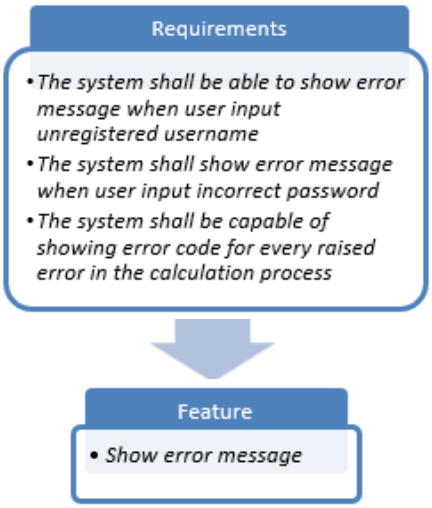
Tabel 2. Beberapa Label *Dependency Parser* yang sering digunakan

Label	Keterangan
aux	*auxiliary*
cc	*coordinating conjunction*
conj	*conjunct*
csubj	*clausal subject*
dep	*unspecified dependency*
dobj	*direct object*
iobj	*indirect object*
neg	*negative*
nsubj	*nominal subject*
pobj	*prepositional object*
prep	*preposition*
root	*root*

4. TEKNIK IDENTIFIKASI

4.1 Pendefinisian Fitur

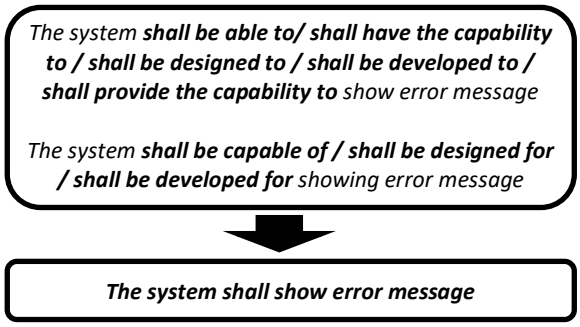
Berdasarkan berbagai definisi fitur yang telah disebutkan pada Tabel 1, definisi fitur sebagai bagian frasa kata kerja utama dan kata benda objek dari kalimat pernyataan kebutuhan yang dituliskan menggunakan *requirement boilerplate,* digunakan dalam penelitian ini. Adapun contoh identifikasi fitur berdasarkan kalimat kebutuhan diilustrasikan pada Gambar 3.



Gambar 3. Ilustrasi Identifikasi Fitur dari Kalimat Pernyataan Kebutuhan (Haris, Kurniawan and Ramdani, 2020)

4.2 Penyederhanaan Kalimat Pernyataan Kebutuhan

Dalam penulisan kalimat pernyataan kebutuhan sering kali didapati beberapa variasi penulisan yang sebenarnya memiliki deskripsi fitur yang sama. Oleh karena itu, perlu dilakukan penyederhanaan kalimat sesuai dengan pola *requirement boilerplate* untuk mempermudah proses segmentasi kalimat selanjutnya menggunakan *dependency parser.* Ilustrasi variasi penulisan kalimat pernyataan kebutuhan dapat dilihat pada Gambar 4.



Gambar 4. Ilustrasi Penyederhanaan Kalimat Pernyataan Kebutuhan

4.3 Identifikasi Fitur dengan NLP *Dependency Parser*

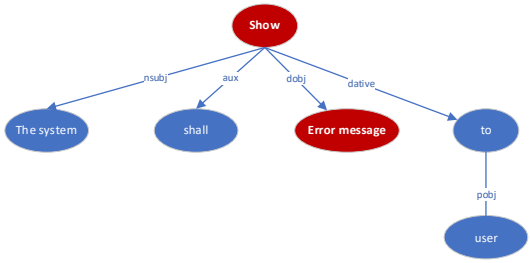
Identifikasi fitur dilakukan dengan cara menganalisis berbagai variasi kalimat pernyataan kebutuhan yang telah melalui proses penyederhanaan kalimat. Kalimat pernyataan kebutuhan yang digunakan adalah kumpulan kalimat pernyataan kebutuhan yang diperoleh dari *dataset* dokumen spesifikasi kebutuhan yang diperoleh pada penelitian kami sebelumnya tentang otomatisasi ekstraksi kalimat kebutuhan dari dokumen *software*

[2] https://spacy.io

requirement specification (Haris and Kurniawan, 2020).

Aturan *dependency parser* dirumuskan dengan cara menggambarkan hasil *parsing* dalam bentuk *tree*, dimana setiap token menjadi *node* dan label relasi hasil *parsing* sebagai *edge*. Jika X adalah label *dependency token* dan Y adalah level *token* dalam *tree*, maka setiap token dalam *tree* bisa dituliskan dengan notasi X_y. Sehingga bisa diilustrasikan dan dirumuskan aturan-aturan sebagai berikut:

Aturan 1 diperoleh dengan cara menganalisis kalimat seperti "*the system shall show error message to user"* yang bisa diilustrasikan pola *dependency parser*-nya pada Gambar 5.



Gambar 5. Pola Kalimat untuk Aturan 1.

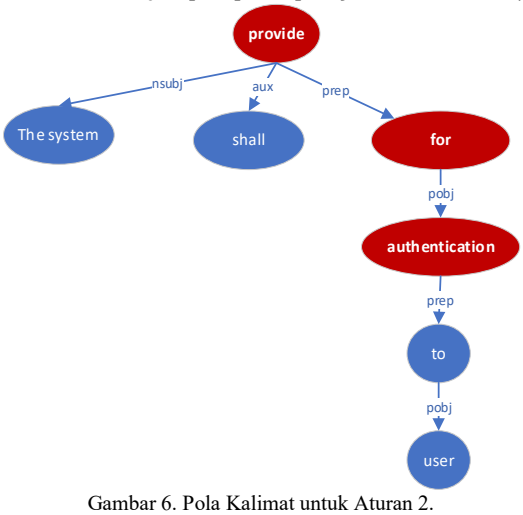
Sehingga bisa dirumuskan aturan pertama untuk identifikasi fitur dari kalimat pernyataan kebutuhan sebagai Persamaan (1).

$$\boldsymbol{Rule1 = root_0 + dobj_1} \quad \textbf{(1)}$$

Aturan 2 diperoleh dengan cara menganalisis kalimat seperti "*the system shall provide for authentication to user*" yang bisa diilustrasikan pola *dependency parser*-nya pada Gambar 6.

Sehingga bisa dirumuskan aturan kedua untuk identifikasi fitur dari kalimat pernyataan kebutuhan sebagai Persamaan (2).

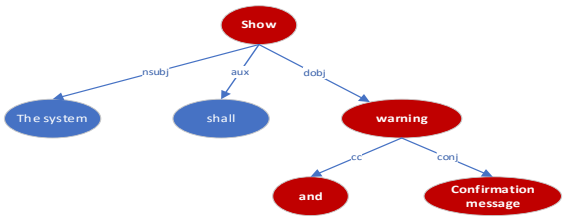
$$\boldsymbol{Rule2 = root_0 + prep_1 + pobj_2} \quad \textbf{(2)}$$



Gambar 6. Pola Kalimat untuk Aturan 2.

[3] https://www.python.org/

Aturan 3 diperoleh dengan cara menganalisis kalimat seperti "*the system shall show warning and confirmation message*" yang bisa diilustrasikan pola *dependency parser*-nya pada Gambar 7.

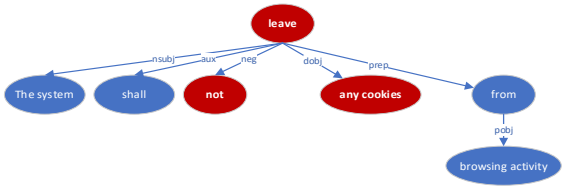


Gambar 7. Pola Kalimat untuk Aturan 3.

Sehingga bisa dirumuskan aturan ketiga untuk identifikasi fitur dari kalimat pernyataan kebutuhan sebagai Persamaan (3).

$$\boldsymbol{Rule3 = (Rule1\ OR\ Rule2) + cc_2 + conj_2} \quad \textbf{(3)}$$

Aturan 4 diperoleh dengan cara menganalisis kalimat seperti "*the system shall not leave any cookies from browsing activity*" yang bisa diilustrasikan pola *dependency parser*-nya pada Gambar 8.



Gambar 8. Pola Kalimat untuk Aturan 4.

Sehingga bisa dirumuskan aturan keempat untuk identifikasi fitur dari kalimat pernyataan kebutuhan dalam Persamaan (4).

$$\boldsymbol{Rule4 = neg_1 + (Rule1\ OR\ Rule2\ OR\ Rule3)} \quad \textbf{(4)}$$

5. HASIL PENGUJIAN

Pada penelitian ini, aturan-aturan yang telah dirumuskan diujikan pada kumpulan kalimat pernyataan kebutuhan yang diperoleh dari *dataset* yang berisi 5 (lima) dokumen spesifikasi kebutuhan yang telah digunakan pada penelitian sebelumnya (Haris and Kurniawan, 2020). Pengujian dilakukan dengan cara mengimplementasikan aturan-aturan *dependency parser* menggunakan bahasa pemrograman *Python*[3] dan *Spacy library* untuk memproses 5 (lima) berkas teks yang berisi daftar kalimat pernyataan kebutuhan hasil ekstraksi. Hasil yang didapatkan dari pemrosesan tersebut adalah daftar frasa gabungan kata kerja utama dan kata benda objek dari tiap kalimat pernyataan kebutuhan.

Hasil pengujian menunjukkan bahwa aturan-aturan *dependency parser* yang diusulkan memiliki efektivitas beragam untuk tiap kelompok data uji.

Hasil persentase efektivitas tertinggi diperoleh pada kelompok data uji *LIS* yang berhasil mengidentifikasi 23 fitur dari 26 kalimat pernyataan kebutuhan atau 88,46% dan hasil terendah diperoleh pada kelompok data uji *e-store* yang hanya berhasil mengidentifikasi 56 fitur dari 85 kalimat pernyataan kebutuhan atau 65,12%. Adapun hasil lengkap yang dilakukan terhadap 5 (lima) kelompok data uji bisa dilihat pada Tabel 3.

Tabel 3. Akurasi Hasil Implementasi Aturan NLP *Dependency Parser* untuk Identifikasi Fitur

Data Uji	nRS	nF	%
CCTNS	37	30	81,08
DH	33	27	81,82
e-Store	85	56	65,12
LIS	26	23	88,46
Puget	36	27	75,00

nRS = jumlah kalimat pernyataan kebutuhan
nF = jumlah fitur yang berhasil diidentifikasi
% = prosentase tingkat keberhasilan identifikasi

6. PEMBAHASAN

Hasil penelitian ini memiliki hasil persentase keberhasilan yang berbeda-beda untuk tiap kelompok data uji. Hal ini dikarenakan terjadi ketidakakuratan identifikasi yang disebabkan oleh beberapa hal, seperti kalimat pernyataan kebutuhan dengan perspektif pengguna yang dinyatakan sebagaimana perspektif sistem. Contohnya adalah beberapa kalimat seperti berikut:

- *The system shall enable user to navigate between the search results*
- *The system shall allow user to create profile and set his credential*
- *The system shall enable the user to enter their reviews and ratings*

Ketiga contoh kalimat di atas adalah kalimat pernyataan kebutuhan dengan perspektif pengguna yang dituliskan dengan pola *requirement boilerplate* seperti halnya perspektif sistem sehingga fitur yang diidentifikasi (frasa yang digarisbawah) menjadi tidak akurat.

Penyebab ketidakakuratan berikutnya adalah karena keterbatasan alat bantu NLP dalam mendeteksi jenis kata sehingga hubungan antar kata dalam aturan *dependency parser* menjadi tidak akurat. Beberapa kalimat pernyataan kebutuhan dengan frasa kata yang tidak bisa diidentifikasi adalah sebagai berikut:

- *The e-store system shall communicate to credit management system*
- *The system shall open a pop-up window displaying information*

Pada kalimat pertama di atas, alat bantu NLP menganggap frasa "*to credit*" adalah kata kerja padahal seharusnya frasa "*credit management system*" adalah kata benda sebagai objek. Sedangkan pada kalimat kedua, alat bantu NLP juga tidak bisa mengidentifikasi frasa "*pop-up windows*" sebagai suatu kata benda sebagai objek.

7. KESIMPULAN

Pada penelitian ini dibuktikan bahwa aturan *dependency parser* NLP yang diusulkan mampu mengidentifikasi fitur perangkat lunak dari suatu kalimat pernyataan kebutuhan. Adapun hasil persentase keberhasilan bervariasi antara 65% sampai dengan 88% dikarenakan pada beberapa kalimat pernyataan kebutuhan ada yang masih menggunakan perspektif pengguna, sedangkan pola *requirement boilerplate* yang digunakan adalah perspektif sistem. Selain itu, ketidakakuratan identifikasi juga disebabkan adanya keterbatasan pendeteksian jenis kata oleh NLP menggunakan *Spacy*, seperti frasa kata benda yang dideteksi sebagai kata kerja.

Penelitian ini masih berfokus pada identifikasi fitur dari kalimat pernyataan kebutuhan dalam bahasa Inggris dengan asumsi bahwa satu kalimat kebutuhan mewakili satu fitur perangkat lunak. Adapun informasi variabilitas, seperti hubungan antar fitur, masih belum teridentifikasi. Pada penelitian berikutnya, implementasi dalam bahasa Indonesia dan hubungan antar fitur akan diteliti agar dapat diperoleh data tingkat keberadaan fitur dalam suatu produk perangkat lunak, apakah bersifat wajib (*mandatory*) atau pilihan (*optional*). Selanjutnya, daftar fitur hasil identifikasi bisa disajikan dalam bentuk diagram *Feature Model* (FM), yang merupakan artefak utama pada proses *domain engineering* dalam pengembangan *software product line.*

DAFTAR PUSTAKA

ACHER, M., CLEVE, A., PERROUIN, G., HEYMANS, P., VANBENEDEN, C., COLLET, P. AND LAHIRE, P., 2012. On extracting feature models from product descriptions. *Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems*, VaMoS '12, pp.45–54.

APEL, S., BATORY, D., KÄSTNER, C. AND SAAKE, G., 2013. *Feature-Oriented Software Product Lines*. Springer-Verlag Berlin Heidelberg.

ARORA, C., SABETZADEH, M., BRIAND, L. AND ZIMMER, F., 2017. Automated Extraction and Clustering of Requirements Glossary Terms. *IEEE Transactions on Software Engineering*, 43(10), pp.918–945.

BERGER, T., LETTNER, D., RUBIN, J., GRÜNBACHER, P., SILVA, A., BECKER, M., CHECHIK, M. AND CZARNECKI, K., 2015. What is a feature? A Qualitative study of features in industrial software product lines. *ACM International Conference Proceeding Series*, 20-24-July, pp.16–25.

BOUTKOVA, E. AND HOUDEK, F., 2011. Semi-automatic identification of features in requirement specifications. *Proceedings of the 2011 IEEE 19th International Requirements Engineering Conference, RE 2011*, pp.313–318.

FENSKE, W., MEINICKE, J., SCHULZE, S., SCHULZE, S. AND SAAKE, G., 2017. Variant-preserving refactorings for migrating cloned products to a product line. *SANER 2017 - 24th IEEE International Conference on Software Analysis, Evolution, and Reengineering*, pp.316–326.

HARIS, M.S. AND KURNIAWAN, T.A., 2020. Automated requirement sentences extraction from software requirement specification document. In: *Proceedings of the 5th International Conference on Sustainable Information Engineering and Technology*. [online] New York, NY, USA: ACM.pp.142–147.

HARIS, M.S., KURNIAWAN, T.A. AND RAMDANI, F., 2020. *Teknik Ekstraksi Katalog Fitur dari Dokumen Spesifikasi Kebutuhan Sebagai Basis Pengembangan Software Product Lines*. Tesis Magister. Universitas Brawijaya.

KASTNER, C., DREILING, A. AND OSTERMANN, K., 2014. Variability mining: Consistent semi-automatic detection of product-line features. *IEEE Transactions on Software Engineering*, 40(1), pp.67–82.

LAGUNA, M.A. AND CRESPO, Y., 2013. A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring. *Science of Computer Programming*, [online] 78(8), pp.1010–1034.

LI, Y., SCHULZE, S. AND SAAKE, G., 2018. Extracting features from requirements: Achieving accuracy and automation with neural networks. *25th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2018 - Proceedings*, 2018-March, pp.477–481.

MARTINEZ, J., ZIADI, T., BISSYANDE, T.F., KLEIN, J. AND LE TRAON, Y., 2017. Bottom-up technologies for reuse: Automated extractive adoption of software product lines. *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering Companion, ICSE-C 2017*, pp.67–70.

MARTINEZ, J., ZIADI, T., BISSYANDÉ, T.F., KLEIN, J. AND LE TRAON, Y., 2016. Automating the extraction of model-based software product lines from model variants. *Proceedings - 2015 30th IEEE/ACM International Conference on Automated Software Engineering, ASE 2015*, pp.396–406.

MILI, A. AND TCHIER, F., 2015. *Software Testing Concepts and Operations*. New Jersey: John Wiley & Sons.

POHL, K. AND RUPP, C., 2015. *Requirements Engineering Fundamentals*. 2nd ed. Rocky Nook Inc.

QUIRCHMAYR, T., PAECH, B., KOHL, R., KAREY, H. AND KASDEPKE, G., 2018. Semi-automatic rule-based domain terminology and software feature-relevant information extraction from natural language user manuals An approach and evaluation at Roche Diagnostics GmbH.

REINHARTZ-BERGER, I. AND KEMELMAN, M., 2020. Extracting core requirements for software product lines. *Requirements Engineering*, [online] 25(1), pp.47–65.

SOMMERVILLE, I., 2015. *Software Engineering*. Tenth Edit ed. *Software Engineering*. Pearson Education Limited.

SREE-KUMAR, A., PLANAS, E. AND CLARISÓ, R., 2018. Extracting software product line feature models from natural language specifications. *ACM International Conference Proceeding Series*, 1, pp.43–53.

STANKOV, E., JOVANOV, M. AND BOGDANOVA, A.M., 2013. Source code similarity detection by using data mining methods. *35th International Conference on Information Technology Interfaces, ITI 2013*, [online] pp.257–262.

WIEGERS, K. AND BEATTY, J., 2013. *Software Requirements, Third Edition*. *Microsoft Press*.S

Halaman ini sengaja dikosongkan