Using Model Scoping with Expected Model Elements to Support Software Model Inspections: Results of a Controlled Experiment

Carlos Gracioli Neto[1,4], Amadeu Anderlin Neto[2,5], Marcos Kalinowski[2],
Daniel Cardoso Moraes de Oliveira[1], Marta Sabou[3], Dietmar Winkler[3] and Stefan Biffl[3]
[1]*Computing Institute, Federal Fluminense University, Niterói/RJ, Brazil*
[2]*Department of Informatics, Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Rio de Janeiro/RJ, Brazil*
[3]*CDL-SQI, Information Systems Engineering, Vienna University of Technology, Vienna, Austria*
[4]*Federal Institute of Education, Science and Technology of Mato Grosso, Rondonópolis/MT, Brazil*
[5]*Federal Institute of Education, Science and Technology of Amazonas, Manaus/AM, Brazil*
carlos.neto@roo.ifmt.edu.br, aanderlin@inf.puc-rio.br, kalinowski@inf.puc-rio.br, danielcmo@ic.uff.br,
marta.sabou@ifs.tuwien.ac.at, {dietmar.winkler, stefan.biffl}@tuwien.ac.at

Keywords: Model inspection, model quality assurance, model scoping, empirical study.

Abstract: ***Context:*** Software inspection represents an effective way to identify defects in early phase software artifacts, such as models. Unfortunately, large models and associated reference documents cannot be thoroughly inspected in one inspection session of typically up to two hours. Considerably longer sessions have shown a much lower defect detection efficiency due to cognitive fatigue. ***Goal:*** The goal of this paper is to propose and evaluate a *Model Scoping* approach to allow inspecting specific parts of interest in large models. ***Method:*** First, we designed the approach, which involves identifying *Expected Model Elements* (EMEs) in selected parts of the reference document and then using these EMEs to scope the model (i.e., remove unrelated parts). These EMEs can also be used to support inspectors during detect defection. We conducted a controlled experiment using industrial artifacts. Subjects were asked to conduct UML class diagram inspections based on selected parts of functional specifications. In the experimental treatment, *Model Scoping* was applied and inspectors were provided with the scoped model and the EMEs. The control group used the original model directly, without EMEs. We measured the inspectors' defect detection effectiveness and efficiency and collected qualitative data on the perceived complexity. ***Results:*** Applying *Model Scoping* prior to the inspection significantly increased the inspector defect detection effectiveness and efficiency, with large effect sizes. Qualitative data allowed observing a perception of reduced complexity during the inspection. ***Conclusion:*** Being able to effectively and efficiently inspect large models against selected parts of reference documents is a practical need, in particular in the context of incremental and agile process models. The experiment showed promising results for supporting such inspections using the proposed *Model Scoping* approach.

1 INTRODUCTION

Software engineering models represent abstractions for different aspects of a software system (e.g., structure, behaviour, or interaction). The quality of such models can be of key importance for completing projects successfully (Lange and Chaudron, 2005). Hence, the verification of models prior to the creation of software is of particular relevance for high-quality information systems analysis.

Verifying the correct representation of domain concepts in software models requires human knowledge of the domain. Software inspection methods (Thelin *et al.*, 2003; Travassos *et al.*, 1999) have been found effective to detect defects in requirements and software models in empirical studies (Elberzhager *et al.*, 2012).

Software model inspection typically requires checking whether a conceptual model correctly and completely represents the content of suitable reference documents, such as systems specifications. In practice, models representing abstractions of large enterprise information systems tend to be large as well (e.g., UML class diagrams for large information systems may have hundreds of domain classes). Unfortunately, model inspection studies have only focused on the inspection of small-to-medium sized models so far.

Hence, an important question is how to address cases where large models need to be inspected against their associated reference documents, i.e., involving inspection materials beyond the size that an inspector can cover within the limitations of a traditional one-

pass inspection process (Laitenberger and DeBaud, 2000), of typically up to two hours.

The strategy investigated in this paper to tackle this problem involves scoping the model for selected parts of the reference documents. It is noteworthy that nowadays software is typically developed following iterative or agile processes (Theocharis *et al.*, 2015), where new specifications (e.g., user stories or use cases and their descriptions) are added incrementally. Hence, being able to effectively and efficiently inspect large models against selected (or incremental) parts of reference documents is a practical need.

We introduce the *model scope* concept as a well-defined model part that acts as a filter or view showing only relevant model elements. Our proposed *Model Scoping with Expected Model Elements* approach consists of identifying *Expected Model Elements* (EMEs) in the selected parts of the reference document and then using these EMEs to: (a) scope the model (remove unrelated parts) and (b) guide the inspectors during detect defection. The idea of identifying EMEs within reference documents has been used to allow supporting inspection with crowdsourcing (Winkler *et al.*, 2017). In this paper, we investigate using the EMEs for model scoping.

We conducted a controlled experiment with students using real industrial artifacts (UML class diagrams and selected parts of functional specifications) aiming to understand how *Model Scoping* would influence the model inspection effectiveness and efficiency. Subjects were asked to conduct UML class diagram inspections based on selected parts of functional specifications for two different modules. In the experimental treatment *Model Scoping with EMEs* was applied and inspectors were provided with the EMEs and the scoped model. The control group used the original model directly, without EMEs.

Applying *Model Scoping with EMEs* prior to inspection significantly increased the inspector defect detection effectiveness and efficiency. While this paper presents the first reference-document-based *Model Scoping* approach and it positively influenced model inspection effectiveness and efficiency, applying it properly requires some effort and being able to correctly identify EMEs in selected parts of the reference document.

The remainder of this paper is organized as follows. Section 2 presents the background and related work. Section 3 describes the *Model Scoping* approach. Sections 4 and 5 present the experimental study and its results. Section 6 discusses the results. Section 7 concerns the threats to validity. Finally, Section 8 concludes and identifies future work.

2 SOFTWARE MODEL INSPECTIONS

Software inspection (Fagan, 1976) is a well-established formal defect detection approach that enables efficient defect detection in early software development phases, e.g., during software design.

Over the years, some research has been reported regarding model inspections. Travassos *et al.* (1999), for instance, introduced a reading technique for inspecting object-oriented UML structure and behaviour models regarding the consistency between models and reference information. Sabaliauskaite *et al.* (2002; 2003; 2004) conducted controlled experiments with UML documents to compare and evaluate the effectiveness and efficiency of reading techniques with different levels of guidance. Thelin *et al*. (2003) introduced usage-based reading to guide inspectors by first prioritizing business scenarios and then checking whether a design model correctly represented the information of the most important business scenarios.

These studies have focused on a complete scope of work, typically involving small-to-medium sized models and their related reference documents, which an average inspector can address in two hours to mitigate risks from fatigue. However, the studies did not consider how to address larger inspection objects (e.g., larger models and/or larger reference documents). Therefore, it is unclear to what extent their findings hold for (parts of) larger software artifacts. Winkler *et al.* (2017a; 2017b; 2018) investigated distributing the effort of model inspection on a group of inspectors by using crowdsourcing. While inspectors had to focus on specific parts of a model, model scoping was not directly considered in these investigations.

The idea of using model scoping to support software inspections was also explored by Briand *et al.* (2014). Their results show a significant decrease in effort and an increase in decisions correctness when models are filtered prior to inspection. However, they focused on a particular problem, extracting *design slices* (model fragments) to support safety inspection. In contrast, the approach proposed in this paper, detailed in the next section, is generic and not restricted to a particular type of requirements.

3 MODEL SCOPING WITH EMEs

The core idea of the approach *Model Scoping with EMEs* is to define a model scope based on a selected part of the reference documents (e.g., a selected part of a large functional specification). The scoping is conducted based on *Expected Model Elements*

(EMEs). The EMEs for the selected part of the reference documents should be identified and used to: (a) scope the model (remove parts that are not related to the EMEs) and (b) guide the inspectors during defect detection.

To help identifying the EMEs in the reference documents, guidelines commonly applied when designing the model could be used. For instance, Larman (2004) presents guidelines for identifying classes, attributes, operations and relationships for UML class diagrams based on requirements. Alternatively, Sabou *et al.* (2018) argue on the feasibility of using expert sourcing for such purpose. While such alternatives could be applied, for simplification purposes, in the scope of this paper, we consider that *Model Scoping with EMEs* is applied by a specific role, which typically could be conducted by someone with skills similar to the ones required for identifying such elements when designing conceptual models (e.g., a requirements analyst).

Figure 1 outlines the context in which the *Model Scoping with EMEs* approach is applied. Inputs are the selected part of the reference documents and the model to be reviewed, outputs are the *list of EMEs* and the scoped model.

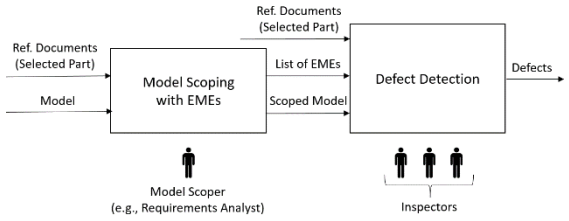

Figure 1: *Model Scoping with EMEs* approach.

Model Scoping with EMEs itself is conducted by following 3 steps:

1. Based on the type of model to be inspected, define the *types of EMEs* to be identified (e.g., for UML class diagrams, EME types are classes, attributes, operations and relationships; for UML state diagrams, EME types are states and transition events).
2. Read the selected part of the reference documents and identify the *list of EMEs* (existing guidelines for identifying EMEs may be used to support this step).
3. Scope the model by removing model elements that are not in the *list of EMEs*. While doing so, to avoid removing relevant contextual (closely related) information, the following elements should not be removed: (a) elements that represent relationships among elements included in the list of EMEs (e.g., an association between classes in an UML class diagram or a transition between two states in an UML state diagram); and (b) elements that are contained in an element included in the list of EMEs (e.g., attributes of a class in an UML class diagram).

During defect detection (not part of the *Model Scoping with EMEs* approach), inspectors can use the *list of EMEs* and the reference documents to verify whether the EMEs are correctly represented in the scoped model as foundation for reporting defects.

4 EXPERIMENT DESCRIPTION

Aiming to understand how *Model Scoping with EMEs* would influence the model inspection effectiveness and efficiency, we designed and conducted an experimental study. The goal, planning (i.e., context, variables, hypotheses, subject selection, design, and instrumentation), and operation of the experiment are detailed in the following subsections.

4.1 Experiment Goal

The experiment goal was defined based on the GQM (Goal-Question-Metric) template (van Solingen *et al.*, 2002). Table 1 presents the experiment goal.

It is noteworthy that, while the approach is intended to be generic (i.e., applicable to any type of model), we instantiated the experiment using UML class diagrams to be inspected with respect to functional information system specifications.

Based on our goal, we derived the following research question: *What is the impact of* Model Scoping with EMEs *on inspection effectiveness and efficiency?* The variables used to answer this research question are described in detail in Subsection 4.3.

Table 1: Experiment goal.

Analyze	the inspection of UML class diagrams using *Model Scoping with EMEs*
for the purpose of	characterization
with respect to	inspection effectiveness & efficiency
from the point of view of	the information systems researcher
in the context of	UML class diagram inspection based on a valid functional specification, conducted by novice inspectors, when compared to not using *Model Scoping with EMEs*.

4.2 Experiment Context

The experiment was conducted in two undergraduate class room trials, representing exact internal replications, involving students enrolled in *Software Engineering* classes at the *Fluminense Federal University*. These students were asked to review UML class diagrams based on correct functional specifications.

In order to make the context more representative, we selected artifacts from a real industrial software project. The project concerned the development of an integrated management system, with several modules. We selected the functional specification and class diagrams of two modules, one module concerning simpler administrative functions and the other module more complex financial billing services. Each functional specification contained an overview description, a list of functional requirements, use case diagrams, and use case descriptions. It is noteworthy that the specifications had been reviewed by professionals and validated by industrial stakeholders.

For the experiment, we selected excerpts of each functional specification related to specific use cases, which were good representatives of use cases to be implemented in a next development cycle and against which the model should be verified before implementation. The excerpt of the administrative module comprised the contextual information (i.e., overview, functional requirements, use case diagram, and use case descriptions) for four small use cases, while the excerpt of the financial billing module comprised the contextual information for two more complex use cases. These specifications were assumed to be correct. We seeded each class diagram with 28 artificial defects related to the respective functional specification excerpts. For distributing the types of defects, we considered the defect taxonomy proposed by Shull (1998), containing the types ambiguity, inconsistency, incorrect fact, omission, and extraneous information. We seeded 7 defects of each type (except inconsistency, given that each task involved inspecting a single model that could therefore not be inconsistent with others). The seeding also considered including defects of different difficulty levels (easy, medium, and hard) for each type.

The first author applied the *Model Scoping with EMEs* activity, building the *lists of EMEs* and the scoped models for both class diagrams using the functional specifications excerpts. The third author reviewed this activity.

4.3 Variables Selection

The independent variable in the inspection experiment is the treatment applied by the groups in order to find defects in the UML model. While both groups used an *ad-hoc* inspection technique (i.e., no specific reading technique), the experimental group received the *Defect Taxonomy*, the *list of EMEs*, and the *scoped model*, and the control group received the *Defect Taxonomy* and the full model.

Regarding dependent variables, we used in the inspection experiment effectiveness and efficiency, defined as follows:

- *Effectiveness* is the ratio between the number of real defects found and the total number of known defects.
- *Efficiency* is the ratio between the number of real defects found and the time spent.

Besides measuring these variables, we collected qualitative feedback in a follow-up questionnaire, inspired by the *Technology Acceptance Model* (TAM) questionnaire (Davis, 1989) regarding the perceived usefulness, ease of use, and behavioral intention of adoption. TAM provides proper theoretical constructs and has been extensively used and validated for this purpose (Turner *et al.*, 2010).

4.4 Hypotheses

Using the variables described in the previous subsection, we defined the following null hypotheses:

- H_{01} – there is no difference in the effectiveness when inspecting UML class diagrams with or without using *Model Scoping with EMEs*.
- H_{02} – there is no difference in the efficiency when inspecting UML class diagrams with or without using *Model Scoping with EMEs*.

4.5 Selection of Subjects

Subjects were intended to represent novice inspectors, to avoid specific knowledge on inspection techniques as a significant confounding factor. We selected students from two undergraduate classes on Software Engineering at the *Fluminense Federal University*, involving 44 daily shift and 10 nightly shift students.

All students filled in a characterization form with objective questions to inform us about their expertise in the topics related to the study: (a) their experience with software development; (b) their experience in object-oriented modeling (UML); and (c) their experience in software inspection. We collected the inspector characterization form from each student and ranked it into: none (N), low (L), medium (M), medium-high (MH), and high (H) experience for each expertise topic.

For instance, regarding experience with software development, a subject was characterized as having:

(N) no experience, if s/he never had contact with software development; (L) low experience, if s/he had contact with software development only in class; (M) medium experience if s/he had contact with programming in an academic project; (MH) medium-high experience, if s/he had contact with programming in an industry project; or (H) high experience, if s/he had industrial experience involving multiple projects. Likewise, the experience in software inspection and in object-oriented modeling (UML) were assigned.

After characterizing the participant's experience, aiming to mitigate threats to validity concerning the distribution of subjects between the groups, we applied the principles of balancing, blocking, and random assignment (Wohlin *et al.*, 2012). For balancing, we attempted to create groups of equal size. However, due to some absences on the day of the experiment execution, one group had a larger number of members. Concerning blocking, we avoided having one team with more experienced subjects than the other. Finally, subjects of equal experience were randomly assigned to the groups.

The experiment comprised two exercises applied on two days (one exercise per day). Students who participated in only one exercise were removed from the analysis. In addition, participants, who found less than 10% of the defects were discarded as outliers because their results were understood as those of students with difficulty understanding the task or who did not engage in the activity for some other reason. Thus, out of 44 participants in the first trial, the data of 32 participants was considered for the data analysis. Regarding the second trial, out of 10 participants, the data of 8 participants was considered.

Table 2 (first trial subjects) and Table 3 (second trial subjects) present the results of the subjects' characterization and the group division. To facilitate understanding of the blocking, we highlighted the most experienced subjects in the tables with grey-tone background filling.

4.6 Experiment Design

The experiment design is a one-factor design with two treatments (ad-hoc with or without *Model Scoping with EMEs*) and two different tasks (artifact inspection exercises, study objects). We adopted a cross-over design to mitigate threats to validity of the experiment concerning: (i) differences among experimental tasks (the influence of the provided exercise matierials in the results); and (ii) the learning effect (the influence of the order in which the treatments are applied on the outcomes). It is noteworthy that the principles applied to distribute the subjects between the groups still enable comparing the results for each individual exercise. The cross-over design is shown in Figure 2.

Table 2: Expertise per participant in the first trial.

Group	ID	Software Development	UML Models	Software Inspection
1	P1	MH	M	M
	P2	L	MH	L
	P3	M	MH	M
	P4	MH	H	L
	P5	M	M	L
	P6	L	M	L
	P7	MH	L	L
	P8	H	H	M
	P9	MH	M	L
	P10	L	M	L
	P11	H	H	L
	P12	L	M	L
	P13	L	MH	M
	P14	L	L	L
	P15	M	M	M
	P16	MH	M	L
	P17	MH	MH	M
	P18	M	L	L
2	P19	H	MH	L
	P20	L	MH	L
	P21	L	L	L
	P22	MH	MH	L
	P23	MH	MH	L
	P24	MH	MH	M
	P25	L	L	M
	P26	L	L	L
	P27	L	L	L
	P28	L	L	L
	P29	MH	MH	L
	P30	M	M	MH
	P31	MH	M	L
	P32	MH	L	L

Table 3: Expertise per participant in the second trial.

Group	ID	Software Development	UML Models	Software Inspection
1	P33	H	H	L
	P34	L	H	L
	P35	M	M	L
	P36	H	H	M
2	P37	H	H	M
	P38	H	H	L
	P39	L	H	L
	P40	M	H	L

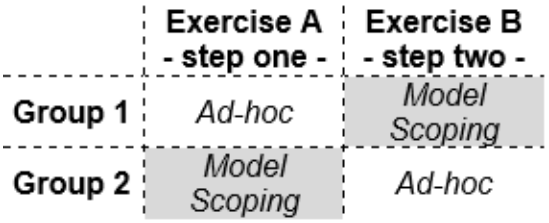

Figure 2: Cross-over experiment design.

4.7 Instrumentation

The instruments used in this experiment were: consent and characterization forms; training material on model inspection; task description, with instructions to perform the inspection, including the defect taxonomy; a defect reporting form; an excerpt of a real industrial functional specification (the overview of the system module to be inspected, the functional requirements, the use case diagrams and their descriptions); and an industrial UML class diagram with defects seeded by the authors. When *Model Scoping with EMEs* was applied, instead of the UML class diagram, subjects received the *scoped UML class diagram* and the *list of EMEs*.

The scoped models were prepared by applying *Model Scoping with EMEs* on the defect seeded UML class diagrams based on the functional specification excerpts. For the administrative module, researchers identified a *list of EMEs* in the contextual information (i.e., overview, functional requirements, use case diagram, and use case descriptions) for four small use cases (maintaining company data, maintaining customer data, maintaining tax information, and maintaining cost centers). Then, using this *list of EMEs*, researchers scoped the UML class diagram by applying *Model Scoping with EMEs* as described in Section 3. This process reduced the domain class diagram for this module from 19 to 12 classes.

Similarly, for the billing module, based on the two (more complex) selected use cases (registering invoices for provided services and registering payments for invoices), the class diagram was reduced from 22 to 16 classes. Figure 3 shows the cut-outs performed during the scoping for the billing module. These cut-outs are related to entities concerning the physical emission of invoices (in communication with the federal invoice emission system) and of other receipts, which are detailed in two other use cases that were not part of the selection. Thus, these model parts would only add complexity to the inspection, as they were not related to the selection against which the model should be verified.

It is noteworthy that, while the scoping allowed cutting out some irrelevant parts for the inspection scope, many classes still remained, given that the selected use cases were very central for each module. It is noteworthy that the researchers did not bias the process, as these artifacts were used as provided from the industrial partner. The excerpt of the functional specification for the inspection task was planned for an inspection of up to 75 minutes (industrial inspections are recommended to take no longer than 2 hours). It is noteworthy that if one of these modules even would have had hundreds of use cases and hundreds of domain classes, the cutout for the selected use cases would still select the same relevant amount of classes, related to the EMEs contained in the excerpt.

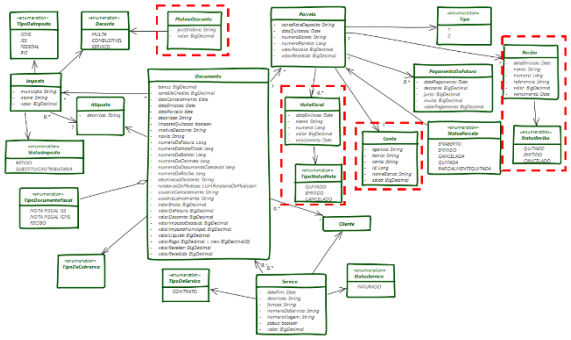

Figure 3: Cut-outs during Model Scoping (dashed rectangles) for the billing module.

4.8 Experiment Operation

The experiment was conducted on three days. On the first day, the participants answered the characterization form in order to allow dividing them into experiment groups. Prior to the execution of the experiment, on the second day, basic concepts of the diagrams and relevant types of defects were reviewed with participants in a training session of 15 minutes. After that, the inspection was conducted as follows.

On the first round (Exercise A), Group 1 inspected the full UML class diagram with 19 classes, based on the requirements document using the ad-hoc treatment. On the other hand, the participants of Group 2 (*Model Scoping with EMEs* treatment) inspected the scoped UML class diagram with 12 classes. Besides the exact same functional specification excerpt, the subjects of Group 2 used the *list of EMEs*. All subjects had to report the defects found in the diagram. In total, the inspectors had 75 minutes to perform the inspection, including reporting the defects found. It is important to mention that communication between participants was not allowed. After the inspection, the participants answered the follow-up questionnaire.

On the last day, the procedures conducted for Exercise A (administrative module) were repeated for Exercise B (billing module). However, the group that was previously assigned to the ad-hoc treatment was now assigned to the *Model Scoping* treatment and vice-versa. We used the exact same procedures and instrumentation for both trials. In the next section, we present the results of the experiment.

5 RESULTS

This section reports on the analysis of quantitative and qualitative data collected in the experiment.

5.1 Quantitative Analysis

The data analyst obtained the quantitative data from the defect reporting form, resulting from the experimental task. The data analyst counted the number of real defects found, false positives, and time used by each subject. Following the design presented in Figure 2, Table 4 presents both the results per subject and the overall result per treatment of the Exercise A and Exercise B, respectively.

The data analyst performed the statistical analyses using the statistical tool SPSS v 25.0.0. For hypothesis testing, the data analyst used the *Mann-Whitney* non-parametrical test with $\alpha = 0.10$. This statistical significance level has been found acceptable for software engineering experiments, which typically involve small sample sizes (Dybå *et al.*, 2006). In addition, to increase our sample size, we decided to aggregate the results of both trials, which does not expose us to additional threats to validity, given that the trials were exact internal replications.

Figure 4 shows descriptive results as boxplots that compare the effectiveness indicator of both exercises (A and B). It is possible to observe that the median for the *Model Scoping with EMEs* treatment in both exercises (A median 0.41, A mean 0.41, B median 0.50, B mean 0.48) is higher than the median for the ad-hoc treatment (A median 0.32, A mean 0.36, B median 0.28, B mean 0.29). The effect sizes (standardized mean difference between two populations) for exercises A and B are respectively 0.47 (medium) and 1.57 (very large). Thus, the group that used *Model Scoping with EMEs* was more effective than the control group. Also, using the *Mann-Whitney* test showed a significant difference between the groups ($p = 0.075$ for Exercise A and $p = 0.001$ for Exercise B).

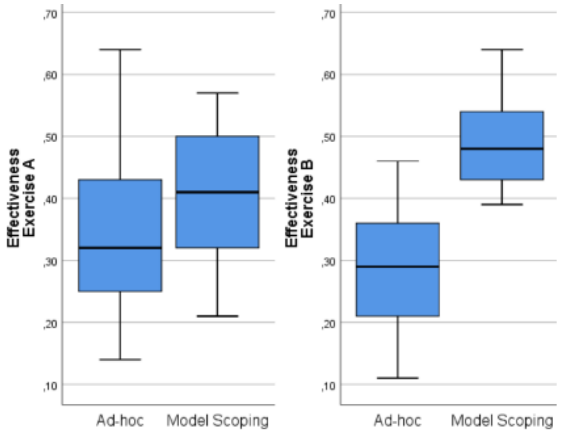

Figure 4: Effectiveness (defects found / seeded defects) indicator for defect detection.

These results suggest that the use of *Model Scoping with EMEs* allowed the inspectors to achieve higher defect detection effectiveness. These results allow rejecting the null hypothesis H_{01}.

Figure 5 shows boxplots comparing the efficiency of the treatments. For efficiency it is also possible to observe that the median of the *Model Scoping* treatment in both exercises (A median 10.64, A mean 10.60, B median 11.05, B mean 11.53) is higher than the median of the ad-hoc treatment (A median 7.71, A mean 8.58, B median 7.67, B mean 7.43). The effect sizes for exercises A and B are respectively 0.65 (medium-to-large) and 1.30 (very large).

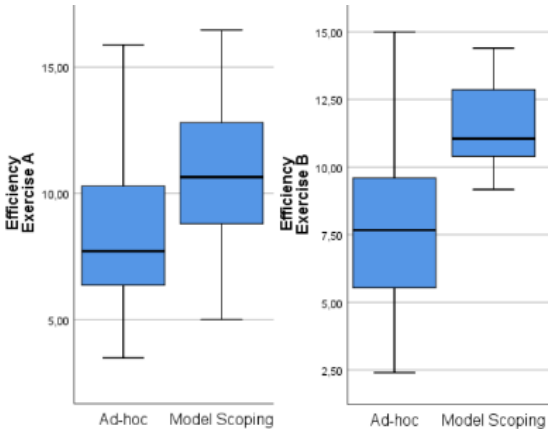

Figure 5: Efficiency (defects found per hour) indicator for defect detection.

Thus, the group that used the *Model Scoping with EMEs* was more efficient when compared to the control group. Again, the *Mann-Whitney* test shows the difference between the groups to be statistically significant ($p = 0.024$ for Exercise A and $p = 0.001$ for Exercise B). Hence, using *Model Scoping with EMEs* allowed inspectors to achieve higher efficiency. These results allow rejecting the null hypothesis H_{02}.

5.2 Qualitative Analysis

The data analyst collected qualitative data from the follow up questionnaires. The participants rated their perception on the complexity of the task (as very complex, complex, simple, or very simple), and noted difficulties they had during the experiment. For participants using the *Model Scoping with EMEs* treatment (which received a *list of EMEs* to support the inspection), participants filled in the TAM questionnaire with a five point *Likert* scale (completely disagree, disagree, neutral, agree, and completely agree) for the questions on perceived usefulness, ease of use, and intention of adoption.

Table 4: Quantitative results per subject and treatment.

		Exercise A					Exercise B				
	ID	**Duration (min)**	**Defects found**	**False positives**	**Effect.**	**Effic.**	**Duration (min)**	**Defects found**	**False positives**	**Effect.**	**Effic.**
Group 1	P1	71	8	4	29%	6.76	73	13	14	46%	10.68
	P2	72	10	4	36%	8.33	75	13	4	46%	10.40
	P3	72	15	8	54%	12.50	75	17	7	61%	13.60
	P4	69	13	6	46%	11.30	73	14	15	50%	11.51
	P5	68	7	11	25%	6.18	66	12	0	43%	10.91
	P6	71	9	4	32%	7.61	78	12	14	43%	9.23
	P7	68	10	6	36%	8.82	75	14	5	50%	11.20
	P8	66	7	4	25%	6.36	72	11	6	39%	9.17
	P9	70	9	5	32%	7.71	75	15	10	54%	12.00
	P10	69	10	4	36%	8.70	75	12	0	43%	9.60
	P11	72	17	9	61%	14.17	76	17	5	61%	13.42
	P12	70	9	12	32%	7.71	70	14	7	50%	12.00
	P13	72	6	13	21%	5.00	63	15	5	54%	14.29
	P14	69	4	8	14%	3.48	72	13	11	46%	10.83
	P15	70	12	9	43%	10.29	75	18	23	64%	14.40
	P16	72	7	7	25%	5.83	75	12	3	43%	9.60
	P17	69	15	13	54%	13.04	73	13	20	46%	10.68
	P18	69	11	9	39%	9.57	70	15	12	54%	12.86
	P33	68	18	4	64%	15.88	75	16	18	57%	12.80
	P34	75	9	7	32%	7.20	75	13	4	46%	10.40
	P35	71	8	4	29%	6.76	75	12	3	43%	9.60
	P36	75	7	13	25%	5.60	75	18	15	64%	14.40
Group 2	P19	60	11	4	39%	11.00	50	7	11	25%	8.40
	P20	67	12	4	43%	10.75	63	4	0	14%	3.81
	P21	51	14	5	50%	16.47	50	8	12	29%	9.60
	P22	71	9	3	32%	7.61	66	10	2	36%	9.09
	P23	72	16	10	57%	13.33	73	12	8	43%	9.86
	P24	72	9	9	32%	7.50	75	8	4	29%	6.40
	P25	71	11	6	39%	9.30	66	4	14	14%	3.64
	P26	57	10	4	36%	10.53	74	4	7	14%	3.24
	P27	63	15	10	54%	14.29	75	12	3	43%	9.60
	P28	60	9	3	32%	9.00	60	10	5	36%	10.00
	P29	67	9	15	32%	8.06	52	13	6	46%	15.00
	P30	72	6	14	21%	5.00	75	7	4	25%	5.60
	P31	61	12	4	43%	11.80	65	6	5	21%	5.54
	P32	62	14	7	50%	13.55	55	9	6	32%	9.82
	P37	75	16	2	57%	12.80	75	8	2	29%	6.40
	P38	68	13	4	46%	11.47	75	3	17	11%	2.40
	P39	75	12	3	43%	9.60	59	8	18	29%	8.14
	P40	75	11	0	39%	8.80	75	9	3	32%	7.20

Regarding the perceived complexity of the task, shown in Table 5, there was a subtle difference between the treatments. While overall the tasks were considered complex by the participants, which was expected as they were handling real industrial artifacts, the amount of participants that perceived the complexity as simple was larger for the *Model Scoping with EMEs* treatment. In particular, it is easy to observe that Exercise B (the billing system, which had more complex business rules) was perceived as more complex by the participants when using the ad-hoc treatment. Thus, the *Model Scoping* and the EMEs seemed to have reduced the perceived complexity. For instance, participant P34 mentioned "*difficulties to understand the description of use cases and then compare against the class diagram*" and suggested that "*the diagram could be […] smaller*". Participant P1 mentioned that "*browsing through the numerous descriptions when searching defects is a complex task […], some kind of guidance is needed*".

Table 5: Complexity of the experimental tasks as perceived by participants.

Treatment	Mod.	Complexity			
		Very Simple	Simple	Complex	Very Complex
Ad-hoc	Adm.	0%	5%	90%	5%
	Billing	0%	12.5%	50%	37.5%
Model Scoping	Adm.	0%	15%	75%	10%
	Billing	0%	22.5%	72.5	5%

The TAM questionnaire was applied only with the *Model Scoping with EMEs* treatment. In general, the inspectors perceived receiving the EMEs to support inspection as useful, easy to use and would like to receive such support. Note that the participants did not know that they were inspecting a scoped model. Indeed, all participants agreed or completely agreed with the TAM questions. Moreover, we noticed that many inspectors, who used the *Model Scoping with EMEs* treatment in the first exercise, mentioned that they missed this kind of support in the second exercise. For instance, participant P32 mentioned that the exercise contained "*a lot of information*" and that it was "*difficult to understand what to do without a guide*". He also requested "*the extra page (EMEs) of the previous exercise*". Participant P28 mentioned difficulties in "*identifying defects in relationships between classes*" and reported that "*the use of EMEs*" could make the task more enjoyable.

6 DISCUSSION

In this section we discuss the results of the quantitative and qualitative analyses of experiment data.

The main goal of the quantitative analysis was investigating how the *Model Scoping with EMEs* approach would affect the effectiveness and efficiency of inspectors when reviewing models based on reference documents. We selected real industrial class diagrams and their related functional specifications as reference documents for two modules of an integrated management system. As input for the experiment, we selected a set of use cases (and their contextual information) for each module and conducted the *Model Scoping with EMEs* activity as described in Section 3.

The experiment results indicate that applying the proposed *Model Scoping with EMEs* approach before the inspection improved both effectiveness and efficiency of the inspectors when reviewing the UML class diagrams against the functional specification excerpts. Moreover, the results were statistically significant and had large effect sizes.

As a complement, the qualitative analysis indicated that inspectors perceive their inspection tasks as less complex if *Model Scoping with EMEs* was applied before the inspection (i.e., they inspect a *scoped model* and receive a *list of EMEs*). They also perceived it useful to receive candidate EMEs to support the verification of the model against the reference documents.

These results indicate applying *Model Scoping with EMEs* before inspections in situations where large UML class diagrams are to be inspected against excerpts (or increments) of functional specifications. This is in line with the results by Briand *et al.* (2014), who also observed effort reductions when scoping models for their specific purposes (addressing safety requirements). While we assume *Model Scoping with EMEs* applicable for any kind of model, we limit the advice and recommendations to the findings of our specific experimental setting.

It is also noteworthy that properly applying *Model Scoping with EMEs* requires some effort and being able to correctly identify EMEs in selected parts of the reference document. The first author took 1 hour to identify the EMEs within each functional specification excerpt and then used the EMEs to scope the model. Still, we believe that the effort is worthwhile in many cases (in particular for large models) considering that inspection teams usually involve three to five inspectors who would perceive their task as less complex and would be more effective and efficient. Approaches for identifying expected model elements in natural text have been investigated (Sabou *et al.*, 2018) and could be used in this context.

7 THREATS TO VALIDITY

In this section, we present and discuss the threats to the validity of the controlled experiment, organized by the categories described by Wohlin *et al.* (2012).

7.1 Internal Validity

The tasks of the experiment were performed by the participants individually and under the supervision of one of the researchers. Communication among the participants was not allowed.

After characterizing the participants' experience, the principles of balancing, blocking and random assignment (Wohlin *et al.*, 2012) were applied to mitigate threats to validity regarding the distribution of subjects between groups.

7.2 External Validity

Regarding artifact representativeness, we used real industrial UML class diagrams and functional specifications. Still, they represent artifacts from one specific organization. For subject representativeness, we used students to represent novice inspectors. Using students for this purpose is a valid abstraction, in particular considering that they have been properly characterized (Falessi *et al.*, 2018).

Still, artifacts are from one specific organization and subjects come from one specific context. Therefore, there are no claims of external validity throughout the paper and the study validity is specific to the context in which it was performed. Indeed, we call for replications involving a variety of artifacts (e.g., models and reference documents) and contexts to reinforce the experimental evidence.

7.3 Construct Validity

Regarding the treatment, the experiment task involved inspections using real industrial artifacts. We employed a cross-over design to isolate confounding factors related to the experimental tasks (i.e., the exercises) and the learning effect. In addition, it is important to note that the metrics used to measure effectiveness and efficiency have been widely used in empirical studies on software inspection.

A potential threat to the construct validity regards the defects seeded in the diagrams. To mitigate this threat, the defects were distributed in a harmonic way as to their types, as well as the levels of difficulty of detection.

7.4 Conclusion Validity

To improve conclusion validity, we aggregated the subjects from both trials to increase the sample size for data analysis. This was possible given that we had two exact internal replications following the same cross-over design. Outliers were carefully removed to avoid influence on the results. We applied appropriate statistical tests and our results were statistically significant with large effect sizes. Based on these results, we are confident that the drawn conclusions are valid for the reported experimental setting.

8 CONCLUSION AND FUTURE WORK

In this paper, we introduced the model scope concept as a well-defined model part that acts as a filter or view showing only relevant model elements. We proposed and evaluated an approach for *Model Scoping with Expected Model Elements*. The approach consists of identifying *Expected Model Elements* (EMEs) in the selected parts of the reference document and then using these EMEs to scope the model and guide inspectors during detect defection.

For evaluation, we conducted a controlled experiment with students using real industrial artifacts aiming to understand how *Model Scoping with EMEs* would influence the model inspection effectiveness and efficiency. The experiment results indicate, with statistical significance and large effect sizes, that applying *Model Scoping with EMEs* before the inspection improved both, effectiveness and efficiency of the inspectors when reviewing UML class diagrams against the functional specification excerpts. Additionally, qualitative data indicated that inspectors perceive their inspection tasks less complex when *Model Scoping with EMEs* has been applied before inspection.

Our takeaway message is that we recommend applying *Model Scoping with EMEs* before inspections in situations where large UML class diagrams are to be inspected against excerpts (or increments) of functional specifications. Nevertheless, further investigations to precisely estimate in which cases *Model Scoping with EMEs* would be (most) worthwhile the upfront investment are needed. We call out to the community for replicating the reported experiment on *Model Scoping with EMEs*, including the use with other diagrams in other contexts, to reinforce experimental evidence and improve external validity.

ACKNOWLEDGEMENT

The financial support by the Christian Doppler Research Association, the Austrian Federal Ministry for Digital & Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged.

REFERENCES

Briand, L., Falessi, D., Nejati, S., Sabetzadeh, M. and Yue, T., 2014. Traceability and SysML design slices to support safety inspections: A controlled experiment. ACM Transactions on Software Engineering and Methodology (TOSEM), 23(1): 43p.

Davis, F.D., 1989. Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. MIS Quarterly, vol. 13.

Dybå, T., Kampenes, V. B., Sj□berg, D. I. K., 2006. A systematic review of statistical power in Software Engineering experiments. Information and Software Technology 48 (8):745-755.

Elberzhager, F., Münch, J., Nha, V.T.N., 2012. A systematic mapping study on the combination of static and dynamic quality assurance techniques. In: *Information and Software Technology*, 54(1):1-15.

Fagan, M.E., 1976. Design and code inspections to reduce errors in program development. IBM Systems Journal, 15(7): 182-211.

Falessi, D., Juristo, N., Wohlin, C., Turhan, B., Münch, J., Jedlitschka, A. and Oivo, M., 2018. Empirical software engineering experts on the use of students and professionals in experiments. Empirical Software Engineering, 23(1): 452-489.

Laitenberger, O., DeBaud, J.M., 2000. An encompassing life cycle centric survey of software inspection. In: *J. of Syst. and Software*, 50(1):5-31.

Lange, C.F., Chaudron, M.R., 2005. Managing model quality in UML-based software development. IEEE International Workshop on Software Technology and Engineering Practice, pages 7-16.

Larman, C., 2004. Applying UML and Patterns. 3rd Edition, Prentice Hall.

Sabaliauskaite G., Matsukawa F., Kusumoto S., Inoue K., 2002. An experimental comparison of checklist-based reading and perspective-based reading for UML design document inspection. Int. Symp. on Empirical Software Engineering, pages 148–157.

Sabaliauskaite G., Matsukawa F., Kusumoto S., Inoue K., 2003. Further investigations of reading techniques for object-oriented design inspection. Information and Software Technology, 45(9): 571–585.

Sabaliauskaite G., Kusumoto S., Inoue K., 2004. Assessing defect detection performance of interacting teams in object-oriented design inspection. Information and Software Technology, 46(13): 875–886.

Sabou, M., Winkler, D., Petrovic, S., 2018. Expert Sourcing to Support the Identification of Model Elements in System Descriptions. In: *SWQD 2018*, pages 83-99.

Shull, F., 1998. Developing Techniques for Using Software Documents: A Series of Empirical Studies. Ph.D. thesis, University of Maryland, College Park.

Solingen, R. van, Basili, V., Caldiera, G., Rombach H. D., 2002. Goal Question Metric (GQM) Approach. In: *Encyclopedia of Software Engineering*.

Thelin, T., Runeson, P., Wohlin, C., 2003. An experimental comparison of usage-based and checklist-based reading. In: *TSE*, 29(8):687–704.

Theocharis, G., Kuhrmann, M., Münch, J. Diebold, P., 2015. Is water-scrum-fall reality? on the use of agile and traditional development practices. International Conference on Product-Focused Software Process Improvement, pages 149-166.

Travassos, G., Shull, F., Fredericks, M., Basili, V., 1999. Detecting defects in object-oriented designs: Using reading techniques to increase software quality. In: *Proc. of OOPSLA*, pages 47-56.

Turner, M., Kitchenham, B., Brereton, P., 2010. Does the technology acceptance model predict actual use? A systematic literature review. Information and Software Technology, vol. 52: 463-479.

Winkler, D., Kalinowski, M., Sabou, M., Petrovic, S., Biffl, S., 2018. Investigating a Distributed and Scalable Model Review Process. CLEI Electronic Journal, 21(1): 4:1–4:13.

Winkler, D., Sabou, M., Petrovic, S., Caneiro, G., Kalinowski, M., Biffl, S., 2017a. Improving model inspection with crowdsourcing In: International Workshop on Crowdsourcing in Software Engineering, ICSE, Buenos Aires, Argentina, pages 30-34.

Winkler, D., Sabou, M., Petrovic, S., Carneiro, G., Kalinowski, M., Biffl, S., 2017b. Improving Model Inspection Processes with Crowdsourcing: Findings from a Controlled Experiment. In EuroSPI, pages 125–137.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., Wesslén, A., 2012. Experimentation in Software Engineering. Berlin, Heidelberg: Springer Berlin Heidelberg.