Comparing Comprehensibility of Modelling Languages for Specifying Behavioural Requirements

Grischa Liebel
Software Engineering Division
Chalmers | University of Gothenburg, Sweden
grischa@chalmers.se

Matthias Tichy
Ulm University, Germany
matthias.tichy@uni-ulm.de

***Abstract*—The selection of a suitable modelling language influences the success of software modelling. Several experiments comparing the comprehensibility of graphical modelling languages have been published. However, no published study comparing the comprehensibility of functional requirements modelled in different graphical modelling languages exists. This paper evaluates how two requirements modelled in a sequence-based notation, Modal Sequence Diagrams, and in a state-based notation, Timed Automata, compare with respect to comprehensibility. A controlled experiment with 22 student from an undergraduate course on software modelling was performed. Our results show no significant differences with respect to the comprehensibility of the two different languages, but subjects who answered the questionnaire for the sequence-based notation completed significantly more answers in the given time limit. These initial results indicate that choosing a modelling language for requirements modelling based on convenience does not significantly affect the understanding of the resulting requirements.**

I. Introduction

Choosing a visual modelling language in practice is typically dependent on previous experience with the modelling language or the availability of the respective modelling tools. While both aspects are certainly reasonable, other criteria are similarly important.

Comprehensibility is a commonly evaluated criteria of visual languages, e.g., for UML diagrams in [1], [2], [3], [4]. However, evaluation results can be contradicting as in [1] and [2]. Similarly to visual languages, the comprehensibility or understandability of software requirements specifications is the most common aspect evaluated in empirical requirements engineering studies [5]. At the same time, their practical value is often questioned [6]. A recent family of experiments by Abrahão et al. reports that providing sequence diagrams together with a natural language specification increases comprehensibility [7]. Additionally, the authors state that possible future work in this area could be "experiments to analyse the effect of different behavioural diagrams in the comprehension of software models".

As a first step in this direction, we conducted a controlled experiment in order to understand which behavioural diagrams perform superior to others for the specific case of modelling functional requirements with respect to comprehensibility. Specifically, we compared two modelling languages, Modal Sequence Diagrams (MSDs) [8] and Timed Automata (TA) [9]. We chose these two languages as they are representatives for sequence-based notations (MSDs) and state-based notations (TAs), as they are similar in terms of expressiveness and hence possible alternatives for expressing the same requirements, and as both have been applied in industrial case studies, e.g. [10], [11], [12]. Additionally, both languages were already used in a joint project with industrial partners. The experiment is based on an extensive and detailed requirements specification by a vehicle manufacturer that defines the behaviour of a software function to be realised by a supplier. Hence, the requirements specification is quite detailed and as such a reasonable candidate for modelling. We used 22 undergraduate students in a course on software modelling as subjects. Our results show no significant difference with respect to the comprehensibility of requirements modelled in the two languages, but requirements modelled in MSDs are significantly quicker to understand. This indicates that the current practice, selecting visual languages based on convenience, is in fact feasible with respect to the comprehension of the resulting requirements specifications. However, it might take longer to understand the requirements depending on the chosen language.

The remainder of this paper is structured as follows. In Section II, related literature is discussed. Section III covers the basics of the two visual modelling languages we used in the experiment. Section IV describes the experiment design, followed by a discussion of validity threats in Section V. Section VI presents the actual results and discusses them in depth. The paper is concluded in Section VII.

II. Related Work

In the context of the UML [13], a number of experimental studies have been published that compare different modelling languages with respect to comprehensibility. The comprehensibility of UML behavioural diagrams, namely sequence, collaboration, and state machine diagrams, in both real-time and management information systems is compared using a controlled experiment by Otero and Dolado [1]. The results show that sequence diagrams are more comprehensible for real-time systems than for management information systems. With respect to the answering speed, their data shows that sequence diagrams perform better than collaboration and state machine diagrams for both domains. As subjects, 31 undergraduate students are used in their study.

In contrast to this, Glezer et al. report that sequence diagrams are more comprehensible for management information systems than for real-time systems [2]. The authors mainly attribute this difference to the previous knowledge of the subjects, who were not experienced in real-time systems. In this study, the 76 student subjects performed the experiment in terms of a mandatory mid-term exam.

Nugroho investigates the impact of detail on the comprehension of UML Class, Sequence, Package, and Use Case diagrams in form of a controlled experiment with 53 graduate students [3]. The author reports that a low level of detail can lead to misinterpretations and that the subjects' knowledge did not have an impact on the comprehension.

Staron et al. report the results from four controlled experiments studying the impact of using UML stereotypes on comprehensibility conducted with 68 students and 4 professionals in total [4]. The studies show that stereotypes indeed improve the comprehensibility and the total and relative times for answering the used questionnaires.

Similar to the UML, comprehensibility is a commonly studied characteristic of requirement specifications. Condori-Fernández et al. present an evaluation of empirical studies until 2008 on requirements comprehensibility [6]. The authors conclude that while comprehensibility studies are common, many of them have practical limitations, such as using made-up examples instead of real specifications.

Kamsties et al. study how different specification techniques affect the comprehensibility of a software requirements specification, using a re-engineered specification of a bicycle computer [14]. The authors report that black-box specification techniques, describing a system by its externally visible behaviour, lead to a faster and more correct answering of the used instrument than white-box specification techniques, where the system is described by the behaviour between its entities.

Finally, there are a number of studies which investigate the comprehensibility of requirements modelled in or enhanced with visual modelling languages. Scanniello et al. study the effect on requirements comprehensibility when using SysML diagrams in addition to natural language, compared to only natural language requirements [15]. The authors use students as subjects in two controlled experiments and report that comprehensibility is increased when SysML diagrams are provided, whereas completion time for the comprehension task is unaffected.

A recent paper by Abrahão et al. reports a family of five experiments on the comprehensibility of functional requirements modelled with sequence diagrams in addition to the natural language specification [7]. Hereby, one experiment uses undergraduate students, two experiments use master students, one experiment uses doctoral students and one experiment uses professionals as subjects. Four out of five experiments show statistically significant support for improved comprehensibility when using sequence diagrams.

In summary, a number of experiments exist that investigate the comprehensibility of visual modelling languages, of requirements specifications, and of requirements represented in visual modelling languages. However, the outcomes vary and are sometimes even contradicting, e.g. in [1] and [2].

Additionally, we are not aware of any experiment comparing requirements represented by behavioural models only. This is a gap in knowledge, as requirements are typically on a more abstract level than for example software design and are, additionally, often intended to be read and understood by non-experts. Particularly, in the automotive domain, it is the usual process that detailed requirements specificions covering the behaviour of software components are defined by the vehicle manufacturer and subsequently sent to the supplier who needs to correctly understand and realise the specified behaviour. We are filling this gap with our contribution in this paper.

III. Background

In the following, we introduce the two compared modelling languages and illustrate them using sample models from our experiment. The models specify the behaviour for the case that a user wants to increase the speed of a wiper by one unit. Since both languages basically employ the same modelling notations to specify real-time aspects, we did not use any real-time aspects but instead focused on non real-time behaviour. Furthermore, a pilot of the experiment showed that the real-time aspects were too difficult to understand for the planned experiment. We plan a future experiment specifically targeting the real-time aspects.

A. *Modal Sequence Diagrams*

Modal Sequence Diagrams (MSDs) [8] are a recent variant of Live Sequence Charts (LSCs) [16] to model the behaviour of a set of objects. MSDs/LSCs are sequence diagrams that, by different modalities assigned to messages and conditions, allow to precisely describe scenarios with liveness (something good must happen) and safety (something bad must not happen) properties. Notably, LSCs and MSDs define how multiple scenarios can be active concurrently and synchronise on common events as well as activate and de-activate MSDs. This allows engineers to flexibly specify systems that fulfill different tasks at the same time.

One key advantage of MSDs/LSCs is that they can be executed with the play-out algorithm, which allows engineers and other stakeholders to understand the behaviour emerging from the interplay of the scenarios [17]. Furthermore, it is possible to analyse whether a set of scenarios can be realised, i.e., it does not contain contradictions or results in deadlocks.

Figure 1 shows a sample MSD used in our experiment. It specifies the communication between a user, a wiper controller as well as the actual wiper actuator. The sequence in the figure describes that (1) a request is sent to the wiper controller to increase the speed, (2) it is checked whether the wiper is in the state active, and (3) the controller sends a message to the actuator to increase the speed by one. If the check in step 2 fails, the MSD will be de-activated and not further executed. Once the first message in an MSD is executed (wiperRequest(WiperRequest::WIPER_INCREASE) in Figure 1), it is called *active*. After the last message, the MSD is

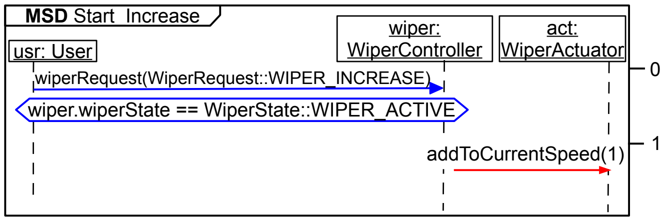


Fig. 1. Sample MSD

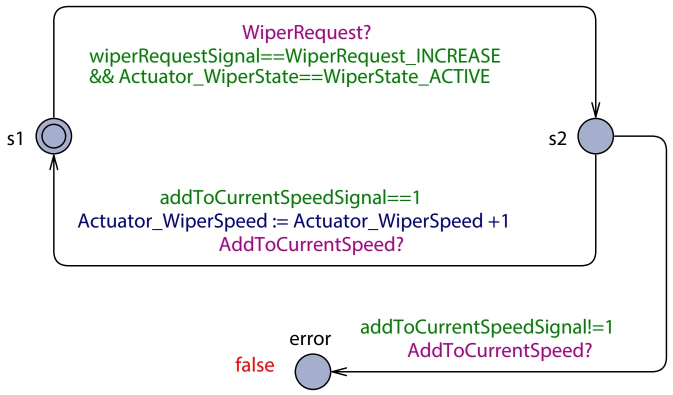

WiperRequest?
wiperRequestSignal==WiperRequest_INCREASE
&& Actuator_WiperState==WiperState_ACTIVE
s1
s2
addToCurrentSpeedSignal==1
Actuator_WiperSpeed := Actuator_WiperSpeed +1
AddToCurrentSpeed?
error
false
addToCurrentSpeedSignal!=1
AddToCurrentSpeed?

Fig. 2. Sample TA

deactivated again. The numbers on the right side describe the so-called *cut*, the positions in which an MSD can be.

The complete MSD model consists of a set of five scenarios covering the communication and conditions for the three mentioned objects.

B. Timed Automata

Timed automata [18] are a state-based formalism which extends finite automata with a set of real-valued variables called clocks as well as various real-time constraints. Several timed automata can be combined into a network of timed automata where different automata synchronise their behaviour by, so called, synchronisation channels. Synchronisation channels can be used as a means to specify synchronous message passing. Timed automata can be both simulated as well as verified for correctness using model checking.

Figure 2 shows the timed automaton for the wiper controller covering the increase wiper speed scenario as described previously for the MSD. It defines that if a wiper request for increasing the speed (condition: wiperRequestSignal==WiperRequest_INCREASE) using the synchronisation channel WiperRequest? is received and the wiper is active (condition: Actuator_WiperState==WiperState_ACTIVE), then the wiper speed is increased by 1 and a helper variable is set to 1 (addToCurrentSpeedSignal:=1). This helper variable is used in another automaton for a long-press functionality.

The complete TA model consists of a network of five timed automata covering the communication and conditions for the three mentioned objects.

IV. Experiment Design

The evaluation of comprehensibility of the two considered modelling languages used for requirements engineering was performed using a controlled experiment. The goal of this experiment is formulated as follows, using the Goal/Question/Metric paradigm [19]:

- Analyse *requirements modelled in two different modelling languages* for the purpose of *comparison* with respect to *comprehensibility* from the point of view of *software developers* in the following context: *application (verification and validation), subjects (students).*

We used a between-subject randomised design with two treatments [20]. The between-subject design was chosen to avoid learning effects. The treatments are the used modelling language, namely *MSD* and *TA*. MSDs, a variant of Live Sequence Charts [21], are sequence diagrams with assigned modalities that allow the expression of liveness and safety properties, and real-time constraints. Timed Automata are a modification of Finite Automata for the specification and verification of real-time systems. Hence, both languages are very similar in terms of expressiveness. However, *MSD*s use a scenario-based description covering multiple objects in one *MSD*, whereas *TA* use a state-based description covering a single object in one *TA*. *MSDs* were chosen in order to have a sequence-based language with executable semantics, in contrast to UML Sequence Diagrams, and with the possibility to model required and forbidden behaviour. In order to not introduce any bias, we chose *TA* as a second modelling language, as the language had not either been introduced during the course. Both languages are used without their timing functionality. Subjects were assigned randomly one of the two treatments. In the following subsections, the details of the experiment design are presented.

A. Subjects

We performed the experiment with 22 students from an undergraduate course on software modelling. This is due to availability reasons, as we had a scheduled university course in the end of 2014 in which we could perform the experiment. The students had basic knowledge of UML, as the experiment was performed towards the end of the course. Both modelling languages were only introduced prior to the experiment in a single 45-minute lecture. However, the students were introduced to similar languages earlier in the course, namely to UML sequence diagrams and to UML state machine diagrams.

B. Instrumentation

As a basis for the experimental objects, which we used in the study, we selected requirements from a real-life project within the automotive domain from an industrial partner. The selected requirements describe joint behaviour, i.e. the requirements are not entirely independent. As these requirements are confidential, we abstracted them and changed their actual content resembling a car wiper specification. However, we ensured that the complexity and the logic is comparable. These requirements were then modelled by the main author of this paper using

MSD and *TA*. The resulting experimental objects consist of two requirements models, S_{MSD} and S_{TA}, consisting of five diagrams each. The diagrams specify the activation of a car wiper in slow mode and in fast mode, the increase of the wiper's speed in two different ways, and the deactivation of the wiper. Additionally, the experimental objects contained a single page describing the context of each treatment. For the MSD specification, this consisted of a UML class diagram and an UML object diagram, and for the TA specification, this page contained the system declarations.

Finally, the instrument contained one page of syntax and semantic explanation additionally to the introduction lecture and a questionnaire. In turn, the questionnaire consisted of a pre-experiment part, collecting demographic data about the subjects (including subjects' knowledge regarding modelling languages), a post-experiment part, collecting subjective judgment, and the actual measurement questionnaire consisting of 12 questions targeting the subjects' understanding. The pre- and post-experiment questionnaires were used to judge whether previous experience, understanding of the introduction lectures, or other factors might have affected the dependent variables. Due to space limitations, we only discuss the data obtained from these questionnaires briefly in Section VI. Each of the 12 questions consisted of an initial state of the system and a number of executed messages or commands. Then, 2 sub-questions were asked. The subjects first had to answer whether the execution violated the requirements or not. Additionally, we asked in which state the system was after the execution (or right before the requirements violation), either by asking for the system's variable values or by asking for the active cuts/states of each diagram. Both sub-questions were awarded with one point each. The second sub-question was only counted if the first sub-question was correct, as it was otherwise already clear that the subject had wrongly executed the requirements. An example question with solutions for both the MSD and the TA model is depicted in Figure 3.

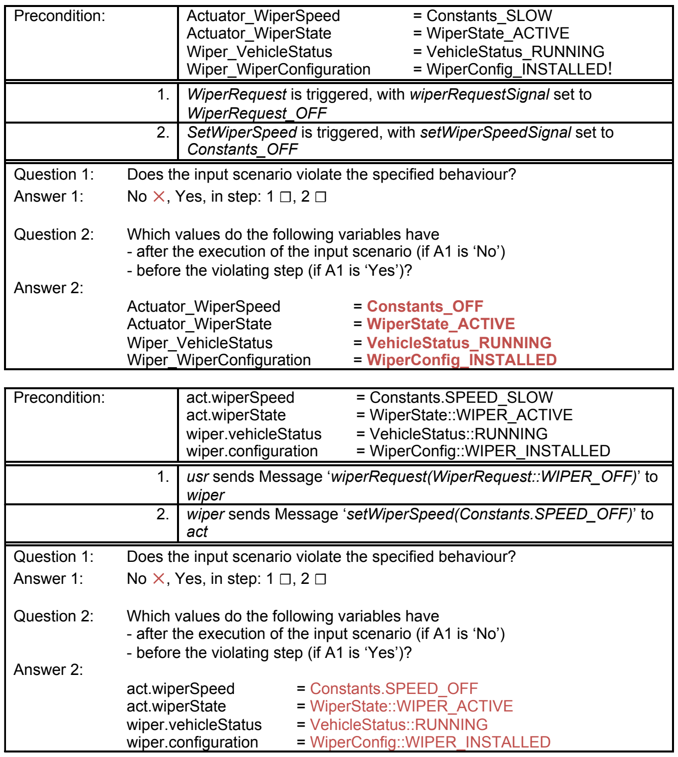
This questionnaire approach has been successfully applied in many similar studies, e.g. in [7], [15], [1]. The instrument, together with the resulting raw data, is published at http://www.grischaliebel.de/data/research/instrument_exp_msd_ta.zip.

C. Variables

There is only a single independent variable in the performed experiment. This is the used visual modelling language with the values *MSD* or *TA*. We measured the comprehensibility of the used requirements specification using three dependent variables:

Answered: The number of answered questions.
AScore: The average score achieved per answered question.
Score: The total score achieved for all 12 questions.

Instead of measuring the time, we decided to design the instrument in a way that it would be difficult to answer all questions in the given time frame. Therefore, we use the number of answered questions, **Answered**, instead of the needed time. We are foremost interested in using modelling languages for verification and validation purposes later on. Therefore, we



Precondition:	Actuator_WiperSpeed	= Constants_SLOW
	Actuator_WiperState	= WiperState_ACTIVE
	Wiper_VehicleStatus	= VehicleStatus_RUNNING
	Wiper_WiperConfiguration	= WiperConfig_INSTALLED!
1.	*WiperRequest* is triggered, with *wiperRequestSignal* set to *WiperRequest_OFF*	
2.	*SetWiperSpeed* is triggered, with *setWiperSpeedSignal* set to *Constants_OFF*	

Question 1: Does the input scenario violate the specified behaviour?
Answer 1: No ×, Yes, in step: 1 □, 2 □

Question 2: Which values do the following variables have
- after the execution of the input scenario (if A1 is 'No')
- before the violating step (if A1 is 'Yes')?

Answer 2:

Actuator_WiperSpeed	= **Constants_OFF**
Actuator_WiperState	= **WiperState_ACTIVE**
Wiper_VehicleStatus	= **VehicleStatus_RUNNING**
Wiper_WiperConfiguration	= **WiperConfig_INSTALLED**

Precondition:	act.wiperSpeed	= Constants.SPEED_SLOW
	act.wiperState	= WiperState::WIPER_ACTIVE
	wiper.vehicleStatus	= VehicleStatus::RUNNING
	wiper.configuration	= WiperConfig::WIPER_INSTALLED
1.	*usr* sends Message '*wiperRequest(WiperRequest::WIPER_OFF)*' to *wiper*	
2.	*wiper* sends Message '*setWiperSpeed(Constants.SPEED_OFF)*' to *act*	

Question 1: Does the input scenario violate the specified behaviour?
Answer 1: No ×, Yes, in step: 1 □, 2 □

Question 2: Which values do the following variables have
- after the execution of the input scenario (if A1 is 'No')
- before the violating step (if A1 is 'Yes')?

Answer 2:

act.wiperSpeed	= Constants.SPEED_OFF
act.wiperState	= WiperState::WIPER_ACTIVE
wiper.vehicleStatus	= VehicleStatus::RUNNING
wiper.configuration	= WiperConfig::WIPER_INSTALLED

Fig. 3. Example Question for TA Model (above) and MSD Model (below)

think that an accurate understanding of a specification is more important than speed. This is why we chose **AScore** as a metric for measuring how correct a question is answered in average. For completeness, we also added **Score**, which is related to the other two metrics by $Score = Answered * AScore$. We opted for comprehensibility instead of letting subjects create diagrams themselves, as this is easier and requires less training. Furthermore, the experiment targets models of functional requirements, not simply behavioural models in general. Therefore, we argue that comprehensibility is of particular importance, as the aim of requirements is to document what a system shall fulfill. Hence, correctly understanding these requirements is crucial.

An additional variable which can influence the outcome of the experiment is the subjects' knowledge regarding modelling languages and their domain knowledge in the automotive domain. While all students are from the same course, they might have different previous knowledge and experience. To address this issue we employed a pre-experiment survey which asked for background information, such as previous courses on modelling taken by the subject.

D. Hypotheses

In the course of the experiment, we used the following null and alternative hypotheses, $\mathbf{H_0}$ and $\mathbf{H_1}$, which we formulated as follows.

- $\mathbf{H_0}$: There is no significant difference between Modal Sequence Diagrams and Timed Automata with respect to comprehensibility of requirements specifications.

- **H_1:** There are significant differences between Modal Sequence Diagrams and Timed Automata with respect to comprehensibility of requirements specifications.

We evaluated the hypotheses separately for each of the dependent variables. Each of the variables was tested for significance using a non-parametric Mann-Whitney U test. Additionally, we tested for equality of variances for each of the variables using a Levene test in order to fulfill the assumptions of the Mann-Whitney U test. For both tests, we used a significance value of 0.05.

E. Operation

The experiment was piloted with two PhD students prior to execution. The instrument turned out to be too complicated and was therefore simplified furthermore to its current form.

The experiment was conducted in a 90-minute lecture. Participation was voluntary and the students received no benefits for the modelling course, such as bonus points or higher grades. In the first 45 minutes, both visual modelling languages were introduced. While this is a rather short time for introducing two new languages, we were limited to this time frame by the course schedule. Additionally, the subjects had previous knowledge in similar languages from the course, so that it was possible to related the newly introduced languages to that knowledge. Prior to the introduction lecture, we already handed out the experimental objects, so that the subjects knew which treatment they would receive and could concentrate on that language during the lecture. Additionally, they could familiarise themselves with the model. The subjects were encouraged not to share or exchange the objects with each other. After the introduction lecture, we handed out the remaining parts of the instrument, namely the questionnaires and the syntax help. Subjects then received 3 minutes for filling out the pre-experiment questionnaire, 40 minutes to fill out the experiment questionnaire, and finally 2 minutes for the post-experiment questionnaire.

V. Validity

We will in the following discuss means which we took in order to ensure validity. We use the four aspects of validity as presented in Wohlin et al. [20].

A. Construct Validity

In order to avoid *inadequate preoperational explication of constructs*, we have explicitly defined what 'comprehensibility' means with respect to our study. Also, it is clearly defined that a higher score in any of the three dependent variables means a better result for that variable. Our dependent variables do not require any human judgment and are therefore objective. *Mono-operation bias* can currently not entirely be ruled out, as we only used one experimental object. We are planning to replicate the experiment with another requirements specification in the future in order to address this. *Mono-method bias* is addressed by asking two sub-questions for each of the 12 experiment questions. While the first of the two sub-questions is a simple yes/no question, it is an additional check whether the subject has correctly understood the model. If this one is already incorrect, we automatically awarded 0 points to the second sub-question as well. Additionally, the second sub-question was much harder to get right by chance.

B. Internal Validity

In order to avoid maturation or learning effects, subjects were only allowed to participate in the experiment once and only in one group, and were not allowed to exchange information with other subjects during the experiment. Additionally, we used a pre-experiment questionnaire in order to assess the subjects domain and modelling knowledge, which might affect the outcome. While all students came from the same course, they had different previous experience with respect to software modelling and requirements engineering. We also assured that the subjects voluntarily participated in the experiment, by not giving rewards in the form of improved course grades or similar, in order to avoid compensation rivalry or demoralisation. However, we can not entirely rule out that some subjects participated to win our appraisal later in the course. The fact that we used volunteers might bias the results, as they could have been more motivated than the average.

C. External Validity

We used parts of a real-life specification instead of a toy example for the experiment instrument. However, the requirements had to be abstracted as the original specification is confidential. Additionally, while modelling the requirements, we had to ensure that both treatments were modelled in the same way and exhibited the same behaviour. This could have lead to one of the treatments being modelled in a way which would not happen in practice, and thus limit generalisability. We tried to reduce this threat by iteratively discussing and improving the instrument among the authors of this paper. Additionally, the fact that we used student subjects possibly limits the generalisability to an industrial context. Finally, the specification is based on an automotive requirements specification, which can limit the generalisability to other domains.

D. Conclusion Validity

We tried to avoid ambiguous wording of questions in the questionnaire by iteratively reviewing and improving it. Additionally, we performed a pilot experiment with two PhD students prior to the actual experiment, in order to improve both the introduction material and the questionnaire. *Reliability of treatment implementation* is given, as the introduction lecture was only given once for the actual experiment. We did only perform statistical tests on the three dependent variables, which were defined up-front, and did not *fish for results* [20].

VI. Results and Discussion

In the following, we will discuss first the demography of the subjects participating in the experiment. Afterwards, we present and discuss the results of the hypothesis testing for the experiment. Finally, we finish with a discussion of the post-experiment questionnaire.

A. Demographic Data

Out of the 22 subjects, 19 are Bachelor students and 3 are Master students. This can be explained through the fact that the course in which we performed the experiment is on Bachelor level, but can be taken as an elective course by first year Master students. All 3 Master students were randomly assigned the *MSD* treatment. Out of 22 subjects, 13 have a secondary school degree, 7 a Bachelor degree, 1 a Master degree, and 1 subject another degree as their highest degree. This means that 5 subjects on Bachelor level are already in possession of a Bachelor degree, and one Master student already has a Master degree. While this is certainly possible, it might also be caused by misunderstanding the question. Most subjects already had previous courses on related topics, such as Object-oriented programming or Software Architecture. Only 6 subjects stated to not have taken any related courses previously. Additionally, we asked the subjects for their professional experience in developing software, in modelling software, and in requirements engineering. In both modelling software and in requirements engineering, only 3 subjects answered that they had previous professional experience, ranging from half a year to three years of experience. In addition to this, 9 subjects stated that they have professional experience in software development, with one subject each stating 0.3 years, 1 year, and 8 years of experience, and 3 subjects each stating 2 and 3 years of experience.

B. Experiment Results

The experiment was conducted on 4th December 2014 at Chalmers University in Gothenburg, Sweden. The answers from the paper questionnaire were afterwards digitalised in order to allow computerised data processing. An overview over both the descriptive statistics and the significance testing for all three variables is depicted in Tables I and II.

TABLE I
DESCRIPTIVE STATISTICS OF THE EXPERIMENT

Treatment	Dependent variable	Mean	Standard deviation
TA	**Answered**	5.667	3.42
	AScore	0.693	0.726
	Score	5.583	7.669
MSD	**Answered**	9.4	3.273
	AScore	0.576	0.45
	Score	6.2	5.453

TABLE II
SIGNIFICANCE TESTING OF THE EXPERIMENT

Dependent variable	Significance Level Levene	Significance Level Mann-Whitney U	H_0 rejected
Answered	$p \approx 0.94$	$p \approx 0.021$	Yes
AScore	$p \approx 0.097$	$p \approx 0.947$	No
Score	$p \approx 0.707$	$p \approx 0.464$	No

The results of the first dependent variable, **Answered**, are depicted in Figure 4 for both treatments. Clearly, subjects in the *TA* group took longer to answer the questionnaire, which led to only one subject finishing all questions. In the *MSD* group, half of the subjects finished all questions. Additionally, four subjects in the *TA* group answered three or less questions, whereas this is only the case for one subject in the *MSD* group. The large difference in the two means for this variable already indicates that the null hypothesis can be rejected, which is confirmed by the significance test with $p \approx 0.021$. Hence, there is a significant difference with respect to the number of answered questions between the two treatments. A possible explanation for this might be the nature of MSDs, compared to TAs. While a single MSD has to be taken into account only once it is activated, each automaton in a TA is 'active' by definition. This means that for each message in a given scenario, all automata need to be studied, while only a subset of the MSDs needs to be considered.

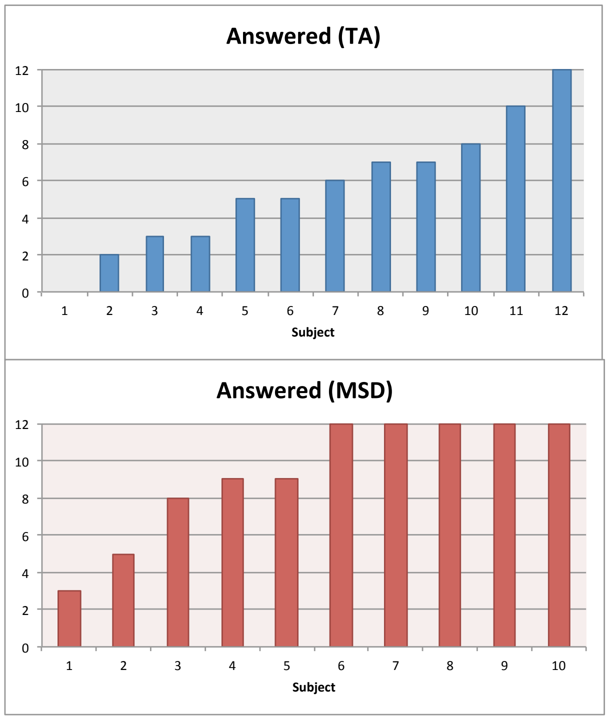


Fig. 4. Answered of TA an MSD treatment

The second dependent variable, **AScore**, is depicted in Figure 5 for both *TA* and *MSD* treatment. Here, in the *TA* treatment there is a much larger variance in the data set, with both very high and very low values. For the *MSD* treatment, there are few values in the extremes. The statistical test results in $p \approx 0.947$, so that the null hypothesis can not be rejected for this variable. We do not have an explanation for the large differences between subjects in the *TA* treatment, but they might be attributed to misunderstandings with respect to the modelling language. Several subjects achieved average scores under 1 point, even though they stated in the post-experiment questionnaire that they were confident in their answers. We plan to replicate the experiment in the future which will include some simple upfront questions in order to measure whether the subjects have really understood the languages well enough and analyse whether this correlates with the self-assessment.

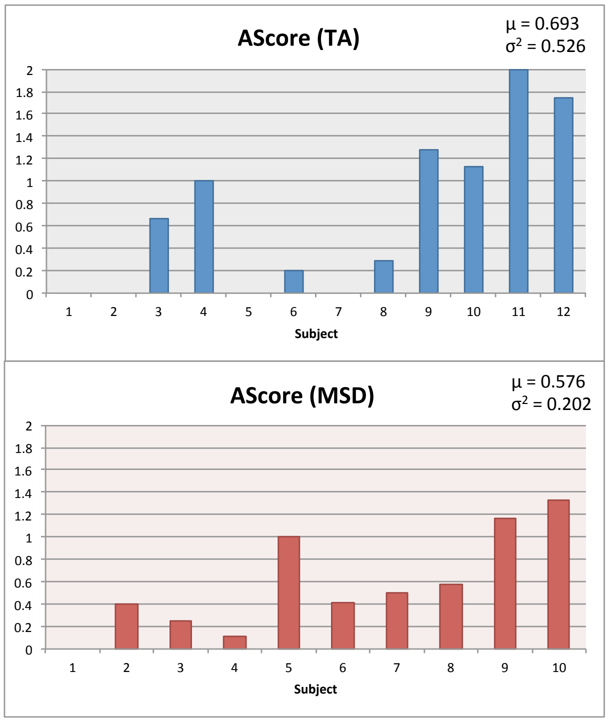


Fig. 5. AScore of TA and MSD treatments

As the third variable *Score* is directly computed from *AScore* and *Answered*, it exhibits a similar pattern (see Figure 6). In the *TA* group, two subjects achieved 20 or more points, close to the maximum of 24. However, many subjects in this group have low total scores. As subjects in the *MSD* group have significantly higher values in the **Answered** metric, their average **Score** values are higher, even though the average score **AScore** is lower for this group.

In summary, we can state that *MSDs* are significantly quicker to comprehend. Therefore, if speed is a relevant factor, *MSDs* should be chosen instead of *TA*. One could argue that speed itself is not relevant, as long as *AScore* is low. Therefore, we plan to replicate the experiment with subjects who are more familiar with the modelling languages, in order to see whether the difference in speed is still present.

C. Correlation between Demographic Data and Dependent Variables

We used the Pearson product-moment correlation coefficient to assess the correlations between the three dependent variables and the **number of related courses** previously taken by students, the **education level (Bachelor/Master)**, and the **subject's confidence** in their answers. The resulting values for Pearson's r and the p-value are depicted in Table III. Assuming an effect size of $r < 0.3$ as small, an effect size of $0.3 \leq r < 0.4$ as medium, and an effect size of $r \geq 0.4$ as large, we see that there is a large correlation between all three dependent variables and the subject's confidence in their results. This result indicates that subjects had, in average, a clear grasp of whether they understood the instrument or not. Interestingly, both the number of previous courses and the education level only show a small correlation with the dependent variables. Similarly, previous experience in Software Development, Software Modelling, and Requirements Engineering has small correlation with the dependent variables, as depicted in Table IV. These results could indicate that the dependent variables were in fact influenced by other factors, such as confusion regarding the newly introduced modelling languages. However, they could also indicate that the understanding of the requirements is not dependent on previous education and experience. Further replications will be necessary in order to answer these questions in a satisfactory manner.

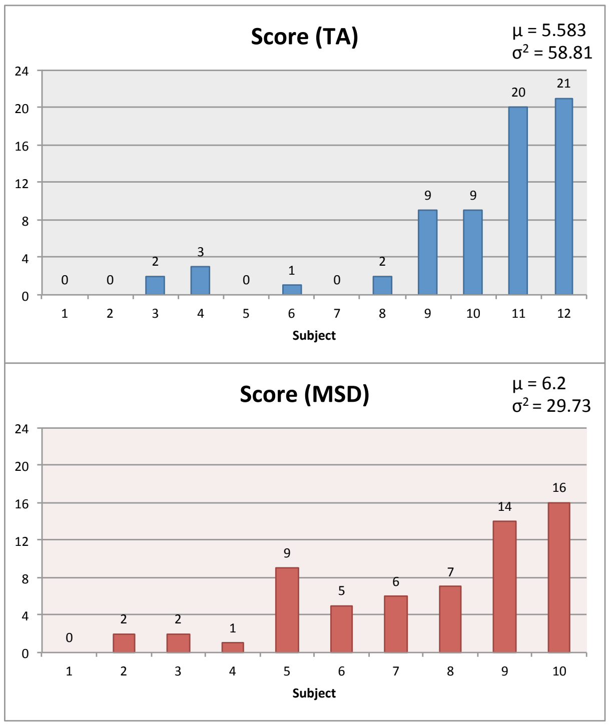


Fig. 6. Score of TA and MSD treatment

TABLE III
CORRELATIONS BETWEEN DEPENDENT VARIABLES AND DEMOGRAPHICS

Dependend Variable	Previous Courses		Bachelor/Master		Confidence	
	r	p	r	p	r	p
Answered	0.207	0.355	−0.247	0.267	0.56	0.007
AScore	0.122	0.588	0.116	0.606	0.513	0.015
Score	0.17	0.45	0.074	0.745	0.557	0.007

TABLE IV
CORRELATIONS BETWEEN DEPENDENT VARIABLES AND EXPERIENCE

Dependend Variable	Software Dev.		Modelling		Req. Eng.	
	r	p	r	p	r	p
Answered	0.08	0.724	0.054	0.811	0.092	0.685
AScore	−0.044	0.846	0.053	0.816	0.063	0.781
Score	−0.09	0.692	0.028	0.901	0.039	0.863

VII. Conclusions and Future Work

In this paper, we have presented the results of a controlled experiment with 22 students in an undergraduate course on software modelling. We studied the comprehensibility of functional requirements modelled in two graphical languages, Modal Sequence Diagrams, a sequence-based notation, and Timed Automata, a state-based notation. Subjects received a model in one of the two languages and a questionnaire with questions testing their understanding of the model. While we can not reject the null hypothesis, that there are no significant differences between the two treatments, for both the average and the total questionnaire scores, subjects receiving the Modal Sequence Diagram specification answered significantly more questions. This indicates that if the speed or the efficiency plays an important role, scenario-based models should be considered instead of the state-based models. However, further studies need to be conducted in order to understand whether this effect persists with more experienced users who achieve higher overall scores.

While our sample of students without a previous knowledge of the used treatments can be seen as a possible threat to validity, this lack of experience is in fact a realistic setup for industrial use in the automotive domain. As requirements specifications are used across organisations and across roles within an organisation, it can not be assumed that the receiver of a specification is always familiar with every detail of the used language. Additionally, receivers are often no experts in modelling, but in other areas such as requirements engineering or system design. Therefore, in contrast to, for example, software development, the receivers of a requirements specification can not be expected to be experts in the used language. Additionally, our results indicate that the current practice, choosing the modelling language based on convenience, is not a threat to the comprehension of the specifications in itself.

In the future, we will replicate the experiment both with different groups of students and with professionals from our industrial partners in order to eliminate possible bias and to assess whether experience and a deeper knowledge of the languages can have a significant impact on the understanding. Additionally, we will aim at generating a theory on which languages are suitable for which kind of task or system when modelling requirements.

Acknowledgement

We would like to express our gratitude to Nadja Marko and Christian Webel, who helped in reviewing and discussing an early experiment design. Additionally, we would like to thank Pariya Kashfi and Vard Antinyan for participating in the pilot experiment. The research leading to these results has received partial funding from the European Union's Seventh Framework Program (FP7/2007-2013) for CRYSTAL-Critical System Engineering Acceleration Joint Undertaking under grant agreement No 332830 and from Vinnova under DIARIENR 2012-04304.

References

[1] M. C. Otero and J. J. Dolado, "Evaluation of the comprehension of the dynamic modeling in UML," *Information and Software Technology*, vol. 46, no. 1, pp. 35–53, 2004.

[2] C. Glezer, M. Last, E. Nachmany, and P. Shoval, "Quality and comprehension of uml interaction diagrams-an experimental comparison," *Information and Software Technology*, vol. 47, no. 10, pp. 675–692, 2005.

[3] A. Nugroho, "Level of detail in uml models and its impact on model comprehension: A controlled experiment," *Information and Software Technology*, vol. 51, no. 12, pp. 1670–1685, 2009.

[4] M. Staron, L. Kuzniarz, and C. Wohlin, "Empirical assessment of using stereotypes to improve comprehension of uml models: A set of experiments," *Journal of Systems and Software*, vol. 79, no. 5, pp. 727–742, 2006.

[5] N. Condori-Fernández, M. Daneva, K. Sikkel, R. Wieringa, O. Dieste, and O. Pastor, "A systematic mapping study on empirical evaluation of software requirements specifications techniques," in *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*. IEEE Computer Society, 2009, pp. 502–505.

[6] N. Condori-Fernández, M. Daneva, K. Sikkel, and A. Herrmann, "Practical relevance of experiments in comprehensibility of requirements specifications," in *Empirical Requirements Engineering (EmpiRE), 2011 First International Workshop on*, Aug 2011, pp. 21–28.

[7] S. Abrahão, C. Gravino, E. Insfran, G. Scanniello, and G. Tortora, "Assessing the effectiveness of sequence diagrams in the comprehension of functional requirements: Results from a family of five experiments," *Software Engineering, IEEE Transactions on*, vol. 39, no. 3, pp. 327–342, March 2013.

[8] D. Harel and S. Maoz, "Assert and negate revisited: Modal semantics for UML sequence diagrams," *Software and Systems Modeling (SoSyM)*, vol. 7, no. 2, pp. 237–252, May 2008.

[9] R. Alur and D. L. Dill, "A Theory of Timed Automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[10] K. G. Larsen, M. Mikucionis, B. Nielsen, and A. Skou, "Testing real-time embedded software using uppaal-tron: An industrial case study," in *Proceedings of the 5th ACM International Conference on Embedded Software*. ACM, 2005.

[11] A. Fehnker, "Scheduling a steel plant with timed automata," in *rtcsa*. IEEE, 1999, p. 280.

[12] J. Greenyer, M. Haase, J. Marhenke, and R. Bellmer, "Evaluating a formal scenario-based method for the requirements analysis in automotive software engineering," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM, 2015.

[13] O. M. Group, "Unified modeling language," http://www.uml.org/, Jun. 2014.

[14] E. Kamsties, A. von Knethen, and R. Reussner, "A controlled experiment to evaluate how styles affect the understandability of requirements specifications," *Information and Software Technology*, vol. 45, no. 14, pp. 955–965, 2003, eighth International Workshop on Requirements Engineering: Foundation for Software Quality.

[15] G. Scanniello, M. Staron, H. Burden, and R. Heldal, "On the effect of using SysML requirement diagrams to comprehend requirements: Results from two controlled experiments," in *18th International Conference on Evaluation Assessment in Software Engineering (EASE)*, May 2014, pp. 433–442.

[16] W. Damm and D. Harel, "LSCs: Breathing life into message sequence charts," in *Formal Methods in System Design*, vol. 19. Kluwer Academic, 2001, pp. 45–80.

[17] D. Harel and R. Marelly, *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine*. Springer, August 2003.

[18] J. Bengtsson and W. Yi, "Timed automata: Semantics, algorithms and tools," in *Lectures on Concurrency and Petri Nets*, vol. 3098. Springer, 2003, pp. 87–124.

[19] V. R. Basili, "Software modeling and measurement: The goal/question/metric paradigm," Tech. Rep., 1992.

[20] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, and B. Regnell, *Experimentation in Software Engineering*. Springer, 2012.

[21] W. Damm and D. Harel, "LSCs: Breathing life into message sequence charts," in *Formal Methods in System Design*, vol. 19. Kluwer Academic, 2001, pp. 45–80.