A Vision of Understanding the Users' View on Software

Hendrik Schrieber[a], Michael Anders[b], Barbara Paech[b] and Kurt Schneider[a]

[a] *Leibniz University Hannover, Welfengarten 1, 30167 Hannover, Germany*
[b] *Ruprecht-Karls-University Heidelberg, Im Neuenheimer Feld 205, 69190 Heidelberg, Germany*

Abstract
Requirements Engineering is focused on eliciting, specifying and validating what customers and potential users say about the software they need. However, we have only little insight how users talk about software; this insight comes from short utterances in user forums and app stores. How do they describe the features of existing software? What are the terms and concepts they use? What do they have in mind and associate with the software? Is it the user interface, underlying functionality, or domain issues? We stipulate that an improved understanding of the user language will also improve the communication between users and developers about new features. Therefore, we propose to study comprehensive user utterances, and to define the so-called user view language. It comprises the concepts and relationships with which users describe their view of software. In this paper we describe our vision of the user view language. Then, we describe planned studies on user utterances and the challenges we see in investigating these utterances with natural language processing techniques. Our data provides an interesting testbed for such techniques and their combination.

Keywords[1]
natural language processing, requirements engineering, user language analysis, user view

1. Introduction: Searching for the User View Language

In requirements engineering (RE), customers and potential users talk about the software they need or envision. The lack of a common language between stakeholders and developers is one of the early key challenges of requirements discovery [1]. When people talk using different terminologies, this is often an indication of conflicting and inconsistent views of the world and in particular of software. This lack is one of the main reasons for misunderstandings and for software that does not meet customer expectations.

Still, we know very little about how users comprehend and explain existing software ("what software does"). Description formats like user stories or use cases have been studied extensively in the context of RE. They have proven useful in practice for specifying requirements, as shown e.g. in [2]. In that study, however, users were not asked how comfortable they were with those techniques that capture software behavior. In many cases in practice, user stories and use cases are not created by users themselves but by developers as an intermediate artifact to support communication. Original user view and user talk are not captured.

In our work, we distinguish the *inside view* of software which encompasses the components and their composite behavior, and the *outside view* of software which encompasses how software is perceived from the outside, e.g. its features or the user interface.

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: hendrik.schrieber@inf.uni-hannover.de (H. Schrieber); michael.anders@informatik.uni-heidelberg.de (M. Anders); paech@informatik.uni-heidelberg.de (B. Paech); kurt.schneider@inf.uni-hannover.de (K. Schneider)

Usually, only the outside view is relevant for users. Therefore, we also call it *user view*. The software engineering community has developed a standardized set of concepts and relationships to talk about the *inside view* of software, in particular the unified modeling language (UML). *But we do not have a standardized set of concepts and relationships to talk about the outside view of software.* In the "User View Language" (UVL) project[2], we want to develop such a UVL and find out which set of concepts and relationships together with an adequate textual and visual notation it should include. This language can then be used to guide users and developers in their communication.

We want to derive UVL by studying original user utterances. In RE, we get to see and exploit original user utterances in the form of user feedback in user forums and app stores. Both contain short and purely textual utterances, often poorly written. There is a lot of research on the mining of such channels [3]. A recent taxonomy shows that product quality, user intention, user experience and sentiment are classified, but not what kind of concepts are used by a user to describe functional or quality features related to all these aspects [4]. Along the same lines, a recent ontology for user feedback focuses on communication in terms of speech acts, but not on their contents [5]. As the utterances are short and only describe individual features, they are not suited to study the comprehension of the software by the users and the concepts and relationships used to express this.

Thus, to achieve our vision of UVL we first need to tackle the challenges of acquiring comprehensive user utterances and of developing adequate natural language processing (NLP) techniques to study them. We focus on NLP4RE techniques that means NLP techniques which have proven useful in the RE context. In the following, we describe (a) our data set, (b) how we want to study it, and (c) which NLP4RE support we need. In order to illustrate this support we discuss the most prominent NLP4RE techniques based on a recent mapping study [6].

2. Goals and Data for Studying User Utterances

We are interested in comprehensive oral or textual user utterances about existing software. In particular we want to study utterances which focus on the software as a whole, for example with different intentions (description or explanation) and in different situations (learning or using software). We are not aware of freely available data sets of this kind. Therefore, we want to acquire this set in the context of a big health project in Heidelberg[3] where several hundred older adults will participate in using and enhancing health technologies over a period of 2 years.

As described in [7], we have several assumptions about the user view. First, users and developers have different mental models of software. Second, users use a lot of domain concepts and terminology, as opposed to generic concepts. There are, nevertheless, also several generic concepts across the domains. Third, users refer to user interface elements, the user tasks, as well as data, actions and quality to talk about software. They do not use many concepts from the inside view. Our core interest for UVL are the generic concepts. Generic concepts can be applied to any software in any context, while domain concepts are specific to a software or an application domain. As UVL should be applicable to any software, its concepts will be generic This will be complemented with domain specific concepts the same way as UML can be adapted to specific domain contexts.Therefore, we want to answer three main research questions:

- RQ1: Which concepts or relations do users use when they describe software?
- RQ2: How are these concepts described by the users?
- RQ3: Which generic concepts about software and its use do users use?

UVL should build on concepts and relations used (RQ1). RQ2 is important to avoid misunderstandings in UVL, e.g. with synonyms and homonyms, examples or metaphors. RQ3 is

[2] https://www.pi.uni-hannover.de/de/se/forschung/projekte/forschungsprojekte-detailansicht/projects/uvl/

[3] https://www.uni-heidelberg.de/en/newsroom/carl-zeiss-foundation-funds-interdisciplinary-practice-study-at-heidelberg-university-with-the-sum

important to define concepts and relationships in UVL so that users cover a variety of software aspects in their utterances, e.g. referring to user interface as well as to functional and quality features.

As the main data set will only evolve starting in 2022, we develop our ideas with a preliminary data set. We conducted exploratory interviews with students from different fields of study at two university locations. For diverse views we included students from non-technical fields as well as computer science students. The software we selected for this study were two e-learning platforms widely used at the participating universities: For Hannover, this is Stud.IP[4] and for Heidelberg it is Moodle[5]. Interviews were conducted online. Participants were asked to verbally describe, what the software is and what it does. To motivate them to give a thorough explanation, we asked them to imagine that they were explaining the software to their grandparents.

In the following, we describe how we analyzed this interview data manually in order to illustrate what kind of NLP support we need. The interviews were analyzed using Grounded Theory analysis in the general sense of [8]. First, we transcribed the video recordings and split the interview text into phrases, e.g. "you can download files". We define a phrase as a linguistic entity of arbitrary length usually expressing a single thought or notion [9]. Each phrase was split up again into smaller sub-phrases where a sub-phrase represents exactly one concept. In the example above, sub-phrases are "files" and "download". We derived the concepts in a bottom-up approach to capture what we think the users talk about (RQ1). For the concepts we chose terms that had been used by the students in their utterances, e.g. "file", to stay as close to their mode of expression as possible. Concepts were grouped with similar concepts, for example "upload" and "download" could form a group. Furthermore, we identified synonyms or metaphors to understand how students name an aspect of the software (RQ2). A student could refer to the concept "file" by using either the term "file" or "data file". Additionally, we also coded special aspects such as examples ("lecture 1 here, for example"), metaphors ("participants can meet in a virtual room"), terms addressing quality ("super structured") and judging terms ("then we have a – very important – search bar").

As a complement to the bottom-up approach of finding concepts, we classified phrases in a top-down approach starting from the software engineering perspective (RQ3). We used a simplified version of the TORE model which is a framework for *Task- and Object-oriented Requirements Engineering* [10]. The TORE model in general is used to assign requirements to one of four different abstraction levels: task level, domain level, interaction level and system level. In our TORE adaptation we merged task and domain level, since our primary interest is whether user utterances are rather specific to the software or to the domain. Furthermore, TORE provides categories for each level,

- domain and task level: tasks, activities, and entities
- interaction level: functions, interactions (such as use cases), abstract GUI concepts (e.g. workspace)
- system level: specific GUI concepts (e.g. buttons) and system internals

Thus, we can distinguish different generic concepts within each category (RQ3). We assigned the sub-phrases we identified before to a TORE category such that each concept is mapped to a category of the TORE model in the given context of a phrase. Within the context of the phrase "you can view the courses you are enrolled in", we map the concept "to view" to the interactions category, and the domain concepts "course" to the entities category and "to enroll in" to activities. By explicitly formulating what we did in the mapping process and verifying this with two researchers we found a set of rules to achieve reproducible results for the mapping process.

From the descriptions above one can see that we need support for the following NLP4RE tasks:

- pre-processing text into sub-phrases
- deriving concepts and relations from sub-phrases (including homonyms and synonyms)
- classifying these concepts and relations according to the TORE categories and levels

[4] https://www.studip.de/

[5] https://moodle.org/

- identifying more fine-grained aspects of expressions like examples and metaphors.

In the next section, we discuss how this fits to the current state of the art in NLP4RE.

3. NLP4RE Techniques for Studying User Utterances

For this section we build on the very helpful mapping study by Zhao et al. [6] about NLP for RE. In figure 9 of this study we can see that from 404 papers only 4 papers study interview scripts as direct user utterances. They do not evaluate their techniques on real life utterances. So, there is very limited previous work on NLP for comprehensive user utterances. Concerning the NLP techniques we can see the following

- As shown in figure 16 of Zhao et al. [6], the most frequently used NLP4RE techniques focus on pre-processing such as POS-tagging, parsing, stop word removal. Thus, general pre-processing will not be a problem.
- For concept and relationship identification and TORE-classification, techniques for the three NLP4RE tasks extraction, modeling and classification will be useful. Below we discuss the challenges we see for our tasks.
- The identification of more fine grained aspects will require more advanced techniques from computer linguistics.

As described in the previous section, we want to identify concepts and relationships as well as the TORE-classification. So one major question is whether it is better for automated NLP to first identify the TORE-categories and -levels and then identify the concepts within the categories and levels, or the other way around.

Independent of this general question, in the following we sketch the most frequently used NLP4RE techniques for semantic tasks and their ranks in [6]. As mentioned in [6] so far mostly word-based techniques are used. On the one hand concept identification can be supported by term extraction (rank 5), TF-IDF to extract important words (rank 9) or frequency analysis (rank 18). We are aware that there is more specific work on abstraction identification in requirements documents, e.g. [11]. However, this has not been applied to user utterances and focused on the domain, whereas we focus on the software aspects. For a more advanced analysis of the relationships semantic role labeling (rank 16) and semantic analysis (rank 19) can be used. However, it will be difficult to distinguish the domain concepts and relationships from the domain-independent concepts and relationships. On the other hand machine learning based classification (rank 14), keyword search (rank 21) or clustering (rank 26) could be used to identify TORE-categories. Keyword search is very restrictive (pre-supposing a certain set of keywords instead of openly exploring the user utterances) and for clustering it will be difficult again to distinguish domain-dependent clusters from domain-independent ones. Thus, - in line with the approaches on user feedback classification collected in [12] - it seems that machine learning techniques, possibly including word embedding (rank > 32), are needed in addition. As mentioned in [6] this requires large annotated data sets.

4. Conclusion

The development of a user view language is an important step for RE. It requires a large scale study of user utterances with NLP. This is beyond the usual NLP4RE studies, because long unstructured original user utterances such as interviews have rarely been studied. Furthermore, in contrast to typical NLP4RE papers, several NLP4RE tasks (such as extraction, modeling, and classification) have to be synchronized and a distinction between domain-dependent concepts and relationships and generic ones for software use is required. Thus, our vision of UVL provides a challenge and a testbed for NLP4RE.

5. References

[1] I. F. Alexander and L. Beus-Dukic, Discovering requirements: how to specify products and services, John Wiley & Sons, 2009.

[2] G. Lucassen, F. Dalpiaz, and JME. van der Werf, The Use and Effectiveness of User Stories in Practice, Int. Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ), LNCS vol. 9619, pp. 205–222, Springer, 2016.

[3] D. Pagano, and W. Maalej, User feedback in the appstore: An empirical study, 21st IEEE international requirements engineering conference (RE), pp. 125-134, IEEE, 2013.

[4] R. Santos, E. C. Groen, and K. Villela, A taxonomy for user feedback classifications, REFSQ Workshops, CEUR Workshop Proc., vol. 2376, 2019.

[5] I. Morales-Ramirez, F. M. Kifetew, and A. Perini, Speech-acts based analysis for requirements discovery from online discussions, Inf. Syst., vol. 86, pp. 94–112, 2019.

[6] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E. V. Chioasca, & R. T. Batista-Navarro, Natural Language Processing (NLP) for Requirements Engineering: A Systematic Mapping Study, 2020. arXiv:2004.01099.

[7] B. Paech and K. Schneider, How Do Users Talk About Software? Searching for Common Ground, 1st Workshop on Ethics in Requirements Engineering Research and Practice (REthics), Zurich, Switzerland, pp. 11-14, 2020. doi: 10.1109/REthics51204.2020.00008.

[8] P. Runeson, M. Host, A. Rainer, and B. Regnell, Case Study Research in Software Engineering: Guidelines and Examples,: John Wiley & Sons, 2012.

[9] OED Online, phrase, n., Oxford University Press, December 2020. URL: https://www.oed.com/view/Entry/142933?isAdvanced=false&result=1&rskey=gTLp6U&.

[10] B. Paech, K. Kohler, Task-Driven Requirements in Object-Oriented Development, in: J.C.S. do Prado Leite, J.H. Doorn (Eds.), Perspectives on Software Requirements. The Springer International Series in Engineering and Computer Science, vol 753, Springer, pp. 45–68, 2004. doi:10.1007/978-1-4615-0465-8_3.

[11] R. Gacitua, P. Sawyer & V. Gervasi, Relevance-based abstraction identification: technique and evaluation, Requirements Eng 16, 251, 2011. doi:10.1007/s00766-011-0122-3.

[12] R. Santos, E. C. Groen and K. Villela, An Overview of User Feedback Classification Approaches, REFSQ Workshops, CEUR Workshop Proc., vol. 2376, 2019.