Requirements Engineering in OTF-Computing

Subjects: Others

Contributor: Frederik Bäumer, Michaela Geierhos

The idea of On-The-Fly (OTF) Computing is to compose and provide software services *ad hoc*, based on requirement descriptions in natural language. Since non-technical users write their software requirements themselves and in unrestricted natural language, deficits occur such as inaccuracy and incompleteness. These deficits are usually tackled by natural language processing methods, which have to face specific challenges in OTF Computing because maximum automation is the goal there. In the following, we present current automatic approaches for solving inaccuracies and incompleteness in natural language requirement descriptions.

Keywords: OTF Computing ; Natural Language Processing ; Requirements Engineering

1. Introduction

The idea of On-The-Fly Computing employs individual software requirements in natural language (NL) provided by users for an automatic composition of individual software services. Simply put, OTF Computing aims at providing tailored software services to individuals. Here, it is challenging that the requirements are given in NL and are accordingly partially incomplete, inconsistent or ambiguous. Since the bidirectional dialog between software developers and end users is omitted in the OTF Computing vision, new ways are needed to obtain missing information from end users regarding the desired software, which cannot be currently found in any existing linguistic resources[1]. Research as well as practical tools for NL requirement refinement are often dedicated to special domains or designed with different guidelines that do not require a fast computation or other OTF-typical standards. However, most approaches are developed for experts rather than for end users or an application in everyday life. Consequently, we want to draw attention to challenges of service description processing in the context of OTF Computing, where a very high degree of automation is required.

2. Natural Language Software Requirements in OTF Computing

Natural Language Processing (NLP) is crucial for the functionality of the OTF vision because, being the only possible form for service descriptions, we have to tackle NL shortcomings while finding and making use of as much information as possible from given NL input. Nevertheless, the individual factors such as the user's knowledge or some expert level know-how can shape a requirement and its accuracy as input for other processes. This leads to service descriptions being characterized as user-generated, informal documents[2]. However, since formal specification languages - even in a weaker form - are not applicable for non-technical end users, we have to face typical NL challenges, among which are a lack of structure and correctness, some grammar and spelling errors as well as occasional ambiguity issues. This is still a tremendous difference compared to the usefulness of formal specification methods already available[3].

2.1. Extraction of Canonical Core Functionalities

It is a challenging task to find important and useful information in NL service descriptions because they are unstructured and contain off-topic information[4]. Apart from that, canonical core functionalities are neither provided in an order nor easy to extract. However, relations among requirements are important in order to figure out which requirements determine others. These steps specify which function an application might have. This requires extracting all specified functionalities and ordering them functionally. The state-of-the-art puts the functions in a temporal or causal order, a way the user might have intended[5].

2.2. Automatic Detection and Compensation of Inaccuracy

In OTF Computing, issues such as ambiguity or inaccuracy have to be tackled due to the lack of expert knowledge on the end users' side. Therefore, we need NLP tools that recognize various types of inaccuracies on the one hand and identify linguistic characteristics leading to inaccuracy on the other hand. These tools have to be able to resolve as many deficits

as possible in the NL descriptions independently. Furthermore, a kind of knowledge base (in the form of linguistic resources) is required, which are quite rare for the OTF Computing domain.

3. Natural Language Processing Approaches

In the following, an insight into the research in the domain of canonical core functionality extraction as well as automatic inaccuracy compensation and detection is given.

3.1. Requirements Extraction

Up to now, there is little research done on extracting software requirements. For example, there is a tool named REaCT[5], which makes use of learning procedures to find phrases belonging to a certain topic. It works on textual requirement descriptions and tries to transfer the most important entities for functional requirements found in a template. From the technical side, this involves dividing description texts into sentences and classifying them into off- and on-topic components. After this step, from the on-topic sentences containing functional requirements, there are attribute-value pairs extracted in order to iteratively fill the template, starting with the most important elements such as subjects, actions, predicates and objects (e.g. indirect objects). Next to this, the idea of extracting requirements without machine learning but rule-based is raised. At any rate, due to the possibly low quality of descriptions, there is still enough to be done. Other approaches focus on high-quality requirements or assume getting high-quality texts, which makes such tools unsuitable for the OTF Computing domain. Apart from that, there exists a study of unstructured and informal requirement descriptions from the Open Source domain[6].

3.2. Multiple Inaccuracy Detection and Compensation

In the following, combined approaches for the detection and compensation of ambiguity and incompleteness are presented. Well-known tools are QuARS[7] and QuARSexpress[8], which can deal with a broad range of inaccuracies, while NL2OCL and SR-Elicitor are tools that reach a low coverage. QuARS is supposed to detect a high number of issues in requirement descriptions, while NL2OCL[9] and SR-Elicitor[10] should detect and compensate issues fully automatically. Another tool, called RESI (Requirements Engineering Specification Improver), is based on a high degree of user interaction when compensating inaccuracies. Here, we want to draw attention to three solutions that we describe in more detail for a better understanding. These tools are NLARE (Natural Language Automatic Requirement Evaluator)[11], RESI[12] and CORDULA (Compensation of Requirements Descriptions Using Linguistic Analysis)[13]. Furthermore, we highlight the discrepancies here. NLARE is a hybrid approach with focus on functional requirements and the detection of ambiguity, incompleteness and atomicity. Among other things, the software employs an atomicity criterion that basically sets the rule that a single sentence must contain a single requirement. Apart from that, incompleteness is seen as to complement information dealing with "W-questions": "Who", "What", "Where", "When." The authors regard ambiguity as given when adverbs and adjectives occur that can be de- or increased. NLARE makes use of regular expressions in order to process NL data. NLARE further applies spelling correction, detects sentence boundaries and tokenizes words. There is no assistance or compensation of inaccuracies. However, there are other tools: RESI has a different aim because it enables (and encourages) user interaction while being flexible and also dealing with linguistic defects. RESI understands requirement specifications as a graph, where it automatically identifies inaccuracies. Each inaccuracy is solved within a dialog between the user and the system.

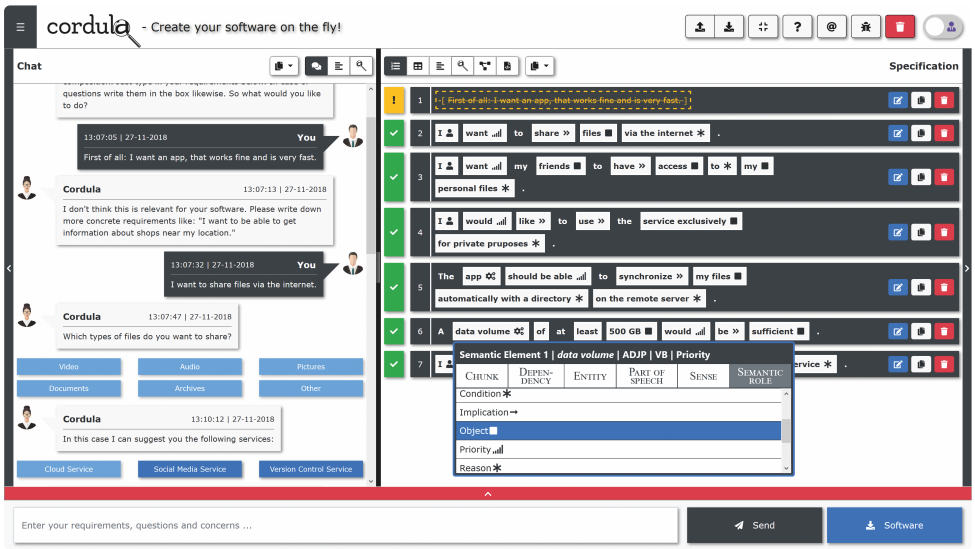

Figure 1. CORDULA tool.

Moreover, there is a tool called CORDULA (cf. Fig. 1), which is in some cases similar to RESI. CORDULA is able to find and compensate "language deficiencies" such as ambiguity, vagueness and incompleteness in written requirement texts produced by end users. Thus, CORDULA is well suited for the OTF Computing domain, for which it was also designed and developed. Furthermore, CORDULA enables users to find suitable software and services that can be used to generate canonical core functionalities from service and requirement description texts. Predefined linguistic features, used as indicators, enable the system to improve text quality individually. This approach is data-driven and aims at current needs as it is driven by a typical text analysis pipeline. However, CORDULA has a considerable disadvantage for the OTF Computing domain: Its slow execution time.

4. Research Outlook

Because of the specific nature of OTF Computing, NLP approaches developed so far do not achieve the required execution times and compensation quality. Therefore, there is a lot to be done: Attention still needs to be paid to the development of methods for extracting requirements as well as to the detection and compensation of inaccuracies. This raises other issues such as the lack of resources but also the lack of interoperability of individual compensation components, ways of efficiently involving end users without overburdening them, and much more. Here, existing as well as future techniques from the Semantic Web research can be applied soon, in particular the Linked (Open) Data techniques. Furthermore, work on modern chat technology to conduct targeted communication with end users as part of the compensation steps is needed. Through this procedure, missing information can be requested and the user is supported with examples. In addition, the results generated during the compensation process can be explained during the offered chat, which increases the usability for end users.

References

1. Geierhos, M.; Bäumer, F.S. How to Complete Customer Requirements: Using Concept Expansion for Requirement Refinement. Proceedings of the 21st NLDB; Métais, E.; Meziane, F.; Saraee, M.; Sugumaran, V.; Vadera, S., Eds.; Springer: Manchester, UK, 2016.
2. Moens, M.F.; Li, J.; Chua, T.S., Eds. Mining User Generated Content; CRC Press: Leuven, Belgium / Beijing, China / Singapur, 2014.
3. Platenius, M.C.; Josifovska, K.; van Rooijen, L.; Arifulina, S.; Becker, M.; Engels, G.; Schäfer, W. An Overview of Service Specification Language and Matching in On-The-Fly Computing (v0.3). Technical Report Tr-ri-16-349, Software Engineering Group, Heinz Nixdorf Institut, Paderborn University, 2016.
4. Geierhos, M.; Schulze, S.; Bäumer, F.S. What did you mean? Facing the Challenges of User-generated Software Requirements. Proceedings of the 7th ICAART; Loiseau, S.; Filipe, J.; Duval, B.; van den Herik, J., Eds.; SCITEPRESS – Science and Technology Publications: Lisbon, Portugal, 2015; Special Session on PUaNLP 2015, pp. 277–283.
5. Dollmann, M.; Geierhos, M. On- and Off-Topic Classification and Semantic Annotation of User-Generated Software Requirements. Proceedings of the Conference on EMNLP; ACL: Austin, TX, USA, 2016.
6. Vlas, R.; Robinson, W.N. A Rule-Based Natural Language Technique for Requirements Discovery and Classification in Open-Source Software Development Projects. Proceedings of the 44th HICSS; IEEE: Kauai, HI, USA, 2011; pp. 1–10.
7. Lami, G. QuARS: A Tool for Analyzing Requirements. Technical Report ESC-TR-2005-014, Carnegie Mellon University, 2005.
8. Bucchiarone, A.; Gnesi, S.; Fantechi, A.; Trentanni, G. An Experience in Using a Tool for Evaluating a Large Set of Natural Language Requirements. Proceedings of the 2010 ACM Symposium on Applied Computing; ACM: New York, NY, USA, 2010; SAC'10, pp. 281–286.
9. Bajwa, I.S.; Lee, M.; Bordbar, B. Resolving Syntactic Ambiguities in Natural Language Specification of Constraints. In Computational Linguistics and Intelligent Text Processing; Gelbukh, A., Ed.; Springer: Berlin / Heidelberg, 2012; Vol. 7181, LNCS, pp. 178–187.
10. Umber, A.; Bajwa, I.S. Minimizing Ambiguity in Natural Language Software Requirements Specification. Proceedings of the 6th ICDIM; IEEE: Melbourn, VIC, Australia, 2011; pp. 102–107.
11. Huertas, C.; Juárez-Ramírez, R. NLARE, a Natural Language Processing Tool for Automatic Requirements Evaluation. Proceedings of the CUBE International Information Technology Conference; ACM: New York, NY, USA, 2012; CUBE'12, pp. 371–378.
12. Körner, S.J.; Brumm, T. Natural Language Specification Improvement with Ontologies. International Journal of Semantic Computing 2010, 03, 445–470.

13. Bäumer, F.S.; Geierhos, M. Flexible Ambiguity Resolution and Incompleteness Detection in Requirements Descriptions via an Indicator-based Configuration of Text Analysis Pipelines. Proceedings of the 51st Hawaii International Conference on System Sciences, 2018; pp. 5746 – 5755.