Improving Model Inspection Processes with Crowdsourcing: Findings from a Controlled Experiment

Dietmar Winkler[1] Marta Sabou[1] Sanja Petrovic[1] Gisele Carneiro[2]
Marcos Kalinowski[2] Stefan Biffl[1]

[1] Vienna University of Technology, Institute of Software Technology, Favoritenstrasse 9-11, 1040 Vienna, Austria
<dietmar.winkler, reka.sabou, sanja.petrovic, stefan.biffl>@tuwien.ac.at

[2] Fluminense Federal University, Graduate Programm in Computing, Niterói, RJ, Brazil
{gcarneiro, kalinowski}@ic.uff.br

Abstract. The application of best-practice software inspection processes for early defect detection requires considerable human effort. Crowdsourcing approaches can support inspection activities (a) by distributing inspection effort among a group of human experts and (b) by increasing inspection control. Thus, the application of crowdsourcing techniques aims at making inspection processes more effective and efficient. In this paper, we present a crowdsourcing-supported model inspection (CSI) process and investigate its defect detection effectiveness and efficiency when inspecting an Extended Entity Relationship (EER) model. The CSI process uses so-called Expected Model Elements (EMEs) to guide CSI inspectors during defect detection. We conducted a controlled experiment on defect detection effectiveness, efficiency, and false positives. While CSI effectiveness and efficiency is lower for CSI inspectors, the number of false positives decreases. However, CSI was found promising for increasing the control of defect detection and supports the inspection of large-scale engineering models.

Keywords: Software Inspection, Software Models, Crowdsourcing, Empirical Study, Process Improvement.

1 Introduction

Verifying software engineering models prior to the creation of software is of particular relevance for database design, creating software architectures, and agreeing on mission critical test cases [1]. In model-based software engineering, the quality of increasingly large models has become a critical issue for deriving mission-critical system parts as software artifacts are directly derived from models [2]. A major challenge is to verify the correct representation of reference documents in corresponding models that were derived from these reference documents.

Software inspection [3] is an important human-based approach to detect defects in software artifacts that are hard to detect automatically. Software model inspection typically focuses on checking whether a conceptual model correctly and completely represents the content of suitable reference documents, such as systems specifications [4]. Although the verification of software models is key to ensure high quality for mission-critical software, this verification faces several challenges [6]. First, large software models and large associated reference documents are challenging to inspect with the limited resources in one inspection session. Second, there are only limited means for the cost-effective coordination of inspector teams to achieve systematic coverage of large inspection materials. Third, best-practice reading techniques for inspection can systematically guide the inspectors in an inspection team, but there is only limited tool support available for the coordinated inspection of models [6].

Based on these challenges, we explore how software model verification can be improved with *Human Computation and Crowdsourcing (HC&C)* methods. HC&C techniques rely on splitting large and complex problems into multiple, small tasks that can be solved quickly by an average contributor in a suitable population and then coordinating the collection and aggregation of individual micro-contributions into a larger result [7]. However, the application of HC&C techniques in traditional inspection processes needs some adaptations on process and method level.

In this paper we present the *Crowdsourced Software Inspection (CSI)* process based on traditional best-practice inspection and HC&C techniques and evaluated the CSI process in a controlled experiment with focus on defect detection performance, i.e., defect detection effectiveness, efficiency, and false positives. The remainder of this paper is structured as follows: Section 2 presents related work and Section 3 presents our study goals and research issues. We present the CSI process in Section 4, the study design in Section 5, and the results in Section 6. Section 7 discusses the results and presents threats to validity. Finally, Section 8 concludes and presents future work.

2 Background and Related Work

This section summarizes background and related work on software inspection and crowdsourcing in Software Engineering.

2.1 Software Inspection

Software inspections enable defect detection in different types of artifacts, such as text documents, models, or software code before delivering them to next software life cycle activities [3]. Fig. 1 presents the traditional process [3] consisting of five steps: (1) *Inspection Planning* for preparation, scheduling, and team composition; (2) *Individual Defect Detection* by inspection team members to identify defects in related artifacts; (3) *Team meetings* to collect, discuss, and aggregate individual defect reports; (4) *Rework* by the authors based on the team defect list; and (5) *Inspection Closure*. Reading techniques can be used to guide inspectors through the defect detection process [8].

However, traditional inspection is time-consuming and involves expensive experts as inspectors. The typical two-hour limit recommended for defect detection and team

meeting activities [4] limits the scope of the inspection artifact and requires appropriate scoping during inspection preparation. Appropriate scoping is particularly challenging for large software models and reference documents. To address high effort for inspection, tool support can help to reduce effort and improve coordination of activities and results. Some tool support has been proposed to support overall inspection process coordination. For instance, the web-based tools presented in [9] and [10] allowed reducing inspection meeting effort by supporting a slightly modified inspection process that replaces the face to face meetings with asynchronous discussions. However, those tools do not support scoping during inspection preparation.

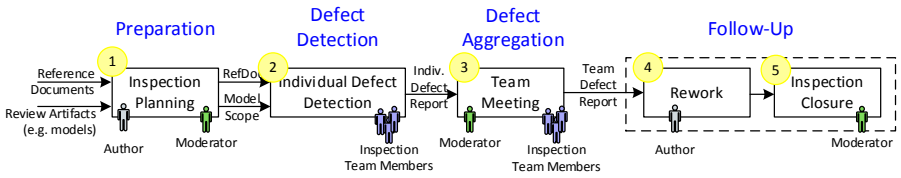

Fig. 1. Traditional Software Inspection Process.

Analyzing traditional inspection processes [11] there is a need for (a) appropriate scoping to enable efficient and effective defect detection for large-scale software artifacts; (b) systematic method support for defect detection, validation of defects, and coordination of inspection activities as these tasks are typically executed manually; and (c) guidelines for defect detection, such as reading techniques for model inspection as traditional reading techniques provide limited support for model inspection.

2.2 Crowdsourcing in Software Engineering

HC&C methods have recently enjoyed a significant uptake to solve a diversity of Software Engineering (SE) tasks [7] and lead to the emerging research area of *Crowdsourced Software Engineering (CSE)* defined as *"the act of undertaking any external software engineering tasks by an undefined, potentially large group of online workers in an open call format"* [7][12]. A plausible reason for the intensified interest in the application of crowdsourcing techniques in SE is the appearance of mechanized labor *(micro-tasking)* platforms such as *Amazon Mechanical Turk*[1] or *CrowdFlower (CF)*[2] . Micro-tasking differs from traditional models of distributed work in SE, such as collaboration and peer-production, by being more scalable due to parallel execution of small task units by non-necessarily expert contributors. Therefore, micro-tasking is promising to address challenges in software inspection, e.g., improved coordination and control (of inspection team members, tasks, and results), increased coverage (as some parts of large artifacts might not be covered with traditional, weakly-coordinated approaches), more diversity (support for dealing with various inspection artifacts), and accelerated inspection processes (by parallelization of small tasks).

[1] Amazon Mechanical Turk: www.mturk.com
[2] CrowdFlower: www.crowdflower.com

In context of SE, Mao *et al.* [12] have recently performed a survey on the application of CSE approaches within the software development life cycle with focus on micro-tasking, collaboration, and peer-production. While the authors identified contributions of CSE to most of the software life cycle phases (i.e., planning and analysis (17 papers), design (4 papers), implementation (26 papers), testing (22 papers), and maintenance (26 papers)) software model inspection tasks have not been addressed by CSE approaches yet. Our work addresses on this gap and explores how HC&C micro-tasking principles can be used to improve software inspection processes through expert-based crowdsourcing, typically available within a community or organization.

3 Research Issues

Main goal of this paper is to improve model inspection with HC&C technologies with focus on improving defect detection capabilities. Main questions are (a) how to adapt/extend a best-practice inspection approach with crowdsourcing and (b) what are the effects of this CSI process approach. We describe the CSI process approach and evaluate this process in a controlled experiment by following the guidelines described by Wohlin *et al.* [13]. For evaluation purposes we applied the traditional Pen&Paper (P&P) approach as control group and derived two research hypotheses.

H1.0. Defect detection effectiveness of the CSI process is similar to traditional software inspection. The alternative Hypothesis H1.1 is that the CSI process is less effective compared to P&P inspection. We define effectiveness as the share of identified true defects and seeded defects.

H2.0 Defect detection efficiency of the CSI process is similar to traditional inspection. The alternative hypothesis H2.1 is that CSI performs better compared to the P&P approach, because CSI inspectors focus on small entities with clear tasks for defect detection. We define efficiency as the number of identified true defects per unit of resources spent, i.e., in the context of this paper defect found per hour.

4 Crowdsourced Inspection (CSI)-Process

To address these challenges, we adapted the traditional Software Inspection process to exploit HC&C capabilities with *Crowdsourcing-based Software Inspection (CSI).*

4.1 Expected Model Elements (EMEs)

To manage small tasks for crowdsourcing application in context of model inspection, related model elements for the inspection process are required. We introduced the concept of *Expected Model Elements (EMEs)*, i.e., key concepts that can be expected in the model under inspection based on inputs from reference documents, i.e., software requirements and specifications [14]. The elicitation of these EMEs from reference documents can be done with a text analysis, executed by human experts or based on natural language processing (NLP) approaches [15]. In context the CSI process, human experts provide a set of EMEs that build an *Extended Entity Relationship*

(EER) model [16], i.e., from the requirements specification they derive entities, entity attributes, relationships, and relationship attributes. These EMEs are further applied to analyze the model and to identify deviations, i.e., defects.

4.2 CSI Process Approach

Based on the traditional inspection approach, we focus on the *Preparation* and *Software Inspection phases* (i.e., individual defect detection and team meeting) as core part of the inspection process. Fig. 2 presents the adapted CSI process that consists of four phases: (1) *Preparation*; (2) *Text Analysis* to identify Expected Model Elements (EMEs); (3) *Model Analysis* to find defects based on EMEs; and (4) *Defect Analysis and Aggregation*: Note that the *Follow-Up Phase* (compared to the traditional inspection process, Fig. 1) has been excluded from Fig. 2 because of readability issues.

In the *Preparation Phase*, the moderator performs inspection planning and takes, in addition, the CSI management role. The author supports the moderator. Main tasks include (a) scoping of inspection artifacts, (b) preparing the crowdsourcing environment, and (c) uploading reference documents and inspection artifacts into the crowdsourcing platform. The *Text Analysis Phase* includes the analysis of the reference document with focus on identifying EMEs, executed by the CSI workers and reported via the crowdsourcing application (2a) and the analysis and aggregation of delivered EMEs, executed by the CSI management (2b). This process step also includes removing duplicate EMEs and the mapping of synonyms.

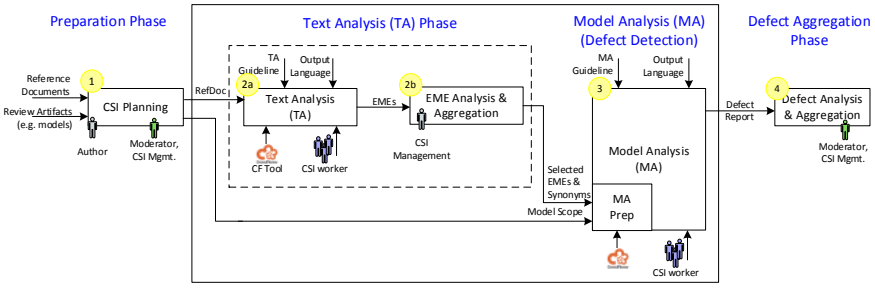

Fig. 2. Crowdsourced Inspection (CSI) Process Approach.

In the *Model Analysis Phase*, the CSI management prepares a selected set of EMEs, derived from manual text analysis and EME aggregation (output of step 2b) for model inspection. Furthermore, the CSI management prepares the model or a sub-model to be inspected. Sub-models are of specific interest if large models have to be inspected. In context of our study, there is no need for scoping the model because of an acceptable model size. For defect detection (i.e., model analysis), CSI workers receive an EME, e.g., an entity attribute, locate it in the model, and report either that the EME was modeled correctly or report at least one defect. In the *Defect Aggregation Phase*, the CSI management aggregates reported defects. Note that the author is not included in the crowdsourced defect detection tasks. He receives the aggregated defect detection reports for the follow-up phase, i.e., rework.

5 Experimental Study

In a controlled experiment we evaluated the effects of the CSI process with focus on defect detection effectiveness, efficiency, and false positives. The study has been initially described in [14] and [17]. The goal of this paper is to extend the evaluation parts with focus on improving the software inspection process.

Following guidelines for conducting controlled experiments [13], the experimental study includes three key phases: *study preparation*, *study execution*, and *data collection and analysis*. The *study preparation phase* focuses on setting up the controlled experimental in terms of material preparation, tool setup for CSI and traditional inspection, study group definition, formal language for reporting EMEs and defects (i.e., "output language" in Fig. 2), and experiment schedule. Material includes reference documents and scenarios, guidelines, list of reference defects (seeded defects), and questionnaires (to capture participant experience and feedback). To test the study setup, we piloted the study in two iterations with domain experts to ensure the basic feasibility of the method and tool support in context of the study material. The *study execution phase* includes tutorials for CSI and P&P inspectors and the experiment. *Data collection and analysis* focused on preliminary data screening, assignment of reported defects to reference defects, and the evaluation of research questions. We analyzed the defect reports according to a list of reference defects, which was extended if a new true defect was found. This paper reports on preliminary findings of the study after the first data screening and data analysis run.

In the following we describe the main aspects of the controlled experiment, i.e., the *study process*, *participants*, *study material*, *tool configuration*, *study variables*, and applied concepts for *data collection & analysis*.

Study Process. Fig. 3 presents an overview of the study process. The experiment was conducted in a classroom setting with 75 participants, who had to solve similar defect detection tasks in two main groups: CSI and P&P inspection. The majority of participants (63 participants, 84%) used the CSI approach while the rest of participants (12 participants, 16%) were invited to use the traditional P&P approach. CSI inspectors were further divided in two sub-groups (A and B), who executed text and model analysis in a different sequence. P&P inspectors were assigned to group C (i.e., the control group). Group assignment was conducted randomly by using a sort card mechanism. Note that CSI and P&P participants were assigned to different rooms to avoid communication and interaction between individual groups.

In the first 30 minutes of the experiment, all inspectors received a tutorial on the processes, assigned to their group, to ensure that they were able to conduct the process steps and could discuss open issues before the experiment. For training purposes we used a different reference document and EER model in the tutorial, i.e., selected scenarios from a parking garage use case with a few defects and EMEs. The results of this tutorial can also give a first indication on the inspector capability for detecting defects in the target setting. The following 120 minutes were dedicated to the experiment with focus on a different application domain, i.e., restaurant scenarios. While

group C applied a traditional (P&P) best-practice software inspection approach, groups A and B spent 60 minutes on the Text Analysis and 60 minutes on the Model Analysis task. We applied a cross-over design for the CSI part of the study, i.e., text analysis (60 min) followed by the model analysis (60 min) for group A and similar tasks in an inverse order for group B. For model analysis, we used a pre-defined set of EMEs to avoid dependencies between different tasks within the groups. These EMEs were provided by the experiment team, i.e., the authors, to focus on model analysis. Note that individual groups focused on a different part of the model.

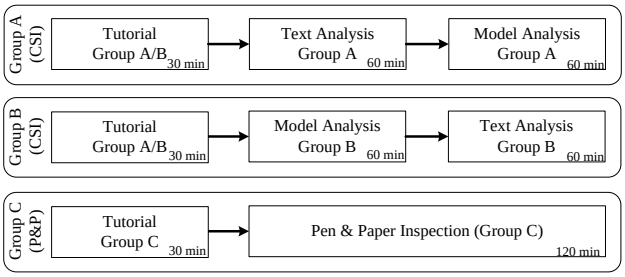

Fig. 3. Experimental Process.

Participants. A total of 75 study participants were recruited from undergraduate students attending the course on "Software Quality Assurance" at the TU Wien. Assignment to groups A, B and C relied on a random distribution of the participants. Since group C was a control group, less participants were assigned (12 inspectors) than to groups A& B (63 inspectors). Given the high number of participants the experiment was run during four different workshops each attended by maximum 20 participants. To assess the qualification of participants we used an experience questionnaire to capture their background skills related to their software engineering experiences and the number of projects they were involved in the past. The initial results showed that only 6 participants (8%) had limited experiences, while 51 (68%) had medium, and 18 (24%) had high experience in software engineering. Thus, the participants can be classified as junior professionals in industry context.

Study Material. Common material for both experiment parts (i.e., P&P inspection and CSI) consists of an experience questionnaire to capture background skills of the participants at the beginning of the study and feedback questionnaires after each step of the experiment process. The experience questionnaire includes demographics, working experiences in software development and quality assurance. The feedback questionnaire captures feedback on the method applied for improvement purposes. Questionnaires were realized using *Google.forms*. We also provided guidelines (as hardcopies) to guide the participants during their specific inspection tasks. We used an *Experiment Management System (EMS)* that holds relevant information for all groups, i.e., electronic documents (if applicable), web links to questionnaires, and links to CSI tasks (CSI inspectors only) in sequence of the task to be solved. To avoid domain-specific knowledge limitations, we applied a well-known application domain,

i.e., typical scenarios of a restaurant process. Study material was a textual reference document, i.e., a system requirements specification including 3 pages in English language, consisting of 7 scenarios and approximately 110 EMEs. During study preparation, the scenarios were split into 33 sentences and numbered as vehicle for defect reporting and referring purposes. System requirements specification was considered to be correct. We used a medium-scale EER Diagram (9 entities, 13 relationships, and 32 attributes) including 33 seeded defects. These seeded defects were introduced by the experiment team (i.e., the authors) based on defects typically occurring during the typical software engineering processes. Furthermore, the participants received selected printed material, i.e., guidelines. For participants, these guidelines were used as supporting material complementary to guidelines provided by the EMS. For the experiment management, we used this guidelines for group assignment and to track the participants. Group C (P&P inspectors) participants also received printed version of the scenario description (i.e., the requirements specification) and the EER model. Participant had to hand-over all hardcopies to the experiment management team when finishing the workshop.

Tool Configuration. For *text analysis (TA)*, each CSI inspector received sentences drawn from half of the reference document, had to identify EMEs and report them via the *CrowdFlower* application. The P&P inspectors did not apply any explicit text analysis. For *model analysis (MA)*, each CSI inspector received up to three batches of EMEs linked to three scenarios, which were different than the scenarios inspected during text analysis to avoid bias. These batches of tasks have been assigned to the CSI participants via links to *CrowdFlower* jobs and the EMS. The CSI inspectors used the *CrowdFlower* platform to report defects. In more detail, for MA, a CSI inspector received (a) a scenario from the reference document to provide a reference context for defect detection and (b) an EME that is related to this scenario. In the study setting, CSI inspectors were encouraged to report all relevant defects related to a given EME (or a synonym) they found in the MA process. Group C inspectors used a spreadsheet solution to report defects and uploaded the final defect reports to EMS.

Variables. We define independent and dependent variables: Independent variables include the seeded defects of the EER model, defect types, tool configuration, and the study treatments. Dependent variables are effort for task execution, reported and true defects, effectiveness (share of reference defects found by a participant), false positives (candidate defects that do not match to seeded defects), and efficiency (reference defects found per hour).

Data Collection and Analysis. Organized by the EMS, we collected data electronically. For P&P inspection we collected all reported defects in the spreadsheet solution via the EMS. For text and model analysis we organized all data in the *CrowdFlower* application, i.e., EMEs and reported detects. To avoid a heterogeneous data input by individuals, we introduced an “output language”, i.e., a data schema and formal language elements to make the analysis of reported EMEs and defects more efficient. Defects were reported as: (a) expected correct model conditions, e.g., a key should be

valid; (b) the deviation in the model from the expectation; and (c) a severity level on a three-level severity scale (i.e., critical, normal, low severity). Questionnaire results were captured with *Google.forms*. All data were exported from the individual application and imported in a MySQL database for further evaluation, including consistency checks and data cleanup. We used a unique identifier to link different data sources. For data analysis, we applied descriptive statistics and the Mann-Whitney Test at a significance level of 95% (two-sided) for hypothesis testing.

6 Study Results

This section summarizes defect detection effort, effectiveness, efficiency, and false positives of the CSI process and P&P.

6.1 Effort

In the context of this study, we extracted the defect detection effort based on the reported starting and end time for P&P inspection and CSI inspectors. The defect detection task of CSI inspectors focuses on model analysis (MA) with a maximum duration of 60 min (CSI-MA). We did not consider the effort for the text analysis in this paper because it is not dedicated explicitly to defect detection (but to some kind of preparation effort). The P&P inspectors spent a maximum duration of 120 min for defect detection. Table 1 presents the descriptive statistics of individual defect detection tasks. The results showed that P&P requires on average less time for defect detection but included a higher standard deviation (SD). We also observed individual outliers for CSI and P&P because (a) some participants had to leave earlier and (b) others required more time to complete ongoing tasks after time has finished.

Table 1. Duration of P&P and CSI Tasks [in min].

Group	#participants	Mean	SD	Min	Max
CSI-MA	63	53 min	11.9 min	28 min	80 min
P&P	12	107 min	27.5 min	28 min	135 min

6.2 Defect Detection Effectiveness

Effectiveness is the share of true defects and seeded defects in [%]. On average, all participants reported 16 defects (SD: 6.8) and 7 true defects (SD: 4.9) resulting in an effectiveness of 21.5% (SD: 14.82%). True defects that were identified more than once were calculated once at the first time of detection. Table 2 presents the results of reported defects, true defects, and effectiveness per study group (we did not separate CSI groups A and B in the evaluation context of this paper).

The observations showed a higher number of reported defects and true defects for the P&P group compared to the CSI group. A main reason could be that the CSI group spent maximum 60 mins for defect detection. Applying the Mann-Whitney-Test at a significance level of 95% (two-sided) showed significant differences for reported defects (p-value: 0.003(S)), reported true defects, i.e., seeded defects

(p-value: 0.015(S)), and effectiveness (p-value: 0.015(S)). Thus, this evaluation showed benefits for P&P inspectors compared to CSI participants. However, the CSI group spent at most 60 min for defect detection (i.e., for model analysis) while P&P inspectors used at most 120 min for defect detection.

Table 2. Reported Defects / True Defects / Effectiveness.

Group	#participants	Reported Defects		True Defects		Effectiveness	
		Mean	SD	Mean	SD	Mean	SD
CSI-MA	63	15	6.5	7	4.9	30%	12.8%
P&P	12	21	5.7	10	4.6	20%	14.4%

6.3 False Positives

An important question is whether or not the EMEs (i.e., pre-requisites for model analysis) can support defect detection by providing a clearer focus on individual model elements and, therefore, reduce false positives (because of that specific focus). In context of this study, we define False Positives (FP) as reported defects that cannot be mapped to a true defect. This mapping process has been executed by the experiment team (i.e., the authors).

Table 3. Absolute Number of False Positives.

Group	#participants	Mean	SD	Min	Max
CSI-MA	63	8	5.0	1	18
P&P	12	11	4.7	5	22

Note that multiple reported defect that map to one true defect were counted once. Table 3 summarizes the absolute number of FP per study group. The results showed a lower number of FP for CSI, i.e., 8 FP on average (SD: 5.0) CSI participants and 11 FP (SD: 4.7) for P&P participants. These results indicate that EMEs can support defect detection processes by driving the inspection process. However, statistical significance was not observed (p-value: 0.055(-)).

6.4 Defect Detection Efficiency

Defect Detection Efficiency refers to true defects found per resource unit spent, i.e., defects per hour. We calculated defect detection efficiency per hour, even if the defect detection effort was less than 60 min. Table 4 contains the descriptive statistics of defect detection efficiency for P&P and CSI with focus on defect detection duration (i.e., max. 120 min for P&P and max. 60 min for CSI).

Table 4. Defect Detection Efficiency with focus on real effort [Defects per hour].

Group	#participants	Mean	SD	Min	Max
CSI-MA	63	7.5	5.29	0	23
P&P	12	5.7	2.17	2	9

The initial results showed a higher defect detection efficiency for CSI participants compared to P&P inspectors. However, the standard deviations for CSI participants is higher. We also observed outliers in the CSI group, i.e., three CSI inspectors did not report any of the seeded defects and three CSI participants achieved an efficiency value of 20 defects per hour (or more). This is a calculated value as some CSI participants did not spend one hour for model analysis. However, applying the Mann-Whitney Test we did not observe any significant differences (p value: 0.379(-)). Based on the results we see a high potential for the CSI process to enable addressing large and complex models and to reduce false positives based on introduced EMEs. CSI is a promising and complementary approach for defect detection.

7 Discussion and Limitations

This section presents the discussion of the main findings related to the research issues and hypotheses. The main goal was to adapt/extend a best-practice inspection approach with crowdsourcing. Fig. 2 presented the CSI process approach with two main activities, a text analysis to capture *Expected Model Elements (EMEs)* and a model analysis task for defect detection. For analysis purposes we executed a controlled experiment to investigate the effect on defect detection performance, i.e., effectiveness, false positives, and efficiency. Defect *detection effectiveness* is the share of true defects found related to the number of seeded defects. The analysis results showed significant benefits for P&P inspectors (p-value: 0.015(S)). Thus, H1.0 that CSI and P&P groups perform similar must be rejected. Main reasons include a smaller defect detection effort for CSI inspectors (60 min for CSI inspectors and 120 min for P&P inspectors). *False positives (FP)*, i.e., reported defects that do not match to a true defect, is important to keep analysis effort low after inspection. As EMEs guide model inspection processes, the results showed benefits for CSI. CSI inspectors report on average less FPs compared to P&P inspectors (p-value: 0.055(-)). *Defect detection efficiency* refers to the number of identified true defects per resource unit spent (i.e., per hour). CSI inspectors achieved on average higher (but not significantly higher) defect detection efficiency levels compared to P&P inspectors (p-value: 0.379(-)). Following this observations, we had to reject H2.0.

We identify and discuss important potential threats to validity of our study and describe how we addressed them [13]. *Internal validity* concerns a causal relationship between the experiment treatment and the observed results, without the influence of potential confounding factors that are not controlled or measured [13]. The authors introduced 33 reference defects in the EER diagram based on typical defects collected during typical software engineering processes. Further analysis steps might raise additional defects which might have been overseen in the preliminary analysis. Domain specific issues have been addressed by selecting a well-known application domain. The experiment package was intensively reviewed by experts to avoid errors. Furthermore, we executed a set of pilot runs to ensure the feasibility of the study design. We applied a random distribution of the group assignment using a sort card algorithm.

The overall duration was limited to 120 min. Individual breaks were allowed; break periods had to be reported. *External Validity* refers to the generalization of the results to a larger population or to environments that differ from the one studied [13]. Interaction of selection and treatment. Participants were 75 undergraduate students at TU Wien. The authors are aware of limitations of student experiments [18].

8 Conclusion and Future Work

To support advancement towards the goal of effective and efficient quality assurance for large software models and associated reference documents, we described in this paper a crowdsourcing-supported inspection (CSI) process. We focused on investigating the effects of CSI model analysis for an Extended Entity Relationship (EER) model with an associated scenario-based requirements document. We reported on a controlled experiment with 75 participants to explore the CSI process. The empirical study provided foundational evidence that the core task of the CSI process, the model analysis task for defect detection guided by EMEs is both, feasible and useful. Thus, the presented process approach can gain benefits for model quality assurance for different types of models (based on the concept of the CSI process) and is promising for large-scale models (based on the crowdsourcing mechanisms, i.e., splitting up large tasks in smaller tasks and increase inspection control for introducing additional CSI workers if needed).

Future work will include (a) more detailed investigation of defect detection capabilities of CSI as preliminary results show promising but no significant results; (b) replication of the controlled experiment in different settings, e.g., large scale models, different model types, and different populations (e.g., in industry setting); (c) investigating the effects of nominal inspection teams and inspector qualification levels; and (d) possible automation that supported inspection processes with tool support to improve the overall inspection process in a distributed setting.

References

1. Parnas, D.L., Lawford, M.: The role of inspection in software quality assurance. IEEE Transactions on Software Engineering 29(8), 674-676 (2003).
2. Brambilla, M, Cabot, J., Wimmer, M.: Model-Driven Software Engineering in Practice. Morgan & Claypool Publishers (2012).
3. Fagan, M.E.: Design and code inspections to reduce errors in program development. IBM Systems Journal 15(7), 182-211 (1976).
4. Aurum A., Petersson, H., Wohlin, C.: State-of-the-art: Software Inspections after 25 Years, Software Testing, Verification and Reliability 12(3), 133-154 (2002).
5. Winkler, D., Wimmer, M., Berardinelli, L., Biffl, S.: Model Quality Assurance for Multi-Disciplinary Engineering, In: Biffl, S., Lüder, A., Gerhard, D. (eds): Multi-Disciplinary Engineering of Cyber-Physical Production Systems, (2017).

6. Travassos, G.H., Shull, F., Carver, J., Basili, V.: Reading techniques for OO design inspections. Tech. Rep., Fraunhofer Center-Maryland, U Maryland (2002).
7. LaToza, T.D., van der Hoek, A.: Crowdsourcing in Software Engineering: Models, Motivations, and Challenges, IEEE Software, 33(1), 74-80 (2016).
8. Shull, F., Rus, I., Basili, V.: How Perspective-Based Reading Can Improve Requirements Inspections. IEEE Software 3(7), 73-79 (2000).
9. Kalinowski, M., Travassos, G.H.: A computational framework for supporting software inspections. In: Proc. of ASE, Linz, Austria, pp. 46-55 (2004).
10. Lanubile, F., Mallardo, T.: Tool support for distributed inspection. In: Proc. of COMPSAC, IEEE (2002).
11. Hernandes, E.M., Belgamo, A., Fabbri, S.: Experimental studies in software inspection process - a systematic mapping. In: Proc. of ICEIS), pp. 66-76 (2013).
12. Mao, K., Capra, L., Harman, M., Jia, Y.: A survey of the use of crowdsourcing in software engineering; Journal of Systems and Software, 28p (2016).
13. Wohlin C., Runeson P., Höst M., Ohlsson, M., Regnell, B. Wessl, A.: Experimentation in software engineering. Springer (2012).
14. Winkler, D., Sabou, M., Petrovic, S., Caneiro, G., Kalinowski, M., Biffl, S.: Improving model inspection with crowdsourcing. In: Proc. Int. Wsh on Crowdsourcing in SE , ICSE, Buenos Aires, (2017) (to appear).
15. Manning, C.D, Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008).
16. Thalheim V.: Extended entity-relationship model. In Encyclopedia of Database Systems, pp.1083-1091, Springer (2009).
17. Winkler D., Sabou, M., Petrovic, S., Caneiro, G., Kalinowski, M., Biffl, S.: Investigating model quality assurance with a distributed and scalable review process. In: Proc. of CIbSE, Buenos Aires, Argentina (2017). (to appear).
18. Runeson, P.: Using students as experiment subjects–an analysis on graduate and freshmen student data. In: Proc of the 7th EASE Conference (2003).