




Can We Design Software as We Talk?

A Research Idea

Marcela Ruiz^(✉)  and Björn Hasselman

Institute of Applied Information Technology, Zurich University of Applied Sciences,
Winterthur, Switzerland
marcela.ruiz@zhaw.ch, hassebj@students.zhaw.ch

Abstract. In the context of digital transformation, speeding up the time-to-market of high-quality software products is a big challenge. **Main challenges.** Software quality correlates with the success of requirements engineering (RE) sessions. RE sessions demand software analysts to collect all relevant material usually specified on written notes, flip charts, pictures, etc. Afterwards comprehensible requirements need to be specified for software implementation and testing. These activities are mostly performed manually, which causes process delays and software quality attributes like reliability, usability, comprehensibility, etc., are diminished causing software devaluation. **Innovative aspects.** This research idea paper proposes a framework for automating the tasks of requirements specification. The proposed framework involves computational mechanisms to enable the automatic generation of software design while requirements are discussed. The innovative aspect of this research comes from digitally transforming the software development life cycle (SDLC) where requirements are generated “on the fly” and virtual reality systems are in place. **Potential to make change.** The proposed framework has the potential to renovate the role of software analysts, which can experience substantial reduction of manual tasks, more efficient communication, dedication to more analytical tasks, and assurance of software quality from conception phases. This research idea paper introduces the framework for automating the task of requirements specification, and report our progress. We conclude the paper by outlining lessons learnt and future lines of work.

Keywords: Requirements engineering · Digital transformation · Software development life cycle · User stories generation

1 Introduction

Living in a digital era, service providers are challenged to offer services to their customers through a wide spectrum of channels. This constant introduction of new devices and technology challenges organisations to provide rapid

This research project is supported by ZHAW Digital and the Digitalisation Initiative of Zürich Universities DIZH.

© Springer Nature Switzerland AG 2020
S. Nurcan et al. (Eds.): BPMDS 2020/EMMSAD 2020, LNBIP 387, pp. 327–334, 2020.
https://doi.org/10.1007/978-3-030-49418-6_22

improvements of their IT infrastructure, while ensuring the highest user experience possible. Digital transformation stimulates the adaptation of existent business models and the creation of new ones; while society adapts to new ways to interact with services. Software systems are omnipresent in digital transformation process; making software quality of crucial value to ensure successful digital transformation [1].

Software quality correlates with the success of requirements engineering (RE) sessions [2], which makes RE a crucial phase of the software development life cycle (SDLC) [3]. The agile movement have proposed user stories as a minimal but complete language for the specification of software requirements [4]. This language has been proven to be successful and widely adopted by software developers [5]. software requirements are collected during RE sessions in the shape of pictures, flip chart notes, documentation etc. Later on, relevant information is digitalised in order to specify a set of comprehensible user stories to be used during development phases. Digitalisation of discussed requirements demands extra effort that has to be undertaken by software analysts [6]. All this complexity is magnified if we consider that software development is not usually taking place in the same geographical location. Big companies make use of software providers located in different continents. Teleworking is posing big challenges at the moment to ensuring collaborative requirements engineering [7].

The main objective of our research endeavour is *to reduce the time-to-market of software products by automating the task of requirements specification while requirements are discussed*. In this research idea paper, we introduce a framework for automating the task of requirements specification (see Fig. 1). We conceive a requirements engineering room where participants discuss requirements that are automatically specified in the shape of user stories, and transformed into software prototypes. In this room, we incorporate virtual reality tools like double robots for embodiment of remote participants, and interactive boards with collaborative tools as they have demonstrated to facilitate access and real-time edition of discussed requirements [8]. By implementing the proposed framework in practical settings, it is expected that SDLC goes through a process of digital transformation where software analysts are going to experience a significant reduction of manual tasks. User stories will be generated on the fly during the session, and virtual reality systems will allow efficient communication. Software analysts get empowered by focusing on meaningful tasks like analysing created user stories and prioritization. Software quality is then assured from the first release. The framework components are still under design and evaluation phases. Particularly, this paper reports our first steps towards automating the specification of user stories “on the fly” during RE elicitation sessions. We discuss and illustrate the use of the DEEP LEARNING CLASSIFIER and ONTOLOGY CRAWLER components presented in Fig. 1. Setting up of the requirements engineering room and software prototype generation are considered part of our short term research plans.

Paper Organisation: After reviewing related work in Sect. 2, we introduce our advances on providing automatic specification of user stores in Sect. 3. We summarise the design of two components: a deep learning classifier in Sect. 3.1, and an ontology crawler in

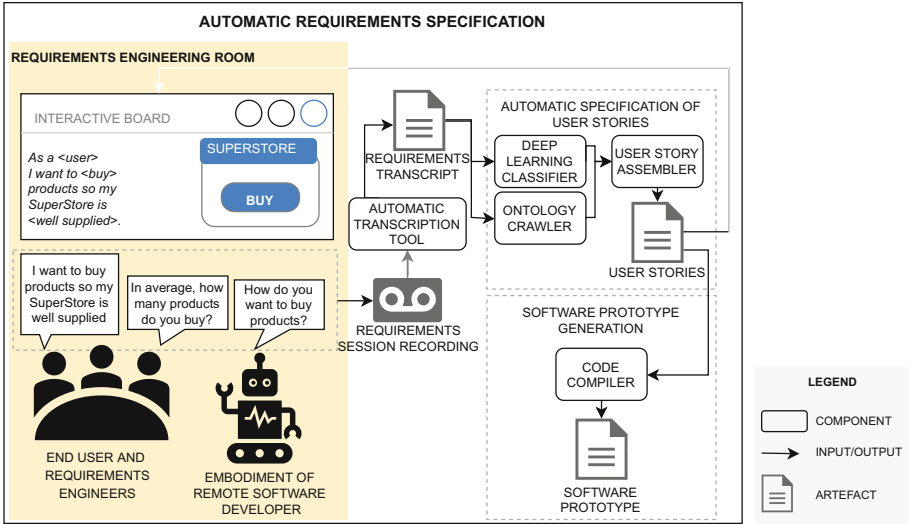


Fig. 1. Framework for automating the task of requirements specification

Sect. 3.2. Finally, we conclude our idea research paper by discussing lessons learnt and future lines of work in Sect. 4.

2 Related Work

In the field of requirements engineering there are several related work that approach the challenge of automate requirements specifications from different angles. We analyse these approaches based on: (a) requirements source: audio recordings/transcripts from requirements meetings, tweets, bug reports, user stories, existing documentation, domain repository; (b) generated requirements specification in the shape of: meeting minutes, knowledge extraction, tweets classification, relevant topics, remedied user stories, meeting summaries, and user stories; (c) Existing validation or evaluation: laboratory demonstration, comparative experiment; (d) Existence or not of tool support; and (e) Whether or has been applied in practice.

Some works focus on supporting software requirements specification by generating meeting minutes. For instance, Kaiya et al. [9] proposes a tool to support requirement elicitation meetings by recording the sessions and providing an assistant tool to manage the recordings and mark the important points via hypertext. Authors conclude that further collaboration mechanisms need to be incorporated to facilitate real-time edition of requirements and knowledge share. Murray et al. [10] developed a natural language processing approach to summarize emails and conversations in general, more projects involving textual sources appeared. Especially in the field of machine learning were multiple techniques developed to extract requirements engineering relevant information from different written origins [4, 11–14].

Rodeghero et al. [13] proposed a machine learning classification algorithm trained to recognise user stories' information [15]. As a conclusion of this study, the authors found out that information about software functionality and requirements rationale can

be identified by means of classification algorithms. Nevertheless, no information about the role can be automatically extracted. Another tool assisted approach to dynamic requirement elicitation was introduced by Abad et al. [14]. The tool extracts relevant snippets and simultaneously uses a third-party API to recognize tone and intentions of statements' providers.

We have taken the aforementioned research works as a reference to cover the gaps in terms of providing complete user stories from spoken software requirements during elicitation sessions (see last row in Table 1).

Table 1. Summary of related research for automatic generation of software requirements.

Ref	Requirements source	Generated requirements specification	Validation or evaluation	Tool support?	Applied in practice?
[9]	Audio recordings from requirements meetings	Meeting minutes, meeting summaries	Comparative experiment	Management tool	Yes
[13]	Audio transcripts from requirements meetings	Partial user stories (functions and rationale)	Comparative experiment	Machine Learning classifier	No
[14]	Audio transcripts from requirements meetings, existing documentation, domain repository	Relevant topics	Expert analysis	ELICA tool	Case study
[11]	Tweets	Tweets classification	Comparative experiment	ALERTme tool	Tested using Twitter
[4]	User stories	Remedied user stories	Supervised learning	AQUSA tool	Tested in University lab
[12]	Bug reports	Meeting summaries	Comparative experiment	Machine Learning classifier	No
Ours	Audio transcripts from requirements meetings	Complete user stories (roles, functions and rationales)	Laboratory demonstration	Yes	No

3 Automatic Specification of User Stories

Our goal is to elicit complete user stories including information related to “Role, Function and Rationale”. Based on the research work presented in Rodeghero et al. [13], we propose to classify software functionality in terms of functional and non-functional requirements; as well as identify requirements' rationale from requirement elicitation

sessions. Our research strategy is summarised in Fig. 2. In short, our research idea is to build a deep learning algorithm that can be further trained by providing labeled requirements elicitation sessions. For identifying missing roles, we propose to make use of existing ontologies that provide information related to typical roles belonging to the context in which software elicitation sessions take place.

In this paper, we summarise the deep learning classifier (see Sect. 3.1) and the ontology crawler components (see Sect. 3.2). For implementation purposes We chose the Java language as it guarantees portability and its popularity results in maintained and tested frameworks we can use. It has a sophisticated deep learning framework available in DL4J¹ and the Java OWL API² for handling Ontology files. The components will later provide the data to be used by the user story assembler component (out of scope of this paper).

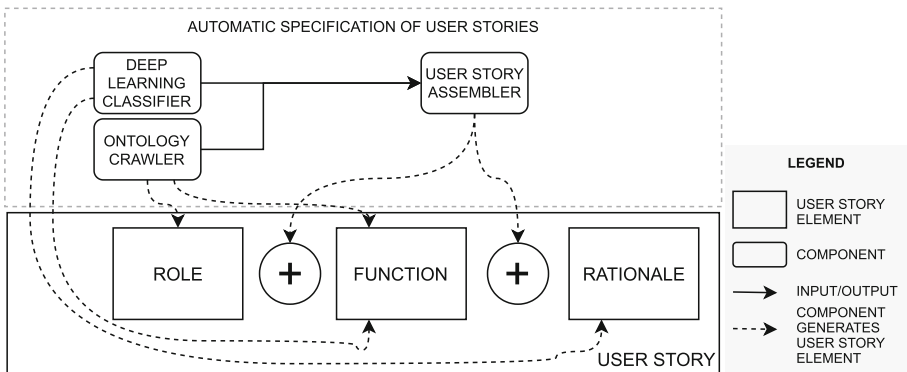


Fig. 2. Research strategy for automatic generation of user stories

3.1 Deep Learning Classifier Component

We propose a deep learning classifier based on the work proposed by [13]. We used deep learning specifically, because our intention is to imitating the classification process that has been done by using machine learning. In this way we can further compare performance values in subsequent experiments.

A **turn** is an established unit of analysis in natural language processing as opposed to using single sentences. It describes, when a person speaks in a conversation in between other speakers. To represent a turn in a learnable format, we use word embeddings provided by the model described in the work of Pennington et al. “GloVe: Global Vectors for Word Representation” [17]. Here, words will be represented as multidimensional, real-value vectors. In a three-dimensional space, similar words would lie ‘closer’ together than those, that semantically differ. Different, pre-trained word embedding models are available. The available representation dimensions depend on the vectors but range from 50 to 300. They also differ in terms of topic and number of tokens.

¹ <https://deeplearning4j.org/>.

² <http://owlcs.github.io/owlapi/>.

For our initial development process, we used the smallest set; the “Wikipedia 2014 + Gigawords”, which consists of 6 billion tokens and a representation of 50 dimensions.

The implementation of the deep learning classifier is available in our public GitHub repository at https://github.com/lmruizcar/requirements_classifier. An example of classification is presented in Fig. 3. The model in its current state performs about as well as random guessing since we need data for training purposes. As it has been mentioned by [16], the lack of data from requirements elicitation sessions is an obstacle in this type of investigations. Our model differentiates between three labels: None (0), Non-Functional (1) and Functional (2). A caveat of this deep learning approach is, that it only cares indirectly for the fact that turns can be both; labelled 1 and 2. Whereas [13] built multiple binary classifiers which each analysed the turn, our approach uses a SoftMax layer for which the output is interpret able as probabilities. A turn that falls into both categories, would have probabilities around 0.5 for both labels which can be interpreted individually, but is not represented in the standard evaluation method of machine learning classifiers.

```

Examples labeled as 0 classified by model as 0: 1 times
Examples labeled as 0 classified by model as 2: 4 times
Examples labeled as 1 classified by model as 2: 5 times
Examples labeled as 2 classified by model as 0: 1 times
Examples labeled as 2 classified by model as 2: 4 times

Warning: 1 class was never predicted by the model and was excluded from average precision
Classes excluded from average precision: [1]

=====Scores=====
# of classes: 3
Accuracy: 0.3333
Precision: 0.4038 (1 class excluded from average)
Recall: 0.3333
F1 Score: 0.3651 (1 class excluded from average)
Precision, recall & F1: macro-averaged (equally weighted avg. of 3 classes)
=====

```

Fig. 3. Results from running the deep learning classifier component

3.2 Ontology Crawler Component

Rodeghero et al. stated, that in conversations in requirement elicitation meetings “only 0.5% discussed role” [13]. Which is why they concluded that it is not feasible to extract role information from transcripts. For a complete user story, this role information is crucial. An established practice in information science is the use of ontologies to organize data and reduce complexity. We propose the ontology crawler component. The ontology crawler searches an ontology for defined entities and their restrictions to identify possible roles in the required context. As input, it takes an ontology from a file formatted in Web Ontology Language format. As output, it generates a list of foundational user stories, which consist of a role, an action and an object. For this, we have implemented a prototype based on Java OWL API. The prototype is available on our GitHub public repository at https://github.com/lmruizcar/ontology_crawler.

Figure 4 exemplifies the generation of foundational user stories for the SmallShop Case. A product manager needs to be able to add products to the shop and remove it, e.g., if they are not in stock anymore. The customers that use the shop want to buy

```

C:\Users\ \Desktop>java -jar FUSCrawler.jar -o test-restrictions.owl
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
As a ProductManager I want to adds Product
As a ProductManager I want to removes Product
As a User I want to buys Product
As a Admin I want to adds User
As a Admin I want to removes User

```

Fig. 4. Example of executing the ontology crawler component in the context of the SmallShop case

products. And for returning customers it is good practice to store relevant information like the shipping address in a user account.

4 Lessons Learnt and Future Work

The current paper presents a research idea for automating the process of requirements specification. We propose a framework consisting of a requirements engineering room, and components to support the automatic generation of user stories and software prototype generation. This paper presents results from the implementation of the components for automatic generation of user stories while requirements are discussed. Our efforts lead to the development of prototypes for a deep learning classifier and ontology crawler. Initial results are promising and proves the feasibility of the proposed research idea. Prototypes are made available on our public GitHub repository to motivate further research in the field.

For the near future, we plan to keep evolving the prototype for user stories assembler component. In this way, we can obtain full user stories from elicitation sessions. For our framework to mature and being implemented in practical settings, we envision to build a flexible environment to support the plug-and-play of components that conform the framework. In this way, we can incorporate alternative components while ensuring a proper interoperability. In addition to evaluating the extent to which our solution improves user stories' generation in terms of efficiency, we plan to evaluate the intention to use and usability of the framework from the perspective of requirements engineers. For our long term plans, we plan to investigate the quality of software engineering recordings to further adjust and improve our framework.

References

1. Gebhart, M., Giessler, P., Abeck, S.: Challenges of the digital transformation in software engineering. In: The Eleventh International Conference on Software Engineering Advances (ICSEA) (2016)
2. Mund, J., Femmer, H., Mendez, D., Eckhardt, J.: Does quality of requirements specifications matter? Combined results of two empirical studies (2017). <https://arxiv.org/pdf/1702.07656.pdf>
3. Chakraborty, A., Baowaly, M., Arefin, A., Bahar, A.: The role of requirement engineering in software development life cycle. *J. Emerg. Trends Comput. Inf. Sci.* **3**, 723–729 (2012)

4. Dalpiaz, F., Brinkkemper, S.: Agile requirements engineering with user stories. In: 26th International Requirements Engineering Conference (RE), Banff, AB, Canada (2018)
5. Wagenaar, G., Overbeek, S., Lucassen, G., Brinkkemper, S., Schneider, K.: Working software over comprehensive documentation - rationales of agile teams for artefacts usage. *J. Softw. Eng. Res. Dev.* **6**, 7 (2018). <https://doi.org/10.1186/s40411-018-0051-7>
6. Wüest, D., Seyff, N., Glinz, M.: FlexiSketch: a lightweight sketching and meta-modeling approach for end-users. *Softw. Syst. Model.* **18**(2), 1513–1541 (2017). <https://doi.org/10.1007/s10270-017-0623-8>
7. Damian, D., Zowghi, D.: RE challenges in multi-site software development organizations. *Requir. Eng. J.* **8**, 149–160 (2003). <https://doi.org/10.1007/s00766-003-0173-1>
8. Wüest, D., Seyff, N., Glinz, M.: Sketching and notation creation with FlexiSketch team: evaluating a new means for collaborative requirements elicitation. In: 23rd International Requirements Engineering Conference (RE), Ottawa (2015)
9. Kaiya, H., Saeki, M., Ochimizu, K.: Design of a hyper media tool to support requirements elicitation meetings. In: Proceedings Seventh International Workshop on Computer-Aided Software Engineering (1995)
10. Murray, G., Carenini, G.: Summarizing spoken and written conversations. In: Conference on Empirical Methods in Natural Language Processing, PA, USA, Stroudsburg (2008)
11. Guzman, E., Ibrahim, M., Glinz, M.: A little bird told me: mining tweets for requirements and software evolution. In: 25th International Requirements Engineering Conference (RE) (2017)
12. Rastkar, S., Murphy, G.C., Murray, G.: Summarizing software artifacts: a case study of bug reports. In: Proceedings of the 32nd International Conference on Software Engineering, NY, USA, New York, vol. 1 (2010)
13. Rodeghero, P., Jiang, S., Armaly, A., McMillan, C.: Detecting user story information in developer-client conversations to generate extractive summaries. In: 39th International Conference on Software Engineering (ICSE) (2017)
14. Abad, Z.S.H., Gervasi, V., Zowghi, D., Barker, K.: ELICA: an automated tool for dynamic extraction of requirements relevant information. In: International Workshop on Artificial Intelligence for Requirements Engineering (AIRE) (2018)
15. Krasniqi, R., Jiang, S., McMillan, C.: TraceLab components for generating extractive summaries of user stories. In: International Conference on Software Maintenance and Evolution (ICSME) (2017)
16. Rodeghero, P.: Behavior-informed algorithms for automatic documentation generation. In: International Conference on Software Maintenance and Evolution (ICSME) (2017)
17. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar (2014)