Tool Support for Restricted Use Case Specification: Findings from a Controlled Experiment

Markus Weninger*, Paul Grünbacher†, Huihui Zhang‡, Tao Yue§, Shaukat Ali¶
*†Christian Doppler Laboratory MEVSS
†Institute for Software Systems Engineering, *Institute for System Software
Johannes Kepler University Linz, Austria
Email: markus.weninger@jku.at
‡School of Computer Engineering, Weifang University, China
§¶Simula Research Laboratory, Norway
§Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China
¶ Information Systems Architecture Science Research Division, National Institute of Informatics, Tokyo 101-8430, Japan

***Abstract*—Evidence has shown that the use of restricted natural languages can reduce ambiguities in textual use case specifications (UCSs). Restricted natural languages often come with specific editors that support particular use case templates and provide enforcement of the language's restrictions. However, whether restriction enforcement facilitates the definition of UCSs as compared to an editor without such support is a fundamental question to answer. To this end, we report results of a controlled experiment in which we compared two approaches for defining restricted UCSs: (i) a specific Restricted Use Case Modeling (RUCM) tool that supports restriction enforcement; and (ii) a general Office Word UCS template without such enforcement. We compared both approaches from multiple perspectives including restriction misuse, understandability, and restrictiveness. Results show that the restriction misuse rates are generally low, which indicates the usefulness of the RUCM, independent of the use of the editors. The results also indicate that the RUCM tool eases the application of more complex restrictions. We also found that the participants profited from extensive training prior to the experiment. The experiment participants further showed their strong willingness to recommend the RUCM tool to others and to use it in the future, which was not the case for the Office Word template.**

***Index Terms*—Use case modeling, restricted natural language, controlled experiment.**

I. Introduction

Use case modeling is one of the main approaches for specifying requirements and a number of use case modeling solutions have been proposed [1]–[3], some of which have also been implemented as open source and commercial tools [4]–[6]. As discussed in [7], use case modeling is one of the most well-applied modeling methods in practice.

A use case modeling solution often consists of use case diagrams and textual UCSs. Use case diagram notations have been standardized in the Unified Modeling Language (UML) [8] in the 1990's and are therefore implemented in most of the modeling frameworks (e.g., Papyrus [4], IBM RSA [5], MagicDraw [6]). However, there is no standardized solution for describing textual UCSs [9]. Many efforts have been made in the past to propose different use case templates [10], [11], to reduce ambiguities of textual UCSs [12], and to enable automation [13]. One such effort resulted in the use case modeling solutions *Restricted Use Case Modeling* (RUCM) [14], [15] and RUCM for Real-Time (RUCM-RT). RUCM has been used for education purposes in several international universities and has also been applied in various industry domains such as avionics [16].

RUCM is composed of a use case template to structure and document UCSs, as well as a set of restrictions. These essentially constrain the way users can write and structure UCSs (on the use of English and applying pre-defined keywords). The motivation of devising RUCM is to reduce ambiguities in textual UCSs and to facilitate the automated generation of downstream artifacts such as analysis models and test cases. In the past, the RUCM template and restrictions have been evaluated in terms of their applicability and the capability of facilitating the generation of UML analysis models via two controlled experiments conducted at Carleton University, Canada [14]. Results of the controlled experiments showed that the RUCM methodology (with its template implemented as tables in Office Word, named *Office Word editor* in the rest of the paper) was easy to apply. The results also showed that a higher quality of the UML analysis models was achieved when RUCM models were used as input compared to a non-restricted use case modeling solution. At the time when these controlled experiments were conducted, no dedicated tool support existed for the RUCM methodology, except for the general Office Word editor.

However, tool support is considered an important factor in promoting software engineering methodologies in practice, as well recognized in the software engineering community, especially in the area of model-based engineering. Therefore, a dedicated RUCM tool has been developed [17] and applied in various settings. However, there is still a lack of evidence showing the usability and applicability of the RUCM tool. We thus conducted a controlled experiment, reported in this paper, to test if the RUCM tool helps to enhance the user experience and applicability of the RUCM methodology and to understand which aspects of the current RUCM tool could further be improved. Our controlled experiment involved 41 students, who received prior training about RUCM in the context of a

compulsory master-level course on Requirements Engineering, which is part of the software engineering program at the Johannes Kepler University Linz, Austria. Results revealed no statistically significant differences between the RUCM tool and Office Word editor regarding the *Misuse* rate of applying the different RUCM restrictions. Still, results indicate that the RUCM tool eases the application of more complex restrictions. Our results also show that the participants' ability to correctly apply the RUCM restrictions strongly profited from extensive training prior to the experiment. Overall, the observation shows that RUCM, in combination with prior training, is generally easy to apply, yet the participants showed their strong willingness to recommend the RUCM tool to others and to use it in the future, which was not the case for the Office Word editor.

The rest of the paper is organized as follows. Section II summarizes the background of the RUCM methodology including the RUCM restrictions. Section III describes the goals and research method of the controlled experiment. Section IV summarizes the results for the investigated research questions. Section V discusses interesting findings and the threats to validity of the experiment. In Section VI we discuss the related work. Finally, we present our conclusions and give an outlook on future work.

II. Background on RUCM

The *Restricted Use Case Modeling* (RUCM) methodology has been proposed by Yue et al. [14], [15] to facilitate the automated and semi-automated generation of other artifacts such as UML analysis models [8], executable test cases [18], and use case scenarios for facilitating requirements inspections [19]. RUCM provides a use case template alongside 26 restrictions to guide the textual specification of use cases [14]. The approach aims to ease the practical uses, to reduce ambiguity, and to facilitate automated analyses and generation of downstream artifacts.

Controlled experiments have been conducted by Yue et al. [14], [15] to evaluate RUCM in terms of its ease of use and the quality of manually derived analysis models. Results showed that RUCM is overall easy to use and results in significant improvements [14] regarding the understandability of UCSs compared to the use of a commonly applied use case template.

A RUCM UCS has one basic flow, which can have one or more alternative flows. An alternative flow always depends on a condition occurring in a reference flow, which is either the basic flow or an alternative flow itself. There are three types of alternative flows: *i)* a *specific alternative flow* refers to a specific step in the reference flow; *ii)* a *bounded alternative flow* refers to more than one step in the reference flow; and *iii)* a *global alternative flow* (called general alternative flow in [20]) refers to any step in the reference flow. For specific and bounded alternative flows, the RFS (Reference Flow Step) keyword specifies the reference flow step numbers.

The 26 restrictions of RUCM are classified into three categories: *i)* 16 restrictions on the use of natural language (R1-R16); *ii)* nine restrictions regarding the use of keywords for specifying control structures (R17-R25); and *iii)* one restriction on that each flow of events should have its own postcondition (R26).

Due to its nature of being a general purpose tool, the Office Word editor does not provide any RUCM-specific support beyond the structure of the template as a table, nor does it enforce any of the RUCM restrictions. The RUCM tool, on the other hand, enforces certain criteria regarding all nine keyword restrictions, as well as R2 ("Describe the flow of events sequentially") and R26 ("Each flow of events should have its own postcondition"), by informing the users about detected restriction violations. Specifically, for the keyword restrictions R17 (INCLUDE <*use case name*>) and R18 (EXTENDED BY <*use case name*>), the tool automatically checks that valid names are specified following the keywords. When applying the RFS keyword (R19), a template enforces that the keyword is followed by the name of a reference flow and step number(s). Regarding R20 (IF-THEN-ELSE-ELSEIF-ENDIF), the tool requires users to specify conditions after IF and ELSEIF. To enhance the user experience, the tool also provides auto-indentation for the inner blocks. The keyword MEANWHILE (R21) has to connect two sentences, and a condition should be specified following VALIDATES THAT (R22). When applying DO-UNTIL (R23), the user needs to specify a condition following UNTIL and a list of steps after DO. Again, auto-indentation is provided. The keyword ABORT (R24) forms a sentence by itself; nothing else should be added to it. The application of RESUME STEP (R25) is very similar to RFS: a template enforces to refer to a step of the same or different flow of events. Whenever one of these constraints is violated, the tool notifies users with a warning.

III. Experiment Planning

We adapted the experiment reporting template proposed in [14]. All key aspects of the experiment are described.

A. Experiment Definition

The overall objective of the controlled experiment was to evaluate RUCM for defining UCSs with the RUCM tool enforcing certain RUCM restrictions (cf. Section II). Specifically, we used the RUCM tool and compared it to the Office Word editor not enforcing restrictions. We investigated different aspects such as *Applicability*, *Understandability*, *Restrictiveness*, and *Learning Effort*. We pursued the overall objective by exploring the following three goals, as formulated by using the Goal-Question-Metric template [21]:

Goal 1: *Analyze* the RUCM tool *for the purpose of* evaluating its applicability for defining UCSs conforming with the RUCM restrictions *from the point of view of* requirements engineers (in the context of graduate students defining use case models). For that purpose we used four dependent variables originally defined in [14] for our experiment: *Restriction Misuse*, *Understandability*, *Applicability*, and *Restrictiveness*.

Table I
EXPERIMENT DESIGN.

Goal	Round	RQ	Task (3 ¼ hours in total)	Group A	Group B
G1 and G3	1	1, 4	Task 1: Answering the Pre-Lab Questionnaire (5 mins) Task 2: Setup (reading specifications, opening tool, etc.) (5-10 mins) Task 3: Specifying UCs (60 mins)	Tool Pre-Lab Q. CS1 + Tool	Word Pre-Lab Q. CS1 + Word
		2	Task 4: Answering the Comprehensive Questionnaire (10 mins)		
		4	Task 5: Answering the Post-Lab Questionnaire (5 mins)	Tool Post-Lab Q.	Word Post-Lab Q.
G1 and G3	2	1, 4	Task 1: Answering the Pre-Lab Questionnaire (5 mins) Task 2: Setup (reading specifications, opening tool, etc.) (5-10 mins) Task 3: Specifying UCs (60 mins)	Word Pre-Lab Q. CS2 + Word	Tool Pre-Lab Q. CS2 + Tool
		2	Task 4: Answering the Comprehensive Questionnaire (10 mins)		
		4	Task 5: Answering the Post-Lab Questionnaire (5 mins)	Word Post-Lab Q.	Tool Post-Lab Q.
G2	3	3	Task 6: Comparison Questionnaire (5 mins)		

We investigated the impact of using the RUCM tool on these variables.

Goal 2: *Analyze* the RUCM tool *for the purpose of* evaluating its perceived applicability *from the point of view of* requirements engineers (in the context of graduate students defining use case models). Specifically, we investigated how the subjects perceive the applicability of RUCM and its tool. We defined five dependent variables: *Learning Effort*, *Ease to Apply*, *Subjective Usefulness*, *Willingness to Use*, and *Willingness to Recommend*.

Goal 3: *Analyze* the RUCM tool *for the purpose of* evaluating the required effort *from the point of view of* requirements engineers (in the context of graduate students defining use case models). In particular, we investigated the time required to use the RUCM tool using the dependent variable time effort as well as the perceived effort for learning and understanding the tool.

For each of the three goals, we defined three independent variables: *Method*, with two treatments corresponding to the use of RUCM_Word and RUCM_Tool; *System*, i.e., the selected case studies; and *Order*, i.e., the sequence of using the two treatments.

B. Context Selection and Experiment Participants

We conducted the experiment during a compulsory master-level course on Requirements Engineering, which is offered as part of the Software Engineering master program at the Johannes Kepler University Linz, Austria. The course covers the role of requirements in the software life cycle and a variety of methods for eliciting, analyzing, negotiating, documenting, and validating requirements. Forty one students registered for the course in the winter term 2016 and participated in the experiment.

Before the experiment, the students attended a lecture (by the first two authors of the paper) explaining the meaning and use of the 26 RUCM restrictions in the requirements specification process. As part of the experiment preparation, the students further had to complete a homework assignment on two different use case scenarios with the goal of specifying UCSs conforming to the RUCM restrictions, once using the RUCM tool and once using the Office Word editor. The scores of the assignment were used to form two equally strong groups of the students for the experiment. A more detailed discussion on how this training influenced the experiment results is given in Section V.

The experiment was conducted as part of a series of compulsory laboratory exercises during the course. We selected the two case study systems *SmartHome* (3 use cases, medium complexity) and *OrderProcessing* (2 use cases, higher complexity), as we regarded their domains as easy to understand by the students. Also, we assumed that the subjects would be able to finish these specification tasks within a 3-hour laboratory session. The subjects had no prior knowledge about the goals of the experiment, the selected case studies, and the use cases. Moreover, the subjects were aware that the tool was not developed by the people grading the lecture.

C. Hypotheses Formulation and Research Questions

We formulate hypotheses for each dependent variable of our research goals and for the two treatments: *RUCM_Word*, denoting the use of the RUCM methodology with an Office Word editor, and *RUCM_Tool*, denoting the use of the RUCM methodology with the RUCM tool. For each dependent variable, the null hypothesis (H_0) to be tested is that there are no significant differences between RUCM_Word and RUCM_Tool. The alternative hypothesis (H_1) for each dependent variable is two-tailed and says that there are significant differences between RUCM_Tool and RUCM_Word.

Specifically, we investigate the following research questions:

- RQ1 – Is RUCM_Tool significantly different with RUCM_Word in terms of enabling the conformance of the devised RUCM models against the RUCM restrictions? (cf. Goal 1)
- RQ2 – Is RUCM_Tool significantly different with RUCM_Word in terms of facilitating the understandability, applicability, and restrictiveness of the RUCM restrictions? (cf. Goal 1)
- RQ3 – Is RUCM_Tool significantly different with RUCM_Word from the subjects' subjective opinions on the learning effort, ease to apply, usefulness, willingness to use, and willingness to recommend? (cf. Goal 2)

- RQ4 – Is RUCM_Tool significantly different with RUCM_Word in terms of required time effort for completing USCs? (cf. Goal 3)

D. Experiment Design

We used a within-subject experiment design [22], i.e., each subject was exposed to more than one of the treatments being tested [23] (in our case two treatments), as opposed to a between-subject design exposing subjects to a single treatment only. A within-subject design is especially suited if the number of subjects is rather low, yet one has to counter possible carryover effects such as a practice effect by counter-balancing the order of treatments or by providing extensive practice prior to administering any treatments [24], which we both ensured in our experiment design.

The experiment design presented in Table I consists of three rounds, in which the experiment subjects were divided into Group A and Group B. Round 1 has five tasks (Task 1 to Task 5), as shown in the table. Each group was given a treatment (i.e., RUCM_Tool for Group A and RUCM_Word for Group B) to first solve the *SafeHome* case study. In Round 2, the two groups were swapped to work on the different treatment (i.e., RUCM_Word for Group A and RUCM_Tool for Group B), this time working on the *OrderProcessing* case study. In Round 3, all subjects were asked to complete a comparison questionnaire, which we used to collect subjective opinions of the participates after having experienced the two treatments in Round 1 and Round 2.

E. Instrumentation

We used the following experiment instruments for the different tasks[1].

Task 1, Task 2, and Task 5: We designed a pre-lab questionnaire with seven four-point Likert scale questions to collect information about the subjects' prior background on use case modeling and the RUCM methodology (Task 1).

After that initial assessment, the participants had 10 minutes to read the experiment guidelines and the specifications of the case studies. The specification of each case study contains a short description of the system and its actors, a use case diagram, as well as an informal description of the events of the involved use cases. The experiment guidelines describe the list of documents provided, the tasks to be completed, and the submission guidelines. For instance, the participants were asked to write down the time required to complete each UCS. The guidelines also reminded them to apply the RUCM restrictions. The subjects also had access to a document summarizing all RUCM restrictions (together with examples and counterexamples of applying them) for reference throughout the experiment.

We used a post-lab questionnaire to collect information about the subjects' perception about the experiment at the end of Round 1 and Round 2 (Task 5). For instance, we asked questions to determine if the students understood the assignment, how they felt when applying the RUCM restrictions using the two treatments (RUCM_Tool vs. RUCM_Word), if they had enough time to finish the tasks, and in which step they lost time in case they could not complete a task.

Task 3: This task required the subjects to document UCSs by applying RUCM. As input, the participants received partially filled UCSs complying with the RUCM use case template. The UCSs contain descriptions for only the following fields of the template: Use Case Name, Brief Description, Primary Actor, Secondary Actor, Dependency, and Generalization. The rationale was to provide an overview of high-level requirements (e.g., thanks to the brief description), to ensure that UCSs were consistent with the use case diagram, and to let the participants then focus on defining the flows of events – the most complex part of UCSs most relevant for the RUCM restrictions.

Task 4: We used the comprehension questionnaire presented in [14] to capture the participants' subjective opinions on the RUCM restrictions. It was designed for the participants to characterize each restriction according to three measures: *Understandability*, *Applicability*, and *Restrictiveness*. The first question is a 'yes/no' question helping to determine whether the participants were able to understand each restriction when performing Task 3. The second question required the participants to assess the straightforwardness of applying the restrictions using a four-point Likert scale. The third statement measures the perceived restrictiveness of each restriction, again using a four-point Likert scale.

Task 6: After the subjects had gained experience with both treatments, i.e., RUCM_Tool and RUCM_Word, we asked them in Round 3 to complete a comparison questionnaire containing eight four-point Likert scale questions and one open-ended question on the applicability of the two treatments. The questionnaire covers the learning effort, the ease of use, the usefulness for applying the RUCM restrictions as well as personal preferences and recommendations regarding the use of the two editors.

F. Evaluation Measurement and Data Collection

Task 3 and Task 4 evaluated each RUCM restriction based on four measures: *Understandability*, *Applicability*, *Restrictiveness* (following the definitions in [14]), and *Restriction Misuse* for evaluating both RUCM_Word and RUCM_Tool. The experiments reported in [14] only evaluated RUCM_Word. Due to space limitation, please refer to Section 3.6.1 of [14] for the definitions of the first three measures. *Restriction Misused* will be introduced in Section IV-A. To evaluate the cost required to apply the two treatments we measured the *Time Effort* spent on specifying each UCS (in minutes).

Task 6 compares the two treatments from the aspects of *Learning Effort*, *Ease to Apply*, *Subjective Usefulness*, *Willingness to Use*, and *Willingness to Recommend*. Recall that these questions were answered on four-point Likert scales, from 1 (Completely disagree) to 4 (Completely agree).

[1]The complete set of questionnaires is available at http://dx.doi.org/10.5281/zenodo.1460462.

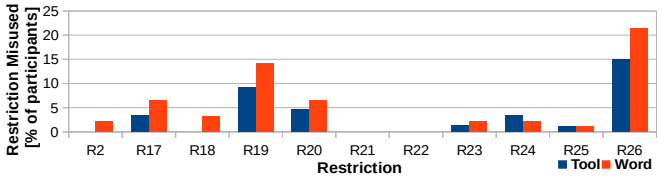

Figure 1. Percentage of participants that misused a given restriction at least once (*Restriction Misused*) when using RUCM_Tool and RUCM_Word.

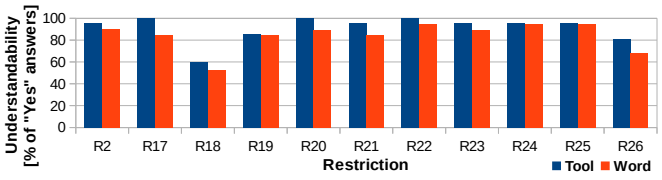

Figure 2. Percentage of *Understandability=true* after round 1 when using RUCM_Tool and RUCM_Word.

IV. Experiment Results and Analyses

In this section, we report and analyze the results based on our four research questions. As explained in Section II, the 26 RUCM restrictions are classified into restrictions on the use of natural language (R1 to R16), restrictions enforcing the usage of keywords for specifying control structures (R17 to R25), and one restriction enforcing that each flow of events should have its own postcondition (R26). The RUCM tool provides syntax-highlighting and auto-completion for the RUCM keywords. However, the tool does not enforce any restrictions on the use of natural language except for R2. Our analyses thus focus on the restrictions R2 as well as R17-R26, which are enforced by the RUCM tool.

A. Results for RQ1

RQ1 aims to answer if RUCM_Tool is significantly different from RUCM_Word in terms of enabling the conformance of the devised RUCM models against the RUCM restrictions. To do so, the participants had to define textual UCSs that conform to the 26 RUCM restrictions during Task 3 of Round 1 and Round 2. Each participant had to work on two case studies, once using RUCM_Word and once using RUCM_Tool (cf. Table I). To evaluate the participants' conformance to each of the 26 RUCM restrictions, we define the boolean metric **Restriction Misused**, which indicates if a given participant misused a particular restriction at least once in a UCS.

Figure 1 shows the *Restriction Misused* metric for the 11 restrictions enforced by the tool. The two restrictions R21 (MEANWHILE) and R22 (VALIDATES THAT) were never misused. Especially the results for R22 are interesting, since this rule had to be applied very often, but has never been misused due to its simple nature.

R2 ("Describe the flow of events sequentially") and R18 (EXTENDED BY) were misused only by participants using RUCM_Word. While it is possible in RUCM_Word to describe the flow of events in any way (e.g., by not using the tabular layout), RUCM_Tool automatically maintains the sequence of a flow of events, thus strictly enforcing R2. Even though R18 was not necessary for defining the UCSs, subjects using RUCM_Word misused the restriction in a few cases. RUCM_Tool prevented the users from wrongly applying this restriction.

All the other restrictions, except for R24 (ABORT), had lower *Restriction Misused* results when using RUCM_Tool. Most of the R24 violations were caused by alternative paths not properly ended with an ABORT statement. Future versions of the RUCM tool may enforce that every alternative path has to either end with ABORT (R24) or RESUME STEP (R25) to further reduce this restriction misuse.

R26, i.e., specifying a postcondition for each flow of events, is the only restriction that has a *Restriction Misuse* value above 15%. We believe that these relatively high misuse rates were caused by the difficulty of eliciting such postconditions when comparing with eliciting event flow steps. Training on how to elicit postconditions should be given to users in the future.

While the data reported above shows that on average the participants made fewer mistakes (average *Restriction Misused* of 4.2% using RUCM_Tool, 5.0% using RUCM_Word) there is no statistically significant difference. We executed Fisher's exact test with $\alpha = 0.05$ based on H_0*: RUCM_Tool and RUCM_Word perform equally in regards to Restriction Misuse* and the alternative hypothesis H_1*: RUCM_Word has a statistically significant different Restriction Misuse than RUCM_Tool*, the null hypothesis H_0 could never be rejected.

Among the 26 restrictions, except R12 and R26, all the other 24 restrictions have been misused by less than 15% of all the participants, 18 of which have even been misused by less than 5% of all the participants. This indicates that the participants, independently of the used treatment, in general only made very few to no mistakes, which also explains why there is no statistically significant difference between both treatments. We believe that this is a result of the students' in-depth training before the experiment. They used both treatments during a training homework assignment on two use case studies of similar complexity to those used during this experiment. Afterwards, they received detailed personal feedback on which restrictions they applied wrongly.

B. Results for RQ2

RQ2 investigates if RUCM_Tool significantly differs from RUCM_Word regarding the *Understandability*, *Applicability*, and *Restrictiveness* of the RUCM restrictions. The participants answered a comprehensive questionnaire in Task 4 of Round 1 and Round 2 to collect information about these metrics.

The **Understandability** reflects the subjective opinion of the participants on the ease of comprehending a particular restriction. It is rated as either *true* or *false*.

We only analyzed Round 1, to avoid bias caused by learning effects from Round 1 to Round 2 regarding *Understandability*. As pointed out in Section III-B we also formed equally strong groups based on the students' grades on the homework assignments, thus avoiding distorted results that may be caused by unbalanced groups.

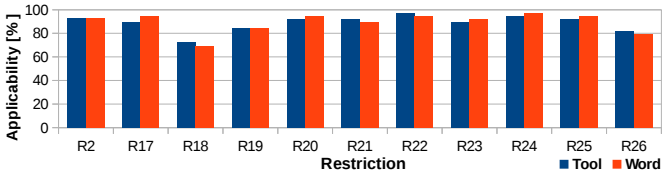

Figure 3. Percentage of *Applicability >= 3* when using RUCM_Tool and RUCM_Word.

Figure 2 shows the understandability score of Round 1 for R2 and R17 to R26, i.e., the 11 tool-enforced restrictions. In general, the understandability after Round 1 can be summarized as *well understood*. 21 of the 26 restrictions had an understandability score of over 90%, which again indicates that the intensive training before the experiment was useful to the participant. Surprisingly, the average understandability score was less than 90% only for one of the natural language domain restrictions, while the average understandability score was less than 90% for four of the nine RUCM keyword restrictions. R18 (EXTENDED BY) is the strongest outlier with regard to understandability. The reason may be that it was not a focus in the homework assignments before the experiment and also did not have to be used in the experiment itself.

Overall, tool support seems to ease the understanding of some of these restrictions. The average understandability score was 94.4% when using RUCM_Tool and 90.5% when using RUCM_Word. Five of the RUCM keyword restrictions had an understandability score of less than 90% when using RUCM_Word, while only two of these restrictions scored lower than that threshold when using RUCM_Tool. For the 11 RUCM restrictions shown in Figure 2, RUCM_Word never surpassed RUCM_Tool in terms of understandability. Yet, we could not determine statistically significant differences for any restriction using Fisher's exact test ($\alpha = 0.05$).

Another interesting aspect detected during data analysis is that the understandability score increased for seven out of nine RUCM keyword restrictions when Group B switched from RUCM_Word in Round 1 to RUCM_Tool in Round 2. On the other hand, none of the understandability scores increased when Group A switched from RUCM_Tool in Round 1 to RUCM_Word in Round 2. This supports our assumption that the tool fosters understanding the RUCM keyword restrictions.

We measure the **Applicability** on a four-point Likert scale ranging from 1 (Completely disagree) to 4 (Completely agree).

20 out of 26 restrictions received an applicability rating of 90% or higher (i.e., 90% of the students rated the applicability of the given restriction with 3 or 4 on the 4-point Likert scale). The figure shows that the applicability score does not differ much when using RUCM_Tool compared to when using RUCM_Word, at least no statistically significant difference can be detected using Fishers's exact test ($\alpha = 0.05$). Yet, we argue that this is again due to the intensive training before the experiment. The students have already gained knowledge about how to apply the restrictions in certain situations, and therefore the treatment did not further affect this applicability during the experiment.

Interestingly, the applicability scores were on average 5% higher in the second round, independently if RUCM_Tool or RUCM_Word was used in Round 1. This indicates some kind of warm-up effect during Round 1, i.e., the participants needed some time to get used to RUCM restrictions again.

Restrictiveness is also measured on a four-point Likert scale from 1 (Completely disagree) to 4 (Completely agree).

As expected, the participants feel more restricted by RUCM_Tool than by RUCM_Word for all RUCM keyword restrictions (Figure 4). While the Office Word editor does not enforce any restrictions, the RUCM tool prevents certain operations in the first place or informs the user in case of errors. Nevertheless, only five restrictions had a restriction rating over 20%, and all were lower than 35%. Also, Fisher's exact test ($\alpha = 0.05$) does not detect a statistically significant difference for any of the restrictions.

In Section V we discuss possible reasons for these higher restriction ratings when using RUCM_Tool. The feedback we received from the students suggests that the currently used mechanism to visualize mistakes and syntactical errors in the tool (a yellow triangle with a white exclamation mark next to the error message) is one of the main reasons for these scores. In particular, the tool currently lacks a feature informing users on how to resolve the reported problem.

C. Results for RQ3

RQ3 investigates if RUCM_Tool is significantly different with RUCM_Word regarding the subjective opinions on the learning effort, ease to apply, usefulness, willingness to use, and willingness to recommend. Data was collected using the comparison questionnaire from Task 6 in Round 1 and Round 2, using seven four-point Likert scale questions.

Learning effort: Learnability is an important aspect of usability [25]. The participants have been asked if it takes significantly more effort (in terms of time) to learn using the RUCM tool than the template for the Office Word editor. About three of four participants (72,5%) disagree with the statement that the RUCM tool takes more time to learn. Considering that the participants have years of experience on how to use the Office Word editor and its table-editing features, yet had no prior experience before the training assignments on how to use the RUCM tool, we regard this as a good result, with possible future improvements further reducing the tool's learning curve.

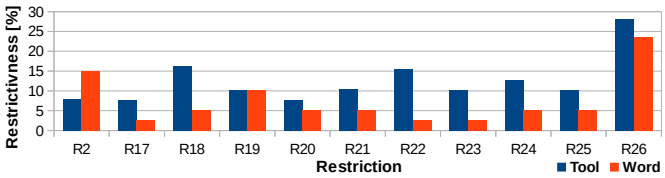

Figure 4. Percentage of *Restrictiveness >= 3* when using RUCM_Tool and RUCM_Word.

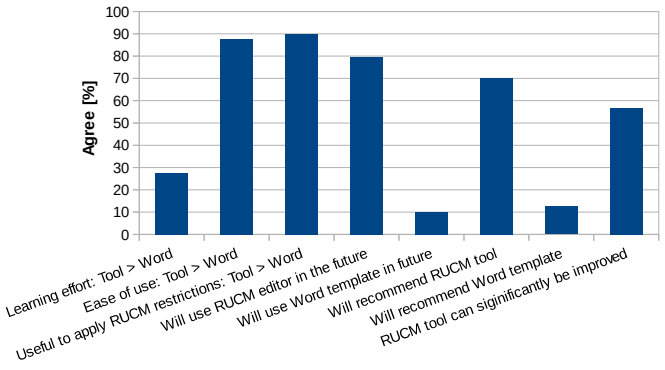

Figure 5. Rating on Learning Effort, Ease to Apply, Usefulness to Apply RUCM Restrictions, Use in the Future and Recommend to Others when using RUCM_Tool and RUCM_Word.

Ease to apply: The ease of use of a tool can greatly affect its future adoption by users [26]. Therefore, the participants were asked if they considered the RUCM tool significantly easier to apply, compared to the Office Word editor template. 87,5% of the participants agreed that the RUCM tool is easier to use than the Office Word editor. In particular, the automatic support for editing structural constructs, such as stacked IF-THEN-ELSE statements, was reported as being helpful in the tool, while the same task was regarded as tedious in the Office Word editor due to formatting problems.

Usefulness to apply RUCM restrictions: The third question evaluates if the RUCM tool is seen as more useful in terms of applying the RUCM restrictions, compared to the template for the Office Word editor. While Section IV-B showed no real higher applicability rating on a per-restriction basis, 90% of the participants agreed that the RUCM tool is more useful for applying RUCM restrictions in general than the Office Word editor. This indicates that although the participants felt well applying the RUCM restrictions independently of the treatment, they preferred using the RUCM tool. As said in the previous section, this may be due to support for structuring specific segments, or highlighting certain syntax elements and restriction violations.

Use in the future: The comparison questionnaire's fourth and fifth question ask if the participants plan to use the RUCM tool or the Office Word editor and its UCS template in the future. Nearly four-fifths (79.5%) of the participants agreed to use the RUCM editor in future to define use case models and specifications, while only one out of ten (10%) of the participants agreed to use the Office Word editor in future, again a clear results showing that the participants prefer the RUCM tool over the tabular Office Word editor.

Recommend to others: Finally, question six and seven asked the participants if they would suggest the RUCM tool or the Office Word editor to others in the future. 70% of the participants agreed to recommend the RUCM tool to others in the future, while only 12.5% of them agreed to suggest the Office Word editor and its template. These numbers are similar to those obtained for the planned future use and again confirm the positive perception of the RUCM tool by the participants.

D. Results for RQ4

We also asked the participants to track the time they took to finish the UCSs, allowing us to detect if RUCM_Tool reduces the time needed to complete the UCSs compared to RUCM_Word.

Unfortunately, the data was inconclusive with regard to this research question. Figure 6 shows a box plot displaying the time needed to finish every UCS (excluding participants that could not finish on time). Looking at the median times, both treatments took nearly the same amount of time for one use case, two use cases could be finished around 10% faster using RUCM_Word, while two use cases were completed around 20% faster using RUCM_Tool. Overall, these numbers give no clear picture if RUCM_Tool or RUCM_Word is to prefer with respect to speed.

We gained additional insights from the post-lab questionnaires. The participants indicated if they could not finish a UCS in time, and also named reasons for not completing it. The number of participants that were able to finish the exercise was nearly 50% higher for RUCM_Tool than for RUCM_Word in case of the *Safe Home* system. For the *Order Processing* system, the number was even 70% higher for participants using RUCM_Tool compared to RUCM_Word. Those who couldn't finish on time were asked if this was due to problems understanding the used treatment (14% for RUCM_Tool, 22% for RUCM_Word) or problems applying the used treatment (19% for RUCM_Tool, 41% for RUCM_Word).

It is further interesting to consider these results together with the complexity of the use cases and the restriction misused results. The Order Processing case study contains two larger use cases, while Safe Home contains three smaller use cases. The participants using RUCM_Tool finished on time and made fewer mistakes on larger and complex use cases than RUCM_Word users. This could be observed, for example, for complex use cases containing stacked IF-THEN-ELSE-ELSEIF-ENDIF constructs, which participants reported as difficult to use in RUCM_Word. We thus expect that tool support for such constructs will increase overall productivity and usefulness.

V. Discussion

A. Lessons Learned

Importance of training: The subjects received extensive training before the experiment for both treatments. They used

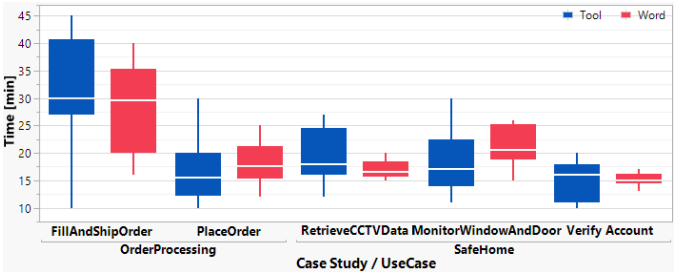

Figure 6. Average Time needed when using RUCM_Tool and RUCM_Word.

RUCM_Tool and RUCM_Word during a training homework assignment. They also received personal feedback in case they misused RUCM restrictions. We see clear signs that such training can have a strong influence in how well RUCM restrictions are used (i.e., it results in a lower restriction misuse) and how well the restrictions are accepted by the participants (e.g., the average applicability ratings are higher). This view is confirmed by the results of R18, which received less attention during the training, with a negative impact on these scores.

Natural language restrictions: Although the students saw the usefulness in using certain keywords (R17-R25) to create more structured UCSs, the students expressed doubts during the training on how putting restrictions on the use of natural language (R1-R16) would help to improve the quality of UCSs. Research has already been conducted on how to derive further artifacts such as analysis models [15] or test cases [18] from restricted use case models. These results show that restrictions on the use of natural language are essential to ease the process of natural language processing and further analyses. We are confident that training on natural language restrictions could be greatly enhanced by presenting tool features for automatically generating downstream artifacts based on a RUCM UCS.

RUCM editor improvements: During Round 3 participants were asked if and how the RUCM_Tool could be further improved. As part of this, 57% agreed that the RUCM tool can still be significantly improved. Additionally, they provided free-text feedback to suggest detailed improvements for RUCM tool capabilities: As already discussed earlier, many students suggested to display additional information in case an error is detected within a UCS. The yellow triangle currently used to signal an error in a certain line of the UCS seemed to irritate rather than encourage users. Suggestions regarding this issue range from simple error messages, as known from common IDEs, to step-by-step instructions on how to resolve common types of errors.

To prevent certain common errors, some participants suggested to add additional checks (i.e., making the tool even more restrictive) or to provide automatic step generation (e.g., automatically generating an alternative flow for every VALIDATES THAT sentence). Further, they requested additional navigation mechanisms (e.g., similar to CTRL + left click in most IDEs). As last point, the participants suggested keyboard shortcuts to ease navigation and to facilitate the editing of UCS.

Many participants complained that the tool is only available as an Eclipse plug-in and they suggested web-based alternatives. This was surprising given that the Eclipse IDE is widely used in different courses at their university. In particular, students criticized the text scaling, the low font resolution of Eclipse's user controls, and compatibility problems of the RUCM plug-in with the versions of Java and Eclipse installed on their machines.

Role of domain familiarity: Participants indicated in the Post-Lab questionnaires of Round 1 and Round 2 whether they had enough time to finish the UCSs. They further provided explanations in case they could not finish a UCSs. 34.6% of the participants said that could not finish the SafeHome system and 60.0% of the participants that could not finish the OrderProcessing system agreed that they spent too much time on understanding the respective case study system. This suggests that familiarity with the use case domain, which is most of the time the case for requirements engineers in the industry, may enable users to use RUCM even more efficiently.

B. Threats to validity

The key internal validity threat in our experiment is the choice of the experiment design. Given that the experiment was conducted as part of a course, we chose a within-subjects design after considering all the practical constraints such as the limited time for the laboratory sessions and the limited number of students.

Our main conclusion validity threat is about the sample size based on which we performed our analyses. In total 41 students participated in the experiment. To maximize the sample size for UCSs, we used two rounds with two different case studies. In this way, we managed to obtain twice the observations given the time constraints and limited availability of the number of students.

In our context, one construct validity threat is related to the comparison questionnaire, which we used in Round 3. The comparison questionnaire includes the same questions for both RUCM_Tool and RUCM_Word. Thus, there was no bias towards any of the treatments. A second construct validity threat is related to the use of measures for comparing the two treatments. In the experiment, we used the same measures for *Understandability*, *Applicability*, and *Restrictiveness* for comparing the two treatments and thus there was no bias introduced during evaluation.

External validity threats are common among controlled experiments, in general, which are related to the generalizability of the experiment results. We conducted our experiment with 41 participants using two different case studies due to practical constraints. Nonetheless, replications of the experiment with additional participants are needed in the future to generalize the results further. Another external validity threat is due to the use of students rather than professionals as the participants of the experiment. However, as studied in [27]–[29], there were no significant differences between trained students (as it was in our case) with professionals regarding various software engineering activities. Therefore, we are confident that this observation is also applicable to our context, i.e., applying the RUCM methodology. In other words, using students as the experiment participants reflects real application contexts, that is professionals applying the RUCM methodology.

VI. Related Work

We focus our discussion of related work on the following relevant aspects: use case modeling, empirical evaluations of use case modeling methodologies, and usability engineering approaches including Human Computer Interaction (HCI).

A. Use Case Modeling

Since first introduced in 1986 [30], use case modeling has been widely applied, which later on contributed to the standardization of the use case diagram notations in UML. However, there is no a standard way of specifying UCSs. In the literature, various types of use case templates (e.g., Cockburn [10], Jacobson et al. [2], Kruchten [3], Kulak et al. [31], and Larman [11]) have been proposed to structure and specify UCSs. In practice, use case modeling often drives the whole system/software development life cycles. Therefore, use case models can be used as an input to derive other artifacts such as analysis models and test cases. Considerable effort has been spent on this research stream such as Liu [32] and Somé [13]. The development of RUCM and RUCM based approaches [14], [15], [18], [33], [34] fall into this category of research streams. Some effort (e.g., Śmiałek et al. [12], Somé [13]) has been also put on proposing restrictions (also called writing guidelines) for specifying UCSs, as summarized in Yue et al. [9], to reduce ambiguities and facilitate automation. Based on the results of the literature review, Yue et al. proposed the 26 restrictions for RUCM (Section II).

B. Empirical Evaluation of Use Case Modeling Approaches

Various empirical studies have been conducted to evaluate the impact of applying restriction rules on the quality of UCSs. For instance, Achour et al. [35] investigated the impact of the CREWS rules on the completeness and structuredness of UCSs. The experiment results show that the application of the rules produced more complete and better structured UCSs. Furthermore, Phalp et al. [36] conducted an empirical study to compare two sets of writing rules: the CREWS rules and CR rules [37] (leaner than the CREWS rules), in terms of seven quality metrics such as coverage (a use case containing all the required information). The experiment results show that the CR rules results in less learning overhead and performs at least as well as the CREWS rules. A similar experiment was also conducted by Anda et al. [38] to compare different sets of guidelines by measuring the resulting UCSs in terms of their understandability, usefulness, and quality. All these experiments however evaluated restriction rules as a whole and none of them evaluated each rule individually.

Two controlled experiments conducted by Yue et al. [14], [15] to evaluate the impact of the 26 restrictions of RUCM individually for assessing the impact of applying RUCM on the understandability of UCSs and the quality of analysis models manually generated from them. Results of the two experiments showed that RUCM (with its template realized as tables for the Office Word editor) was easy to apply and achieved higher quality UML analysis models when RUCM models were used as the input, when comparing with a non-restriction based use case modeling solution. In [39], two recent studies were reported to assess eight use case templates (including the RUCM template) from two aspects, i.e., comprehension and learnability. Their results showed that RUCM's use case template is one of the three that have showed significantly better than the others evaluated during the study in terms of helping experiment participants to make more correct changes.

We, however, aimed to evaluate, via the controlled experiment reported in this paper, the usability and applicability of the RUCM tool (which implements the RUCM template and enforces some restrictions to a certain extent, see Section II for details), when comparing with the RUCM template given in the Word editor. We aim to test if the RUCM tool helps to enhance the user experience and applicability of the RUCM methodology and to understand which aspects of the current RUCM editor need to be improved.

C. Usability Engineering and Tool Evaluation

Our work is also related to the evaluation of usability, i.e., the "capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions" [40], [41]. The field of HCI distinguishes inspection-based approaches and test-based techniques to usability evaluation [42]: inspection methods aim at improving the usability of an interface design by checking it against some standard such as Nielson's five usability characteristics [43] or the Cognitive Dimensions framework [44]. For instance, several software engineering tools have been assessed using such an inspection-based approach [45]–[47]. However, compared to the tools assessed in these studies, the RUCM editors used in our experiments have comparably low complexity and interactivity. Thus, we decided to evaluate RUCM using a test-based approach directly involving end users. Our aim was to reveal differences between two editor variants, thus suggesting an experimental design with questionnaires.

VII. Conclusions and Future Work

Restricted natural languages with specialized tools claim to facilitate textual use case specifications by reducing ambiguities and facilitating automated generation of other artifacts, while at the same time maintaining the expressiveness of such methodologies. However, evidence is needed to support this claim. Towards this direction, we reported a controlled experiment, in which we compared a tool dedicated to Restricted Use Case Modeling (RUCM) with a RUCM template implemented as tables in an Office Word editor. There were three main findings: first, the usability of the dedicated RUCM tool needs to be improved to reduce perceived restrictiveness of the RUCM restrictions. Second, the participants showed strong willingness to use the RUCM tool and recommend it to other users in the future. Third, regarding errors, there were no significant differences between the RUCM tool and the Word template. Finally, from the results, we observed that in-depth training eases the application of RUCM, regardless of the editor. But, the RUCM tool proved more useful for complex specifications with complex structures and a large number of sentences. However, further experiments are needed to confirm this observation. The results of our experiments can also be used to improve the RUCM tool in the future.

ACKNOWLEDGMENTS

Tao Yue and Shaukat Ali were supported by the Zen-Configurator and MBT4CPS projects from the Research Council of Norway. Shaukat Ali was also supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST. Huihui Zhang is supported by Weifang Science and Technology Bureau (no. 2018GX004) and the PhD scholarship of Weifang University (no. 2018BS11).

REFERENCES

[1] B. Dobing and J. Parsons, "How UML is used," *Communications of the ACM*, vol. 49, no. 5, pp. 109–113, May 2006.

[2] I. Jacobson, *Object-oriented Software Engineering: A Use Case Driven Approach*. Pearson Education India, 1993.

[3] P. Kruchten, *The Rational Unified Process: An Introduction*, 3rd ed. Addison-Wesley Professional, 2003.

[4] "Papyrus (Eclipse foundation)," Available at https://www.eclipse.org/papyrus/index.php (02.02.2018).

[5] "IBM Rational Software Architect," Available at https://www.ibm.com/developerworks/downloads/r/architect/index.html (02.02.2018).

[6] "Magicdraw," Available at https://www.nomagic.com/products/magicdraw (02.02.2018).

[7] J. Hutchinson, J. Whittle, M. Rouncefield, and S. Kristoffersen, "Empirical assessment of MDE in industry," in *Proceedings 33rd International Conference on Software Engineering*. IEEE, 2011, pp. 471–480.

[8] "Unified Modeling Language version 2.5.1 (Object Management Group)," Available at http://www.omg.org/spec/UML/2.5.1.

[9] T. Yue, L. C. Briand, and Y. Labiche, "A systematic review of transformation approaches between user requirements and analysis models," *Requirements Engineering*, vol. 16, no. 2, pp. 75–99, 2011.

[10] A. Cockburn, *Writing Effective Use Cases*. Addison-Wesley, 2000.

[11] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Prentice Hall, 2004.

[12] M. Śmiałek, J. Bojarski, W. Nowakowski, A. Ambroziewicz, and T. Straszak, "Complementary use case scenario representations based on domain vocabularies," in *Proceedings Int'l Conference on Model Driven Engineering Languages and Systems*. Springer, 2007, pp. 544–558.

[13] S. S. Somé, "Supporting use case based requirements engineering," *Information and Software Technology*, vol. 48, no. 1, pp. 43–58, 2006.

[14] T. Yue, L. C. Briand, and Y. Labiche, "Facilitating the transition from use case models to analysis models: Approach and experiments," *ACM Trans. Softw. Eng. Methodol.*, vol. 22, no. 1, pp. 5:1–5:38, 2013.

[15] ——, "aToucan: an automated framework to derive UML analysis models from use case models," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 24, no. 3, p. 13, 2015.

[16] H. Zhang, T. Yue, S. Ali, J. Wu, and C. Liu, "A restricted natural language based use case modeling methodology for real-time systems," in *Proceedings of the 9th International Workshop on Modelling in Software Engineering*. IEEE Press, 2017, pp. 5–11.

[17] "Zen-RUCM Editor," Available at http://www.zen-tools.com/rucm/Editors.html (02.02.2018).

[18] T. Yue, S. Ali, and M. Zhang, "RTCM: a natural language based, automated, and practical test case generation framework," in *Proceedings of the 2015 International Symposium on Software Testing and Analysis*. ACM, 2015, pp. 397–408.

[19] H. Zhang, S. Wang, T. Yue, S. Ali, and C. Liu, "Search and similarity based selection of use case scenarios: an empirical study," *Empirical Software Engineering*, vol. 23, no. 1, pp. 87–164, 2018.

[20] K. Bittner, *Use case modeling*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[21] V. Basili, G. Caldiera, and D. Rombach, "Goal/Question/Metric paradigm," in *Encyclopedia of Software Engineering*, J. Marciniak, Ed. New York: John Wiley and Sons, 1994, pp. 528–532.

[22] Within-subjects designs. [Online]. Available: https://web.mst.edu/~psyworld/experimental/within_subjects.html

[23] G. Charness, U. Gneezy, and M. A. Kuhn, "Experimental methods: Between-subject and within-subject design," *Journal of Economic Behavior & Organization*, vol. 81, no. 1, pp. 1 – 8, 2012.

[24] A. Greenwald, "Within-subjects designs: To use or not to use?" *Psychological Bulletin*, vol. 83, no. 2, pp. 314–320, 3 1976.

[25] T. Grossman, G. Fitzmaurice, and R. Attar, "A survey of software learnability: Metrics, methodologies and guidelines," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09. New York, NY, USA: ACM, 2009, pp. 649–658.

[26] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS Quarterly*, vol. 13, no. 3, pp. 319–340, 1989.

[27] M. Höst, B. Regnell, and C. Wohlin, "Using students as subjects—a comparative study of students and professionals in lead-time impact assessment," *Empirical Software Engineering*, pp. 201–214, 2000.

[28] D. I. Sjöberg and E. Arisholm, "Evaluating the effect of a delegated versus centralized control style on the maintainability of object-oriented software," *IEEE TSE*, pp. 521–534, 2004.

[29] R. Holt, D. Boehm-Davis, and A. Shultz, "Mental representations of programs for student and professional programmers," in *Empirical Studies of Programmers: Second Workshop*, M. Gary, S. Sylvia, and E. S., Eds. Ablex Publishing Corp., 1987, pp. 33–46.

[30] I. Jacobson, "Object-oriented development in an industrial environment," in *ACM SIGPLAN Notices*, vol. 22, no. 12. ACM, 1987, pp. 183–191.

[31] D. Kulak and E. Guiney, *Use Cases: Requirements in Context*. Addison-Wesley, 2003.

[32] D. Liu, K. Subramaniam, B. H. Far, and A. Eberlein, "Automating transition from use-cases to class model," in *Proceedings IEEE Canadian Conference on Electrical and Computer Eng.*, 2003, pp. 831–834.

[33] C. Wang, F. Pastore, A. Goknil, L. Briand, and Z. Iqbal, "Automatic generation of system test cases from use case specifications," in *Proceedings of the 2015 International Symposium on Software Testing and Analysis*. ACM, 2015, pp. 385–396.

[34] M. Zhang, T. Yue, S. Ali, B. Selic, O. Okariz, R. Norgre, and K. Intxausti, "Specifying uncertainty in use case models," *Journal of Systems and Software*, vol. 144, pp. 573–603, 2018.

[35] C. B. Achour, C. Rolland, C. Souveyet, and N. A. Maiden, "Guiding use case authoring: Results of an empirical study," in *Proceedings of the 4th IEEE International Symposium on Requirements Engineering*. Washington, DC, USA: IEEE Computer Society, 1999, pp. 36–43.

[36] K. T. Phalp, J. Vincent, and K. Cox, "Improving the quality of use case descriptions: empirical assessment of writing guidelines," *Software Quality Journal*, vol. 15, no. 4, pp. 383–399, 2007.

[37] K. Cox, "Heuristics for use case descriptions." Ph.D. dissertation, Bournemouth University, 2002.

[38] B. Anda, D. Sjøberg, and M. Jørgensen, "Quality and understandability of use case models," in *European Conference on Object-Oriented Programming*. Springer, 2001, pp. 402–428.

[39] S. Tiwari and A. Gupta, "Investigating comprehension and learnability aspects of use cases for software specification problems," *Information and Software Technology*, vol. 91, pp. 22–43, 2017.

[40] A. Abran, A. Khelifi, W. Suryn, and A. Seffah, "Usability meanings and interpretations in iso standards," *Software Quality Journal*, vol. 11, no. 4, pp. 325–338, Nov 2003.

[41] "ISO 9126-1:2001 – Software engineering – Product quality," International Organization for Standardization, Tech. Rep., 2001.

[42] A. Holzinger, "Usability engineering methods for software developers," *Communications of the ACM*, vol. 48, no. 1, pp. 71–74, Jan. 2005.

[43] J. Nielsen, *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[44] A. F. Blackwell and T. R. G. Green, "Notational systems — the cognitive dimensions of notations framework," in *HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science*, J. M. Carroll, Ed. Morgan Kaufmann, 2003, p. 103–134.

[45] D. Mapelsden, J. Hosking, and J. Grundy, "Design pattern modelling and instantiation using DPML," in *Proceedings of the 40th International Conference on Tools Pacific: Objects for Internet, Mobile and Embedded Applications*. Australian Computer Society, Inc., 2002, pp. 3–11.

[46] R. Rabiser, M. Vierhauser, and P. Grünbacher, "Assessing the usefulness of a requirements monitoring tool: a study involving industrial software engineers," in *Proceedings of the 38th International Conference on Software Engineering, (Companion Volume)*, 2016, pp. 122–131.

[47] R. Rabiser, P. Grünbacher, and M. Lehofer, "A qualitative study on user guidance capabilities in product configuration tools," in *IEEE/ACM Int'l Conference on Automated Software Engineering*, 2012, pp. 110–119.