

---

# Concept Extraction in Requirements Elicitation Sessions: Prototype and Experimentation

Tjerk Spijkman<sup>b,a</sup>, Boris Winter<sup>a</sup>, Sid Bansidhar<sup>a</sup> and Sjaak Brinkkemper<sup>a</sup>

<sup>a</sup>Dept. of Information and Computing Sciences, Utrecht University. Utrecht, the Netherlands

<sup>b</sup>fizor. Utrecht, the Netherlands,

## Abstract

**[Introduction]** Requirements elicitation is an important, yet complex step in the software development life cycle. It is vital for business analysts to differentiate between configuration and customization requirements, as they require a vastly different implementation approach. **[Objectives]** This paper explores the design and evaluation of a key abstraction extraction NLP tool that is able to extract concepts which indicate configuration or customization needs from transcripts of elicitation sessions to aid business analysts. **[Method]** Our research is structured according to Wieringa's engineering cycle. We drafted an ontology for an existing software product, designed a prototype NLP tool and dashboard to identify known and unknown concepts based on the ontology. We then validated the results through domain expert interviews. **[Results]** The prototype outputs and domain expert validation show that the tool can provide insight on the contents of a elicitation session through a presentation of discussed known and unknown concepts. **[Conclusion]** Through the tool design and NLP prototype we build a promising foundation for further investigation and development, and identify the need for providing additional context information.

## Keywords

Requirements Engineering, Ontology, Natural Language Processing, Business Analysis, Case Study

## 1. Introduction

Requirements elicitation (RE) is a crucial step in the software development process, as it ensures accurate stakeholder understanding [1]. During this phase, requirements are elicited to ensure the software design, or an existing product meets the needs of the customer [2]. While many elicitation techniques exist, they are mostly applied in a conversational setting [3]. These meetings are subject to challenges, including domain knowledge, time-constraints and other obstacles associated with stakeholder and analyst interaction [4]. Transcribing and post-processing of the lengthy recordings from these elicitation sessions is an administrative burden [3, 5].

Software products are used by a variety of users, with their own requirements. Most products offer variability options for configuration of the software to support different situations [6]. Additionally, software products are often customized by business partners or by the customers themselves [7]. According to Shin [8], up to 60% of ERP project costs are exhausted during setup, implementation and

---


In: F.B. Aydemir, C. Gralha, S. Abualhaija, T. Breaux, M. Daneva, N. Ernst, A. Ferrari, X. Franch, S. Ghanavati, E. Groen, R. Guizzardi, J. Guo, A. Herrmann, J. Horkoff, P. Mennig, E. Paja, A. Perini, N. Seyff, A. Susi, A. Vogelsang (eds.): *Joint Proceedings of REFSQ-2021 Workshops, OpenRE, Posters and Tools Track, and Doctoral Symposium, Essen, Germany, 12-04-2021*

EMAIL: tjerk@fizor.io (T. Spijkman); b.winter@uu.nl (B. Winter); s.p.bansidhar@uu.nl (S. Bansidhar); s.brinkkemper@uu.nl (S. Brinkkemper)

ORCID: 0000-0003-2726-3065 (T. Spijkman); 0000-0002-2977-8911 (S. Brinkkemper)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

customization. Similarly, Somers and Nelson [9] observed that 37% of companies make small customizations to their software products and 5% of companies customize core applications to fit their business needs.

Depending on market position, companies that supply software products often have to make customizations to meet customer requirements [10]. Ali et al. [11] discuss customization in SaaS applications and resulting challenges like performance issues, increasing software complexity and the need to maintain each tenant's customization code.

In this paper, we discuss and design an aid for RE practitioners, that extracts key concepts from RE transcripts and compares them to a software product ontology. We describe a software product ontology as: “A set of concepts and relationships in a software product domain, which describe functionalities, artifacts and related software systems”. To this end, we investigate the following research question:

**RQ** : “*To what extent can the ontology of a software product be utilized to recognize customization and configuration requirements in a elicitation session?*”

Our investigation of this research question is guided based on the expectation that comparing a transcript to the software product ontology can indicate these requirements from whether or not the concepts can be recognized. We design and evaluate a prototype key abstraction extraction tool [12] with the aim of detecting both unknown and known concepts in a requirements elicitation session. Specifically, we hypothesize that based on a software-product ontology:

**H1** *Unknown concepts indicate the need for customization of the product.*

**H2** *Known concepts indicate the need for configurations of existing features in the product.*

To illustrate these hypotheses we provide an hypothetical example in the domain of a route planning product, as we expect most readers are familiar with this type of software:

“Our company provides shipping of goods on multiple types of routes. We utilize our trucks for national shipping, and our *Cargo ships* to ship goods internationally. For all of our shipping we need to ensure efficient **fuel** use.”

For illustrative purposes, we highlighted one presumed *unknown term* in italics, and one **known term** that might influence the implementation of the product. The known concept, *fuel*, indicates that efficiency as a key factor of the configuration (e.g. compared to delivery time). The term *cargo ships* can indicate the need to support naval route planning as a potential customization.

Through this research, we aim to contribute valuable knowledge to the RE field. There is little uniformity within the RE field about the practices involved and practitioners often have to deal with a considerable amount of administrative burden [13]. Utilization of ontologies to detect unknown and known concepts can aid in determination of configuration and customization needs. Furthermore, we position the findings as a pilot study for automated extraction of requirements from elicitation transcripts, and share our vision. Defining the important concepts is intended to help focus an NLP approach to detect requirements relevant information for the concepts and use this to generate user stories in the Connextra template.

The remainder of this paper will be structured as follows: Section 2 covers related work to position our research and inform about similar research. Section 3 presents our research method. Section 4 describes the design of the concept extraction tool. Section 5 elaborates on the validation of our prototype and its output based on a case study. We share our research vision on automated user story generation from transcripts in Section 6, followed by a discussion and conclusion in Section 7.

## 2. Related Work

In this section, we discuss works in the key fields for our research. First we cover NLP in RE to position our research and cover state of the art NLP papers. Second, we provide a brief overview of use of ontologies in RE, and finally we note a few works on software product customization.

**NLP in RE** By having users express their information needs and knowledge domain in natural language, they are not required to familiarize themselves with formal languages [14]. Natural Language Processing (NLP) makes up a range of techniques to analyze and represent natural language. While NLP research has traditionally focused on lexical semantics, it has recently shifted towards compositional semantics [15, 16] and spoken dialogue systems [16].

NLP has been a powerful tool in the Requirements Engineering domain. For example state of the art approaches include; Lucassen et al. who provide an NLP based evaluation of a set of user stories checking for common issues [17]. Tooling by Gacitua et al. enable extracting abstractions from a document [18]. Wang et al. position an approach for analyzing app reviews. And Martens and Maalej provide an NLP based approach for extracting context information from social media.

Abualhaija et al. [21] propose an approach with goals similar to our vision of identifying requirements through NLP, though their artifact focus is on requirements specifications instead of transcripts. Kumar et al. [22] propose a method to automatically build an ontology from domain specific text by employing statistical methods and computational linguistics techniques. Finally, Arora et al. [23] make use of NLP to extract glossary concepts from natural language requirements.

**Domain ontology** As people have different backgrounds, ideas and needs during communication, their viewpoints and assumptions regarding the subject matter can differ. This lack of a shared understanding can lead to poor communication and errors in requirements elicitation [24, 25]. Since an ontology describes all concepts and relationships in a domain [24] it can aid in achieving a shared understanding by eliminating confusion about terminology and concepts.

**Customization** During requirements elicitation, it is possible that requirements can change or evolve. As a result, software designs, and software products change and become more complex. In certain domains, such as engineering, variant-rich systems are common [6]. Customization can play a role in adapting software to customer specific requirements, but if handled incorrectly can break the architectural integrity of a software product family [26]. Ali et al. [11] conclude that configuration, composition and extension are the most common solutions to enabling customization.

## 3. Research Method

Our research is structured according to the four phases of the engineering cycle by Wieringa [27]:

**Phase I: Problem Investigation:** This research explores how the ontology of a software product can be utilized in combination with the transcript of a requirements elicitation session to improve post-processing of an analysis for a software product. To this end, we perform a combination of an exploratory, deductive case study [28] and design science [27].

The case study is focused on SCANMAN, an accounts payable automation software product. The data was made available as the first author is embedded in the case company. For this software product, we constructed an ontology containing the concepts and relationships within the product domain. An example of this is shown in Fig. 2. By comparing this ontology to the transcripts, we hypothesize that customer unique concepts can be discovered.

**Phase II: Treatment Design:** For our tool design, we explored an industry case study. This enabled the use of an ontology belonging to an existing product and validation with domain experts. The

specifics of the tool design are discussed in Sec. 4

**Phase III: Treatment Validation:** To ensure that the designed treatment is valid and can be applied to achieve the goals of the research, we consider the validity threats defined by Wohlin et al. [29]. For this research, the main *internal validity* threat is that the outcome is specific to the studied case, as the treatment will also be designed based on that data. To mitigate this threat, we will also evaluate the tool outputs based on a different cases for the software product resulting in data triangulation. For the *external validity* we are aware that there is limited generalisability in our research, as it is based on a single software product. However, our goal is to have a first phase of the design cycle, which can be used to roll out and apply the treatment to different software products. Additionally, we perform the case study based on the guidelines of Wohlin et al. [29] to ensure the treatment *construct validity*.

**Phase IV: Treatment Implementation:** After validating our output of the case and comparing it to the documentation resulting from the analysis, the output is tested against a different data sets from other cases for the software product, improving the internal validity. The data used in this research is real data, consisting of transcripts gathered from RE sessions. These outputs are then validated through semi-structured interview meetings with domain experts on the software product. The experts are presented a output for a case they are familiar with, and asked to reflect on the outputs of the tool, and their perceived usefulness of the treatment.

## 4. Design of the Key Concept Extraction Tool

The tool that has been created in order to extract key concepts was coded in Python, and uses the following non-native Python packages: NLTK, TextBlob and Pandas. As of this paper, the tool is in an early stage of development and requires some human intervention. A description of the process in pseudocode is shown in Fig. 1.

<p><b>Input:</b> T: transcript of requirements elicitation session, O: ontology of the software product</p> <p><b>Output:</b> E: set of extractions. Each <math>e \in E</math> is a tuple of <math>\langle \text{concept}, \text{frequency}, \text{speaker} \rangle</math> categorized in known and unknown concepts tables, for example <math>e = \langle \text{voucher}, 37, \text{spk}_2 \rangle</math></p> <p><i>function preprocessing;</i>  <b>for T do</b>              split by speaker tag s;              remove timestamps;              remove punctuation &amp; stop words;              initiate function known concepts;              initiate function unknown concepts;  <b>end</b></p>	<p><i>function known concepts;</i></p> <p><b>for speaker s</b>            <b>for words in s do;</b>              compare words to concepts in O;              <b>if known</b>                add to dataframe;                increase concept count;              <b>else</b>                discard words;            combine dataframes;            export combined dataframe to E (known);  <b>end</b></p>	<p><i>function unknown concepts;</i>          remove concepts in ontology O from transcript T  <b>for speaker s</b>            <b>for noun phrases in s do</b>              determine number of mentions;              <b>if mentions &gt; configured frequency</b>                add to table;                increase noun phrase count;              <b>else</b>                discard noun phrase;            combine dataframes;            export combined dataframe to E (unknown);  <b>end</b></p>
--	--	--

**Figure 1:** Pseudocode of the key abstraction extraction tool prototype

For our case studies, the transcripts were generated using AWS Transcribe. The ontology file created through a manual investigation of documentation and functionality of the product. Extracting the concepts starts with initial text processing. In this step timestamps, punctuation and stop words are removed from the transcript document, and the text is split by speaker. After this text processing two functions are executed simultaneously, for extraction of the known concepts and the unknown concepts respectively. For the known concepts, the tool iterates through each word in the text matching it with the ontology for each of the speakers. If the word is found in the ontology, it will be added to a table. After this function is complete, all tables will be then merged into one table.

For the unknown words, the Noun Phrase property of the TextBlob package is utilized. This returns a combination of words that occur together in a sentence. These Noun phrases can be useful for indicating discussion topics in a transcript and assist in pointing out requirements. These noun phrases are collected in a table for the associated speaker. At the end of the process, two tables are generated, one with known concepts and one with (potentially) unknown concepts. These are presented in a format consisting of the concept, their occurrence frequency and their associated speaker.

## 5. Tool Validation

The tool was applied to transcripts from a industry case, containing 11 hours of recordings for requirements elicitation session between an analyst and customer. The data-set contains 84.500 words, spanning 97 pages without spacing. While this was our primary case study for our own analysis and evaluation of the tooling, two additional cases have been studied for the evaluation sessions in section 5.2. These cases contained transcripts totaling 48.000 and 21.000 words respectively.

An example of a partial ontology in the case, and the impact of recognized concepts can be seen in figure 2. The discovered concepts can indicate requirements as hypothesized, for instance if we consider the following quote from the main industry case:

**“Purchase orders** that we do in **JDE** are currently only **supplies** and **facilities**. Some supplies which are called *minor equipment*. They are requested by **facilities**, if it’s a *slow speed* it goes through **JDE** if it’s *high speed* it goes through *service channel*.”

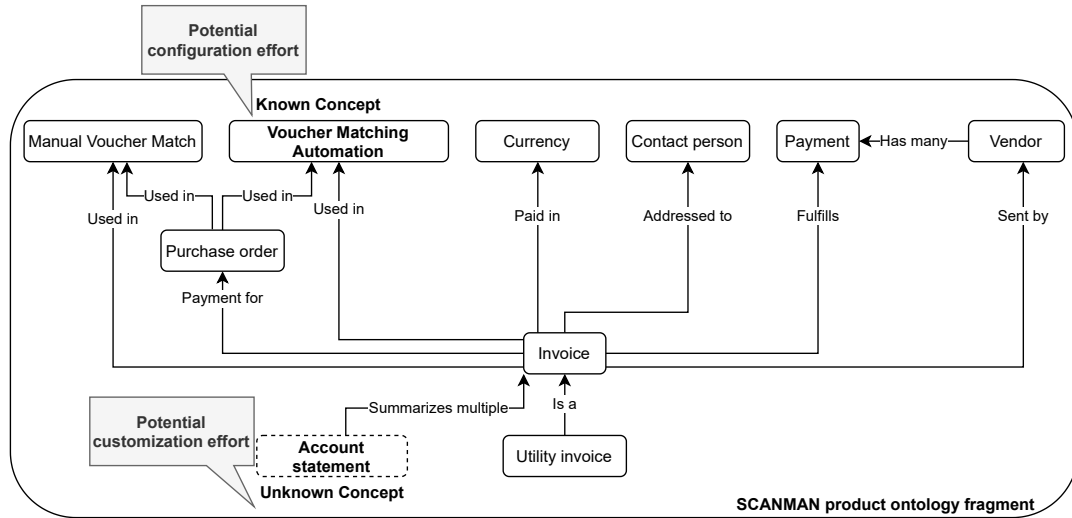
This quote contains in **bold** a number of concepts known within the domain, and in *italics* two unknown concepts. For purchase orders, they imply specific processes that need to be configured in the software product. In contrast, a unknown concept like service channel can indicate some challenges, as there is a different source of purchase orders and they might not be available for the software product. The tool outputs a collection of the extracted concepts, without the context added in the quote to improve understandability.

### 5.1. Primary Case Evaluation

We evaluated the outputs of the tool from the transcripts of the main case. The first author was involved in this case and could provide reflection from a industry point of view. While the output of the tool contained concepts that are not directly useful, for instance those very common in the domain (e.g. invoices in an accounts payable domain). Other concepts were deemed interesting. For example a known term within the ontology, Voucher Match Automation requires additional configuration and is an optional functionality of the software product. Recognizing this was often discussed in a conversation makes it likely that this configuration is in the scope for that customer. Similarly, the unknown concept, account statement, indicates that the customer receives a summary of multiple invoices. Automatic processing of these statements is not supported from the out-of-the box functionality, and can be linked to a identified customization for the case. Similarly, 6 out of 11 of the focus topics could be linked to the presented concepts.

Of the 205 identified concepts extracted from the main case study transcripts when they occurred at least two times, 53 were identified as known concepts, out of which 14 were tagged as indication for configuration. For the 152 unknown concepts, the distribution is as follows: 17 are tagged as additions for the ontology, 114 are irrelevant, 5 indicate a customization need and 16 are specific to the customer domain. The irrelevant words include filler words not included in the filtering by the tool (e.g. “hey”, and “right”), automated transcription errors (“dh”), and frequent irrelevant concepts like months

or names. Furthermore, the Textblob package did not always create comprehensible Noun Phrases (such as “whole pdf file title email subject invoice number”, “copy pdf excepto people”). Further improvements to the NLP tooling improve the precision of the concept outputs.



**Figure 2:** Case study ontology fragment with emphasis on one known and one unknown concept.

A set of dashboard mock-ups that present the vision and outputs of the tool and the source code can be found at the Github Page of this project <sup>1</sup>.

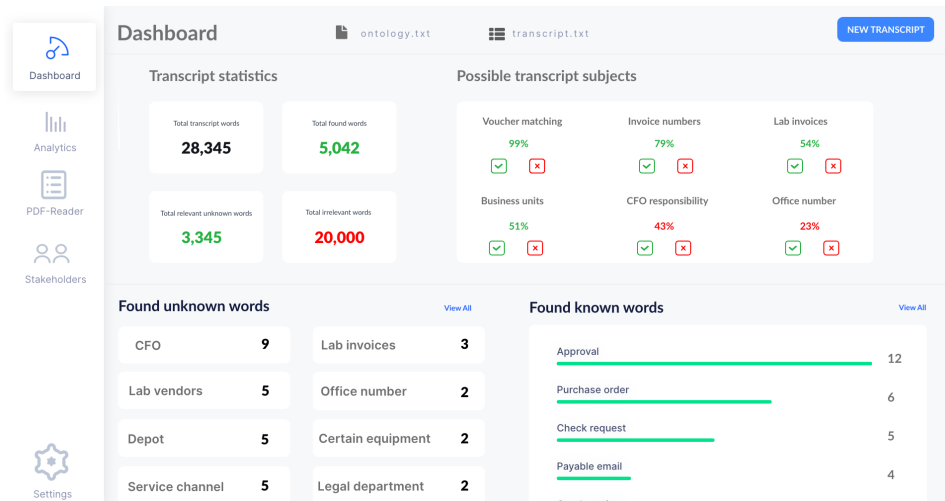
## 5.2. Domain Expert Validation

The results from the prototype were evaluated through four interviews with domain experts involved in different aspects of the software product. The evaluation sessions spanned approximately one hour and consisted of two parts; a presentation of the research plus outputs and a semi-structured interview. The domain experts were presented with outputs from transcripts of cases they were familiar with.

The experts were presented with the dashboard mock-up presented in Fig 3. Three experts mentioned the tool dashboard would be beneficial, and they would use a further developed prototype. One expert found it particularly useful that it could provide an objective view of the interview. Two of the experts mentioned that they did not recognize some of the unknown concepts, and would have liked to view them in context of the conversation. They mentioned it could mean something was missed in processing the session. Three experts mentioned that introduction of context functionality is crucial for the tool. This provides the ability to detect the location of concepts in the text version of a transcript.

While the feedback was mostly positive, one of the experts saw it only as a nice to have. They mentioned their current approach of using search functionality on the transcript can provide most of the value the tool would bring. The expert also argued that the analysis sessions might not be the best reference as other sources play a big part like product documentation, implementation guides, demos and presentations. Their view was that during analysis sessions, speakers could mention factually incorrect information leading to unreliable results. While we acknowledge the limitations of analysis sessions as a source, providing discussed positions on information can also expose communicated factually incorrect information. While the outputs are meant to aid the business analyst we still rely on their expertise regarding the product and domain.

<sup>1</sup><https://bowis.github.io/keyextractor/>



**Figure 3:** Dashboard mock-up of the tool outputs, and implied future functionality through the tabs.

For future improvements to the tool, the experts were prompted for their ideas and proposed:

1. Adding word count for each speaker to show the importance of certain requirements related to the expertise of the speaker.
2. Match concepts with the ontology visualization to provide a larger domain context.
3. Linking the concepts directly to their relevant transcript content.

## 6. Research Vision: From Concepts to User Story Generation

In order to investigate the vision of user story generation from the transcripts and discussed concepts, we performed a targeted search on four concepts within the transcripts. These concepts were expected to match to a configuration or customization requirement. For these, context was collected and we determined how we could (manually) transform these to a user story.

For the term CFO, we formulated the following requirement: “As a CFO, I want to approve invoices by email, so that I don’t have to use the ERP system” from a combination of input from the business analyst and the transcript segment. Reflecting on this requirement, we noticed that the “so that” part could not directly be traced back to the transcript, and was recreated from the memory of the analyst. Additionally, the following assumptions were made:

1. The current situation that was described, was also what was desired in the future situation upon use of the software product.
2. The available approval options in the software product were not satisfactory for the stakeholder.

For the other investigated concepts, the excerpts did not provide us with a sufficient amount of information to establish requirements that matched the actual requirements for the case. Many parts discussed the current process, and to generate requirements from these segments, we either need to assume that no change is desired, or rely heavily on the recollection of the analyst.

Therefore, we observe that the outputs can assist an analyst in formulating requirements and contain parts of the requirement information. However, further research into the source of the requirements and their presence in the transcripts is required. From the context, we observed that the transcripts themselves do not contain all the needed information to generate the requirements. For our tool, the

initial further development will focus on providing business analysts with excerpts for a specific concept to provide the context to aid in requirement formulation. The outputs can be eventually be used to play part in automated requirements generation.

## 7. Discussion & Conclusion

In this research we investigated concept extraction through a key abstraction NLP tool prototype applied to requirements elicitation transcripts. The aim of this approach is to decrease administrative burden of RE practitioners. The prototype enables exploration of the concepts discussed in requirements elicitation sessions thus jogging the minds of the business analyst and reducing chance of missed information.

We provide an answer to the research question: *“How can the ontology of a software product be utilized to recognize customization or configuration requirements in a elicitation session transcript?”* by extracting and categorizing the concepts discussed in a elicitation session. The concepts are compared to a software product ontology to differentiate between known and unknown concepts. The case study gives preliminary confirmation that this is useful information for RE practitioners. Furthermore the observations provide proof for the hypotheses that unknown concepts can indicate need for customization while known concepts can guide configuration of a software product.

While prototype results are promising, a number of additional crucial functionalities required to adequately support business analysts were discovered through the evaluation. Our results indicate that the developed tool is able to identify concepts that are known and unknown to an analyst. However, in the current version of the tool, irrelevant concepts need to be manually filtered out in order to obtain relevant results. Visualizing the found concepts in the form of a dashboard could aid an analyst in processing elicitation session transcripts as it provides a context of the concepts that are reflected in the transcript. Through future developments and usability improvements, the tool can also be used by staff other than analysts. This is useful for sharing knowledge and outsourcing certain tasks, which increases cost effectiveness. Building on this initial knowledge extraction from elicitation sessions we expect to provide valuable information to business analysts and can eventually be leveraged as foundation in an approach to automatically extract requirements from requirements elicitation sessions.

**Acknowledgements:** We would like to thank Fabiano Dalpiaz for his input and ideas for the tool design. And extend our gratitude to Richard van de Bospoort, Niels van der Helm, Ralitsa Bezlova, and Philip Noppen for their valuable feedback and validation of the designed prototype.

## References

- [1] H. Kitapci, B. Boehm, Formalizing informal stakeholder decisions—a hybrid method approach, in proc. of HICSS (2007).
- [2] A. Aurum, C. Wohlin, Requirements engineering: setting the context, in: Engineering and managing software requirements, Springer, 2005, pp. 1–15.
- [3] D. Zowghi, C. Coulin, Requirements elicitation: A survey of techniques, approaches, and tools, in: Engineering and managing software requirements, Springer, 2005, pp. 19–46.
- [4] M. G. Pitts, G. J. Browne, Improving requirements elicitation: an empirical investigation of procedural prompts, ISJ 17 (2007) 89–110.
- [5] L. Cao, B. Ramesh, Agile requirements engineering practices: An empirical study, IEEE Software 25 (2008) 60–67.



- [6] T. Berger, J.-P. Steghöfer, T. Ziadi, J. Robin, J. Martinez, et al., The state of adoption and the challenges of systematic variability management in industry, *ESE* (2020).
- [7] Z. Zhang, M. K. Lee, P. Huang, L. Zhang, X. Huang, A framework of ERP systems implementation success in china: An empirical study, *Int. J. of Production Economics* (2005) 56–80.
- [8] I. Shin, Adoption of enterprise application software and firm performance, *Small Business Economics* 26 (2006) 241–256.
- [9] T. M. Somers, K. Nelson, The impact of critical success factors across the stages of enterprise resource planning implementations, in: *Proc. of ICSS, IEEE*, 2001.
- [10] T. Spijkman, F. Dalpiaz, S. Brinkkemper, Requirements elicitation via fit gap analysis, in: *Proc. of CAiSE, Springer*, Forthcoming 2021.
- [11] A. Q. Ali, A. B. M. Sultan, A. A. Ghani, H. Zulzalil, A systematic mapping study on the customization solutions of software as a service applications, *IEEE Access* (2019).
- [12] D. Berry, R. Gacitua, P. Sawyer, S. F. Tjong, The case for dumb requirements engineering tools, in: B. Regnell, D. Damian (Eds.), *Proc. of REFSQ*, 2012, pp. 211–217.
- [13] A. Sutcliffe, P. Sawyer, Requirements elicitation: Towards the unknown unknowns, *IEEE*, 2013, pp. 92–104.
- [14] A. Bernstein, E. Kaufmann, Gino—a guided input natural language ontology editor, in: *International semantic web conference, Springer*, 2006, pp. 144–157.
- [15] E. Cambria, B. White, Jumping NLP curves: A review of natural language processing research, *IEEE CIM* 9 (2014).
- [16] J. Hirschberg, C. D. Manning, *Advances in natural language processing, Science* (2015).
- [17] G. Lucassen, F. Dalpiaz, J. M. E. van der Werf, S. Brinkkemper, Improving agile requirements: the quality user story framework and tool, *Proc. of RE* 21 (2016) 383–403.
- [18] R. Gacitua, P. Sawyer, V. Gervasi, On the effectiveness of abstraction identification in requirements engineering, in: *Proc. of IREC*, 2010.
- [19] L. Wang, H. Nakagawa, T. Tsuchiya, Opinion analysis and organization of mobile application user reviews., in: *Proc. of REFSQ Workshops*, 2020.
- [20] D. Martens, W. Maalej, Extracting and analyzing context information in user-support conversations on twitter, in: *Proc. of RE, IEEE*, 2019, pp. 131–141.
- [21] S. Abualhaija, C. Arora, M. Sabetzadeh, L. C. Briand, M. Traynor, Automated demarcation of requirements in textual specifications: a machine learning-based approach, *ESE* 25 (2020).
- [22] N. Kumar, M. Kumar, M. Singh, Automated ontology generation from a plain text using statistical and NLP techniques, *Int. J. Systems Assurance Engineering and Management* 7 (2016) 282–293.
- [23] C. Arora, M. Sabetzadeh, L. Briand, F. Zimmer, Automated extraction and clustering of requirements glossary terms, *TSE* (2016).
- [24] M. Uschold, M. Gruninger, et al., *Ontologies: Principles, methods and applications, University of Edinburgh AIAI Technical Report* (1996).
- [25] H. Kaiya, M. Saeki, Using domain ontology as domain knowledge for requirements elicitation, in: *Proc. of RE, IEEE*, 2006, pp. 189–198.
- [26] C. Riva, C. Del Rosso, Experiences with software product family evolution, in: *Proc. of IWPSE, IEEE*, 2003, pp. 161–169.
- [27] R. J. Wieringa, *Design science methodology for information systems and software engineering, Springer*, 2014.
- [28] P. Runeson, M. Host, A. Rainer, B. Regnell, *Case study research in software engineering: Guidelines and examples, John Wiley & Sons*, 2012.
- [29] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, *Experimentation in software engineering, Springer Science & Business Media*, 2012.