Accepted for International Workshop on Requirements Engineering: Foundation
for Software Quality (REFSQ'02), 9-10 September 2002, Essen, Germany

A Controlled Experiment on the Understandability of Different Requirements Specifications Styles

Erik Kamsties[1], Antje von Knethen[2], and Ralf Reussner[3]

[1]Software Systems Engineering, University of Essen,
Altendorferstr. 97-101, D-45117 Essen, Germany
kamsties@cs.uni-essen.de

[2]Fraunhofer Institute for Experimental Software Engineering
Sauerwiesen 6, D-67661 Kaiserslautern, Germany
vknethen@iese.fhg.de

[3]Distributed Systems Technology Center (DSTC)
Monash University, Caulfield Campus
900 Dandenong Road, Caulfield East, VIC 3145, Australia

Abstract

In this paper, we report on a controlled experiment, in which we compared two different requirements specification styles. Following the traditional black-box style, a system is described by its externally visible behavior, any design detail is omitted from the requirements. Following the white-box style, which was popularized by object-oriented analysis, a system is described by the behavior of its constituent entities, e.g., objects. In the experiment, we compared the understandability of two requirements specifications of the same system each written in a different style.

The appropriate choice of a specification style depends on several factors including the project characteristics, the nature of the requirements at hand, and the intended readers. In this paper, we focus on the last factor, and investigate understandability from the viewpoint of a customer.

The results of the experiment indicate that it is easier to understand black-box requirements specifications from a customer point of view. Questions about particular functions and particular behavior of the specified system were answered by the participants faster and more correct. This result suggests using the black-box specification style when communication with customers is important.

1. INTRODUCTION

It is a common belief in requirements engineering that the requirements should describe only the externally visible behavior of a software system, e.g. [Dav93]. However, when it comes to the development of requirements models, often requirements specification languages (RSLs), such as UML, suggest describing the behavior of a software system by the behavior of its constituting components. In this paper, we report on a controlled experiment in which we investigate the understandability of these two styles of describing a system from the viewpoint of the customer. The question whether to stick to the black-box style when developing requirements model or to follow the white-box style as suggest by most RSLs has not been investigated so far in a controlled experiment.

The communication of a requirements model back to the customer is essential to validate it, i.e., to ensure that it reflects the customer's intention, in order to build the right system. There are two basic ways to validate a requirements model with the customer, simulation (i.e., execution) and inspection (e.g., walkthroughs). Simulation is a very powerful technique, because the customer can see what he or she will get in particular usage situations. On the contrary, simulation requires a quite detailed requirements model. Simulation has the same limitations as testing, that is, only the presence of

defects can be shown but not their absence. Inspection requires reading a requirements model, i.e., theoretically, one can observe all defects in a model. In practice, however, customers often have some difficulties to read and understand a requirements model. Therefore, a combination of both is required to ensure that a requirements model is correct and complete.

Our aim is to investigate how different specification styles affect the understandability of requirements models from the customer's point of view. Following the *black-box style,* a system is described by its externally visible behavior, any design detail is omitted from the requirements. Following the *white-box style,* a system is described by the behavior of its constituent entities (e.g., objects, blocks, or modules). We focus on embedded systems and behavioural requirements.

We expect that it is easier for a customer to understand a black-box than a white-box requirements model. More precisely, we expect that the required *time* to answer a question about the function or behavior of a system differ and, additionally the *correctness* of the answers may differ depending on the experience of the customer with the RSL. For a less experienced customer, we expect differences in the correctness of answers; for an experienced customer, we do not expect differences.

The reason why we expect differences is that the end-to-end behavior of a system as specified can be determined easier, i.e., faster and perhaps more correct, because the information to answer a question is less scattered over a model when the system behavior is not specified by a collection of interacting entities, but with as little internal details as possible.

This paper is structured as follows. First, the applied specification styles are briefly described. Second, the previous empirical research is reviewed. Then, the evaluation framework is discussed, followed by the design of the study. Finally, the results are presented, threats to validity are discussed, and conclusions are drawn.

2. REQUIREMENTS SPECIFICATION STYLES

In the remainder of this paper, we subsume under the term *requirements specification languages* requirements modeling languages and formal methods for describing requirements. A *requirements modeling language* offers a graphical language with a formal syntax, that is, a set of diagram elements, and a semi-formal semantics, which is typically stated in natural language. Examples of requirements modeling languages include the Unified Modeling Language (UML) [UML99]. A *formal method* offers a language with a formal syntax and formal semantics. In most cases, this language is mathematical, but also graphical and tabular languages have been proposed. A formal method allows describing requirements rigorously and allows analyzing them extensively. Examples of formal methods include SCR [HJL96], SDL [ITU93], VDM [Jon90], and Z [Spi92]. A *requirements model* is a set of requirements that is represented using a RSL. It is a formalized statement of requirements.

Concerning the aim of the experiment it seems to be sufficient to use one RSL that allows for both specification styles, e.g., UML. However, if we would use just one RSL then learning effects could take place. For example, if a UML requirements model of a system is read twice (once in the black-box style, and once in the white-box style), the understanding gained from the black-box version can be readily applied to the white-box version. Therefore, we chose two different RSLs, SCR and UML, the former supports black-box style, and the latter supports while-box style. We discuss the trade-offs of this decision in more depth in the section on the experimental design.

The SCR (Software Cost Reduction) requirements specification language was developed by the Naval Research Laboratory (NRL) of the US Navy in 1978 [Hen80]. Recently, the NRL presented a formal semantics of SCR [HJL96] and developed a CASE tool called SCR* [HBGL95]. The Unified Modeling Language (UML) was presented by Booch, Rumbaugh, and Jacobson in 1997. The CASE tool RationalRose was used to create the requirements model.

SCR was chosen, because it enforces a black-box style of specifying a system. SCR does not provide on purpose any structuring mechanisms such classes, modules, or blocks, or the like, that would allow to shade some light into the design of a system. A system is defined by a set of variables (which specify the inputs and the outputs of a system), a set of relations between input and output variables, and a set of finite state machines (which allow describing state-dependent behavior).

UML was chosen, because it supports the white-box style of specifying a system. Using the UML, usually the behavior of a system is described by providing a state chart diagram for each class. In general, object-oriented analysis does not necessarily result in a white-box style requirements model. For example, in the OCTOPUS method [AKZ96], on the requirements level, a state chart diagram is created for the whole system, not for each class. The allocation of behavior to classes takes place in later phases. UML, as a pure notation, allows for both specification styles. However, the usual way to use UML is to provide a state chart diagram for each class.

Our separation of black- and white-box specification style roughly correlates with the separation of what- and how-specifications. Because *what* and *how* are relative to one's viewpoint we prefer the former terminology.

In earlier days of requirements engineering, there was a dispute on whether requirements specifications should focus on the external visible behavior or should also include information about the internal details of a system [Dav93]. Today, it is clear that at least some design information is necessary. It would result in artificial and unusuable requirements if one tries to leave this information out of the requirements as argued by Sommerville and Kovitz [SS97, Kov98]. Some types of requirements cannot be described in a useful way without specifying the *how*, e.g., non-functional requirements, as pointed out by Berry [Ber00]. The right degree of design information depends on several factors including the project characteristics, the nature of the requirements at hand, and the intended usage of the requirements.

This empirical study contributes to the question what style should be chosen if the requirements model must be communicated to an external stakeholder, such as a customer or a subcontractor. This situation is typical for the domain of embedded systems where the development of the software/hardware is often outsourced. Of course, other factors, as the ones mentioned in the previous paragraph, must be taken into account for the final decision on the specification style.

3. PREVIOUS EMPIRICAL RESEARCH

This section reports on empirical studies that investigated understandability (also called comprehensibility or readability) of software artefacts in general, because there are very few studies on the understandability of RSLs. However, much can be learned from empirical studies of other artefacts.

Two strategies for measuring understandability can be distinguished. One strategy is recall and reconstruction. Participants are asked to study an artifact and then reconstruct it from memory. The amount of information recalled reflects the understanding of the artifact. This strategy was applied in early program comprehension experiments. Gause and Weinberg recommend this strategy to assess the understandability of requirements specifications [GW89].

The other strategy is to present questions to the participants, which require that they study the artifact in order to determine an answer. The response time and correctness of the answers reflect the understanding of the artifact. Curtis et al. applied this strategy to investigate the understandability of various forms of program documentation [CSK+89]. The authors hypothesized that understandability is affected by two characteristics of a documentation format, type of symbology (e.g., narrative text, constrained language, or ideograms) and spartial arrangement of information on a page of an artifact [CSK+89]. The authors conducted an experiment with professionals in which they treat the type of symbology and spartial arrangement as distinct independent variables. The main dependent variable was response time. The analysis of results was separated for different types of questions:

- Forward-tracing questions: the task is to search forward the control flow, e.g., to answer a question regarding the sequence of steps that occur for a given set of input conditions and a given point in program execution.
- Backward-tracing questions: the task is to search backwards through a program, e.g., to answer a question regarding the conditions that must have held and the events that have occurred in order to reach a given state.
- Dataflow questions: the task is to trace a variable's path through a program, i.e., to answer

questions regarding the values of particular variables at a given point in program execution depending on particular input data.
- Abstraction questions: the task is to abstract from the program statements and to synthesize information, i.e., to answer questions regarding the overall functions of a program.

The results indicate that even well written natural language program documentation requires more time to answer forward- and backward-tracing questions. The authors did not detect any significant effects of the documentation format on the correctness of answers.

Briand et al. investigated the influence of design principles on the understandability of structured and object-oriented design documents by asking questions [BBDD97], [BBD00]. In two experiments with students, the authors found significant differences in the correctness of answers: good object-oriented designs are easier to understand than bad object-oriented designs, and bad structured designs are easier to understand than bad object-oriented designs. Von Knethen [vKn02] investigated the influence of traceability guidelines on the understandability of requirements and design documents by asking questions. She found significant differences in the correctness of the answers of student subjects. Software documents with integrated traceability are easier to understand than software documents without such information. Britton et al. performed an experiment on the understandability of symbols of different RSLs including Z and UML [BJ99].

4. EVALUATION FRAMEWORK

The specification style, i.e., black-box or white-box style, is the controlled variable of interest in this experiment. The style affects the spartial arrangement of the requirements model. Using a black-box style, the information necessary to understand the end-to-end behaviour is more coherently presented as when using a white-box style.

We chose the strategy of asking questions to measure understandability, because memorization and comprehension are not the same processes. Not everything that is understood is remembered and vice versa.

In our evaluation framework, we adapted the types of questions suggested by Curtis et al. discussed above. The questions were originally developed for testing the understandability of programs, and we adapted them to test the understandability of requirements models. We developed three types of questions to test the understanding of function and behavior:

- Forward-tracing questions: the task is to trace an event from a given state and to determine state changes and output events. Example: “The system is in the following state: the bicycle computer is in the elapsed time mode. The display shows the current speed. What happens when the SET and the MOD buttons are pressed simultaneously and held for 5 seconds?”. Note that the questions are provided in terms of externally visible things. Thus, these things must be mapped onto terms of the requirements model first. The answer must be given using terms of the requirements model, e.g., a change in the current mode “CDispFunction: Vtm → Voff”.
- Backward-tracing questions: the task is to trace a particular state of the system or output event back to a given previous state by providing the shortest sequence of input events. Example: “Please specify the shortest sequence of user inputs that lead to the state: the bicycle computer is in the distance mode.” Again, the answer must be given in terms of the requirements model.
- Dataflow questions: the task is to elicit from the requirements model how particular data is computed. Example: “Explain how the total distance is computed.” Again, the answer must be given in terms of the requirements model.

Abstraction questions do not make sense in our setting, because we want to measure the understanding that a customer gains from the requirements and we assume that a customer already has an abstract understanding of the requirements.

We consider an answer as correct if all affected or relevant items are mentioned and no unaffected or irrelevant items are mentioned.

5. EXPERIMENTAL STUDY

This section describes hypotheses, design, instruments, preparation, execution, and data validation.

Hypotheses. The hypotheses followed the aims of the experiment as stated in the beginning of this paper. Our first (alternative) hypothesis is

H_1: There is a difference between a black-box and a white-box requirements model in the effort required to answer the given questions.

We expect that it takes less time in a black-box requirements model to answer questions, i.e., to determine all affected entities and relations in a requirements model. In a black-box requirements model, the end-to-end behavior is described by very few entities and the relations between them are kept to a minimum, while in a white-box requirements model the end-to-end behavior is provided by a not necessarily minimal set of classes or blocks and, consequently, a higher number of relations between them. In case of UML, it is more important that the classes reflect entities of the real world. Our second hypothesis is

H_2: There is a difference in the number of correctly answered questions between a black-box and a white-box requirements model.

Similar to the rationale for hypothesis H_1, the more entities and relations exist, the higher the danger to overlook some of them.

Subjects. The experimental subjects were graduate students from the Computer Science Department of the University of Kaiserslautern. Twenty-two students participated in the empirical study. All students were enrolled in a basic Software Engineering class lasting one semester. The lectures taught software engineering principles and different requirements specification styles. We distributed a questionnaire to measure motivation and experience. The students were highly motivated because they were interested in gaining practical experience – median response 4 "good" (min 2, max 5). The students were of relatively uniform degree of experience. They had little experience with textual requirements specifications – median response 2 "relatively weak" (min 2, max 3) and SCR/UML – median response 2 "relatively weak" (min 2, max 3).

Design. The independent and dependent variables, their variables, and the degree of control that we had are shown in Table 1.

Independent Variables	**Dependent Variables**
• Style (BB, WB): *controlled* • Symbology (SCR, UML): *confounded* • Group (A, B): *controlled* • Type of system investigated: *not controlled*	• Time to complete questionnaire ([min.]) • #Correctly answered questions ([0..6])

Table 1: Dependent and Independent Variables

The main independent variable of interest is the specification style, black-box (BB) or white-box (WB), which implies different spartial arrangements. The symbology is confounded with the specification style as discussed below. We randomly assigned the participants into two groups (A, B). Group A had 12 members, and group B had 10 members. Thus, the second independent variable is the group. Finally, the type of system, we used a specification of an embedded system, is an influencing variable that we did not control. The dependent variables to measure the understandability of a requirements model are the time required to complete the whole questionnaire (incorrectly answered questions are included) and the number of correctly answered questions.

We chose a 2 x 2 within-subjects factorial design as shown in Table 2 (see e.g., [Spe81]). The experimental subjects were not told about the hypotheses, i.e., we performed a blind experiment. In a within-subject design, each subject uses both treatments on the same object. With the help of this type of design, the statistical power of the experiment is increased and strong variations in human capabilities are controlled. The main problems with this design are learning effects between the two runs. We tried to avoid learning effects in two ways. First, we used two different RSLs rather than one. Second, we performed an extensive training of the participants before the experiment took place.

We employed two requirements models of the same system, written in different styles. That is, the documents of the first run, the black-box (D_{BB}) and white-box style (D_{WB}) requirements model, are used also in the second run. The use of the same example system in both runs was possible, because the two requirements models are *very different*. Knowing the SCR black-box specification does not help to understand the UML white-box specification, and vice versa.

Run / Requirements Model	**D_{BB}**	**D_{WB}**
1	A	B
2	B	A

Table 2: Experimental Design

However, since we used two different RSLs, the different types of symbology, tabular vs. graphical, could have an influence too. Because of restrictions in terms of time and number of participants, we could not rule out this factor by a more appropriate design. We discuss these issues in the Section "Threats to Validity" in more detail.

Instrumentation. Two requirements specifications of the system were used in the experiment. Each document consisted of two parts, a textual statement of requirements and the black-box or white-box requirements model. Both documents had the same section outline. The specified system was a bicycle computer, because this domain is familiar to most students. The functionality was reengineered from a commercial bicycle computer for professional bikers, which measures speed, distance, altitude, and the heart rate (with programmable alarm zone). The textual part was 9 pages long, and the black-box requirements model was 12 pages long. The white-box requirements model was 20 pages long and consisted of a class diagram, a use case diagram, a sequence diagram for each use case, a collaboration diagram, and several statechart/activity diagrams. Note that the level of detail was the same in both requirements models, even though the size was different.

The two requirements specifications were written by two different authors. Thus, the two versions of the system are completely different, even the used acronyms for states and events differ. This procedure further reduces the chance of learning effects.

Data Collection. All data collection was done on paper-based forms. The motivation and the experience of participants were collected on a form at the beginning of the experiment. The time to answer the given questions was recorded only for the whole set of questions using a debriefing form. We tried to measure also the time for each question, but only a minority of participants provided such detailed data. The questions and templates for the answers were provided on a third form. This form contained 6 questions, for each question type identified in previous section, two questions were given. The experiment materials, i.e., the requirements specifications and the forms, were tested by two students, who did not participate in the experiment.

Preparation. We conducted an extensive preparation phase. The students received training in reading SCR and UML requirements models. Moreover, the students were introduced to the textual requirements and the experiment forms. We did a dry run, in which they received the textual requirements of the bicycle computer, a requirements model written using the Box Structure Method by Harlan Mills [Mil88], and the above-mentioned forms. The preparation phase lasted 3 hours.

Execution. The execution phase lasted for another 3 hours and consisted of two runs of 1.5 hours each. The runs took place at different days in order to avoid fatigue effects. At the beginning of each run, the participants received the forms and the requirements specification. All materials must be handed back to the experimenters at the end of each run, in order to avoid exchange of information. It is impossible to communicate the answers to the questions without having the requirements model as a basis. In addition, the subjects were told not to discuss the experiment. Participation was of course voluntary, and all participants returned for the second run, which is an indication for high motivation.

Data Validation. We checked the collected data for completeness, plausibility, and outliers. A number of suspicious data points were identified, discussed, and clarified with the participants.

Data Analysis Procedure. We assumed that a black-box requirements specification is easier to understand than a white-box specification. Therefore, a one-tailed test was applicable Because of our

within-subject design (also called paired comparison design [WRH+00]), we chose one of the two following tests. If the data was normal distributed, we would use the Paired t-test to analyze the effects of treatments. If the data was not normally distributed, we would use the Wilcoxon matched paired test. We tested a normal distribution of the data by the Shapiro-Wilks' W Test.

Testing hypotheses involves different types of risks. These risks are referred to as type-I-error and type-II-error. A type-I-error occurs in the experiment in the case that a statistical test indicates a better understanding of black-box requirements models even if there actually is no real better understanding. The significance level α defines the chosen risk of committing a type-I-error. From a scientific perspective, a high level of confidence and, therefore, a low level of significance is necessary (usually quoted as $\alpha = 0.05$). A type-II error occurs in the case that a statistical test does not indicate a better understanding of black-box requirements models even if there actually is a better understanding. Such a result may have two possible interpretations. First, the effect does not exist and H_0 is retained or, second, the experiment does not have sufficient statistical power to detect the effect investigated (usually because the sample size is too small).

Statistical power is defined as the probability that a statistical test will correctly reject a false null hypothesis [MDW+94]. Statistical power is a function of three components, the sample size N, the significance level α, and the effect size γ. The effect size is determined based on comparable studies. For this experiment, the effect size could not be estimated because there was no experience from comparable studies. Therefore, the statistical power could not calculated before executing the experiment. Knowledge about the statistical power allows the sample size or the significance level to be varied in such a way that the effect investigated is detected. In this experiment, we did not have the choice of increasing the sample size or the significance level. There were no more subjects to participate and a level of significance had to be chosen that is of scientific relevance. We calculated the statistical power and the effect size for each statistical test based on the experiment results. The effect size can be used in replications of our experiment or in comparable studies to estimate the effect size and to calculate the statistical power.

6. RESULTS

Table 3 presents a descriptive summary of the dependent variables for the two investigated specifications styles. The table shows the number of subjects (N), the mean (x), the median (m), the minimum (min.), the maximum (max.), and the standard derivation (s) of the samples. In addition to the total time ("$Time_T$") to complete the questionnaire, we introduced a derived variable that captures the average time spent on a question ("$Time_{PQ}$"). Each run of the experiment was limited to 95 minutes and we encountered a ceiling effect. That is, not all participants were able to complete the questionnaire in 95 minutes. In order to cope with this effect, we introduced $Time_{PQ}$. This variable is computed from $Time_T$ divided by the number of questions that were answered, no matter whether correct or wrong.

	Black-box Style						**White-box Style**					
Variable	N	x	m	Min	Max	s	N	x	m	Min	Max	s
Correctness	22	4.14	4	1	6	1.05	22	3.27	4	0	6	1.8
$Time_T$	22	73.82	76	45	95	13.41	22	77.23	79	57	95	9.85
$Time_{PQ}$	22	14.97	13.33	7.5	47.5	8.08	22	17.31	16.3	9.5	28.33	5.89

Table 3: Descriptive Analysis for Dependent Variables

The descriptive analysis supports the hypotheses. Both variables are in the predicted direction.

Data Anomalies. The data sets show different outliers. We identified that some outliers result from subjects that were not able to understand the requirements specification because of their English skills. Two subjects mentioned that they had serious problems to understand the requirements specification or could not understand anything. We exclude the data of these two subjects from our statistical evaluation.

H_1- Time. Table 4 shows the results of the statistical tests performed. N is the size of the sample, W and t are the specific values of the applied test and p represents the p-value. In addition, the table shows the effect size γ and the Power P. The Shapiro-Wilks' W Test showed that data of both groups is normally distributed for the dependent variables $Time_T$ and $Time_{PQ}$. Therefore, we applied the Paired t-test.

	Shapiro-Wilks' W Test						Paired t-test		Effect Size	Power
	Black-box Style			White-box Style						
Variable	N	W	p	N	W	p	t	p	γ	P
$Time_T$	20	0.97	0.77	20	0.91	0.07	-1.0	0.32	1.8	0.17
$Time_{PQ}$	20	0.95	0.34	20	0.94	0.22	-2.01	0.045	1.52	0.53

Table 4: Statistical Analysis for Dependent Variable "Time"

The results of the statistical analysis show no significant difference for $Time_T$, because the total time spent on the questionnaire was more or less the same since the time was limited to 95 minutes. However, only about 50% of the participants were able to answer all 6 questions in time. A significant difference was observed for $Time_{PQ}$. Thus, we conclude that questions can be answered faster based on a black-box requirements model than on a white-box requirements model. Most participants did not record the time to answer each question; therefore, a more detailed evaluation cannot be given.

H_2 – Correctness. Table 5 shows the results of the statistical tests performed. The Shapiro-Wilks' W Test showed that the data sets of both groups are normally distributed for the dependent variable "Correctness". Therefore, we applied the Paired t-test.

	Shapiro-Wilks´ W Test						Paired t-test		Effect Size	Power
	Black-box Style			White-box Style						
Variable	N	W	p	N	W	p	t	p	γ	P
Correctness	20	0.91	0.06	20	0.93	0.24	2.07	0.045	0.45	0.52

Table 5: Statistical Analysis for Dependent Variable "Correctness"

The results of the statistical analysis show that there is a significant difference in the correctness of the answers given by the participants. More correct answers were given based on the black-box requirements mode than were given based on the white-box requirements model. Figure 1 gives a detailed look at the correctness of answers for each question.

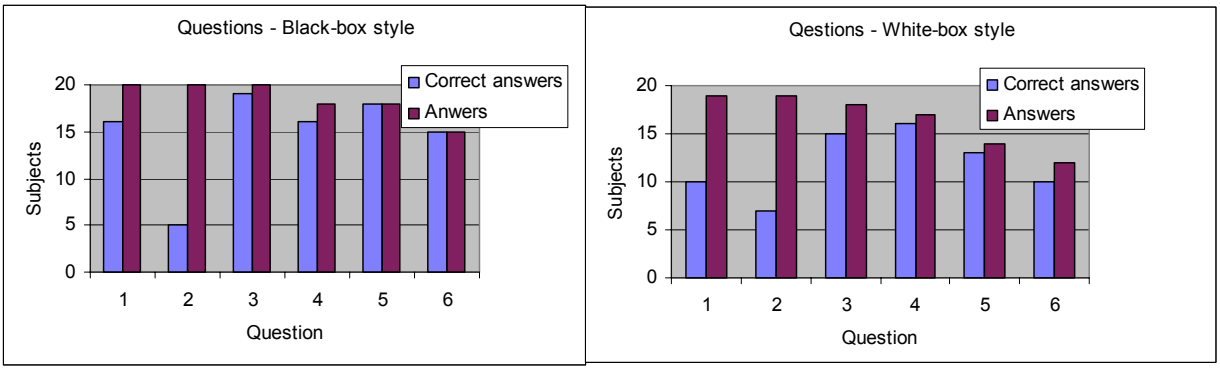

Figure 1: Numbers of Answers and Correct Answers

First of all, one can clearly see the ceiling effect. The decrease of answered questions in relation to the number of the question is quite obvious. Due to the low number, two, of questions in each category of questions we cannot apply statistical tests. Thus, we discuss Figure 1 informally.

There are no big differences between black-box and white-box style in the difficulty to answer a question. A question that is difficult to answer is difficult to answer with both styles. This fact shows

that the set of questions is not biased towards a particular style. Forward-tracing questions are most difficult to answer in both specification styles, i.e., the difference between *correctly* answered and answered questions is bigger than for the other question types. The other types of questions are less difficult to answer correctly.

The overall trend that higher numbers of correct answers are given for black-box requirements models, applies also to each type of question. This fact supports our design decision to apply a statistical test on the whole set of questions.

Analysis Summary. We found significant difference in time and correctness of answers depending on what specification style is used. Based on a black-box requirements model, questions were answered faster and more correct. This result is consistent with the previous empirical research as far as student subjects are concerned. Briand et al. also reported statistically significant differences in the correctness of answers in case of student subjects [BBDD97]. The procedure to avoid learning effects, as described in the previous section, was successful. No learning effects actually took place between the first and the second run. In conclusion, the results of our empirical study indicate that a black-box requirements model is easier to understand from a customer viewpoint than a white-box requirements model.

7. THREATS TO VALIDITY

The following possible threats to the validity of this study were identified:

- In the design of the experiment, two influencing factors, namely *Style* and *Symbology*, are correlated, i.e., black-box style is associated with SCR symbology and white-box style is associated with UML symbology. It could be that the observed differences are caused by inherent differences between SCR and UML symbology. One inherent difference is syntax. SCR has a tabular notation, while UML has a graphical notation. It is possible that one type of notation is easier to understand than the other type. Therefore, we asked the students for their subjective impression. Twice as many students deemed UML more understandable than SCR. That is, if there is a bias in understandability from a syntactical point of view than it is towards UML. The second inherent difference is semantics. As mentioned previously, both SCR and UML are based on finite state machines. The main semantic difference is the hierarchical decomposition of states that is possible in UML and not possible in SCR. It is commonly accepted that this feature eases understandability if not used too extensively. If there is a bias in understandability from a semantics point of view than it is towards UML. The fact that SCR black-box specification performed significantly better in our statistical tests leads us to the conclusion that the factor *Symbology* had no significant effect on the results.
- The requirements models might not be representative in terms of size and complexity. Both requirements models were written and reviewed by experts. Thus, their quality was good. Larger systems expose a certain complexity that cannot be wished away by methodological choices. Larger requirements models thus need some design to handle complexity. Therefore, it could be the case that the black-box style loose its benefit of better understandability if a requirements model gets larger.
- The subjects who participated in the experiment, students, are unlikely to be representative of real customers. Therefore, we cannot generalize the results to that population. Unfortunately, access to real customers is very difficult if not impossible. On one hand, computer science students are attuned to specification languages from their educational background. Thus, the results could be overoptimistic. On the other hand, if these students understand a back-box specification better, as the results indicate, then this should be true also for people with less command of specification languages. In general, we believe that student experiments are useful as a pilot for later industrial experiments. For example, we can test hypotheses in a student setting in order to decide whether it is worth investigating them further in industrial settings.

8. CONCLUSIONS

This paper presented the results of a controlled experiment on the understandability of black-box and white-box requirements models. The participants answered questions about a given requirements model. We analyzed the response time and correctness of these answers. We found significant difference in response time and correctness of answers depending on what specification style is used. Based on a black-box requirements model, questions were answered faster and more correct. The significant difference in the correctness results from the use of student subjects. Professional subjects can be expected to be 100% correct [CSK+89].

We can confirm the general advice that requirements should describe only the externally visible behavior of a system [Dav93] as far as understandability of requirements models is concerned. The results of our empirical study show that a black-box requirements model is easier to understand from a customer viewpoint than a white-box requirements model. That is, the black-box specification style should be chosen when communication with customers is important. Based on these results, we make the following recommendation for the use of RSLs in RE processes: at the beginning of a new project, if a choice can be made between several requirements specification languages, a language that explicitly supports this style should be chosen. If the language is already chosen, then a black-box style should be followed as far as possible.

We would like to encourage researchers to perform empirical research in requirements engineering to further investigate the impacts of RSLs in RE in order to increase the body of empirical knowledge in this area of RE. Understandability is just one of a lot of interesting questions around the use of RSLs, which all require more empirical research.

ACKNOWLEDGEMENTS

We would like to thank Klaus Pohl for his comments on previous versions of this paper. We thank Dieter Rombach for providing us a time slot for conducting the experiment in his lecture. Moreover, we thank the students from the University of Kaiserslautern for their willingness to participate in this experiment.

REFERENCES

[AKZ96] Maher Awad, Juha Kuusela, and Jurgen Ziegler. Object-Oriented Technology for Real-Time Systems: A Practical Approach Using OMT and Fusion. Prentice Hall, 1996.

[BBD01] L. C. Briand, C. Bunse, J. Daly, "A Controlled Experiment for Evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs", IEEE Transactions on Software Engineering, Vol. 27, No. 6, pp. 513-529, June 2001.

[BBDD97] L.C. Briand, C. Bunse, J.W. Daly, and C. Differding. An Experimental Comparison of the Maintainability of Object-Oriented and Structured Design Documents. *Empirical Software Engineering* 2, 291-312, 1997.

[Ber00] Berry, D.M., ``What, Not How? When Is `How' Really `What'? and Some Thoughts on Quality Requirements", Technical Report, University of Waterloo, Waterloo, ON, Canada, July, 2000

[BJ99] Carol Britton and Sara Jones. The Untrained Eye: How Languages for Software Specification Support Understanding in Untrained Users**.** *Human-Computer Interaction*, 14(1/2), 191-244, 1999

[CSK+89] B. Curtis, S.B. Sheppard, E. Kruesi-Bailey, J. Bailey, and D.A. Boehm-Davis. Experimental Evaluation of Software Documentation Formats. *Journal. of Systems and Software* 9, 167-207, 1989.

[Dav93] Alan Davis. Requirements Engineering: Objects, Functions, States. Prentice-Hall, 1993

[GW89] Donald C. Gause and Gerald M. Weinberg. *Exploring Requirements: Quality Before Design*. Dorset House, 1989.

[HBGL95] C. Heitmeyer, A. Bull, C. Gasarch, and B. Labaw. SCR*: A toolset for specifying and analyzing requirements. In *Proceedings of the 10th Annual Conf. on Computer Assurance*,

pages 109–122, Gaithersburg, MD, USA, June 1995.

[Hen80] K.L. Heninger. Specifying software requirements for complex systems: New techniques and their application. *IEEE Transactions on Software Engineering*, SE-6(1):2–13, January 1980.

[HJL96] C.L. Heitmeyer, R.D. Jeffords, and B.G. Labaw. Automated consistency checking of requirements specifications. *ACM Transactions on Software Engineering and Methodology*, 5(3):231–261, July 1996.

[ITU93] *Recommendation Z.100, Specification and Description Language (SDL)*. ITU, 1993.

[Jon90] C. B. Jones. *Systematic Software Development Using VDM*. Prentice Hall, New Jersey, USA, 1990.

[Kov98] Benjamin L. Kovitz. Practical Software Requirements - A Manual of Content & Style. Manning, 1998.

[MDW+94] J. Miller, J. Daly, M. Wood, M. Roper, and A. Brooks. Statistical Power and its subcomponents - missing and misunderstood concepts in empirical software engineering research. Technical Report EFOCS-15-94. Dept. Computer Science, University of Strathclyde, Scotland, 1994.

[Mil88] Harlan D. Mills. Stepwise refinement and verification in box-structured systems. *IEEE Computer*, June 1988.

[Spe81] P. Spector, Research Designs. Quantitative Applications in the Social Sciences. Sage Publications, 1981.

[Spi92] J. M. Spivey. *The Z Notation – A Reference Manual*. Prentice-Hall, 1992.

[SS97] I. Sommerville and P. Sawyer. *Requirements Engineering – A good practice guide*. Wiley, 1997.

[UML99] OMG Unified Modeling Language. Technical report, Rational Software Corporation, June 1999. Version 1.3.

[vKn02] A. von Knethen, "Change-Oriented Requirements Traceability. Support for evolution of Embedded Systems", PhD Theses in Experimental Software Engineering, Vol. 9, Fraunhofer IRB, 2002.

[WRH+00] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering.* The Kluwer International Series in Software Engineering, Kluwer Academic Publishers, Boston, Dordrecht, London, 2000.