Research on NLP for RE at Utrecht University: A Report

Fabiano Dalpiaz
RE-Lab, Utrecht University
Utrecht, The Netherlands
f.dalpiaz@uu.nl

Sjaak Brinkkemper
RE-Lab, Utrecht University
Utrecht, The Netherlands
s.brinkkemper@uu.nl

Abstract

[Team Overview] The Requirements Engineering Lab at Utrecht University conducts research on techniques and software tools that help people express better requirements in order to ultimately deliver better software products. *[Past Research]* We have focused on natural language processing-powered tools that analyze user stories to identify defects, to extract conceptual models that deliver an overview of the used concepts, and to pinpoint terminological ambiguity. Also, we studied how reviews for competing products can be analyzed via natural language processing (NLP) to identify new requirements. *[Research Plan]* The gained knowledge from our experience with NLP in requirements engineering (RE) triggers new research lines concerning the synergy between humans and NLP, including the use of intelligent chatbots to elicit requirements, the automated synthesis of creative requirements, and the maintenance of traceability via linguistic tooling.

1 Team Overview

The Requirements Engineering Lab (RE-Lab) is a research group in the Department of Information and Computing Sciences at Utrecht University. The RE-Lab members are involved in several research directions with the common objective to help people express better requirements in order to ultimately deliver better software products. The lab's research method involves the use of state-of-the-art, innovative techniques from various disciplines (computer science, logics, artificial intelligence, computational linguistics, social sciences, psychology, etc.) and to apply them to solve real-world problems in the software industry.

We study *software products* and have substantial experience in market-driven RE [RB05]. We closely collaborate with a network of companies in the Netherlands who provide empirical data about their software products. Many students at the master's and Ph.D. level spend considerable time in those companies and conduct experiments and case studies to validate their research artifacts. The RE-Lab maintains and evolves its network through higher education and professional training about requirements engineering and software product management.

The main themes of the RE-Lab—with or without the use of NLP—include, from the most to the least recent (i) interactive tools to foster participation in RE, e.g., via gamification [LDLB16]; (ii) crowd-based requirements engineering [GSA+17]; (iii) agile requirements engineering via user stories [LDvdWB16c]; (iv) software product management, including next release planning [ADB+18] [ZHO+18]; and and (v) requirements modeling languages, especially goal-oriented ones like the iStar 2.0 language [DFH16].

2 Research on NLP in the RE-lab

Before the RE-lab was established, the researchers had considerable experience in applying linguistic techniques various settings [BBM88] [KBHW97] [oDGBR05]. The improved quality of NLP tooling becoming available for research and industry motivated us to study the application of these tools for RE research questions. We organize the discussion of research on NLP for RE in our lab into multiple categories; within each category, the work is presented in chronological order.

2.1 Authoring high-quality user stories with QUS and AQUSA

Triggered by the increasing popularity of user stories as a simple language for expressing requirements in agile development contexts [LDvdWB16a], we studied whether the linguistic formulation of user stories would affect the quality of the resulting software products. The main research artifacts from this endeavor are the Quality User Story (QUS) framework [LDvdWB16c] and the Automatic Quality User Story Artisan (AQUSA) tool[1]. QUS is a collection of heuristic rules for authoring high-quality user stories. The AQUSA tool complements QUS by automatically checking a collection of user stories against some of these heuristics; in technological terms, AQUSA is a Python tool that makes use of regular expressions and part-of-speech tagging as delivered by the Natural Language Toolkit (NLTK) library.

The heuristics of QUS are organized into three categories: (i) *syntactic*: on the textual structure of a user story regardless its meaning; (ii) *semantic*: on the meaning of the user story; and (iii) *pragmatic*: on how to effectively communicate the requirements through a collection of user stories. AQUSA is able to check syntactic and pragmatic criteria for which we envisioned it is possible to achieve 100% recall with good-enough precision.

Consider the syntactic rule *atomic*, which postulates that a user story shall express a requirement for exactly one feature. AQUSA detects violations to atomicity by looking for conjunctions like 'and', '&', 'or', etc. within a user story. An example of a non-atomic story is "As a user, I want to print and save my results", because of the use of the 'and' conjunction. Consider the pragmatic rule *uniform*, stating that all user stories in a collection shall follow roughly the same template. AQUSA checks uniformity by inferring the most frequent template (e.g., "As a, I want to, so that"). If a user story exists in the collection that has a different template, such as "As a, I'm able to, so that", AQUSA would indicate a uniformity defect for that set of user stories.

We evaluated the precision and recall of AQUSA on 18 user story data sets from the software industry—mostly, Dutch software product companies writing their requirements in English—, which included 1023 user stories in total [LDvdWB16c]. AQUSA could test the criteria atomic, minimal, well-formed, uniform, and unique. The overall precision was 72.2% and the overall recall was 93.8%. Although far from the 100% recall objective, these results prompted us to make some changes to AQUSA to cover some cases that had not been encountered yet.

After extensive interaction with some industrial partners from our network, we set off to investigate the effectiveness of QUS and AQUSA on productivity and work deliverable quality in a longitudinal study. We involved three companies, with a total of thirty practitioners using our approach, over a period of two months [LDvdWB17]. Besides improving AQUSA based on earlier results, we offered it as a service that could be integrated with the project management platform Jira.

Our study included a baseline measurement of project management variables like recidivism rate, velocity, and number of issues; a training session for the participants; data collection during the experimental period; intake and exit surveys; and follow-up interviews. Despite an improvement of the intrinsic user story quality, the practitioners did not perceive such a change, although they argued that more constructive user story conversation took place after the treatment was introduced. Project management metrics, on the other hand, did not show any improvement [LDvdWB17].

2.2 From user stories to conceptual models with Visual Narrator

When the number of user stories grows, as typical for software products that are developed through multiple releases, it becomes difficult to determine the key terms that are being used, and to get an overview of the entities and relationships that the requirements refer to. With this problem in mind, we proposed the Visual Narrator[2], a tool that automatically extracts conceptual models from a collection of user stories [LRD+17].

The Visual Narrator uses a selection of heuristics from the literature for the extraction of conceptual models from domain descriptions, and adapts them to the context of user stories; our expectation was that the extraction

[1] `https://github.com/RELabUU/aqusa-core`
[2] `https://github.com/MarcelRobeer/VisualNarrator`

process would be facilitated by the well-defined structure of the user stories. For example, the story "As a visitor, I want to choose an event, so that I can book a ticket for that event" would result in a conceptual model with three entities: visitor, event, ticket; and two relationships: choose(visitor, event) and book(visitor, ticket). The heuristics are implemented via a combination of regular expressions and part-of-speech tagging; we employed the libraries offered by the spaCy toolkit.

After a preliminary evaluation that resulted in an improved version of the tool, the accuracy of the Visual Narrator has been tested on four industrial cases [LRD+17] and delivered very good results in the 85%–95% range both for precision and recall. The major NLP challenges we encountered concerned the interpretation of compound words, the ability to link the verb to the right object, and human mistakes like grammar errors and typos. While the latter issue can be addressed via grammar checkers, the first two challenges require either advances in NLP toolkits, or their training on domain-specific corpora. Practitioners found the tool especially useful for getting newcomers acquainted with the product domain.

Further experience soon highlighted one major issue: while the tool was quite effective at extracting entities and relationships, their number would grow quickly as the size of the user story collection increases, thereby making the a box-and-arrow visualization of the concepts too cluttered. In other words, we were shifting the complexity from the textual representation to the graphical model. To cope with this, we resorted to principles from information visualization and proposed new, interactive ways to interact with the extracted models.

First, we developed a tool that employed semantic similarity to couples of the extracted concepts, and used clustering algorithms to groups the concepts based on their similarity [LDvdWB16b]. The user could then decide whether to see all concepts or only the clusters. Second, we built the Interactive Narrator, a web application for navigating conceptual models through a set of filters (e.g., by role, by sprint, etc.) [SDBL18]. As we are writing, the latter tool—although less sophisticated—is our preferred choice when delivering trainings about our approach, for the similarity+clustering algorithms do not perform very well in absence of domain knowledge.

2.3 Terminological ambiguity in requirements

Our experience with tools like the Visual Narrator and the Interactive Narrator have also pointed out that, especially for software products that evolve over the years through multiple releases and development teams, it is hard to ensure the use of consistent terminology. For example, two different teams may be using the verbs "browse" and "navigate", but are they really interchangeable, or is there a subtle difference in the terms?

We have investigated *terminological ambiguity*: those situations in which near-synonyms are used, and the reader cannot tell whether the two words do actually represent the same concept. Our research first led to the REVV tool [DvdSL18], which takes as input the outputs of the Visual Narrator, and creates a Venn diagram-inspired visualization (see Fig. 1) where roles are represented as sets, the extracted entities from a user story are denoted as elements in a set, and those elements are colored based on the likelihood that they are terminologically ambiguous with another term. To do this, besides the NLP embedded in the Visual Narrator, REVV measures the semantic similarity between two terms via an off-the-shelf technique called semantic fingerprinting [DSW15].

To test the effectiveness of REVV, we have built an improved version called REVV-Light[3] [DvdSB+19] and conducted a controlled quasi-experiment with 28 real-world data sets (2067 user stories in total). We tested whether human analysts assisted by REVV-Light would be better than manual taggers in identifying potential terminological ambiguities. The comparative results were somewhat disappointing: in a time-constrained setting, human taggers performed slightly better, also due to the low maturity of our tool and the high usability expectations of the participants.

Among our main findings, it became manifest that analyzing terminological ambiguity is time consuming, and that recognizing true ambiguity requires domain knowledge to avoid ill-informed guesses. In particular, the off-the-shelf semantic fingerprinting technique showed limitations that can be overcome only by re-training the algorithm with domain-specific data. REVV-Light should therefore be seen a first example of a tool that builds on the synergy between NLP and human analysis.

2.4 New requirements from competing products through user feedback mining

An increasing number of researchers are proposing algorithms for processing app store reviews that extract features, determine the sentiment in the reviews, etc. Especially for products with a large use base, app reviews are a rich, real-time source of user needs and wishes. We recently entered this arena with a study of how to

[3] https://github.com/RELabUU/revv-light

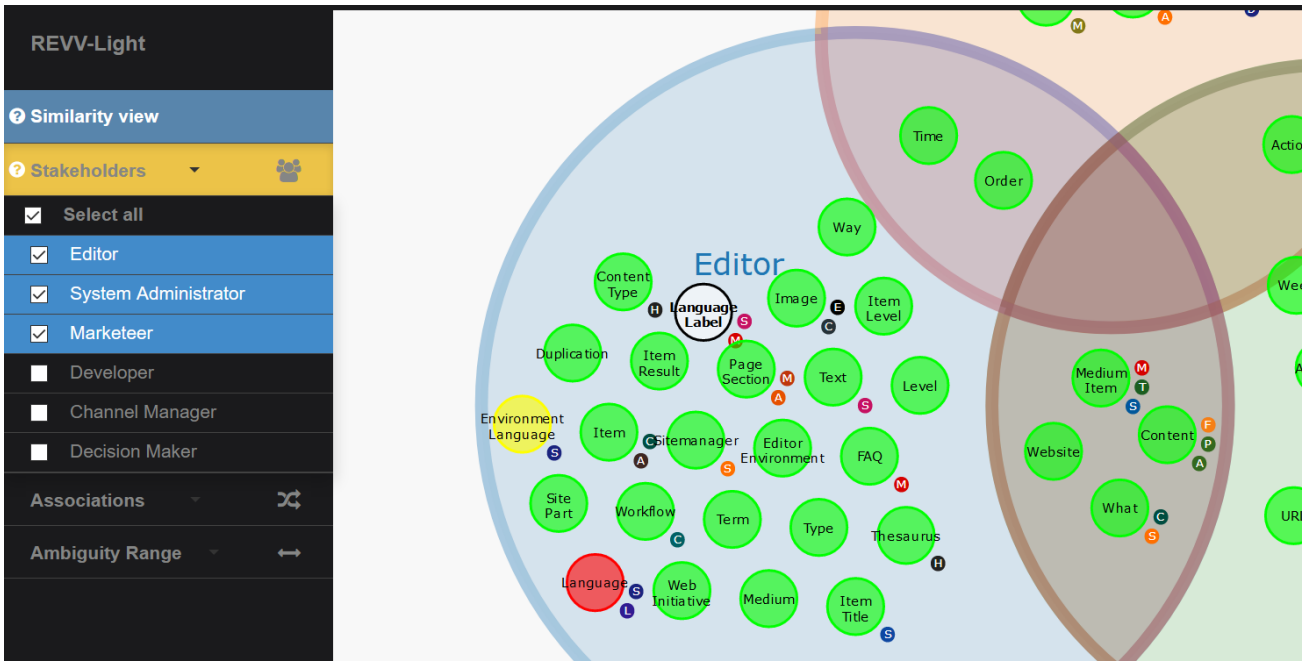

Figure 1: The REVV-Light tool highlights possible terminological ambiguity in a user story data set.

identify requirements from competitive products. Thus, instead of analyzing the product in isolation and with respect to a hypothetical fully satisfied user base, we are investigating the product in comparison to others.

Our approach is called RE-SWOT[4] [DP19], and it automatically converts app reviews to a dashboard through the use of NLP. RE-SWOT is inspired by the famous Strength-Weakness-Opportunity-Threat analytical framework that is used by organizations to compare themselves, or their products, to the competitors in the market. The idea is similar; through an automated analysis of the reviews for a reference app and its competitor apps, RE-SWOT identifies the main features of the apps within the same market and determines whether those features are strengths, weaknesses, opportunities, or threats from the perspective of the reference app. For example, a feature is a strength when the average sentiment for that feature as implemented by the reference app is positive, and when such a value is higher than the average sentiment for that feature in the competing apps.

Under the hood, RE-SWOT uses two NLP algorithms. First, it uses collocations and regular expression matching to identify user reviews that are likely to refer to features. Second, it employs sentiment analysis to determine whether the average sentiment toward that feature. These outputs are used to perform the SWOT classification, which is then rendered graphically as a dashboard that clearly distinguishes the four categories and also highlights the feedback volume for each feature and app. Although RE-SWOT is still a young prototype, and many improvements are possible, interviews with three case companies have revealed that the concept of RE-SWOT could be highly beneficial for defining the roadmap of an app.

3 Research Plan on NLP for RE

Besides strengthening our previous tools and conducting additional empirical evaluation, we are planning to investigate the synergies and the interactions between humans and NLP; some notable directions are the following:

- *Assisted requirements elicitation via chatbots*: in our past research, we have mostly explored the restricted notation of user stories. Although having a template is perceived as helpful by the practitioners [LDvdWB16a], this does inevitably constrain how they can express their requirements. We have started investigating how the elicitation process can be assisted through the use of chatbots that use NLP in a requirements-related conversation to obtain unambiguous and well-detailed requirements. A key challenge for the chatbot is to interact with the users in a natural fashion, without over-constraining the human, e.g., by enforcing the use of a semi-controlled language (this constraint was set, e.g., by other authors in [FBG18]). This research direction will require us to move away from the classic information retrieval metrics like precision and recall,

[4] `https://github.com/RELabUU/RE-SWOT`

and focus on the human-bot interaction: when and why is a requirements-related conversation better than another? The ability to uncover clear and useful requirements is not enough; we need to do so while keeping the conversation short and without annoying the human with the elicitation bot.

- *Automated synthesis of creative requirements*: our recent work [DP19] has started exploring the potential of identifying new requirements by automatically analyzing strengths and weaknesses of competitive software products. One may see RE-SWOT as a primitive tool that supports combinational creativity, for our algorithms pinpoint possible features to implement in the next releases by extracting features from other products. We intend to conduct further research in this field by focusing on exploratory creativity; for example, we could use semantic relatedness applied to nouns and verbs to suggest unique requirements that do not appear elsewhere. For instance, "generate schedule" is a feature that could be derived from existing ones like "create appointment", "export calendar", etc, for 'generate' is similar to 'create' and 'schedule' is related to 'calendar'. The challenge is how to identify features that are novel yet represent a sensible feature.

- *Linking requirements with architectures via linguistic traces*: we are experimenting, through case studies, a tool-assisted method for refining high-level requirements (*epics* in agile environments) to *software features* through intermediate layers: epics → user stories → modules → features. The method, which is called RE4SA (Requirements Engineering for Software Architecture), includes the use of NLP to identify and suggest possible trace links between the various artifacts through the similarity of the text phrases used in the artifacts. To shape our proposal, we started from the bottom up by studying software companies that follow a disciplined process to refine their requirements into specific software features [MDM$^+$19].

Acknowledgements

We would like to thank the former members of the RE-Lab: Garm Lucassen, Govert-Jan Slob, and Fatma Başak Aydemir; thanks to all the master's students who joined the lab temporarily, including Laurens Müter, Micaela Parente, Marcel Robeer, and Ivor van der Schalk.

References

[ADB$^+$18] Fatma Başak Aydemir, Fabiano Dalpiaz, Sjaak Brinkkemper, Paolo Giorgini, and John Mylopoulos. The Next Release Problem Revisited: A New Avenue for Goal Models. In *Proc. of RE*, 2018.

[BBM88] Sjaak Brinkkemper, N. Brand, and J. Moormann. Deterministic modelling procedures for automatic analysis and design tools. In *Computerized Assistance During the Information Systems Life Cycle*, pages 117–160, 1988.

[DFH16] Fabiano Dalpiaz, Xavier Franch, and Jennifer Horkoff. iStar 2.0 language guide. arXiv:1605.07767 [cs.SE], 2016.

[DP19] Fabiano Dalpiaz and Micaela Parente. RE-SWOT: From User Feedback to Requirements via Competitor Analysis. In *Proc. of REFSQ*, 2019.

[DSW15] Francisco De Sousa Webber. Semantic folding theory and its application in semantic fingerprinting. *arXiv preprint arXiv:1511.08855*, 2015.

[DvdSB$^+$19] Fabiano Dalpiaz, Ivor van der Schalk, Sjaak Brinkkemper, Fatma Başak Aydemir, and Garm Lucassen. Detecting Terminological Ambiguity in User Stories: Tool and Experimentation. *Information & Software Technology*, 2019.

[DvdSL18] Fabiano Dalpiaz, Ivor van der Schalk, and Garm Lucassen. Pinpointing Ambiguity and Incompleteness in Requirements Engineering via Information Visualization and NLP. In *Proc. of REFSQ*, 2018.

[FBG18] Edwin Friesen, Frederik Simon Bäumer, and Michaela Geierhos. CORDULA: software requirements extraction utilizing chatbot as communication interface. In *Proceedings of First Workshop on Natural Language Processing for Requirements Engineering*, 2018.

[GSA+17] Eduard C. Groen, Norbert Seyff, Raian Ali, Fabiano Dalpiaz, Joerg Doerr, Emitza Guzman, Mahmood Hosseini, Jordi Marco, Marc Oriol, Anna Perini, and Melanie Stade. The Crowd in Requirements Engineering: The Landscape and Challenges. *IEEE Software*, 34(2):44–52, 2017.

[KBHW97] Marnix Klooster, Sjaak Brinkkemper, Frank Harmsen, and Gerard Wijers. Intranet facilitated knowledge management: A theory and tool for defining situational methods. In *Advanced Information Systems Engineering, 9th International Conference CAiSE'97, Barcelona, Catalonia, Spain, June 16-20, 1997, Proceedings*, pages 303–317, 1997.

[LDLB16] Philipp Lombriser, Fabiano Dalpiaz, Garm Lucassen, and Sjaak Brinkkemper. Gamified Requirements Engineering: Model and Experimentation. In *Proc. of REFSQ*, volume 9619 of *LNCS*, pages 171–187, 2016.

[LDvdWB16a] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn van der Werf, and Sjaak Brinkkemper. The Use and Effectiveness of User Stories in Practice. In *Proc. of REFSQ*, volume 9619 of *LNCS*, pages 205–222, 2016.

[LDvdWB16b] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn van der Werf, and Sjaak Brinkkemper. Visualizing User Story Requirements at Multiple Granularity Levels via Semantic Relatedness. In *Proc. of ER*, volume 9974 of *LNCS*, pages 463–478. Springer, 2016.

[LDvdWB16c] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn E. M. van der Werf, and Sjaak Brinkkemper. Improving Agile Requirements: the Quality User Story Framework and Tool. *Requirements Engineering*, 21(3):383–403, 2016.

[LDvdWB17] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn van der Werf, and Sjaak Brinkkemper. Improving User Story Practice with the Grimm Method: A Multiple Case Study in the Software Industry. In *Proc. of REFSQ*, volume 10153 of *LNCS*, pages 235–252, 2017.

[LRD+17] Garm Lucassen, Marcel Robeer, Fabiano Dalpiaz, Jan Martijn van der Werf, and Sjaak Brinkkemper. Extracting conceptual models from user stories with Visual Narrator. *Requirements Engineering*, 22(3):339–358, 2017.

[MDM+19] Laurens Müter, Tejaswini Deoskar, Max Mathijssen, Sjaak Brinkkemper, and Fabiano Dalpiaz. Refinement of User Stories into Backlog Items: Linguistic Structure and Action Verbs. In *Proc. of REFSQ*, 2019.

[oDGBR05] Johan Natt och Dag, Vincenzo Gervasi, Sjaak Brinkkemper, and Björn Regnell. A linguistic-engineering approach to large-scale requirements management. *IEEE Software*, 22(1):32–39, 2005.

[RB05] Björn Regnell and Sjaak Brinkkemper. Market-driven requirements engineering for software products. *Engineering and Managing Software Requirements*, pages 287–308, 2005.

[SDBL18] Govert-Jan Slob, Fabiano Dalpiaz, Sjaak Brinkkemper, and Garm Lucassen. The interactive narrator tool: Effective requirements exploration and discussion through visualization. In *Proc. of the Poster Track of REFSQ*, 2018.

[ZHO+18] Yuanyuan Zhang, Mark Harman, Gabriela Ochoa, Guenther Ruhe, and Sjaak Brinkkemper. An empirical study of meta- and hyper-heuristic search for multi-objective release planning. *ACM Trans. Softw. Eng. Methodol.*, 27(1):3:1–3:32, 2018.