Improving Efficiency of Change Impact Assessment Using Graphical Requirement Specifications: An Experiment

Niklas Mellegård and Miroslaw Staron

Department of Applied IT,
Chalmers Tekniska Högskola, Göteborgs Universitet
SE-412 96 Gothenburg, Sweden
{niklas.mellegard,miroslaw.staron}@ituniv.se

Abstract. *Objective*: Graphical requirements representation is often considered needed to advance model-driven development. Dedicated modelling languages include formalisms for graphically representing requirements, and together with new methods for structuring requirements, graphical modelling promises improvements such as more efficient change management. This paper examines whether the use of a graphical notation of a requirements affects the task of assessing the impact of a proposed change to a requirements specification.
Method: The efficiency of using a graphical requirements representation was examined through an experiment – using 18 student subjects. Time, perceived confidence and accuracy were measured as dependent variables.
Result: The results showed that using a graphical representation decreased the time required and increased the perceived confidence, but the accuracy decreased. However, the statistical analysis of the results showed that only the difference in time was significant. Furthermore, there was a large difference in variance within the dependent variables between the groups.

Keywords: Requirements Engineering, Visualization, Efficiency, Experiment.

1 Introduction

Model Driven Engineering (MDE) [1] is an established software analysis and design paradigm, bringing software engineering even closer to other engineering disciplines [2, 3]. Despite the numerous advantages of the state-of-the-art modelling techniques (e.g. UML [4], DSL[1]s [5], SysML [6]) engineers still struggle to efficiently link requirements to design models for the purpose of documentation, traceability, or later change impact assessment. The traditional 'use case driven' approach rooted in Objectory [7] is well suited for capturing the functional, scenario-like requirements, whereas they are not suitable for other kinds of requirements (e.g. non-functional, pure text based). One of the domains where text-based requirements are commonplace is the automotive domain in which the requirement specifications are often used

[1] Domain-specific Language.

M. Ali Babar, M. Vierimaa, and M. Oivo (Eds.): PROFES 2010, LNCS 6156, pp. 336–350, 2010.

when handshaking development between the car manufacturers and their subcontractors [8-11]. The complexity and volume of the requirement specifications are usually problematic for understanding of the specifications. The problems with understanding and incompleteness of the specification [12] may lead to quality problems with the final products or timeliness of development projects (when the quality has to be improved before the release).

In this paper, we evaluate whether using a graphical way of structuring requirements leads to improved quality of the design models during development projects. In particular, we address the following research question:

> *Does using a graphical representation of requirements result in more correct and more efficient change impact assessments in model-driven design?*

In order to address this question we conducted an experiment with students as subjects. The objects of the experiment were inspired by the research project that we conduct together with Volvo Car Corporation (VCC) [13]. In order to control the environment we created a dedicated domain specific modelling language [14] that was integrated with the existing requirement engineering practices and tools – e.g. IBM/Rational RequisitePro. The dedicated modelling language was also chosen as we in the future work intend to investigate whether adding more informal information about requirements (as advocated by [15]) lead to improved requirement specification, thus making the requirements model as the core requirements artefact in model-driven projects. The proprietary model for structuring textural requirements at VCC was replaced in the experiment with the Requirement Abstraction Model (RAM) [16] and our implementation of RAM as a graphical Domain-Specific Language (DSL) called gRAM [14], without the loss of generality of the results[2].

The results show – with statistical significance – that using a graphical representation of the requirements hierarchy decreased the time required to assess the impact of a proposed change – in our experiment it decreased with 37%. The results also indicate, although without statistical significance, that the accuracy of the assessments may deteriorate with the use of a graphical representation.

This paper is structured as follows; Section 2 presents work related to our research. Section 3 briefly outlines the requirements specification formats. Section 4 details the experiments we conducted as well as the results, section 5 contains discussions about the result and section 6 concludes the paper.

2 Background and Related Work

The intended main contribution of the experiment reported in this paper was to evaluate what effect a graphical representation of a model. In particular, the experiment focused on the representation of requirements specification and its effect on the efficiency of assessing the impact of a proposed change. Hence, the work related to this paper concerned the evaluation of different model notations and their effect on the efficiency of using the models. Additionally, as we chose to comply with the Requirements

[2] The replacement was made in order to avoid biasing the generality of the study with the proprietary model for requirements structuring. RAM was found to be good enough to approximate the proprietary model.

Abstraction Model (RAM), its effectiveness was also of interest. Moreover, in the experiment we evaluated the requirements' representation by having the subjects perform tasks related to change impact assessment – as our industrial partner has expressed this as a significant challenge – work related to assessing the impact of a proposed change was also of interest.

The work presented in this paper was part of our ongoing research (outlined in [17]) within the research project ASIS, done in cooperation with Volvo Car Corporation [13]. One part of the project aimed at improving the way requirements were specified, and in particular, the extent to which requirement specification can be reused with a minimum of effort. As part of this research, a model for the requirements specification process was developed (gRAM [14]) with the intention of finding areas where Model-Driven Engineering (MDE) approaches may improve efficiency. This paper contributes to that research by examining to what extent a graphical model of the requirements affect the efficiency of assessing the impact of a change request to a specified system.

Much of the empirical studies done on modelling – in both system modelling and requirements engineering fields – have been with the focus on investigating and improving aspects of specific approaches. Maiden et al.'s CREWS experiment [18] and its replications [19] proposed and evaluated whether templates and style guidelines improve the quality of use-case descriptions. Although the replications found some contradictions, both studies provided evidence of that the use of guidelines improved the quality of the use-case descriptions. Phalp et al. [20] extended the CREWS research by comparing their approach with a leaner set of guidelines and found that it performed at least as well as the original approach. Gravino et al. [21] examined, through a controlled experiment, whether dynamic modelling and UML sequence diagrams provided an accurate account of stakeholder requirements, with the focus on evaluating whether a behavioural modelling approach improved the comprehension of software requirements. In their study, they found no evidence of any significant differences in the comprehension of system requirements by using dynamic modelling, even though the subjects perceived the use of dynamic modelling as useful, thus showing a difference between the perceived usefulness of a given method and effective advantage of using it. Our study examined the use of a graphical representation of the RAM with the traditional text based one, isolating and exploring what effect a visual representation had on the comprehension of requirements as well as traceability to design and implementation. Thus, our study examined the effect of introducing a graphical representation in an earlier phase of the development cycle.

In their paper [22] Lange and Chaudron performed a similar study to ours, in the sense that they measured correctness and the effort required to comprehend a software system. Lange and Chaudron compared four novel graphical views of a set of UML diagrams to the representation used by traditional UML modelling tools. The study found statistically significant improvements in both time and correctness (20% and 4.5% respectively) when using the alternative representation. Our study compared a graphical and a textual representation of requirements – with linking to high-level design – in order to evaluate specifically what influences the graphical representation had on the comprehension of the specification in the context of assessing the impact of a proposed change to the requirements or to some underlying software component.

There have been numerous comparisons of the efficiency of different modelling approaches, e.g. De Lucia et.al [23] comparing the comprehension of a data model represented in ER and UML diagrams, in which they found that the use of UML significantly improves comprehension. Otero and Dolado [24] examined the effect of different notation types with respect to comprehension of dynamic modelling, by comparing the use of UML and OML in a design document, and found evidence that the use of OML improved the semantic comprehension and required less time. In contrast, our study was intended not to be dependent on any particular modelling notation, but rather to evaluate the effect of graphical representation itself.

Studies to validate the effectiveness of the RAM approach have been done in e.g. [16, 25, 26] and in our paper we intended to extend these studies by investigating change impact assessment and using graphical representation. In that context, we evaluated whether adding a graphical representation for a RAM-structured requirement specification can lead to further improvements. However, we also considered time as one of the factors, thus we focused on efficiency, not only effectiveness.

In the light of the paper by Wong and Sun [27], where they examined how diagram layout affected the comprehension of the programs they represent, we have chosen to design the gRAM to as closely as possible resemble the original RAM, in order to assure that our results can be generalized to the same contexts as the RAM itself.

Noppen et al. [12] showed that creating requirement specifications was an iterative process and subject to frequent changes. Therefore, it was important that the time required to identify what changes need to be made was short. Our experiment showed that by using a graphical language the time required to assess the impact of a proposed change can be substantially reduced, which means that using the graphical language can lead to quite substantial improvements in iterative SRS development.

Lindvall [28] examined the accuracy of predicting the impact of introducing or changing a requirement prior to design and implementation by examining real data in best-of-practice projects, and found an under-prediction factor of 3.1 showing evidence of the need to improve change impact predictions. The study by Lindvall was done mainly to explore the accuracy of state-of-practice approaches to change impact assessment, and did not take the perspective of requirements representation, nor did it take traceability between requirements and high-level design into consideration, as done in our study.

Arisholm et al. [29] examined the cost effectiveness of model-driven development with UML by studying – in two consecutive controlled experiments – what impact the presence of UML models in design and implementation documentation had on the task of system maintenance, in terms of effort and correctness of performing post-release changes. They concluded that when considering only the time required making code changes, the UML documentation did help save effort but when also considering the time required to change the UML documentation accordingly, no savings were visible. They also concluded, however, that in terms of functional correctness, the use of UML documentation had a positive effect on the most complex tasks. Our evaluation examined a similar research question, but from the perspective of a graphical requirements model, and what influence the graphical representation had on the correctness and effort required to assess the impact of a change.

In the context of our research (i.e. product line oriented, large, complex embedded software systems), reuse was commonly achieved by modification of a requirements

specification of a similar existing system. Additionally, it was commonplace in many business areas to manage requirements using structured text documents, thus, the effect of the representation of requirements, and their linking to design artefacts, on change impact assessment were of interest to examine.

3 Requirement Specification Format

In this section, we briefly describe the requirement specification format that was used to create the domain specific graphical notation. The requirement specification format is an established one with published evidence that this specification format is indeed improving industrial requirements engineering practice [25].

3.1 Requirements Abstraction Model

The Requirement Abstraction Model (RAM) [16] has the goal of ensuring consistency and traceability among requirements in order to increase the overall quality of requirement specifications. The RAM defines a number of abstraction levels to which each requirement is classified, and checklists to ensure that the requirements are assigned their proper level. In their original paper Gorschek and Wohlin [16] suggest, but do not limit their model to, four different abstraction levels:

- *Product*: Product level requirements have a goal-like nature, very high-level descriptions of the desired functionality of the product.
- *Feature*: Feature level requirements describe the features that fulfil the product level goals.
- *Function*: Function level requirements define which functions should be provided by the system in order to provide the features.
- *Component*: Component level requirements describe how something should be solved, i.e. bordering to design information.

RAM ensures traceability between requirements through all levels of abstraction by enforcing that, with the exception of the product level, no requirement may exist without a link to a requirement one level of abstraction higher. The rationale for this rule is that no requirement may exist unless there is a clear and unambiguous reason for its existence motivated by higher-level requirements, and conversely, high-level requirements should be traceable to the lower-level requirements that satisfy them.

3.2 gRAM – DSL for Modelling Requirements

gRAM is a Model-Driven Engineering (MDE) language with the purpose of creating an easy to use requirements management environment for directly manipulating a requirements structure, from which other documents can be automatically generated through translational semantics. The gRAM is a formalized graphical Domain Specific Language[3] (DSL) complying with the RAM, where validation rules (i.e. static semantics) built into the gRAM ensures that the model and the resulting requirement specification are syntactically correct and well formed according to the RAM.

[3] "Domain specific" refers to the horizontal domain of requirements engineering.

In addition to the traceability link RAM defines between requirements at adjacent abstraction level – which gRAM represents with an *Owns/Satisfies* link – gRAM also adds the *Depends-on* traceability link, which indicates that there is a dependency between two requirements within the same abstraction level.

In [14] we provide a more detailed description of gRAM, and the full set of experiment material – including a textual and a gRAM requirements specification – is available from [30].

4 Experiment Design

The experiment presented in this paper, was designed to compare the use of a requirements specification represented with the gRAM language, with a textual representation written according to RAM. The objective of the experiment was to examine whether a graphical representation of requirements (as advocated by MDE) increases the efficiency and effectiveness of assessing the impact of a change to the requirements from the perspective of system designers. This section presents the design of the experiment, which was conducted as a standard two-group design – the control group (using the textual specification) and the test group (using the gRAM specification).

4.1 Population and Sample

The subjects in this experiment were students – i.e. convenience sampling. A total of 18 subjects participated in the experiment, of which 14 were first and second year master students (i.e. in their 4^{th} and 5^{th} year of university studies) attending Software Engineering and Management programme, and 4 were 3^{rd} year bachelor students (i.e. their 3^{rd} year of education) from the same programme.

Blocking – in order to assign subjects to experiment groups – was done based on the subjects prior knowledge of UML, requirements engineering, industrial experience and experience with projects.

The population of this experiment is software designers working with implementation of software requirement specifications and systems analysts creating/maintaining these specifications. Most of the participating master students had over one year of industrial experience prior to their studies, which makes them representative for junior designers in industry.

4.2 Instrumentation

Objects

The experiment objects shown to the subjects in both groups consisted of:

- Generic description of a toy software system
- Detailed design of the toy system (a class diagram)
- Requirement specification for the toy system complying with the RAM:
    - For the control group: the textual requirements specification
    - For the test group: the graphical representation (gRAM) of the requirements

The toy system used in the experiment was a simulator of a power steering function in a car with customizable algorithm for automated power steering. The simulator was implemented in Java prior to the experiment and the requirements were traced (linked) to the software components of the simulator. The toy system was inspired by the real-world systems our partners work with, which could not be used due to confidentiality and the complexity of the systems.

The requirements specification consisted of 56 requirements, out of which 29 were at the lowest level of abstraction. The high-level logical design view – represented by a class diagram of the implemented power steering simulator – consisted of 10 classes and 15 associations.

The full set of experiment material is available from [30].

Data Collection

Five tasks were prepared and used in the experiment, and were concerned with:

- listing requirements related to some functionality or having some, by the task, defined property
- listing components in the logical view that implement a given requirement
- listing components which may be affected by a given change request

The instruments of data collection were (i) a form with the tasks and (ii) a questionnaire surveying the background of the subjects. We used a separate answer sheet for each task to collect the data, on which the subjects were asked to note the time when they began the task, write their answer and finally note the time the task was finished. We also asked the subject to note how confident they were in their answer; the 5-point Likert scale was used for that purpose.

Additionally, we conducted informal, semi-structured interviews with subjects from both groups in order to acquire qualitative data about how they perceived the experiment. The interview questions were concerned with how the subject used the material and what they found difficult.

Independent and Dependent Variables

There was only one independent variable in the experiment: the type of the requirement specification with the values – graphical (`gRAM`) and textual (`TEXT`).

The following direct dependent variables were used for each task and subject:

- *Tx_SCORE* The percentage of correctly identified requirements/ components (%) for task x
- *Tx_FP* The absolute number of falsely identified requirements / components for task x
- *Tx_TIME* The time in seconds spent on task x
- *Tx_CONF* The perceived confidence of the answer for task x (LIKERT scale 1-5)

The following variables were derived from the collected variables for each subject:

- *AVG_SCORE* The subject's average score over all tasks (percentage)
- *TOT_FP* The subject's total number of false positives for all tasks
- *TOT_TIME* The total amount of time the subject spent on the tasks

- *TOT_CONF* The sum of the subject's confidence level over all tasks
- *EFF* The efficiency of the change impact assessment process, calculated as *AVG_SCORE* / *TOT_TIME*

In the Analysis, We Used the Derived Variables. Hypotheses
The hypotheses posed in the study were tested at a 2-tailed confidence level of 95% ($p <= 0.05$). For each task, we posed the following null hypotheses[4]:

- H-AS_0: $\mu_{AVG\_SCORE\_TEXT} = \mu_{AVG\_SCORE\_gRAM}$
- H-TF_0: $\mu_{TOT\_FP\_TEXT} = \mu_{TOT\_FP\_gRAM}$
- H-TT_0: $\mu_{TOT\_TIME\_TEXT} = \mu_{TOT\_TIME\_gRAM}$
- H-TC_0: $\mu_{TOT\_CONF\_TEXT} = \mu_{TOT\_CONF\_gRAM}$
- H-EF_0: $\mu_{EFF\_TEXT} = \mu_{EFF\_gRAM}$

Each null hypothesis had a corresponding two-sided alternative hypothesis.

Analysis Methods
The collected data was analysed using both descriptive and inferential statistical methods; box plots were used to identify outliers (complemented with Little's MCAR test [31] for analysis of missing values) and extreme values, mean values and standard deviations were used to characterize the data set.

For the inferential statistics, we used Shapiro-Wilk test [32] to check whether the variables fit the normally distribution, and Mann-Whitney U-test [33] for testing the hypotheses stated in the previous section.

Validity Evaluation
The main threats to the validity of the study – as described by Wohlin et al. [34] – are analysed below.

Internal Validity – As blocking was made by us based on our experience of the subjects – all subjects were at some point students in courses taught by us – we assessed the threat to internal validity by collecting background information using a survey. After analysis we found no significant difference between the groups.

The introductory lecture was given to the two groups by different presenters, which might have affected the internal validity of the study. This threat was minimized by (i) having a common set of slides, which only differed in the presentation of the treatment for each group, and (ii) supplying the same information presented at the lecture in written format, which the subjects were allowed to study for 15 minutes before the test started.

External Validity – The main threat to the external validity is the use of student subjects, which may limit the ability to generalize the result to an industrial situation. The study was mainly done to evaluate the format of the requirements specification, and we do not make any conclusions about its applicability in an industrial situation yet. Eventhough these subjects are not completely representative of this population we could consider them the worst-case scenario sample, in the light of our previous research [35, 36].

[4] μ denotes the mean value for all subjects in the group.

An industrial evaluation of gRAM is planned for the future in the same way as an industrial evaluation presented in [35].

Construct Validity – The following bullets state, in our opinion, the main threats to construct validity according to Wohlin et.al. [34] and how we have avoided those threats

- *Inadequate preoperational explication of constructs*. All variables, as well as the correct answer to all experiment tasks, were clearly defined prior to the experiment.
- *Mono-method bias* and *Restricted generalizability across constructs*. We collected a number of different measurements; time, correctly given answers, false positives and the subjects perceived confidence of the given answer. By contrasting these to each other, we believe to have minimized this threat to the validity of the experiment.
- *Mono-operation bias.* The experiment was conducted only once and with one set of instruments, which poses a threat to construct validity. In order to confirm the findings in this evaluation, replication experiments, using different instrumentation is planned for the future, after an industrial case study on the applicability of this method has been done.

Conclusion Validity – The statistical power of the conclusions is quite low due to the small sample size. This threat to validity limits the strength of the conclusions drawn from the study. Rather than stating firm conclusion, we limit ourselves to *indications* and *tendencies*, and keep in mind that the results that showed no statistical significance may be due to random variation in the sample (the ρ-value is also reported for each hypothesis test). As the post-experiment interviews were, few they are not used in the result analysis. Instead, they are only used in the discussion section as evidence to support findings from the statistical analyses.

The original data set contained a number of missing data points (as reported in section 4.3 below). Due to our low sample size, we chose to impute the missing data. The imputation of the missing values was done using the Estimation-Maximization method [31] that may inflate the correlation between variables, which however, does not influence the statistical tests used in our study.

Testing of the collected data showed that several variables did not fit to the normal distribution – reported in section 4.3 below. For this reason, non-parametric tests were chosen for the inferential analyses, which further decrease the statistical power of the results but avoids the risk of violating assumptions and introducing further threats to the conclusion validity.

4.3 Analysis of Results

Normality Tests

The second column in Table 1 (Norm.(ρ)) shows the ρ-value for the Shapiro-Wilk test – a value below 0.05 indicates that the variable fits the normal distribution. The results show that none of the variables fit the normal distribution. Thus, the less powerful non-parametric Mann-Whitney U-test was used to analyse the posed hypotheses.

Missing Data and Extreme Values

The following data points were missing in the collected data: (i) one subject omitted to note the finish time on task 3; (ii) two subjects omitted to note the confidence on task 4; (iii) one subject omitted to note the finish time on task 5; (iv) one subject omitted to note the confidence on task 5; and (v) one subject did not submit the background survey at all.

Little's MCAR test could not reject that the values are missing completely at random, indicating that methods for data imputation may be used. All missing values were imputed using the Expectation-Maximization method [31].

Two outliers were found and removed from the $\texttt{EFF}_{\texttt{gRAM}}$ variable. The removed values were excluded pair-wise in the subsequent analyses.

Descriptive Statistics

The descriptive statistics for the derived variables are shown in Table 1. The *Diff*-column in Table 1 shows the difference between the text group and the gRAM group, using the text group as baseline. The descriptive statistics indicate that the gRAM group was 37% faster than the text group (TOT_TIME). On the other hand, the results also show that the text group scores 23% better (AVG_SCORE) and produce 26% less false positives (TOT_FP) than the gRAM group. There is also an indication of a 9% higher perceived confidence level of the answers (TOT_CONF) in the gRAM group.

Table 1. Descriptive statistics

Variable	Norm. (ρ)	Mean	Std. Dev.	Diff	Better
$\text{AVG\_SCORE}_{\text{TEXT}}$	0.845	49.07	12.16	-23%	Text (higher score)
$\text{AVG\_SCORE}_{\text{gRAM}}$	0.131	40.00	17.53		
$\text{TOT\_FP}_{\text{TEXT}}$	0.814	17.75	11.25	26%	Text (less false positives)
$\text{TOT\_FP}_{\text{gRAM}}$	0.344	22.40	11.29		
$\text{TOT\_TIME}_{\text{TEXT}}$	0.335	3113.00	253.49	-37%	gRAM (faster)
$\text{TOT\_TIME}_{\text{gRAM}}$	0.192	1965.50	741.65		
$\text{TOT\_CONF}_{\text{TEXT}}$	0.369	14.13	1.64	9%	gRAM (more confident)
$\text{TOT\_CONF}_{\text{gRAM}}$	0.498	15.4	4.67		
EFF_{TEXT}	0.580	0.01572	0.0035	4%	gRAM (more efficient)
EFF_{gRAM}	0.153	0.01629	0.0047		

The results also show that the difference is time between the groups does not imply a higher efficiency, as the gRAM group has a lower score; the difference in the efficiency variable (EFF) is only 4%.

There is, furthermore, a large difference in the standard deviation between the groups for all variables except TOT_FP variable – Fig 1 and Fig 2 show boxplots for the TOT_TIME and EFF variables respectively. This variance within the gRAM group may indicate that the difference between the groups is not statistically significant.

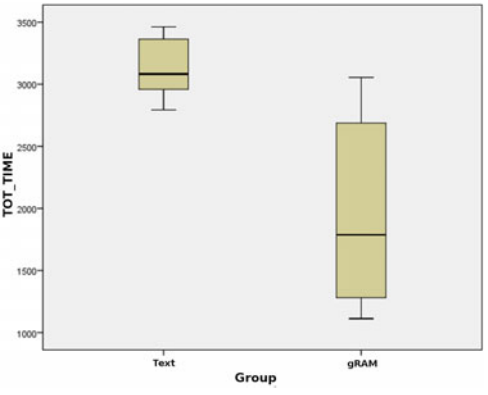

Fig. 1. TOT_TIME variable

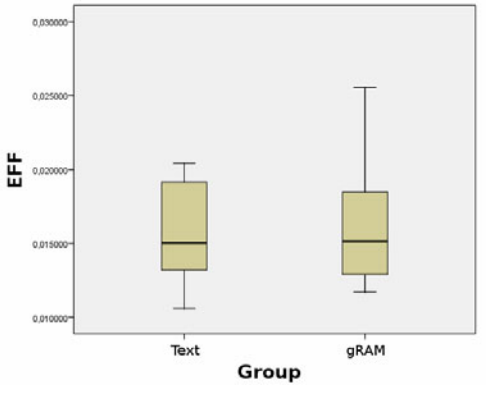

Fig. 2. EFF variable

Hypotheses Tests

The result of the hypotheses tests (Table 2) using the Mann-Whitney U-test shows that only the hypothesis $H-TT_0$ could be rejected. Fig 1 shows a box plot for the total amount of time spent on the tasks (the unit on the y-axis is seconds). The descriptive statistics show that the time required to perform a change impact assessment is in this experiment 37% shorter when using gRAM (see Table 1).

Table 2. Hypotheses tests

Hypothesis	ρ-value	MWW U	H_0 rejected
H-AS_0	0.264	27.50	No
H-TF_0	0.351	29.50	No
H-TT_0	0.002	5.00	Yes
H-TC_0	0.501	32.50	No
H-EF_0	0.834	30.00	No

Interviews
When asked how the requirement specification was used, subjects in the groups with the textual specification stated that they constructed a hierarchical structure of requirements similar to the gRAM, either mentally or on paper. Subjects from the text group stated that the model they drew was revised many times during the experiments, making them doubt whether they answered earlier tasks correctly.

Subjects in the gRAM group stated that they mainly used the graphical representation; in the interview one subject said: "*When I read the task, I had an initial idea about what the answer would be, and a quick look at the diagram confirmed it. I felt no need to read the detailed description*".

The interviews suggest that the text group, while constructing a visualization of the requirements themselves (either mentally or by drawing), read through the detailed description of the requirements more thoroughly than the gRAM group. The gRAM group seemed content with drawing their conclusions based on the graphical model, turning to the details only when in doubt. The large standard deviation in required time (`TOT_TOME`) and score (`AVG_SCORE`) within the gRAM group, however, might indicate that some subjects did read the detailed requirements description more thoroughly than others did.

5 Discussion

The results of our experiment show with statistical significance that the use of a graphical representation reduces the time required to perform the tasks. During the post-experiment interviews, subjects from the text group stated that they tried to construct visualizations of the textual document themselves, and some even stated that the structure they created was similar to the representation used in the gRAM. This suggests that there is a justification for creating such structure as part of making the specification; if it is not done, it will result in redundant work each time the text specification is used. Moreover, statements from the text group suggest that they had doubts whether the structure they created was correct, resulting in revising it during the course of the experiment, which may contribute to the extra time spent by the textual group.

On the other hand – although not statistically significant in our experiment – the textual group had higher score and fewer false positives (variables `AVG_SCORE` and `TOT_FP` in Table 1). This might be explained by interview statements from the group presented with the gRAM representation, which show that they had an initial idea of an answer and used the graphical structure to confirm it; they mainly used the detailed description when in doubt. This indicates that they put a lot of trust in the material provided, while the textual group – knowing that they created the graphical structure themselves – were more inclined to double-check their answer. This suggests that a graphical representation promotes quicker decisions, while the textual representation forces the subject to study the material more closely.

Furthermore, the statements made by the gRAM group – saying that they mainly turned to the detailed requirement description when in doubt – suggest the importance of clearly defining what information is shown in the diagram. The graphical representation might end up being misleading if in fact the detailed description is needed in

order to fully understand the specification. This may explain the large variance in time (TOT_TIME) and score (AVG_SCORE) within the gRAM group – the subjects may have used the detailed requirement description to different degrees.

It should be noted that the experiment was done using student subjects, and that in a real world situation the repercussions of making a mistake would be much more severe than in our superficial case. Furthermore, only one of our five hypotheses could be confirmed with statistical significance, possibly due to the large variation in the gRAM group – which is not adequately explained by the experiment. For these reasons, we plan to do a larger experiment and include subjects from the industry in order to verify our conclusions.

6 Conclusions

Graphical modelling of requirements has been considered in several modelling languages like SysML (the notion of requirement) or UML (the notion of use case). Nevertheless, not much empirical evidence is provided whether graphical modelling of requirements improves typical requirements engineering activities like elicitation, packaging, validation or change management. In this paper we present results from an experiment performed at academia with the objective to verify whether a graphical model of requirements is better than a textual one. As a basis for the experiment, a state-of-the-art method was used – Requirements Abstraction Model – to specify the requirements in a textual form, whereas a dedicated modelling language based on the Requirements Abstraction Model was used for the graphical specification.

The results from the experiment show that the time required to assess impact of a change was substantially shorter for the graphical notation. Aspects such as accuracy of the assessment and the confidence in the result (as perceived by the subjects) were found to be within the limit of statistical error (i.e. statistically insignificant).

In our future work, we plan to replicate the study in an industrial context and to further experiment with such aspects as time required to create the requirement specification and its correctness.

Furthermore, we plan to examine the consistency among the different abstraction levels of gRAM as well as the effectiveness of mapping the requirements specification to design model – i.e. traceability correctness.

Acknowledgments

This research is partially sponsored by VINNOVA under the V-ICT program and the ASIS (Algorithms and Software for Improved Safety) project. We would also like to thank the students who participated in the experiment. Additionally, we like to thank the anonymous reviewers for their valuable comments.

References

1. Kent, S.: Model Driven Engineering. Integrated Formal Methods, 286–298 (2002)
2. France, R., Rumpe, B.: Model-driven Development of Complex Software: A Research Roadmap. In: 2007 Future of Software Engineering, pp. 37–54. IEEE Computer Society, Los Alamitos (2007)

3. Ludewig, J.: Models in software engineering – an introduction. Software and Systems Modeling 2, 5–14 (2003)
4. Object Management Group, http://www.omg.org/
5. Greenfield, J., Short, K.: Software factories: assembling applications with patterns, models, frameworks and tools. In: Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, pp. 16–27. ACM, Anaheim (2003)
6. SysML - Open Source Specification Project, http://www.sysml.org/
7. Jacobson, I.: Object Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley Professional, Reading (1992)
8. Hänninen, K., Mäki-Turja, J., Nolin, M.: Present and future requirements in developing industrial embedded real-time systems - interviews with designers in the vehicle domain. In: 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems, ECBS 2006, p. 9 (2006)
9. Kallenbach, R.G., Emig, R.: Automotive Electronics - What Makes It So Special? Presented at the October 1 (2004)
10. Salzmann, C., Stauner, T.: Automotive software engineering: an emerging application domain for software engineering. In: Languages for system specification: Selected contributions on UML, systemC, system Verilog, mixed-signal systems, and property specification from FDL 2003, pp. 333–347. Kluwer Academic Publishers, Dordrecht (2004)
11. Broy, M., Kruger, I., Pretschner, A., Salzmann, C.: Engineering Automotive Software. Proceedings of the IEEE 95, 356–373 (2007)
12. Noppen, J., van den Broek, P., Aksit, M.: Imperfect Requirements in Software Development. In: Sawyer, P., Paech, B., Heymans, P. (eds.) REFSQ 2007. LNCS, vol. 4542, pp. 247–261. Springer, Heidelberg (2007)
13. ASIS - Algorithms and Software for Improved Safety, http://www.ait.gu.se/english/research_groups/se_management/research_projects/ASIS_Active_Safety_Systems/
14. Mellegård, N., Staron, M.: A Domain Specific Modelling Language for Specifying and Visualizing Requirements. In: The First International Workshop on Domain Engineering, DE@CAiSE, Amsterdam (2009)
15. Winkler, S.: Information Flow Between Requirement Artifacts. Results of an Empirical Study. In: Sawyer, P., Paech, B., Heymans, P. (eds.) REFSQ 2007. LNCS, vol. 4542, pp. 232–246. Springer, Heidelberg (2007)
16. Gorschek, T., Wohlin, C.: Requirements abstraction model. Requir. Eng. 11, 79–101 (2006)
17. Mellegård, N., Staron, M.: Methodology for Requirements Engineering in Model-Based Projects for Reactive Automotive Software. In: Vitek, J. (ed.) ECOOP 2008. LNCS, vol. 5142. Springer, Heidelberg (2008)
18. Maiden, N., Minocha, S., Sutcliffe, A., Manuel, D., Ryan, M.: Co-operative scenario based approach to acquisition and validation of system requirements: how exceptions can help! Interacting with Computers 11, 645–664 (1999)
19. Cox, K., Phalp, K.: Replicating the CREWS use case authoring guidelines experiment. Empirical Software Engineering 5, 245–267 (2000)
20. Phalp, K., Vincent, J., Cox, K.: Improving the quality of use case descriptions: Empirical assessment of writing guidelines. Software Quality Journal 15, 383–399 (2007)
21. Gravino, C., Scanniello, G., Tortora, G.: An Empirical Investigation on Dynamic Modeling in Requirements Engineering. In: Czarnecki, K., Ober, I., Bruel, J.-M., Uhl, A., Völter, M. (eds.) MODELS 2008. LNCS, vol. 5301, pp. 615–629. Springer, Heidelberg (2008)

22. Lange, C., Chaudron, M.: Interactive views to improve the comprehension of UML models - An experimental validation. In: Proceedings - ICPC 2007: 15th IEEE International Conference on Program Comprehension, pp. 221–230 (2007)
23. De Lucia, A., Gravino, C., Oliveto, R., Tortora, G.: Data model comprehension an empirical comparison of ER and UML class diagrams. In: Proceedings of the 16th IEEE International Conference on Program Comprehension, ICPC, pp. 93–102 (2008)
24. Otero, M., Dolado, J.: An empirical comparison of the dynamic modeling in OML and UML. Journal of Systems and Software 77, 91–102 (2005)
25. Gorschek, T., Garre, P., Larsson, S., Wohlin, C.: Industry evaluation of the Requirements Abstraction Model. Requirements Engineering 12, 163–190 (2007)
26. Mohammad, N., Vandewoude, Y., Berbers, Y., Feldt, R.: Suitability of Requirements Abstraction Model (RAM) Requirements for High-Level System Testing. International Journal of Computer and Information Science and Engineering, 2
27. Wong, K., Sun, D.: On evaluating the layout of UML diagrams for program comprehension. Software Quality Journal 14, 233–259 (2006)
28. Lindvall, M.: Evaluating Impact Analysis - A Case Study. Empirical Software Engineering 2, 152–158 (1997)
29. Arisholm, E., Briand, L., Hove, S., Labiche, Y.: The impact of UML documentation on software maintenance: An experimental evaluation. IEEE Transactions on Software Engineering 32, 365–381 (2006)
30. gRAM Experiment Material, `http://www.ituniv.se/~miroslaw/ram-dsl_experiment/`
31. Little, R., Rubin, D.: Statistical Analysis with Missing Data. Wiley, Chichester (2002)
32. Altman, D.: Practical Statistics for Medical Research. Chapman-Hall, Boca Raton (1991)
33. Bowerman, B., O'Connell, R., Murphree, E.: Business Statistics in Practice. McGraw-Hill, New York (2008)
34. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslèn, A.: Experimentation in Software Engineering: An Introduction. Kluwer Academic Publisher, Boston (2000)
35. Staron, M., Kuzniarz, L., Wohlin, C.: Empirical assessment of using stereotypes to improve comprehension of UML models: A set of experiments. Journal of Systems and Software 79, 727–742 (2006)
36. Staron, M.: Using Experiments in Software Engineering as an Auxiliary Tool for Teaching – A Perspective of Students' Learning Process. In: Borsler, J., Eriksson, J. (eds.) 6th Conference on Software Engineering Research and Practice, Sweden, pp. 29–38. Umeå University, Umeå (2006)