Towards an Experiment Line on Software Inspection with Human Computation

Stefan Biffl
Institute of Information Systems Eng., Information & Software Engineering, Technische Universität Wien
A-1040 Vienna, Austria
+43 1 58801 18810
Stefan.Biffl@tuwien.ac.at

Marcos Kalinowski
Experimental and Applied Software Engineering Research Group
Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
22451-900 Rio de Janeiro, Brazil
+55 21 (21) 3527-1510
Kalinowski@inf.puc-rio.br

Dietmar Winkler
Christian Doppler Laboratory for Security and Quality Improvement in the Production System Lifecycle, Technische Universität Wien
A-1040 Vienna, Austria
+43 1 58801 18810
Dietmar.Winkler@tuwien.ac.at

ABSTRACT

Software Inspection is an important approach to find defects in Software Engineering (SE) artifacts. While there has been extensive research on traditional software inspection with pen-and-paper materials, modern SE poses new environments, methods, and tools for the cooperation of software engineers. Technologies, such as Human Computation (HC), provide tool support for distributed and tool-mediated work processes. However, there is little empirical experience on how to leverage HC for software inspection. In this vision paper, we present the context for a research program on this topic and introduce the preliminary concept of a theory-based *experiment line* to facilitate designing experiment families that fit together to answer larger questions than individual experiments. We present an example feature model for an experiment line for *Software Inspection with Human Computation* and discuss its expected benefits for the research program, including the coordination of research, design and material reuse, and aggregation facilities.

ACM CCS keywords: Software verification, Software defect analysis, Empirical studies; Collaborative and social computing, Experimentation.

Keywords

Software Inspection, Human Computation, Empirical Software Engineering, Experiment Lines.

1. INTRODUCTION

Software Inspection is an important Software Engineering (SE) approach to find defects in SE artifacts [6], such as software requirements or models. Inspection effectiveness and efficiency heavily depend on human expertise, such as semantic understanding or specific real-world knowledge. For instance, finding defects in models requires human expertise for checking model semantics that require world/domain knowledge that is hard to model explicitly in the required detail and level of correctness [5].

Some major challenges of software inspections are (a) limited attention span/effort for intensive inspection in a typical inspection session and (b) limited method and tool support for software inspection coordination, in particular, for distributed inspection support, for scaling up inspections to large artifacts, and for ensuring the quality of inspection contributions [15]. There has been extensive research on traditional software inspection [1] with pen-and-paper materials. Unfortunately, these approaches do not seem to address these challenges properly, in particular, in the context of large artifacts.

Modern SE poses new environments, methods, and tools for SE and the cooperation of software engineers. There are new technologies, such as *Human Computation* (HC), that can provide method and tool support for distributed and tool-mediated work processes. Indeed, our initial investigations indicated the feasibility of using such a technology towards a distributed and scalable inspection processes [16][17]. However, there is still little empirical experience reported on how to leverage HC for software inspection and there is a need for further investigations, potentially conducting several experiments in the context of an experiment family to build relevant knowledge on the topic [3].

In this vision paper, based on our initial investigation experiences in Austria and Brazil, we present the context for a research program on the topic of *Software Inspection with Human Computation*, i.e., to investigate how traditional pen-and-paper inspection (process, method, and tool support) can be adapted to modern SE practices and technologies, such as *Human Computation*. We introduce the concept of a theory-based *experiment line* to allow discussing and designing experiment families, based on potential configurations of cause-effect theory constructs for experiment treatments, that can fit together to answer research questions beyond the possibilities of individual experiments. We focus on two main research issues:

RI-1. How could an experiment line for Software Inspection with Human Computation address variability for planning experiments? To support the discussion, we envision and explain an example concept of such an experiment line.

RI-2. What are the expected benefits of an experiment line for Software Inspection with Human Computation? The discussion emphasizes research coordination (e.g., identification and distribution of relevant investigation efforts), design and material reuse, and aggregation facilities (e.g., providing a further understanding on contributions to the theory and threats to validity).

ICSE-WSESE 2018, May, 2018, Gothenburg, Sweden.

DOI: http://dx.doi.org/10.1145/12345.67890

2. BACKGROUND AND RELATED WORK

This section summarizes the background on *Software Inspection*, *Human Computation*, and recent work on *Software Inspection with Human Computation*.

2.1 Software Inspection

Software Inspection improves software product quality by analyzing software artifacts, detecting defects for removal before delivering these artifacts to later software life cycle activities. The traditional inspection process by Fagan [6] involves a moderator planning the inspection, inspectors reviewing the artifact, a team meeting to discuss and register defects, passing the defects to the author for rework, and a final follow-up evaluation by the moderator on artifact quality assessment and the need of a new inspection.

There has been extensive research on traditional software inspection with pen-and-paper materials [4], investigating several software inspection aspects by varying levels of specific experiment factors. Web-based tools [8] have been proposed, but typically provide limited support for reading techniques and fall short in coordination support when compared to modern SE capabilities. Theoretical contributions to the area of inspections (e.g. [7][9][10][14]) reveal theory constructs that help defining typical experiment factor variation options and typical outcomes. Such variation options include artifacts (e.g., requirements, design, or code), inspector characteristics (e.g., qualification, experience), process (e.g., inspection with or without a group meeting), and method (e.g., reading techniques). Typical observed outcomes are effectiveness and efficiency of the inspection on individual and team level.

2.2 Human Computation

Human Computation provides method and tool support to coordinate a potentially large group of workers for addressing small tasks and for aggregating their results to address larger SE problems [11]. Mao *et al*. report on a survey on the use of crowdsourcing in SE [12]. While crowdsourcing mechanisms, e.g., splitting up large-scale tasks into smaller pieces of work, are frequently used in open source environments and quality assurance has been tackled in context of software testing, software inspection had not been covered by crowdsourcing at the time of their survey [12]. However, splitting up large-scale documents into smaller pieces, inspecting these small tasks within a group of experts, and aggregating identified candidate defects might support scalability of inspection artifacts, improve coordination between different experts (i.e., inspectors), and increase effectiveness and efficiency of defect detection. Therefore, we recently took a first attempt in this direction [15].

2.3 Software Inspection with Human Computation

Increasing needs for scalable and tool-supported software inspection and the availability of *Human Computation* approaches led to the idea of *Crowdsourced Software Inspection (CSI)* [15]. Recently, we conducted several experiments on *Software Inspection with Human Computation* to gain initial experience on the benefits and limitations of human computation in a software inspection process. The CSI process consists of five main phases [15]: (a) Planning of the inspection process; (b) Text Analysis and Aggregation (Figure 1, step 1), executed by a crowd of experts to derive relevant building blocks of the model (i.e., *Expected Model Elements - EMEs*) of a SE diagram; (c) Model Analysis to identify candidate defects (Figure 1, step 2); (d) Analysis and Aggregation of candidate defects (Figure 1, step 3); and (e) rework of inspection artifacts. Figure 1 presents the main phases related to defect detection with the CSI process, i.e., text analysis, model analysis, and candidate defect analysis and aggregation: Step 1: the CSI management prepares the defect detection task by providing EMEs and the model scope; Step 2: a group of inspectors (i.e., the CSI workers) identifies model defects based on EME-based verification micro tasks and report defects if needed. Step 3: the CSI management analyses individual defect reports and aggregates candidate defects towards an aggregated defect list. Note that we use *Crowdflower (CF)*[1] for defect detection and defect analysis and aggregation.

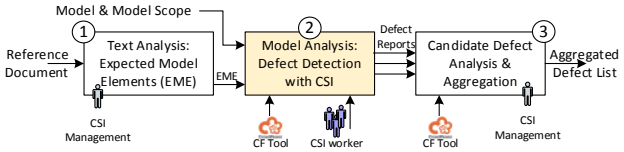

Figure 1. Defect Detection with Crowdsourced Software Inspection.

Initial results [15][16][17] showed that the CSI process is a promising approach for supporting model inspection, e.g., by addressing different aspects of the inspection artifact and focusing on different defect types. Although initial results are promising, there is a need for further investigations on variations of the CSI process. In context of a *research program*, such variations can include (a) different artifacts (e.g., model types and sizes), (b) CSI process variants and extensions (e.g., defect validation tasks), and (c) method/tooling to better support inspectors in defect detection in distributed environments and to address scalability issues.

3. RESEARCH ISSUES

The main goal of this vision paper is to present the context for a research program on the topic of *Software Inspection with Human Computation* and to envision the concept of an *experiment line* to support the design of experiment families that could help to coordinate a research program. We discuss two research issues:

RI1. How could an experiment line for Software Inspection with Human Computation address variability for planning experiments?

In a broader software engineering context, *Software Product Lines* (SPL) provide means to efficiently design, produce, and maintain multiple similar software variants, exploiting their common properties and managing their variabilities [13]. We identified similar challenges for families of experiments in a research program, as their designs also involve common properties and variabilities.

Therefore, we suggest that the variabilities of an experiment line could be represented using feature models, which have been widely used in the SPL context [2]. In the context of an experiment line, features are theory constructs considered in the experiment design to define the scope or to represent factors (i.e., independent variables), factor levels, or response variables (i.e., dependent variables). The variations between experiments in the context of a research program consists of investigating different factors (or factor levels) and measuring different response variables on a similar topic. We draft and explain an example of such experiment line feature model for software inspection with human computation by revisiting typical inspection theory constructs used in inspection experiments.

RI2. What are the expected benefits of an experiment line for Software Inspection with Human Computation?

[1] Crowdflower: www.crowdflower.com

We discuss three relevant expected benefits of using an experiment line to structure such research program. First, *research coordination* within and beyond a research group. We believe that focusing on the theory and analyzing potential investigation variabilities can support the identification and distribution of relevant investigation efforts. Second, *reuse* of the experiment design and material (e.g., artifacts) among experiment variants. Third, better understanding the variabilities and their relation to theory can provide *aggregation facilities*. Indeed, if variation is planned, the data coming from the experiments in the experiment line may be aggregated more easily.

4. A VISION OF AN EXPERIMENT LINE

In this section, we provide an example of how an experiment line could look like for *Software Inspection with Human Computation*.

We represent the variabilities of the experiment line using a feature model, which has been widely used to represent common properties and variabilities in the SPL area [2].

To build the feature model, we revisited typical software inspection theory constructs (e.g., the ones presented in [7][8][10][14]) that have been used to define the scope, factors, factor levels or response variables in inspection experiments. Thereafter, we complemented the model discussing new constructs that could be of interest for the particular investigation context of *Software Inspection with Human Computation.* Figure 2 shows an excerpt of the resulting feature diagram. Due to space constraints, we did not represent the theory constructs used for response variables (e.g., defect detection effectiveness, efficiency, and coordination effort).

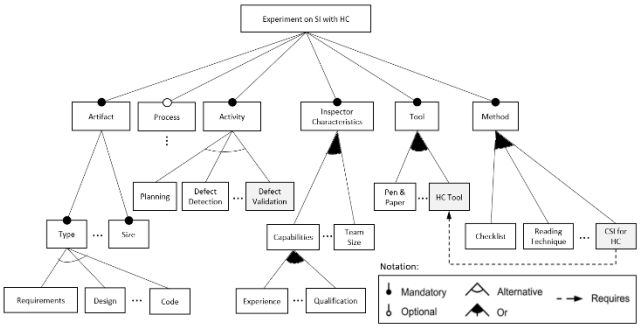

Figure 2. Feature Model for Experiments on Software Inspection with Human Computation.

While Figure 2 is not supposed to be complete, it shows that an inspection experiment of the represented experiment line involves (mandatory): an inspection target *artifact*, a specific inspection *activity* to be conducted by inspectors with certain *inspector characteristics* using *tools* and *methods*. Figure 2 also shows that, while this type of experiment tends to focused on specific activities, there is an option on investigating how activities are performed in the context of different inspection *processes* (e.g., inspection with or without a group meeting, distributed).

In Figure 2, it is easy to see that, inspection experiments in the designed family have a scope defined with fixed ('*alternative'* options) artifact types and activities. Inspector characteristics, tools and methods, on the other hand, can be chosen as experiment factors, given that they can have more than one options that can be applied in the context of an experiment ('*or'* options). These options should be considered as factor levels.

The feature model also allows adding constraints. An example of such constraint is that experiments using the method *CSI for HC* (the human computation approach) require human computation tool support. This context provides a new process and new options for method and tool support to investigate different activities. An activity that deserves particular attention in this context is defect validation, given that several inspectors will report defects using HC tool support, leading to the need of assuring the quality of these contributions. This context also affects the response variables (not depicted) of interest. For instance, a new variable that becomes relevant in this context is the overall inspection coordination effort. Indeed, it is easy to see that a complete investigation of this context would involve several experiments, ideally conducted in the context of a research program involving several research groups.

After the feature model has been discussed and built, valid experiment configurations can be identified and used as input for collaborations options within and beyond a research group. Each experiment configuration involves a combination of theory construct selections that can be stated as theoretical propositions to be further investigated. Hence, the experiment configurations are directly related to the underlying theory (where the constructs came from). After identifying experiment configurations of interest, they should be analyzed based on the practical relevance, cost to conduct the experiment and data collection, and the risk of research (including threats to validity).

An example experiment configuration could involve comparing "*design* inspection *defect detection* by *novice inspectors*, using *pen and paper* support to conduct *scenario-based reading*" with "*design* inspection *defect detection* by *novice inspectors*, using a *HC tool* (e.g., *Crowdflower*) support to follow the *CSI for HC method*" in terms of effectiveness and efficiency.

Besides the variation possibilities, the feature model also allows identifying common properties between experiments, which can be exploited to encourage experiment design and material reuse and facilitate research outcome aggregation.

5. BENEFITS OF AN EXPERIMENT LINE

The main expected benefits of using the experiment line concept are related to *research program coordination, design and material reuse,* and providing *aggregation facilities.* A brief discussion on the rationale behind each of these expected benefits follows.

Research program coordination. Collaborating researchers can jointly build the experiment line by discussing common properties and variabilities of experiments based on relevant theory constructs and existing experiments and materials. Thereafter they can use the experiment line to compare different configuration options and to evaluate which configurations could be particularly relevant. Once the relevant experiment configurations have been identified, the researchers can divide the overall effort strategically among themselves collaborating on different experiments, considering the costs and risks to conduct each experiment. In this way, the researchers can jointly operate families of experiments that can fit together to help answering research questions beyond the possibilities of individual experiments.

Design and material reuse. One of the main benefits of software product lines is related to reuse [13]. Similarly, we believe that cooperating researchers (or research groups) would benefit from a common platform to share research designs and materials. Experiments involving similar constructs are candidates for sharing design elements and materials. For instance, experiments on design inspections could reuse models that are being inspected, experiments considering inspector characteristics could share characterization questionnaires, and experiments measuring the same response variables may reuse definitions and statistical analysis procedures. More systematic reuse approaches for experiment replica-

tions are challenging and relevant especially in the context of experiments conducted with human subjects, in which undesired variations can lead to serious threats to validity.

Aggregation facilities. If variations between experiments are planned, data coming from the different experiments may be aggregated more easily. Better understanding the variabilities between different experiments facilitates proper aggregation, mitigating relevant threats to validity. For instance, if experiments share similar constructs (e.g., exact replications) they may be aggregated to improve internal and conclusion validity. Experiments with planned construct variations, on the other hand, might be helpful to generalize results and improve construct and external validity. If experiments families are designed in an unplanned way and constructs vary widely, the overall common view may be limited and aggregations might lead to conclusions with serious threats to validity.

6. CONCLUSION AND FUTURE WORK

In this vision paper, we presented the context for a research program on the topic of *Software Inspection with Human Computation*. Based on this context, we introduced the concept of a theory-based *experiment line* to allow designing experiment families that can fit together with the purpose of answering research questions beyond the possibilities of individual experiments.

Based on this context, we discussed the following two research issues: (a) *RI-1. How could an experiment line for Software Inspection with Human Computation address variability for planning experiments?* and (b) *RI-2. What are the expected benefits of an experiment line for Software Inspection with Human Computation?*

To address the research issues, we revisited traditional theory models on software inspection to derive typical theory constructs that warrant investigation and complemented those constructs in the light of software inspection with human computation. Based on these constructs we built a variability model expressed as a feature model and discussed potential benefits of working with families of experiments as part of an experiment line.

Main expected benefits comprise (a) supporting strategically planning a family of studies to facilitate the cooperation of research groups, (b) supporting systematic experiment design and material reuse, and (c) helping to provide a further understanding to allow the proper aggregation of individual study results, in particular of studies involving human subjects, where undesired variations can lead to serious threats to validity. Given these potential benefits for experimental collaborations within and beyond research groups, we believe that the concept of experiment lines should be further investigated and evolved.

7. ACKNOWLEDGMENTS

The financial support by the Austrian Federal Ministry for Digital, Business and Enterprise and the National Foundation for Research, Technology and Development is gratefully acknowledged.

8. REFERENCES

[1] Aurum A., Petersson H., Wohlin C.: “State-of-the-Art: Software Inspection after 25 years”, Software Testing Verification and Reliability, 12(3), pp. 133- 154, 2002.

[2] Acher, M., Lopez-Herrejon, R.E. and Rabiser, R.: “Teaching software product lines: A snapshot of current practices and challenges”, ACM Trans. on Computing Education (TOCE), 18 (1), 31 pages, 2017.

[3] Basili, V.R., Shull, F. and Lanubile, F.: “Building knowledge through families of experiments”, IEEE Trans. on Software Engineering, 25(4), pp.456-473, 1999.

[4] Biffl S., Freimut B., Laitenberger O.: “Investigating the cost-effectiveness of reinspection in software development”, In: Proc. of ICSE, pp.155-164, 2001.

[5] Brambilla M., Cabot J., Wimmer M.: “Model-Driven Software Engineering in Practice”, Morgan & Claypool publishers, 2017.

[6] Fagan, M.E., “Design and Code Inspection to Reduce Errors in Program Development”, IBM Systems Journal, vol. 15, no. 3, pp. 182-211, 1976.

[7] Jeffery R: “Empirical Methods and Theory Research in Software Engineering: State of the Art and State of Practice”, Keynote, Asia-Pacific Software Engineering Conference, 2017.

[8] Kalinowski, M., Travassos, G.H.: “A computational framework for supporting software inspections”, In: Int. Conf. on Automated Software Engineering (ASE), pp. 46-55, 2004.

[9] Laitenberger, O., DeBaud, J.-M.: “An encompassing life cycle centric survey of software inspection”, *Journal of Systems and Software*, vol. 50 (1), pp. 5–31, 2000.

[10] Land L.P.W., Wong B., Jeffery R.: “An extension of the behavioral theory of group performance in software development technical reviews”, In: Proc. of APSEC, pp.520-530, 2003.

[11] LaToza T.D., van der Hoek A.: “Crowdsourcing in Software Engineering: Models, Motivations, and Challenges”, IEEE Software, vol. 33 (1), pp. 74-80, 2016.

[12] Mao K., Capra L., Harman M., Jia Y.: “A survey of the use of crowdsourcing in software engineering”, In: Journal of Systems and Software, 28p, 2016.

[13] Pohl, K., Böckle, G., van der Linden, F.: “Software Product Line Engineering: Foundations, Principles, and Techniques”, Springer, 2005.

[14] Sauer C., Jeffery D.R., Land L., Yetton P.: “The effectiveness of software development technical reviews: a behaviourally motivated program of research”, IEEE Transactions on Software Engineering (TSE), 26(1), pp.1-14, 2000.

[15] Winkler D., Sabou M., Petrovic S., Carneiro G., Kalinowski M., Biffl S.: “Improving Model Inspection with Crowdsourcing”, In: International Workshop on Crowdsourcing in Software Engineering (CSI-SE), ICSE, pp.30-34, 2017.

[16] Winkler D., Sabou M., Petrovic S., Carneiro G., Kalinowski M., Biffl S.: “Investigating Model Quality Assurance with a Distributed and Scalable Review Process', In: Ibero-American Conf. on Software Engineering (CIBSE), pp.141-154, 2017.

[17] Winkler D., Sabou M., Petrovic S., Carneiro G., Kalinowski M., Biffl S.: “'Improving Model Inspection Processes with Crowdsourcing: Findings from a Controlled Experiment”, In: European Conference on Systems, Software & Service Process Improvement and Innovation (EuroSPI), CCIS, volume 748, pp.125-137, 2017.