Winkler, Jonas Paul; Vogelsang, Andreas

Using tools to assist identification of non-requirements in requirements specifications

A controlled experiment

This is a post-peer-review, pre-copyedit version of an article published in Lecture Notes in Computer Science (10753). The final authenticated version is available online at: https://doi.org/10.1007/978-3-319-77243-1_4.

Winkler, Jonas Paul; Vogelsang, Andreas (2018). Using tools to assist identification of non-requirements in requirements specifications. A controlled experiment. In: Lecture Notes in Computer Science (pp. 57-71). Cham: Springer. https://doi.org/10.1007/978-3-319-77243-1_4.

Using Tools to Assist Identification of Non-Requirements in Requirements Specifications – A Controlled Experiment

Jonas Paul Winkler and Andreas Vogelsang

Technische Universität Berlin, Germany
{jonas.winkler,andreas.vogelsang}@tu-berlin.de

Abstract. [**Context and motivation**] In many companies, textual fragments in specification documents are categorized into requirements and non-requirements. This categorization is important for determining liability, deriving test cases, and many more decisions. In practice, this categorization is usually performed manually, which makes it labor-intensive and error-prone. [**Question/Problem**] We have developed a tool to assist users in this task by providing warnings based on classification using neural networks. However, we currently do not know whether using the tool actually helps increasing the classification quality compared to not using the tool. [**Principal idea/results**] Therefore, we performed a controlled experiment with two groups of students. One group used the tool for a given task, whereas the other did not. By comparing the performance of both groups, we can assess in which scenarios the application of our tool is beneficial. [**Contribution**] The results show that the application of an automated classification approach may provide benefits, given that the accuracy is high enough.

Keywords: Requirements Engineering, Machine Learning, Convolutional Neural Networks, Natural Language Processing

1 Introduction

Requirements specifications are used in many requirements engineering (RE) processes to document results. The purpose of these documents is to define the properties that a system must meet to be accepted. Moreover, in contexts, where one company or department acts as a customer and another company acts as a supplier, the requirements specification also defines liability between the partners (i.e., what must be achieved to fulfill the contract). For this reason, requirements specifications should undergo a rigorous quality assessment process especially in industries where systems are created by a collaboration of many suppliers (e.g., automotive).

Besides actual and legally binding requirements, requirements specifications usually contain auxiliary content (e.g., explanations, summaries, examples, and references to other documents). These content elements are not requirements, which must be fulfilled by the supplier but they may facilitate the process of

understanding requirements and their context. To distinguish this auxiliary information from legally binding requirements, one of our industry partners annotates all content elements in their requirements specifications with specific labels for *requirements* and *information*. However, this manual labeling task is time-consuming and error-prone. By analyzing a set of requirements specifications from our partner, we observed that labels (i.e., requirement and information) are often not added when the content is created. This impedes the usage of these documents for following activities, such as creating a test specification based on a requirements specification. Adding the labels at a later stage is expensive since every content element has to be read and understood again.

To assist requirements engineers in performing this task, we have created a tool that automatically classifies the content elements of requirement specifications and issues warnings if the actual label deviates from the automatically predicted one. This tool is used by requirements authors and reviewers for creating new requirements or inspecting already existing requirements.

The tool uses neural networks to classify content elements as either information or requirement. This neural network is trained on a large corpus of reviewed requirements taken from requirements specifications of our industry partner. As with all neural networks, performance is not perfect and thus the tool will sometimes issue warnings on correctly labeled items and will sometimes ignore actual defects. In earlier evaluations, the classifier achieved an accuracy of 81% [1]. This might impede the usefulness of our tool. Thus, we currently do not know whether using the tool actually helps increasing the classification quality compared to not using the tool.

Therefore, we have conducted a controlled experiment with computer science students trained on requirements engineering to evaluate the usefulness of our tool for the given task. The students were split into two equally sized groups. Both groups performed a given task independently. One group used our tool, whereas the other did not. In this paper, we present the goals, setup, and results of this experiment.

The results indicate that given high accuracy of the provided warnings, users of our tool are able to perform slightly better than the users performing manual review. They managed to find more defects, introduce less new defects, and did so in shorter time. However, when many false warnings are issued, the situation may be reversed. Thus, the actual benefit is largely dependent on the performance of the underlying classifier. False negatives (i.e., defects with no warnings) are an issue as well, since users tend to focus less on elements with no warnings.

2 Background

At our industry partner, documentation and review of requirements are independent processes. After creation, requirements documents are reviewed during quality audits. Each requirement is assessed as to whether it is necessary, conflict free, well written, etc. Some assessments are automatically checked by a requirements specification analysis tool using predefined rule sets (e.g., is the require-

ment phrased using certain modal verbs, weak work analysis, are the required attributes set). However, most of the assessments require context knowledge of the requirements engineer and thus cannot be performed by such simple analysis methods. The task of separating information and requirements is one example of such an assessment.

In our previous works [2, 1], we have presented a method to perform this task automatically. At its core, our approach uses a convolutional neural network as presented in [3]. The network is trained on requirement content elements and information content elements taken from requirements specifications of our industry partner. The approach has been integrated in the aforementioned requirements specification analysis tool.

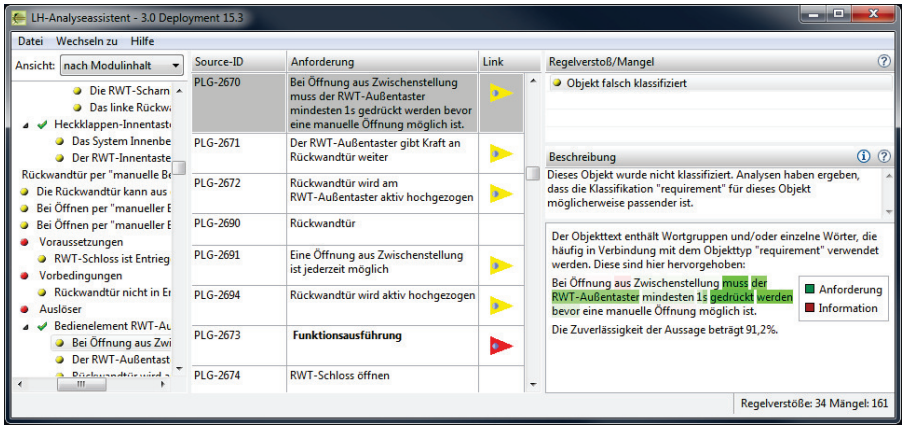


Fig. 1: Screenshot of the tool

Fig.1 shows a screenshot of the tool. It closely resembles the requirements engineering tool used at our industry partner (IBM Rational DOORS), featuring a tree view on the left and a tabular view of the requirements in its center. The tool issues warnings (yellow markers) and errors (red markers) on content elements where the predicted classification differs from the actual one. On the right hand side, an explanation of the error is provided: Words and groups of words leading to the classification decision are identified and highlighted using a back tracing technology [4]. Additionally, content elements for which no class could be reliably detected are also marked. These might need to be rephrased.

By explicitly pointing out content elements with questionable phrasing and/or classification, we expect that requirements engineers will identify more issues within their documents and may do so in shorter time. This will shorten the time spent during quality audits and hopefully reveal more issues compared to fully manual reviews. However, using such a tool also bears the risk of hiding actual errors. If requirements engineers start to trust the tool and rely on it, it is less likely that they identify defects not found by our tool.

3 Research Methodology

In order to assess the impact of our tool on the task of reviewing requirements/information classification, we conducted a controlled experiment with students. We followed the guidelines provided in Ko et al. [5] and Jedlitschka et al. [6].

3.1 Research Questions

The overall goal of our experiment is to examine whether and how the use of a tool improves the process of finding *defects* in requirements documents compared to completely manual review. In this paper, a defect is a misclassified content element (i.e., requirement marked as information or information marked as requirement). As there are various ways of improving this process, we aim to analyze different aspects. Therefore, we followed five research questions.

RQ1: Does the usage of our tool enable users to detect more defects? This is the primary goal of our tool. By focusing the attention of users on possibly misclassified content elements, we assume they will be able to detect more defects within their documents.

RQ2: Does the usage of our tool reduce the number of defects introduced by users? Requirements engineers tend to make errors during quality audits (e.g., dismissing a requirement as an information). By decreasing the focus on possibly correctly classified content elements, we assume they will less likely edit those elements and introduce less defects into their documents.

RQ3: Are users of our tool prone to ignoring actual defects because no warning was issued? As our tool issues warnings to focus the attention of users, it is possible that they will tend to skip elements with no warnings. If these content elements contain defects, users are likely to miss them. Thus, we need to analyze whether users miss more unwarned defects when using our tool.

RQ4: Are users of our tool faster in processing the documents? One of our primary goals is to allow requirements engineers to work more efficiently. Therefore, we analyze whether users of our tool are able to work faster.

RQ5: Does our tool motivate users to rephrase requirements and information content elements? Our tool also shows explanations for each issued warning, i.e., which words caused the internal neural network to decide on either requirement or information. If an actual requirement was classified as information by our tool due to bad phrasing, these explanations could lead users into rethinking the phrasing and reformulating it, thus improving the quality of the requirement.

3.2 Experiment Design

We utilized a two-by-two crossover design [7], using two sessions and two groups of subjects (see Table 1). The treatment group worked within our tool environment that we described in Section 2, later referred to as the tool-assisted group (TA), while the control group was working without the help of our tool. We refer to the control group as the manual group (M). The difference between sessions

is the requirements specification that was used. In the first session, we used a requirements specification of a wiper control system and in the second session, we used a requirements specification of a window lift system.

Table 1: Experimental Design

	Group 1	Group 2
Session 1 (Wiper Control)	M	TA
Session 2 (Window Lift)	TA	M

3.3 Participants

The experiment was conducted as part of a university masters course on automotive software engineering at TU Berlin. The participants of this course were undergraduate students in their last year. The majority was enrolled for the study programs computer science, computer engineering, or automotive systems. The course included lectures on basic principles of requirements and test engineering. As a result, the students understood what requirements engineering is used for and how requirements should be documented. They were especially aware of the consequences of bad requirements engineering on subsequent development steps.

The experiment was announced beforehand. We especially emphasized that a large number of participants would be crucial for acquiring useful results. We motivated the students to take part in the experiment by telling them that they would gain insight into real world requirements engineering. At the time of the experiment, 20 students were present, which reflects about two-thirds of all students enrolled in the course.

3.4 Experimental Materials

The experiment was conducted using real-world requirements documents available at our industry partner. We selected two documents describing common systems in any modern car: the wiper control system and the window lift system. The documents contain requirements in a tabular format. Each row contains one content element, consisting of its identifier, the content text, and its object type. Three object types were present in these documents: heading, requirement, and information.

The documents are very long, containing about 3000 content elements each. Since the students cannot possibly read, understand, and find defects in the entire document within the time limit (see Sec. 3.7), the documents were truncated to a reasonable size. Also, as per request of our industry partner, certain confidential information such as the names of persons, signals, and other systems were replaced by generic strings (e.g., “SIGNAL-1”, “SYSTEM-3”).

To assess whether the students with or without tool perform better, we created a gold standard by identifying all defects the students had to find in the two documents by ourselves. This gold standard serves as reference for comparing the performance of the groups.

Each document was then prepared in two different formats: a csv like format readable by our tool for assisted review and an MS Excel version for unassisted review. Both formats contain exactly the same data. Colors and font sizes in the Excel spreadsheet were selected to mimic the tool as close as possible.

Table 2 lists the relevant characteristics of the documents, such as number of elements, number of defects, numbers about warnings issued by our tool, and overall accuracy of the tool on this document. The Wiper Control document has many obviously misclassified elements and many of the false warnings are easily dismissible as such. On the Window Lift document, our tool issued many false warnings due to an inconsistent writing style within the document.

Table 2: Characteristics of the used requirements specifications

	Wiper Control	**Window Lift**
Total Elements	115	261
Total Requirements	85	186
Total Information	30	75
Total Defects	20	17
Total Warnings	24	70
Correct Warnings	12	12
Unwarned Defects	8	5
Accuracy	82.6%	75.8%

3.5 Tasks

The task given to the students was designed to resemble the procedure taken during actual quality audits. Each student had to read and understand the requirements specifications and correct defects within these documents. The students were instructed to search for the following defects:

- Requirement content elements incorrectly classified as information
- Information content elements incorrectly classified as requirements
- Badly phrased requirements (i.e., ambiguous, missing modal verb, ...)

The students were asked to fix the defects by either changing the object type, the phrasing, or both.

3.6 Data Analysis Procedure

We perform the analysis of our research questions using metrics defined in this section and formulate working hypotheses about what outcome we expect. The

independent variable in our experiment is the review method used by the student, which is either *Manual*, or *Tool-Assisted*.

RQ1: Does the usage of our tool enable users to detect more defects? We evaluate this question by calculating the *Defect Correction Rate (DCR)*:

$$DCR = \frac{DefectsCorrected}{DefectsInspected}$$

DefectsCorrected is the number of defects identified and corrected by a student, *DefectsInspected* is the number of defects examined by the student. We do not base this metric on the total number of defects in the document because a student might not have had the time to review the whole document. For the DCR, we are only interested in the likelihood that a defect is identified and corrected if the respective object has at least been examined by a student. We expect that the warnings issued by our tool help students to identify and correct defects. Thus, we expect a higher DCR:

$$H_1 : DCR(\text{Tool-Assisted}) > DCR(\text{Manual})$$

RQ2: Does the usage of our tool reduce the number of defects introduced by users? Similar to RQ1, we evaluate this question by calculating the *Defect Introduction Rate (DIR)*:

$$DIR = \frac{DefectsIntroduced}{ElementsInspected}$$

where *DefectsIntroduced* is the number of modified elements that were originally correct and *ElementsInspected* the total number of elements examined by the student. We expect that

$$H_2 : DIR(\text{Tool-Assisted}) < DIR(\text{Manual})$$

RQ3: Are users of our tool prone to ignoring actual defects because no warning was issued? For evaluating this question, we only consider elements on which our tool issued no warnings. The *Unwarned Defect Miss Rate (UDMR)* is defined as

$$UDMR = \frac{UnwarnedDefectsMissed}{UnwarnedDefectsInspected}$$

where *UnwarnedDefectsInspected* is the number of examined defects for which the tool did not give any warnings and *UnwarnedDefectsMissed* is the subset of these that were not corrected. Since we suspect that the users of our tool will be more focused on the elements with warnings, we expect the following (which would be a negative property of using the tool):

$$H_3 : UDMR(\text{Tool-Assisted}) > UDMR(\text{Manual})$$

RQ4: Are users of our tool faster in processing the documents? This question is answered by examining how much time the users spent on each element. The *Time Per Element (TPE)* is calculated as follows:

$$TPE = \frac{TotalTimeSpent}{ElementsInspected}$$

TotalTimeSpent is the time the students needed to complete the document or the total time of the experiment if they did not finish. We suspect that users of our tool will be faster in processing the documents:

$$H_4 : TPE(\text{Tool-Assisted}) > TPE(\text{Manual})$$

RQ5: Does our tool motivate users to rephrase requirements and information content elements?

$$ERR = \frac{ElementsRephrased}{ElementsInspected}$$

This metric captures how many content elements are rephrased by users. We did not inspect whether the change improved the requirement or not. We expect that users of the tool may be more eager to rephrase content elements since the tool points to linguistic weaknesses by providing visual explanations of its decisions.

$$H_5 : ERR(\text{Tool-Assisted}) > ERR(\text{Manual})$$

3.7 Procedure

The experiment was scheduled to take 90 minutes. The time available was divided into four segments:

1. **Introduction, setup, data distribution, and group assignment (20 minutes):** The session was started with a presentation on requirements quality, how our industry partner performs quality audits, the importance of differentiation between requirements and information, details on the structure of the experiment itself, and details on the documents necessary to understand them.
   After that, we randomly divided the students into two groups and distributed the requirements documents to them. The tool was distributed to the students a week before the experiment without any data to reduce the time needed for setup.
2. **Session 1: Wiper Control (20 minutes):** During the experiment, students worked through the document from top to bottom and made modifications where they thought it is necessary. We allowed them to form teams of two or three students of the same group. This way, they were able to discuss their opinions, much like requirements engineers will do during real quality audits. We prohibited them from sharing information between teams or groups. After time was up, the students were asked to mark the position they were at.
3. **Session 2: Window Lift (30 minutes):** The second run was executed exactly like the first but with switched groups and with a different document.

4. **Conclusions (10 minutes):** After the second run, we collected the modified documents and presented how we are going to evaluate the data and what kind of results we expect.

3.8 Piloting

Prior to performing the actual experiment, we simulated the experiment. Some of our co-workers were briefed and performed the same tasks as the students in the experiment. We used the results of the experiment to adjust certain parameters of the experiment, such as the size of the documents and allocated time for each session. The test run also allowed us to verify that our planned evaluation methodology yields usable results.

4 Study Results

For the first document, we received a total of 14 reviews, 7 reviews with tool usage and 7 reviews without tool usage. We received less reviews for the second document (3 with tool usage and 4 without tool usage) because some students had to leave. We also had to discard 2 reviews because one did not contain any changes and the other was done by a student who had major difficulties in understanding the documents due to language barriers.

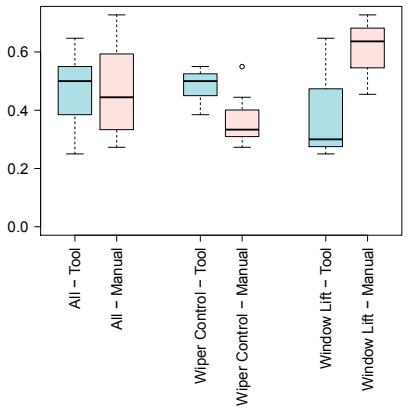
An overview of all collected data is available online[1]. Fig. 2 shows boxplots of the calculated metrics over all reviews and for each review document separately. In the following, we discuss our research questions based on these results.

4.1 Discussion

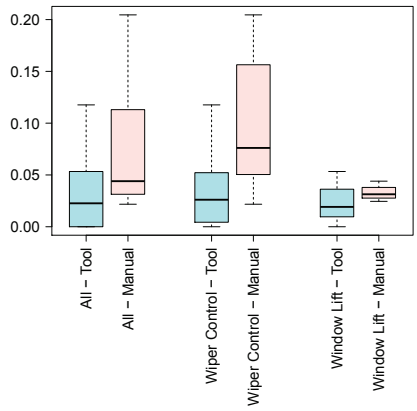
In Fig. 2a, the Defect Correction Rate (DCR) is displayed for each document and review method. Regarding the Wiper Control document, the students with the tool performed better than the students without tool support. The average correction rate is 11% higher. However, on the Window Lift document, results were opposite: The students doing manual review corrected 61% of all examined defects, whereas the students doing assisted review only corrected 45%. One explanation for this could be the lower quality of the warnings issued by our tool (see the difference in accuracy in Table 2) due to the low linguistic quality of the Window Lift document. Therefore, it is possible that the students were misled by the false warnings of the tool.

Fig. 2b shows the Defect Introduction Rate (DIR), i.e., how many new defects were introduced per examined element by changing content elements with no defect. The students doing the assisted review performed better on both documents, introducing only half as many new defects on average as the students without tool. We assume that students refrained from changing content elements if no warning was issued by the tool.

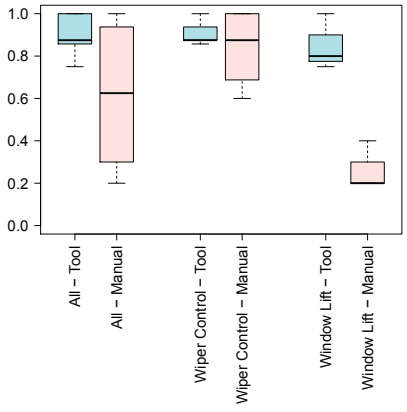
[1] https://doi.org/10.6084/m9.figshare.5469343.v1



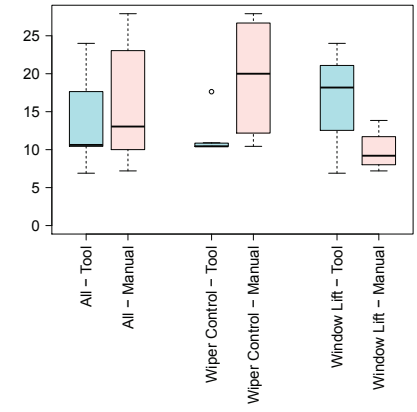
(a) Defect Correction Rate



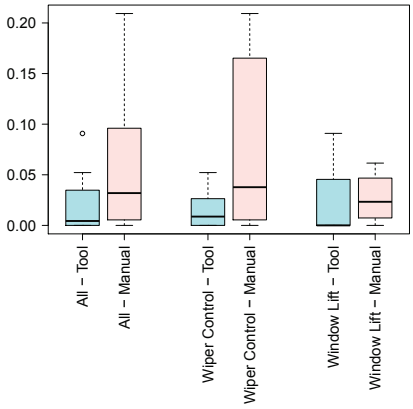
(b) Defect Introduction Rate



(c) Unwarned Defect Miss Rate



(d) Time Per Element



(e) Element Rephrase Rate

Fig. 2: Study results

An unwarned defect is a defect for which our tool did not issue a warning. We analyzed how likely it is that these defects are missed by tool users. We compare this with the performance of the manual review group on the same set of defects (those without warnings). Of course, students in the manual group did not know which defects had warnings in the tool. Fig. 2b shows that if the tool is used, 90% of defects without a warning are not corrected. As expected, the group doing manual review performed better, missing only 62% of all unwarned defects. This is in line with our expectation that students with tool support will focus less on elements without warnings.

The time spent by the students on each element is shown in Fig. 2d. Students spent less time on each element in the Window Lift document (mean: 10.8 s) than on the Wiper Control document (mean: 13.9 s). This may be the result of a learning effect: Students became used to the task and learned for which information they need to look. On the first document, the students performing assisted review were considerably faster (11.2 s per element on average compared to 16.6 s for the manual review). In addition, 4 out of 7 teams using the tool completed their review, whereas only 1 out of 7 teams finished using manual review. On the second document, the students using the tool were slower, most likely because they analyzed the false warnings and tried to decide whether to change or not to change a content element.

Fig. 2e shows how many content elements were rephrased. Overall, only 3.8% of the examined content elements were changed. In 8 out of 21 reviews, no element was changed at all. We expected more changes, considering that the overall text quality of the documents was rather low. The students not working with the tool changed more content elements, especially on the Wiper Control document. We assume that the students working with the tool were more focused on the warnings than changing the text of content elements.

To summarize the discussion, we provide answers to our research questions:

RQ1: Users of our tool may be able to detect and fix more defects than users without the tool. However, this depends on the accuracy of our tool. Bad accuracy may even have a negative effect on defect identification.

RQ2: If our tool is used, less new defects are introduced during a review.

RQ3: Our students missed more unwarned defects (i.e., false negatives) if warnings were present.

RQ4: Given that the accuracy of the tool is high enough, users of our tool may be able to complete the task much faster.

RQ5: In our experiment, usage of the tool did not motivate users to rephrase more content elements.

5 Threats to Validity

In this section, we discuss the various threats to construct, internal and external validity of our experiment.

Number of participants [construct]. A major threat to our results is the low number of participants. Since we allowed students to work in teams, the

number of results is even smaller. This allowed them to engage in discussions within the team, which, in our opinion, is more important for the experiment setup than having a larger sample size. On the other hand, the small sample size forbids making any statistical tests on the hypothesis described in this paper. Therefore, we do not claim that we can reject or support any of the hypothesis with our results. Our goal was to check and refine the working hypothesis that we came up with to see which (additional) parameters might influence the results.

Definition of gold standard [construct]. We compared the results of the two review methods with a gold standard that we created ourselves, i.e., we defined what a defect is in the documents. This definition has an impact on the performance assessment of the review methods. The authors of this article are working on this classification problem for more than 3 years in close collaboration with an industry partner. Therefore, we claim that the created gold standard is close to what the industry partner would consider as truth.

Differences in knowledge between students [internal]. We assumed that the students have no prior knowledge in requirements engineering apart from what was taught during the lecture. Some students may have more knowledge in requirements engineering than others and thus may perform better at the task. We diminished the effects of this by having each students perform the task with both review methods.

Maturation [internal]. Maturation is an effect that occurs over time and may change a subject's behavior due to learning, fatigue, or changes in motivation. The students in our experiment may have learned from the first session of the experiment and applied that knowledge in the second session. It is also possible that students have lost motivation or performed worse due to fatigue after completing the first session.

Communication between groups [internal]. We have especially stated during the experiment that it is important not to share information about defects between groups. However, since the experiment was conducted in a classroom setting and students were able to discuss within the group, information may have been shared between groups nonetheless. As such, not all reviews may be independent.

Time limit [internal]. The time limit was set for two reasons: First, the time in actual quality audits is limited as well, and second, we only had a total of 90 minutes available. We told the students that it is not necessary to complete a document within the time limit. However, the students could have aimed for completing the review nonetheless and thus may have performed worse than without a time limit.

Students are no RE experts [external]. Compared with people who actually perform quality audits, students are no requirements engineering experts. They lack both general knowledge about the processes in which requirements specifications are involved in as well as special knowledge about the documents themselves. However, students may inspect the documents more carefully whereas RE experts may tend to dismiss possible defects either due to them being the authors or due to process constraints (changes may induce additional

costs). Falessi et al. state that controlled experiments with students are as valid as experiments with experts [8].

The most relevant threat to validity is the number of participants. Our sample size is not sufficiently large to be used for statistical significance tests and therefore, experiments on larger groups of participants may show different results. An experiment on a larger user base should be performed next.

6 Related Work

Machine learning techniques are applied for many requirements engineering tasks, especially for classification. A few of these works are outlined here.

Hayes et al. [9] present a tool that integrates with Weka and provides a convenient way for users to perform classification tasks. For example, their tool is able to differentiate between temporal and non-temporal requirements.

Huang et al. [10] present and approach to classify different types of non-functional requirements, achieving 81% recall and 12% precision on their dataset, averaged above all classes.

Ott [11] presents an approach to increase the efficiency of requirements specification reviews by assigning requirements to topics (e.g., temperature, voltage). He argues that a block of requirements belonging to the same topic may be reviewed faster than requirements of mixed topics. However, no validation of that claim is provided.

Perini et al. [12] use a prioritization algorithm based on machine learning techniques to sort software requirements by their importance. This allows stakeholders to discern important and less important requirements more easily. Its effectiveness is demonstrated using empirical evaluation methods.

There is currently a discussion in the community around the empirical investigation of the effectiveness of automated tools for RE tasks. In an earlier paper, Berry et al. [13] claim that in some scenarios, for some tasks, any tool with less than 100% recall is not helpful and the user may be better off doing the task entirely manually. In fact, our experiment supports this claim by indicating that the accuracy of the tool may have an effect on the observed performance. In a follow-up paper [14], Berry relaxes his first claim by saying that a human working with the tool on the task should at least achieve better recall than a human working on the task entirely manually. Our experimental setup follows this idea by comparing tool-assisted and manual reviews.

7 Conclusions

At our industry partner, each content element of a requirements specification document needs to be classified as either requirement or non-requirement ("information"). A requirement is legally binding and needs to be tested. This does not apply to non-requirements. This classification is currently performed manually. We have built a tool that classifies content elements of specification documents

as either information or requirement and issues warnings when the classification seems to be wrong. We assume that by using our tool, RE experts will be able to perform this classification more effectively and efficiently.

In this paper, we have presented the results of a controlled experiment, showing the benefits and limitations of our tool. Two groups of students analyzed requirements specification documents and were asked to fix any defects in them. One group used the tool, whereas the other did not.

The results show that, given high accuracy of the provided warnings, users of our tool are able to perform slightly better than the users performing manual review. They managed to correct more defects, introduce less new defects, and did so in shorter time. However, when many false warnings are issued, the situation may be reversed. Thus, the actual benefit is largely dependent on the performance of the underlying classifier. False negatives (i.e., defects with no warnings) are an issue as well, since users tend to focus less on elements with no warnings.

The sample size used in our experiment is not high enough to underpin our conclusions with measures on statistical significance, as we were limited to the students visiting the lecture. We plan to perform the experiment again with more students. However, the results presented in this paper already show that improvements can be achieved by using our tool.

Since the tool is based on machine learning algorithms, achieving perfect accuracy, or at least perfect recall, is impossible. Therefore, our tool may not be needed when a requirements engineer is doing a complete review of a specification document and is able to detect all defects. However, in the real world, humans do errors due to various reasons such as fatigue and inattention. Our approach may help them to do fewer errors and achieve higher quality specification documents (with regard to requirement vs. information classification) compared with manual review.

To assess which accuracy or recall the tool must provide to outperform a completely manual review is an interesting question that we want to follow in future experimental setups.

References

1. Winkler, J.P., Vogelsang, A.: Automatic Classification of Requirements Based on Convolutional Neural Networks. In: 3rd IEEE International Workshop on Artificial Intelligence for Requirements Engineering (AIRE). (2016) 39–45
2. Winkler, J.P.: Automatische Klassifikation von Anforderungen zur Unterstützung von Qualitätssicherungsprozessen. In Mayr, H.C., Pinzger, M., eds.: INFORMATIK 2016. Lecture Notes in Informatics (LNI), Bonn (2016) 1537–1549
3. Kim, Y.: Convolutional Neural Networks for Sentence Classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). (2014) 1746–1751
4. Winkler, J.P., Vogelsang, A.: "What Does My Classifier Learn?" A Visual Approach to Understanding Natural Language Text Classifiers. In: Proceedings of the 22nd International Conference on Natural Language & Information Systems. NLDB (2017) 468–179

5. Ko, A.J., LaToza, T.D., Burnett, M.M.: A practical guide to controlled experiments of software engineering tools with human participants. Empirical Software Engineering **20**(1) (2015) 110–141
6. Jedlitschka, A., Ciolkowski, M., Pfahl, D.: Reporting Experiments in Software Engineering. In Shull, F., Singer, J., Sjøberg, D.I.K., eds.: Guide to Advanced Empirical Software Engineering. Springer London, London (2008) 201–228
7. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering. Springer Science & Business Media (2012)
8. Falessi, D., Juristo, N., Wohlin, C., Turhan, B., Münch, J., Jedlitschka, A., Oivo, M.: Empirical software engineering experts on the use of students and professionals in experiments. Empirical Software Engineering (2017) 1–38
9. Hayes, J.H., Li, W., Rahimi, M.: Weka meets TraceLab: Toward Convenient Classification: Machine Learning for Requirements Engineering Problems: A Position Paper. In: 1st IEEE International Workshop on Artificial Intelligence for Requirements Engineering (AIRE). AIRE (2014) 9–12
10. Cleland-Huang, J., Settimi, R., Zou, X., Solc, P.: Automated classification of non-functional requirements. Requirements Engineering **12**(2) (2007) 103–120
11. Ott, D.: Automatic Requirement Categorization of Large Natural Language Specifications at Mercedes-Benz for Review Improvements. In Doerr, J., Opdahl, A.L., eds.: Requirements Engineering: Foundation for Software Quality: 19th International Working Conference, REFSQ 2013, Essen, Germany, April 8-11, 2013. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg (2013) 50–64
12. Perini, A., Susi, A., Avesani, P.: A Machine Learning Approach to Software Requirements Prioritization. IEEE Transactions on Software Engineering **39**(4) (2013) 445–461
13. Berry, D., Gacitua, R., Sawyer, P., Tjong, S.F.: The Case for Dumb Requirements Engineering Tools. In Regnell, B., Damian, D., eds.: 18th International Working Conference on Requirements Engineering: Foundation for Software Quality. Lecture Notes in Computer Science, Berlin Heidelberg, Springer (2012) 211–217
14. Berry, D.M.: Evaluation of Tools for Hairy Requirements and Software Engineering Tasks. In: 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW). (2017) 284–291