Summer 6-15-2016

CAN LAYMEN OUTPERFORM EXPERTS? THE EFFECTS OF USER EXPERTISE AND TASK DESIGN IN CROWDSOURCED SOFTWARE TESTING

Niklas Leicht
University of St.Gallen, niklas.leicht@unisg.ch

Marcel Rhyn
University of St. Gallen, marcel.rhyn@unisg.ch

Georg Hansbauer
Testbirds GmbH, g.hansbauer@testbirds.de

CAN LAYMEN OUTPERFORM EXPERTS? THE EFFECTS OF USER EXPERTISE AND TASK DESIGN IN CROWDSOURCED SOFTWARE TESTING

Research in Progress

Niklas Leicht, University of St. Gallen, St. Gallen, Switzerland, niklas.leicht@unisg.ch

Marcel Rhyn, University of St. Gallen, St. Gallen, Switzerland, marcel.rhyn@unisg.ch

Georg Hansbauer, Testbirds GmbH, Munich, Germany, g.hansbauer@testbirds.de

Abstract

In recent years, crowdsourcing has increasingly gained attention as a powerful sourcing mechanism for problem-solving in organizations. Depending on the type of activity addressed by crowdsourcing, the complexity of the tasks and the role of the crowdworkers may differ substantially. It is crucial that the tasks are designed and allocated according to the capabilities of the targeted crowds. In this paper, we outline our research in progress which is concerned with the effects of task complexity and user expertise on performance in crowdsourced software testing. We conduct an experiment and gather empirical data from expert and novice crowds that perform different software testing tasks of varying degrees of complexity. Our expected contribution is twofold. For crowdsourcing in general, we aim at providing valuable insights for the process of framing and allocating tasks to crowds in ways that increase the crowdworkers' performance. Secondly, we intend to improve the configuration of crowdsourced software testing initiatives. More precisely, the results are expected to show practitioners what types of testing tasks should be assigned to which group of dedicated crowdworkers. In this vein, we deliver valuable decision support for both crowdsourcers and intermediaries to enhance the performance of their crowdsourcing initiatives.

Keywords: Crowdsourcing, Complexity, Expertise, Performance, Software Testing

1 Introduction

With the advent of the digitization and the rise of web 2.0 technologies, crowdsourcing has increasingly gained attention as a powerful sourcing mechanism for problem-solving in organizations (Jeppesen and Lakhani, 2010, Afuah and Tucci, 2012). Crowdsourcing describes "the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call" (Howe, 2006). It represents a fundamental shift in the way in which businesses may acquire resources and extends the frontiers of available skill or knowledge beyond the boundaries of their organization (Geiger et al., 2012). With crowdsourcing, it is possible to mobilize the expertise and creativity distributed among a large panel of people in order to achieve a certain set of tasks (Schenk and Guittard, 2011). This allows businesses to efficiently cope with growing complexity, shorter innovation cycles, and tightening resource constraints as encountered in highly dynamic and competitive environments (Zogaj et al., 2014).

As a new form of human computer interaction, crowdsourcing has already been adopted for a vast array of value creation activities ranging from early stages of research and development to later stages of testing and after-sales services (e.g., Vukovic, 2009, Leicht et al., 2015, Schlagwein and Bjørn-Andersen, 2014, Blohm et al., 2011). However, depending on the type of activity addressed by crowdsourcing, the role of the contributors and the characteristics of their contributions may differ substantially (Doan et al., 2011, Geiger et al., 2012). Some activities require the processing of simple

tasks of high granularity whose effort intensity and structural complexity tend to be low. In this context, the value of crowdsourcing stems from the efficient and cost-effective completion of the tasks on a large scale. Other activities revolve around complex and intellectually more demanding tasks that specifically ask for expertise in a certain domain (Rouse, 2010, Schenk and Guittard, 2011). Crowdsourcing then provides the opportunity to leverage the "wisdom of the crowd" and grants access to a pool of diverse resources that entail the required knowledge (Zwass, 2010).

Consequently, these distinct types of tasks place different demands on the crowd. There are crowds that can consist of potentially any individual, crowds that need to resemble a particular profile or crowds that require experts with unique abilities, specializations, or skills (Zwass, 2010, Geiger et al., 2012). Thus, it is crucial for crowdsourcing initiatives that the tasks are assigned to suitable crowdworkers. The performance in crowdsourcing is believed to be highly dependent upon both the complexity of the tasks and the crowdworkers' degree of expertise with regard to the problem that needs to be solved (Zogaj et al., 2014, Rouse, 2010). However, the implications of different task and crowdworker configurations for the performance in crowdsourcing remain mostly unclear. More specifically, there is still a lack of empirical data that show how tasks of different degrees of complexity should be designed and how they should be allocated to appropriate crowdworkers in order to leverage performance in crowdsourcing (Kittur et al., 2013, Mäntylä and Itkonen, 2013, Dolstra et al., 2013). Hence, in our research, we address the following question: How do task complexity and crowdworker expertise jointly affect performance in crowdsourcing?

In order to close this research gap, we conduct an experiment and gather empirical data from expert and novice crowdworkers that perform functional and usability-based software testing tasks of varying degrees of complexity. The underlying theory on task design and task complexity is grounded on the widely applied works of Hackman (1969), Wood (1986), and Chomsky (1957). We expect to contribute to existing literature in two ways: For crowdsourcing in general, our results may provide valuable insights for the process of designing and allocating tasks to crowdworkers in ways that increase the crowdworkers' performance. For software testing, which represents a major application field of crowdsourcing in practice (Zogaj et al., 2014, Dolstra et al., 2013, Tung and Tseng, 2013, Hoßfeld et al., 2014), we aim to support the decision process for configuring the crowdtesting initiatives. The results are expected to show what types of testing tasks should be assigned to which group of testers. These findings may help to decide whether a certain form of testing can be crowdsourced or whether it should rather be tested with alternative testing approaches.

In this paper, we outline our research in progress and present first preliminary results that are to be expanded in the future. The remainder of the paper is structured as follows: First, section 2 outlines the conceptual background of our research and introduces crowdsourced software testing as an application of crowdsourcing in practice. Secondly, section 3 and 4 are dedicated to the theoretical background and the development of the research hypotheses. In this part, existing literature on task complexity, user expertise, and human performance in a task setting will be discussed. Thirdly, section 5 describes the research methodology and explains the experimental setting with all related variables and measures in detail. Finally, in section 6 and 7, we provide preliminary results of the experiment, discuss expected contributions from our research and outline the next steps to reach these objectives.

2 Conceptual Background: Crowdtesting

Crowdtesting, or crowdsourced software testing, is a specific application of crowdsourcing in the domain of software development (Zogaj et al., 2014). It refers to the outsourcing of software testing activities to the crowd. Following Myers et al. (2011), we define software testing as "the process of executing a program with the intent of finding errors" (p. 6). These errors (also malfunctions or bugs) are deviations from the expected behavior (Beizer, 2003). In this vein, software testing can be considered a form of quality assurance that encompasses a wide spectrum of different activities, such as functional software testing or usability software testing (Bertolino, 2007).

As with other crowdsourcing applications, companies can either directly interact with the crowd or they can use intermediaries who provide this service for a fee (Chanal and Caron-Fasan, 2010). These intermediaries act as brokers who connect the organizations that apply crowdsourcing with potential crowdworkers. They play a key role in designing the tasks, providing the necessary technical infrastructure for testing, and managing the crowd. However, depending on the type of testing (e.g., functional testing, usability testing), these tasks as well as the targeted crowds can be very diverse (Stol and Fitzgerald, 2014).

Crowdtesting represents an excellent example for the study of task design, task allocation, and task performance – not only for the domain of software testing but also for crowdsourcing in general. Crowdsourced software testing encompasses a broad variety of potential testing tasks with different purposes (Zogaj et al., 2014). These tasks may vary greatly in terms of their complexity and require the crowds to have specific expertise or characteristics in order to achieve the desired results. Thus, software testing may act as a microcosm for crowdsourcing insofar that it requires crowdsourcing to adapt to different degrees of task complexity and different forms of crowdworker expertise. With crowdtesting, it is possible to design comparable types of tasks while still being able to consider the effects of complexity and expertise on performance. In this vein, software testing allows performance to be measured and analyzed by objective software testing quality metrics. Hence, there is grounded truth regarding the performance in crowdsourced software testing (Bonabeau, 2009).

3 Theoretical Background: Task Complexity and Expertise

Crowdsourcing that revolves around the collective undertaking of value creation activities is typically based on a task setting – especially in software development and software testing (Stol and Fitzgerald, 2014, Rouse, 2010, Geiger et al., 2012). As described previously, however, these tasks may vary greatly in terms of their complexity and expertise requirements. The following sections aim at providing a more in-depth theoretical background on task complexity and user expertise in crowdsourcing in order to derive research hypotheses and outline the framework for the experiment.

3.1 Tasks and Complexity in Crowdsourcing

Tasks can generally be defined as patterns of stimuli impinging on individuals and describe "behavioural responses a person should emit in order to achieve some specific level of performance" (Wood, 1986, p. 62, Hackman, 1969). According to Chomsky's (1957) work in linguistics, that has been widely applied in performance research (e.g., Chi et al., 1981a, Chi et al., 1981b, Ericsson and Charness, 1994, Schenk et al., 1998), there are two layers of representation in a task: the surface structure and the deep structure. The surface structure refers to the superficial characteristics of a task. These are immediately available through the stimulus material and represent the task's objective characteristics. The deep structure, on the other hand, refers to the underlying principles and constructs of a task. This layer is "not directly observable in the stimulus material but is inferable to those with sufficient knowledge of the task" (Haerem and Rau, 2007). Hence, it is argued that the task performers' subjective information-processing of objective task inputs (e.g., the way the tasks are designed and described) may lead to a different perception and execution of the same task (Byström and Järvelin, 1995, Campbell, 1988). This is especially relevant to crowdsourcing, where the role of the contributors and the characteristics of their contributions may be very diverse (Doan et al., 2011, Fähling et al., 2011, Geiger et al., 2012). In some cases, crowdsourcing revolves around the processing of simple tasks whose effort intensity tend to be low. In other cases, the tasks are intellectually more demanding and ask for expertise in a certain domain (Blohm et al., 2016, Rouse, 2010, Schenk and Guittard, 2011). As mentioned previously, crowdsourced software testing represents an excellent example for this (Zogaj et al., 2014)

In this context, task complexity describes the demands placed on the knowledge, skills, and resources of the task performers and thus constitutes a major determinant for human performance in a task setting (Wood, 1986, Maynard and Hakel, 1997). Task complexity can be defined either in terms of purely objective task qualities or as a subjective person-task interaction (Campbell, 1988). In this vein, the notion of "critical complexity" describes the complexity embodied in the resolution path that minimizes the individual's amount of information processing to complete a task (Haerem and Rau, 2007, Sonnentag, 1998, Frese and Zapf, 1994). It can be used to distinguish between different types of tasks that vary in their degree of complexity.

Based on the work of Haerem and Rau (2007), we define simple tasks as tasks whose critical complexity resides in the task's surface structure. In order to solve these tasks in the most efficient manner, it is sufficient to focus on the inputs and outputs of the tasks (i.e., the objective task characteristics). As their effort intensity and the individual impact of a contribution tend to be low, related crowdsourcing systems usually revolve around the processing of these small, decomposed tasks in large quantities (Zwass, 2010). On the other hand, we define complex tasks as tasks whose critical complexity is embedded in their deep structure. In order to complete these tasks, it is necessary to focus on the task process and consider the underlying principles of the problem. As these crowdsourcing tasks revolve around sophisticated problem-solving skills, the intellective demands, the structural complexity, and the effort intensity tend to be high (Zwass, 2010). A detailed discussion about how we specified these types of tasks in our crowdsourced software testing experiment can be found in section 5.2 below.

3.2 Expertise and Human Performance in a Task Setting

Depending on the degree of complexity, tasks in crowdsourcing place different demands on the characteristics or, more specifically, the expertise of the crowdworkers. In this context, one of the central mechanisms that affects human performance is expertise. Expertise is generally defined as the result of acquiring vast amounts of knowledge and procedural skill in a particular domain with the ability to perform its pattern-based retrieval in a task setting (Chi et al., 1982). Differences in the degree of expertise have multiple implications for human performance (Chi, 2006).

On the one hand, experts and novices differ in the way their body of knowledge is organized and how it is recalled when executing a task (Chi et al., 1981a). For example, Chi et al. (1981a) argue that – even though both skill groups use the same descriptions and features of a task – the cues themselves and their interactions engage greater tacit knowledge for the experts than the novices. Moreover, the problem schemata of experts contain a great deal of procedural knowledge with explicit conditions for applicability. Novices, on the other hand, may have declarative knowledge about the features of a task but they lack abstracted solution methods (Chi et al., 1981a). McKeithen, Reitman et al. (1981) found that these observations also apply to experts and novices in computer programming and software development.

On the other hand, Ericsson and Charness (1994) stress that experts frequently do not outperform other people in many relevant tasks in their domains of expertise. For example, several studies indicate that novices outperform experts on tasks that require memory for the surface structure of task features (Wiley, 1998). While experts excel at understanding and remembering the deep structure of a problem, they are prone to overlook details and fail to recall surface features (Chi, 2006). In this context, Adelson (1984) found that novices perform better than experts in answering concrete questions about computer programs, whereas experts outperformed novices on abstracts questions. This implies that experts may have learned that paying attention to abstract elements of a task is more important than paying attention to low-level details (Adelson, 1984) and that novices habitually pay more attention to the surface structure of a problem (Haerem and Rau, 2007). Moreover, expertise may lead to functional fixedness and biased mental sets. A number of studies indicate novices outperform experts when a task runs counter to highly proceduralized behaviors (Wiley, 1998). It is suggested that expertise may inhibit creative problem solving and that novices are more likely to come up with creative solutions for certain problems.

4 Development of Research Hypotheses

Based on this theoretical background, it is possible to derive a number of research hypotheses regarding the performance of experts and novices on different types of crowdsourced software testing tasks. As discussed previously, it is to be assumed that experts acquired vast amounts of knowledge and procedural skill in software testing. Hence, they are expected to have in-depth understanding of the underlying principles in software development and software testing. This should especially affect their performance on complex task addressed by functional software testing. Thus, for our experiment presented in this paper, we hypothesize as follows:

H_{1a} Experts outperform novices on both complex and simple functional testing tasks.

H_{1b} Differences between the performance of experts and novices on functional testing tasks are higher for complex tasks than for simple tasks.

On the other hand, existing literature shows that novices perform better than experts in answering concrete questions about software and that they habitually pay more attention to the surface structure of a problem. This may lead to the expectation that they are able outperform experts on tasks that run counter to highly proceduralized processes. Hence, for the full evaluation of our experiment in the future, we hypothesize as follows:

H_{2a} Novices outperform experts on usability testing tasks.

H_{2b} Differences between the performance of experts and novices on usability testing tasks are higher for simple tasks than for complex tasks.

5 Methodology

Given the lack of preliminary data, the research methodology of experimentation is particularly useful for addressing the research question of this paper (Montgomery, 1984, Box et al., 2005). By the means of an experiment, it's possible to gather empirical insights on how expert and novice crowdworkers perform on simple and complex tasks in crowdtesting. The following sections outline the application of this methodology.

5.1 Experimental Design

We employed a 2x2 between-subjects factorial design in which we vary crowdworker expertise and task complexity to test our hypotheses H_{1a} and H_{1b} (see Table 1). In order to ensure high external validity and examine the effects in a real world setting, the experiment has been conducted as a field experiment in cooperation with a German-based crowdtesting intermediary. The company ranks among Europe's leading providers of crowdtesting services with more than 100'000 crowdworkers worldwide. The stimulus object of the experiment consisted of a website for which the participants had to conduct a software test. The experiment was conducted in August 2015.

	User Expertise	
Task Complexity	*Novices*	*Experts*
Simple Tasks (Surface Structure)	32 Crowdworkers	21 Crowdworkers
Complex Tasks (Deep Structure)	32 Crowdworkers	20 Crowdworkers

Table 1. 2x2 between-subjects factorial design of the experiment

5.2 Tasks

In order to manipulate the complexity of the crowdsourced software tests, we designed two sets of functional testing tasks. Both sets contained the same basic type of use cases that guided the partici-

pants through the software application and its features. However, one set of use cases had been framed to represent simple tasks whose critical complexity was embedded in their surface structure while the other set of use cases had been framed to represent complex tasks whose critical complexity was embedded in their deep structure. More precisely, simple tasks were concerned with the superficial characteristics of the software. In order to complete these tasks, the testers typically needed to focus on the (user) interface, as surface structure tasks address bugs that can be identified by purely relying on visible inputs and outputs that are immediately available through the software (e.g., typographical errors, broken links or buttons, incorrect alignment of tables or images, error messages). Complex tasks, on the other hand, were concerned with the underlying structures and processes of software. The testers typically needed to think about how an application should behave when certain steps or operations are carried out. The bugs addressed by complex tasks were issues that produced inconsistent or unintended results (e.g., malfunctions in more complex parts of the software like a messaging system). These tasks generally required the participants to experiment with specific functions, use different combinations and settings of certain tools or test the software and its underlying processes in a more rigorous way. Additionally, we designed usability testing tasks for our second set of hypotheses (i.e., H_{2a} and H_{2b}). Participants were asked open questions that allowed them to freely provide feedback about the website. However, as the evaluation as well as the coding of the usability feedback have yet to be completed, we did not include this part in our research in progress. It will be part of our full research.

As a manipulation check, we verified that these tasks did indeed differ in terms of their complexity by using a card-sorting approach (Upchurch et al., 2001, Fincher and Tenenberg, 2005, Faiks and Hyland, 2000) with two independent professionals from the intermediary used for our experiment. For this card-sorting approach, the professionals received a randomized list with all tasks and were asked to put the separated tasks faced down on their table. One by one, they classified the tasks as either simple or complex testing tasks. The results of this procedure with the two independent professionals validated that our tasks did indeed differ in terms of their complexity.

5.3 Participants

In order to create two groups with crowdworkers of low and high expertise, we used two proxies for expertise: professional experience in software testing (cf. Faraj and Sproull, 2000) and the number of completed (crowdsourced) software tests on the platform (cf. Bandura, 1977, Quińones et al., 1995, Maynard and Hakel, 1997). Crowdworkers were considered experts if they either test software professionally or if they have completed more than 20 tests on the platform. If the potential participants did not have professional experience in software testing and if they had completed less than 5 crowdtests in the past, they were considered novices. According to these criteria, the intermediary randomly invited and assigned potential participants from their pool of crowdworkers to their respective groups until a minimum of 20 participants for each group of experts and novices had been acquired.

A total of n=105 crowdworkers participated in the experiment. With regard to their nationality, 102 of these crowdworkers were from Germany, Austria, and Switzerland. The age of the crowdworkers ranged from 15 to 58 years with an average of 27.9 years and a standard deviation of 8.09 years. The 64 novices in our experiment were 25.9 years old on average (SD: 7.66). The 41 experts in our experiment were 31.0 years old on average (SD: 7.80). 54.3% of all participants were male, 25.7% were female. The remaining 20.0% of the crowdworkers provided no answer to this question. In general, these characteristics represent the target demographic for the website and thus suggest that the participants are appropriate subjects for the experiment.

To ensure that the software tests were performed in similar technical environments, all participants were required to use the same operating system (Windows 7) and browser (Google Chrome 46) during the experiment. These requirements were based on the system specifications that are most prevalent amongst crowdworkers registered on the intermediary's platform. Furthermore, the participants were compensated according to the compensation system employed by the crowdtesting intermediary. It grants the crowdworkers a fixed base pay for completing the software test and adds a variable compo-

nent based on the number and type of bugs found during the testing process. Duplicates, issues that cannot be reproduced, issues that are out of scope (i.e., not addressed by the tasks), or reports that include intended behaviour (i.e., website behaviour that users perceived as malfunctions but that were actually within the specifications) are declined. There are several reasons for adopting this compensation system: First, the fixed base pay reimburses the testers for completing all tasks and performing the software test. Secondly, the variable component acts as an incentive and warrants that differences in performance are reflected in the crowdworkers' financial rewards. Finally, using a compensation system as employed by the intermediary allows the experiment to be conducted under real conditions and ensures high external validity of the results. This compensation system represents the industry standard and thus allows for a better generalizability of the results.

Finally, we employed a manipulation check in order to verify that crowdworkers did indeed differ significantly in terms of their expertise. We analysed their subjective knowledge and experience in software testing. We adapted the 5 items developed by Flynn and Goldsmith (1999) to measure the participants' software testing knowledge on a 7-point Likert scale and the 4 items developed by Griffin, Babin and Attaway (1996) to measure their software testing experience on a 5-point Likert scale. Both scales proved to be internally reliable for our experiment with $\alpha = .828$ and $\alpha = .827$ respectively (Nunally and Bernstein, 1978). We calculated the means for these scales and compared the values for experts and novices using independent-samples t-tests. The results show that experts had significantly higher software testing knowledge ($p < .05$) and experience ($p < .01$). In consequence, it can be said that we successfully manipulated our crowdworker expertise factor.

5.4 Measures

For the functional testing tasks, we use three measures to capture the crowdworkers' performance (Farooq et al., 2011, Lee and Chang, 2013, Nirpal and Kale, 2011). First, we measure the number of bugs that have been submitted and accepted. In the final stage of our research, we are interested in the number of implemented bugs which represents a key performance indicator for functional testing. Additionally, we measure the severity of the bugs and the acceptance rates (i.e., the number of accepted bugs divided by the number of submitted bugs per tester). The process of reviewing the bugs was conducted by two independent professionals who both used a consistent review scheme employed by the intermediary in practice.

For the analysis of the test reports and the usability feedback, which is still in progress, we conduct a content analysis using category-based coding (Huberman and Miles, 1994). Additionally, we use the number of implemented usability issues and suggestions as a quantitative measure.

6 Preliminary Results

In the following, we will present our preliminary results regarding the performance of experts and novices on different tasks in crowdsourced software testing. These preliminary results focus on the descriptive statistics of the functional software test and provide first insights into potential findings for H_{1a} and H_{1b}. At this stage, we are not yet able to discuss the final statistical evaluation of the experiment for two reasons: As defined earlier, one of the most important measures in software testing is the number of implemented bugs that ultimately represent the relevant outputs of the software tests for website providers. However, this final implementation of the bug fixes is not completed yet. Secondly, the evaluation and coding of the usability feedback are still in progress.

The descriptive statistics depicted in Table 2 are concerned with the quantity of the test results (i.e., bug reports). With regard to both H_{1a} and H_{1b}, the preliminary results show that expert crowds found more bugs and reached higher acceptance rates than novice crowds on both simple and complex tasks in functional software testing – even though novice crowds were around 60% larger in the experiment than the expert crowds. As suggested by the literature and our hypotheses in section 4, the difference between the performance of experts and novices with regard the number of accepted bugs is more ap-

parent for complex tasks than for simple tasks. The statistics depicted in Table 3 are concerned with the severity of all accepted bugs. A comparison between the crowds hints at two findings. First, expert crowds surpass novice crowds with regard to the number of bugs found on simple and complex tasks across all severity ratings. Secondly, the (relative) discrepancy between the number of valid bugs reported by experts and novices rises with the severity of the bugs. Unsurprisingly, this discrepancy is larger on complex tasks than on simple tasks: the number of critical bugs found by the expert crowd on complex testing tasks is a lot larger than the number of critical bugs found by the novice crowd on the same tasks. Overall, the preliminary results indicate support our hypotheses H_{1a} and H_{1b}. As for our future work, it will be interesting to see whether novices were able to outperform experts on usability testing tasks.

		Bugs				**Accepted Bugs**				**Declined Bugs**			
		Min	Max	Sum	Mean (SD)	Min	Max	Sum	Mean (SD)	Min	Max	Sum	Mean (SD)
Simple Task	Novice	0	51	189	5.91 (9.31)	0	47	119	3.72 (8.41)	0	14	70	2.19 (2.75)
	Expert	0	40	229	10.91 (12.92)	0	33	168	8.00 (10.51)	0	12	61	2.90 (3.48)
Complex Task	Novice	0	15	110	3.44 (4.33)	0	14	64	2.00 (3.31)	0	11	46	1.44 (2.05)
	Expert	0	64	146	7.30 (14.45)	0	58	103	5.15 (12.94)	0	10	43	2.15 (2.80)
TOTAL		**0**	**64**	**674**	**6.42 (10.42)**	**0**	**58**	**454**	**4.32 (8.99)**	**0**	**14**	**220**	**2.10 (2.74)**

Table 2. Number of defects reported during the crowdsourced software test

Severity of Accepted Bugs	**Simple Task**								**Complex Task**							
	Novice				Expert				Novice				Expert			
	Min	Max	Sum	Mean (SD)	Min	Max	Sum	Mean (SD)	Min	Max	Sum	Mean (SD)	Min	Max	Sum	Mean (SD)
Low	0	43	80	2.50 (7.68)	0	17	101	4.81 (6.02)	0	9	36	1.13 (2.31)	0	40	57	2.85 (8.97)
Medium	0	5	21	0.66 (1.10)	0	12	36	1.71 (3.05)	0	4	18	0.56 (0.91)	0	9	20	1.00 (2.18)
High	0	3	11	0.34 (0.70)	0	6	23	1.10 (1.79)	0	2	6	0.19 (0.54)	0	5	12	0.60 (1.23)
Critical	0	2	7	0.22 (0.49)	0	4	8	0.38 (0.973)	0	1	4	0.13 (0.34)	0	4	14	0.70 (1.03)

Table 3. Severity of defects reported during the crowdsourced software test

7 Expected Contributions and Next Steps

In recapitulating our research in progress, there are several interim conclusions to be drawn. It has been discussed that task complexity and expertise jointly impact performance in crowdsourcing. The preliminary results indicate that expert crowdworkers perform better than novice crowdworkers on functional testing tasks. This tendency seems to be even more prominent when tasks are complex. Future research with our data will include more sophisticated statistical analyses with a more valid performance measure which is based on the number of implemented defects. Furthermore, the content analysis of the qualitative user reports will show whether the outcomes change for usability measures as theory would suggest (see H_{2a} and H_{2b}). Hence, our expected contribution is twofold. First, we contribute to crowdsourcing research by providing valuable insights for the process of designing and allocating tasks to crowds in ways that increase the crowdworkers' performance. Secondly, we investigate the performance of crowdsourced software testing as a novel approach for software testing. For practitioners, the results are expected to show what types of testing tasks should be assigned to which group of dedicated crowdworkers in order to achieve the best results possible. Thereby, we deliver valuable decision support for both crowdsourcers and intermediaries to enhance the performance of crowdsourcing initiatives.

As for our next steps, we intend to statistically test our hypotheses H_{1a} and H_{1b} as soon as the key performance indicator (i.e., amount of implemented bugs) for functional testing is available. Furthermore, we aim to analyse the qualitative data by applying a content analysis (Miles and Huberman, 1994) and determine our key performance index for the usability testing results to test hypotheses H_{2a} and H_{2b}.

References

Adelson, B. (1984), "When novices surpass experts: The difficulty of a task may increase with expertise", *Journal of Experimental Psychology: Learning, Memory, and Cognition,* 10 (3), p. 483.

Afuah, A. and Tucci, C. L. (2012), "Crowdsourcing as a solution to distant search", *Academy of Management Review,* 37 (3), pp. 355-375.

Bandura, A. (1977), "Self-efficacy: toward a unifying theory of behavioral change", *Psychological review,* 84 (2), p. 191.

Beizer, B. (2003), *Software testing techniques*, Dreamtech Press.

Bertolino, A. (2007), "Software testing research: Achievements, challenges, dreams", in *2007 Future of Software Engineering*, pp. 85-103.

Blohm, I., Bretschneider, U., Leimeister, J. M. and Krcmar, H. (2011), "Does collaboration among participants lead to better ideas in IT-based idea competitions? An empirical investigation", *International Journal of Networking and Virtual Organisations,* 9 (2), pp. 106-122.

Blohm, I., Riedl, C., Füller, J. and Leimeister, J. M. (2016), "Rate or Trade? Identifying Winning Ideas in Open Idea Sourcing", *Information Systems Research,* 27 (1), pp. 27-48.

Bonabeau, E. (2009), "Decisions 2.0: The power of collective intelligence", *MIT Sloan management review,* 50 (2), pp. 45-52.

Box, G. E., Hunter, J. S. and Hunter, W. G. (2005), *Statistics for Experimenters: Design, Innovation, and Discovery,* Hoboken, New Jersey: Wiley.

Byström, K. and Järvelin, K. (1995), "Task complexity affects information seeking and use", *Information processing & management,* 31 (2), pp. 191-213.

Campbell, D. J. (1988), "Task complexity: A review and analysis", *Academy of management review,* 13 (1), pp. 40-52.

Chanal, V. and Caron-Fasan, M.-L. (2010), "The difficulties involved in developing business models open to innovation communities: the case of a crowdsourcing platform", *M@n@gement,* 13 (4), pp. 318-340.

Chi, M., Glaser, R. and Rees, E. (1982), "Expertise in problem solving", *Advances in the Psychology of Human Intelligence (pp. 7-77)*, Hillsdale, NJ: Lawrence Erlbaum.

Chi, M. T. (2006), "Two approaches to the study of experts' characteristics ", *The Cambridge Handbook of Expertise and Expert Performance (pp. 21-30)*, Cambridge: Cambridge University Press.

Chi, M. T., Feltovich, P. J. and Glaser, R. (1981a), "Categorization and representation of physics problems by experts and novices", *Cognitive science,* 5 (2), pp. 121-152.

Chi, M. T., Glaser, R. and Rees, E. (1981b), "Expertise in problem solving", DTIC Document.

Doan, A., Ramakrishnan, R. and Halevy, A. Y. (2011), "Crowdsourcing systems on the world-wide web", *Communications of the ACM,* 54 (4), pp. 86-96.

Dolstra, E., Vliegendhart, R. and Pouwelse, J. (2013), "Crowdsourcing GUI Tests", in *Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on*, pp. 332-341.

Ericsson, K. A. and Charness, N. (1994), "Expert performance: Its structure and acquisition", *American psychologist,* 49 (8), p. 725.

Fähling, J., Blohm, I., Krcmar, H., Leimeister, J. M. and Fischer, J. (2011), "Accelerating customer integration into innovation processes using Pico Jobs", *International Journal of Technology Marketing,* 6 (2), pp. 130-147.

Faiks, A. and Hyland, N. (2000), "Gaining user insight: a case study illustrating the card sort technique", *College & research libraries,* 61 (4), pp. 349-357.

Faraj, S. and Sproull, L. (2000), "Coordinating expertise in software development teams", *Management science,* 46 (12), pp. 1554-1568.

Farooq, S. U., Quadri, S. and Ahmad, N. (2011), "Software measurements and metrics: Role in effective software testing", *International Journal of Engineering Science and Technology*, 3 (1), pp. 671-680.

Fincher, S. and Tenenberg, J. (2005), "Making sense of card sorting data", *Expert Systems*, 22 (3), pp. 89-93.

Flynn, L. R. and Goldsmith, R. E. (1999), "A short, reliable measure of subjective knowledge", *Journal of business research*, 46 (1), pp. 57-66.

Frese, M. and Zapf, D. (1994), "Action as the core of work psychology: A German approach", *Handbook of industrial and organizational psychology*, 4, pp. 271-340.

Geiger, D., Rosemann, M., Fielt, E. and Schader, M. (2012), "Crowdsourcing Information Systems - Definition Typology, and Design", in *33rd International Conference on Information Systems (ICIS)*, Orlando, USA.

Griffin, M., Babin, B. J. and Attaway, J. S. (1996), "Anticipation of injurious consumption outcomes and its impact on consumer attributions of blame", *Journal of the Academy of Marketing Science*, 24 (4), pp. 314-327.

Hackman, J. R. (1969), "Toward understanding the role of tasks in behavioral research", *Acta psychologica*, 31, pp. 97-128.

Haerem, T. and Rau, D. (2007), "The influence of degree of expertise and objective task complexity on perceived task complexity and performance", *Journal of Applied Psychology*, 92 (5), p. 1320.

Hoßfeld, T., Keimel, C., Hirth, M., Gardlo, B., Habigt, J., Diepold, K. and Tran-Gia, P. (2014), "Best practices for QoE crowdtesting: QoE assessment with crowdsourcing", *Multimedia, IEEE Transactions on*, 16 (2), pp. 541-558.

Howe, J. (2006), "The rise of crowdsourcing", *Wired magazine*, 14 (6), pp. 1-4.

Huberman, A. M. and Miles, M. B. (1994), "Data management and analysis methods".

Jeppesen, L. B. and Lakhani, K. R. (2010), "Marginality and Problem-Solving Effectiveness in Broadcast Search", *Organization Science*, 21 (5), pp. 1016-1033.

Kittur, A., Nickerson, J. V., Bernstein, M. S., Gerber, E. M., Shaw, A., Zimmerman, J., Lease, M. and Horton, J. J. (2013), "The Future of Crowd Work", *Computer Supported Cooperative Work 2013*, San Antonio, USA.

Lee, M.-C. and Chang, T. (2013), "Software measurement and software metrics in software quality", *International Journal of Software Engineering & Its Applications*, 7 (4), pp. 15-34.

Lees, R. B. and Chomsky, N. (1957), "Syntactic Structures", *Language*, 33 (3 Part 1), pp. 375-408.

Leicht, N., Durward, D., Blohm, I. and Leimeister, J. M. (2015), "Crowdsourcing in Software Development: A State-of-the-Art Analysis", *28th Bled eConference*, Bled.

Mäntylä, M. V. and Itkonen, J. (2013), "More testers–The effect of crowd size and time restriction in software testing", *Information and Software Technology*, 55 (6), pp. 986-1003.

Maynard, D. C. and Hakel, M. D. (1997), "Effects of objective and subjective task complexity on performance", *Human Performance*, 10 (4), pp. 303-330.

McKeithen, K. B., Reitman, J. S., Rueter, H. H. and Hirtle, S. C. (1981), "Knowledge organization and skill differences in computer programmers", *Cognitive Psychology*, 13 (3), pp. 307-325.

Miles, M. B. and Huberman, A. M. (1994), *Qualitative data analysis: An expanded sourcebook*, Sage.

Montgomery, D. C. (1984), *Design and analysis of experiments (8th ed.)*, New York: Wiley.

Myers, G. J., Sandler, C. and Badgett, T. (2011), *The art of software testing*, John Wiley & Sons.

Nirpal, P. B. and Kale, K. (2011), "A brief overview of software testing metrics", *International Journal on Computer Science and Engineering*, 3, pp. 204-211.

Nunally, J. C. and Bernstein, I. H. (1978), "Psychometric theory", New York: McGraw-Hill.

Quińones, M. A., Ford, J. K. and Teachout, M. S. (1995), "The relationship between work experience and job performance: a conceptual and meta-analytic review", *Personnel Psychology*, 48 (4), pp. 887-910.

Rouse, A. C. (2010), "A Preliminary Taxonomy of Crowdsourcing", in *Australian Conference on Information Systems (ACIS)*, 1-3 Dec 2010, Brisbane.

Schenk, E. and Guittard, C. (2011), "Towards a characterization of crowdsourcing practices", *Journal of Innovation Economics & Management*, (1), pp. 93-107.

Schenk, K. D., Vitalari, N. P. and Davis, K. S. (1998), "Differences between novice and expert systems analysts: What do we know and what do we do?", *Journal of Management Information Systems*, pp. 9-50.

Schlagwein, D. and Bjørn-Andersen, N. (2014), "Organizational Learning with Crowdsourcing: The Revelatory Case of LEGO", *Journal of the Association for Information Systems*, 15 (11), pp. 754-778.

Sonnentag, S. (1998), "Expertise in professional software design: a process study", *Journal of Applied Psychology*, 83 (5), p. 703.

Stol, K.-J. and Fitzgerald, B. (2014), "Two's company, three's a crowd: a case study of crowdsourcing software development", in *ICSE*, pp. 187-198.

Tung, Y.-H. and Tseng, S.-S. (2013), "A novel approach to collaborative testing in a crowdsourcing environment", *Journal of Systems and Software*, 86 (8), pp. 2143-2153.

Upchurch, L., Rugg, G. and Kitchenham, B. (2001), "Using card sorts to elicit web page quality attributes", *IEEE software*, 18 (4), pp. 84-89.

Vukovic, M. (2009), "Crowdsourcing for enterprises", in *Services-I, 2009 World Conference on*, pp. 686-692.

Wiley, J. (1998), "Expertise as mental set: The effects of domain knowledge in creative problem solving", *Memory & cognition*, 26 (4), pp. 716-730.

Wood, R. E. (1986), "Task complexity: Definition of the construct", *Organizational Behavior and Human Decision Processes*, 37 (1), pp. 60-82.

Zogaj, S., Bretschneider, U. and Leimeister, J. M. (2014), "Managing crowdsourced software testing: a case study based insight on the challenges of a crowdsourcing intermediary", *Journal of Business Economics*, 84 (3), pp. 375-405.

Zwass, V. (2010), "Co-creation: Toward a taxonomy and an integrated research perspective", *International Journal of Electronic Commerce*, 15 (1), pp. 11-48.