When is Crowdsourcing Advantageous? The Case of Crowdsourced Software Testing

Conference Paper · June 2016

DOI: 10.13140/RG.2.1.1833.6240

5 authors, including:

Niklas Leicht
University of St.Gallen
12 PUBLICATIONS **23** CITATIONS
SEE PROFILE

Nicolas Knop
University of St.Gallen
3 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Christoph Müller-Bloch
IT University of Copenhagen
4 PUBLICATIONS **18** CITATIONS
SEE PROFILE

Ivo Blohm
University of St.Gallen
94 PUBLICATIONS **750** CITATIONS
SEE PROFILE

WHEN IS CROWDSOURCING ADVANTAGEOUS? THE CASE OF CROWDSOURCED SOFTWARE TESTING

Research

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

Nicolas Knop, University of St. Gallen, St. Gallen, Switzerland, nicolas.knop@unisg.ch

Ivo Blohm, University of St. Gallen, St. Gallen, Switzerland, ivo.blohm@unisg.ch

Christoph Müller-Bloch, Copenhagen Business School, Copenhagen, Denmark, chmu14ab@student.cbs.dk

Jan Marco Leimeister, University of St. Gallen, St. Gallen, Switzerland, janmarco.leimeister@unisg.ch & University of Kassel, Kassel, Germany, leimeister@uni-kassel.de

Abstract

Crowdsourcing describes a novel mode of value creation in which organizations broadcast tasks that have been previously performed in-house to a large magnitude of Internet users that perform these tasks. Although the concept has gained maturity and has proven to be an alternative way of problem-solving, an organizational cost-benefit perspective has largely been neglected by existing research. More specifically, it remains unclear when crowdsourcing is advantageous in comparison to alternative governance structures such as in-house production. Drawing on crowdsourcing literature and transaction action cost theory, we present two case studies from the domain of crowdsourced software testing. We systematically analyze two organizations that applied crowdtesting to test a mobile application. As both organizations tested the application via crowdtesting and their traditional in-house testing, we are able to relate the effectiveness of crowdtesting and the associated costs to the effectiveness and costs of in-house testing. We find that crowdtesting is comparable in terms of testing quality and costs, but provides large advantages in terms of speed, heterogeneity of testers and user feedback as added value. We contribute to the crowdsourcing literature by providing first empirical evidence about the instances in which crowdsourcing is an advantageous way of problem solving.

Keywords: Crowdsourcing, Software Testing, Case Study

1 Introduction

New information technologies in connection with the advent of Web 2.0 have contributed to the rise of new sourcing approaches. As costs of mass communication decrease, companies are increasingly beginning to interact with large numbers of external sources (Zogaj et al., 2014). Companies can now tap into the resources of the masses (Vukovic, 2009) by taking a task or a function „once performed by employees and outsourcing it to an undefined […] network of people in the form of an open call" (Howe, 2006). This type of sourcing is called 'crowdsourcing'. Crowdsourcing has become a popular sourcing form, and an approach to effectively solve business problems (Brabham, 2008). Furthermore, crowdsourcing may create value for the firm that uses crowdsourcing by turning distant search into local search (Blohm et al., 2013, Afuah and Tucci, 2012). Hence, firms confronted with business problems need to decide whether to turn to the new option of crowdsourcing or to use proven governance structures such as in-house production (Afuah and Tucci, 2012). While firms would like to choose the most advantageous option for solving a given type of problem, it is not clear when crowdsourcing is

more advantageous compared to other governance structures (Afuah and Tucci, 2013, Bernstein et al., 2012, Zhao and Zhu, 2014).

To address this gap, we introduce a conceptual model that illustrates the relative advantage of crowdsourcing as a governance structure. We devise this contribution by merging research on the mechanisms of how crowdsourcing creates value (Afuah and Tucci, 2012) with the concept of transaction cost economics (TCE). Thus, we can construct a palpable conceptualization of the relative advantage of crowdsourcing. We argue that the conceptualization of the relative advantage of crowdsourcing is inextricably linked to the effectiveness of crowdsourcing and to the costs associated with crowdsourcing. While there is extant research on the effectiveness and costs of crowdsourcing (Schenk and Guittard, 2009, Stol and Fitzgerald, 2014, Zaidan and Callison-Burch, 2011), TCE provide a complementary lens for our purpose, as transaction costs such as costs for monitoring, coordinating, and controlling occur as a component of crowdsourcing value creation processes (Williamson, 1979). In addition, building on TCE augments the theoretical grounding of the conceptual model, thus enhancing the validity of the model. For our second contribution, we present two case studies of crowdsourced software testing to illustrate the conceptual model. Both case organizations tested the application via crowdtesting and their traditional in-house testing approaches. Thus, we are able to relate the effectiveness of crowdtesting and the associated costs to the effectiveness and costs of in-house testing, ultimately leading to instances in which crowdsourced software testing is advantageous compared to in-house production.

For scholars, our paper contributes to the knowledge base, as the conceptual model explores the mechanisms of the relative advantage of crowdsourcing. Closing this breach is a crucial step towards explaining when firms should use crowdsourcing as a governance structure, instead of employing internal or outsourced problem solving. Furthermore, our paper illustrates the relative advantage of crowdsourcing in the domain of software testing. Hence, we illustrate the specific factors that influence how companies profit from using crowdsourced software testing. In doing so, we shed light on the instances in which crowdtesting is an advantageous mode of value creation and in which it is not. For practitioners, our paper contributes knowledge regarding when crowdsourcing should be used or not.

The remainder of this paper is organized as follows. In section two, we present the theoretical background of our conceptual model. In the subsequent section, we summarize the methodology employed in this study. Afterwards, we present our case studies. Following that, we discuss and elaborate our findings. Finally, we provide theoretical as well as practical implications of our research.

2 Relative Advantage of Crowdsourcing

In this paper, we intend to combine crowdsourcing research and TCE in order to develop a conceptualization of the relative advantage of crowdsourcing from a crowdsourcer's perspective. Overcoming problems such as functional fixedness or local knowledge searches, crowdsourcing may provide more effective problem solutions than other governance structures (Afuah and Tucci, 2012, Bonabeau, 2009, Leimeister, 2010, Malone et al., 2010). However, to conceptualize the relative advantage of crowdsourced problem solutions, we need to consider not only the effectiveness of the crowdsourced problem solution, but also its costs. We posit that the relative advantage of crowdsourcing is inextricably linked to the effectiveness and costs associated with crowdsourcing (cf. Table 1).

According to TCE, costs associated with the sourcing of a problem can be subdivided into *direct* and *extra costs* (Williamson, 1981). For crowdsourcing, direct costs are the remuneration of the crowdsourcees and the payment of the intermediary (Carmel and Nicholson, 2005). Extra costs occur while managing the crowdsourcing process. Hence, the effectiveness of these problem solutions has to be related to these potential fees for the crowdsourcing intermediary as well as potential transaction costs that may occur for managing crowdsourcing and integrating these problem solutions. The term

"relative" implies that something is compared and set in relation to something else. Consequently, for our case, the *effectiveness of crowdsourcing* has to be put in relation to the *effectiveness of the in-house problem solutio*n which is the same for the *relative costs of crowdsourcing.*

Concept	*Definition*
Relative effectiveness of crowdsourcing	Effectiveness of crowdsourcing / effectiveness of alternative problem solution
Effectiveness of crowdsourcing	Capability of crowdsourcing to produce desired results
Relative costs of crowdsourcing	Costs of crowdsourcing / costs of alternative problem solution
Direct costs of crowdsourcing	Expenses incurred due to contractual obligations
Extra costs of crowdsourcing	Expenses incurred due to managing crowdsourcing

Table 1. Definition of key concepts

In the next three sections, we will elaborate on the extant literature on the relative effectiveness of crowdsourcing and relative costs of crowdsourcing. Furthermore, we also include and elaborate on literature of the effectiveness and costs of crowdsourced software testing in order to ensure the case studies are grounded in literature.

2.1 Relative Effectiveness of Crowdsourcing

We identified three main categories that have an impact on the effectiveness of crowdsourcing, i.e., quality, risk, and time that can contribute to the effectiveness of the crowdsourced problem solution. Regarding quality, crowdsourcing can benefit from network effects, as a growing number of contributors adds to the reputation of the problem-solving realm, attracting even more contributors who provide better solutions and increase heterogeneity (Schenk and Guittard, 2011). Thus, problem solvers of highly different backgrounds and knowledge can be engaged. By contrast, outsourcing organizations can actively choose to whom a certain task is allocated. For the context of software testing, Bonabeau (2009) recommends assessing the quality of solutions by examining the number as well as the quality and scope of issues that were not expected to be uncovered during the testing. Broad participation means that a deep pool of talented workers self-select the problems they solve to match their skills and expertise, hence creating high-quality solutions (Stol and Fitzgerald, 2014). Furthermore, the crowdsourcees may create a variety of potentially valuable solutions. This variety should be taken into account when assessing the quality of crowdsourcing (Estellés-Arolas and González-Ladrón-de-Guevara, 2012, Fähling et al., 2011). In a similar vein, Poetz and Schreier (2012) and Lakhani et al. (2013) found that problem solutions generated by the crowd may outperform the quality of solutions that have been generated by other governance structures.

Regarding risk, firms reduce the dependence on single providers, as tasks are not outsourced to a single provider (Schenk and Guittard, 2009). In comparison to crowdsourcing, outsourcing always poses the imminent risk of excessive dependence on the provider (Gonzalez et al., 2005). Moreover, outsourcing resembles an a priori supplier selection process, whereas crowdsourcing rather reflects an ex post selection of problem solutions (Afuah and Tucci, 2012, Kleemann et al., 2008). Another aspect that reduces the risk for the crowdsourcer is the nature of an open call. Since many solutions for the respective problem are being provided in this context, the risk of not receiving satisfactory input can be deemed relatively limited (Schenk and Guittard, 2009).

Regarding time, Stol and Fitzgerald (2014) found that software development projects tapping the resources of the crowd are characterized by a faster time-to-market. Crowdsourcing allows access to a large crowd of software developers, capable of performing this complex task across time zones as well as working simultaneously on decomposed tasks, and often willing to work on weekends as well. For

software testing, Mäntylä and Itkonen (2013) found that time-pressured crowds may deliver superior performance, as time-pressured individuals yield better bug detection than non-time-pressured. For usability testing in specific, Liu et al. (2012) found that crowdsourced testing can be conducted faster than lab testing.

2.2 Relative Costs of Crowdsourcing

In general, one advantage of crowdsourcing for the crowdsourcer are low costs that arise for payments to the crowdsourcing intermediary for managing the process and thus include the remunerations of the crowdsourcees. For instance, Zaidan and Callison-Burch (2011) found that non-professional translators were able to provide translations of high quality that were significantly cheaper than those by professional translators. Stol and Fitzgerald (2014) also found that using crowdsourcing reduces costs for using software developers, especially as extra cost overheads often incurred in hiring software developers can be avoided. For usability software testing, Liu et al. (2012) also found low cost to be an advantage compared to lab usability testing. Despite the generally rather minor costs, the remunerations crowdsourcees receive vary significantly. While some crowdsourcees work voluntarily, remunerations go from micro-payments to multi-million dollar payments (Byko, 2004). Often, non-professionals consider crowdsourcing as a source of additional income, and remunerations are rather low. Nevertheless, there are some crowdsourcing schemes that rely on non-financial incentives (Hippel and Krogh, 2003, Lerner and Tirole, 2002).

While being a significant cost factor, direct costs are not the only costs associated with crowdsourcing. In addition to direct costs, extra costs also occur. There is already some extant research on extra costs in the crowdsourcing context. Extra costs are all costs beyond the actual payments to the crowdsourcing intermediary and the crowdsourcees (Carmel and Nicholson, 2005). These extra costs incur as the crowdsourcer manages the crowdsourcing project, and can offset the cost savings from lower direct costs of crowdsourcing in comparison to other governance structures (Dibbern et al., 2008). According to Dibbern et al. (2008), there are several types of extra costs that arise in crowdsourcing context: search costs, contract/negotiation costs, specification costs, knowledge transfer costs, coordination costs, and control costs. Specification cost is defined as all costs associated with the process of defining what is needed as a result from the crowd or the intermediary and includes the design of the tasks being broadcasted to crowd. Ill-defined tasks may lead to contributions that do not add value and increase relative costs of crowdsourcing (Schenk and Guittard, 2011). The importance of decomposing and preparing problems to support the work of the crowdsourcees also holds for the case of crowdsourced software testing (Chen and Luo, 2014). The knowledge transfer cost is related to transmitting knowledge from one party to the other enabling it to execute as intended (Ko et al., 2005). When problem solvers are few, such as it is the case with in-house or outsourced problem-solving, feedback loops facilitate the alignment of task and problem solution. However, this is usually not possible when using crowdsourcing for problem solving (Schenk and Guittard, 2011). Interaction with crowdsourcees can also be difficult in the case of crowdsourced software testing. The coordination cost reflect the cost associated with integrating and combining the resources of the project owner and the crowd so the desired objectives can be achieved (Van de Ven et al., 1976). Finally, the control cost cover the cost related to measures taken by the project owner ensuring the crowd executes the tasks as planned (Sabherwal and King, 1995). Crowdsourcees tend to have uneven abilities. Hence, the results submitted by them should be inspected to assess their effectiveness (Liu et al., 2012).

3 Research Setting: Crowdsourced Software Testing

Crowdsourced software testing or crowdtesting 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, i.e., crowdsourcees. As with other crowdsourcing applications, companies, i.e. the crowdsourcer, can either directly interact with the crowd or they can use intermediaries who pro-

vide this service for a fee (Chanal and Caron-Fasan, 2010). These intermediaries act as brokers who connect the organizations that want to apply crowdsourcing with potential crowdsourcees and manage the crowdsourcing process (Zogaj et al., 2014, Leicht et al., 2015). Depending on the type of testing (e.g., functional testing, usability testing, security testing), these tasks as well as the targeted crowds can be very diverse (Stol and Fitzgerald, 2014).

In order to gain access to a crowd with the required skills, both case companies used the services of the same crowdtesting intermediary. The intermediary is one of the biggest crowdtesting intermediaries in Europe. Founded in 2011, the company has grown significantly since then and now counts about 100.000 registered users respectively testers on their platform. The company offers crowdsourced functional as well as usability testing for basically all types of software but is specialized in web and native mobile applications. The intermediary uses a self-developed platform for the intermediation process and offers a variety of services as well as a project manager who provides full support throughout an entire project.

To elaborate on the relative effectiveness of crowdsourcing and its cost and benefits it is crucial to have an object of comparison. Since our unit of analysis is a single testing project it was necessary that the project was replicated and also conducted by the in-house testing unit. Thus, the crowdsourcees received a build version which was also tested in-house. This redundant approach allows to compare test results and to ultimately gain important insights regarding the effectiveness and relative advantage of crowdsourced software testing.

The motivation for choosing crowdsourced software testing for our case studies is threefold. First, software testing is increasingly conducted externally. As firms are accustomed to having software testing performed externally, the shift to using crowdsourcing becomes less complex. Thus, we can focus our analysis on the characteristics of using crowdsourcing, excluding the general effects of shifting towards using external problem-solving from our analysis. Second, software testing may act as a microcosm for crowdsourcing insofar that it requires crowdsourcees to adapt to different degrees of task complexity and different forms of crowdsourcee expertise. Thus, this might enhance the generalizability of our results. Third, and most important, there is a grounded truth regarding the quality of work, as it is possible to measure the amount and quality of bugs uncovered during the testing (Bonabeau, 2009).

4 Methodology

4.1 Case Study Research

A case study design is useful when the phenomenon under scrutiny has not yet received appropriate attention in the extant literature, and existing knowledge with regard to the issue is vague and ambiguous (Eisenhardt and Graebner, 2007). As this is the case regarding the topic of the relative advantage of crowdsourcing, we believe a case study is a suitable approach for investigating the research problem. We conducted instrumental case studies, since the companies were selected to gain deeper knowledge about crowdtesting and its value determinants in organizations (Yin, 2013). For qualitative studies the sample size depends on the judgment of the researcher (Pare, 2004, Yin, 2013). We decided to conduct two case studies in order to collect sufficient data and to increase the generalizability of our results on an analytical level since two cases offer the possibility to cover different frame conditions for software testing in companies. First, the industry and thereby varying importance and know-how of corporate IT can be covered. Accompanied to that, the department size and the corresponding governance structures and mechanism vary largely and may influence the effectiveness of crowdsourced software testing. Another very important difference we wanted to cover with two cases is the software development paradigm, i.e., traditional waterfall model versus agile development.

The selected cases represent *common cases* which illustrate and represent the circumstances and conditions of everyday business situations. To ensure comparability, the cases were selected based on the following criteria: (1) The crowdsourcing companies have comparable experience with crowdsourcing, especially in software testing. (2) The test object for which crowdtesting has been applied is a mobile application for costumers. These types of applications represent a very common case since almost every company has a consumer application or similar. (3) The application to be tested is tested in-house until now and testing is not sourced out. (4) The application is tested redundant in-house to ensure comparability of results for both crowd- and in-house production. While the concept of relative advantage can be examined from different levels of analysis (Lepak et al., 2007), we focus on the test of the software as unit of analysis, as the case studies comprise a single project of crowdsourced software testing in each case.

4.2 Data Collection

For data collection, case studies typically feature multiple methods and data sources (Meredith, 1998). Data sources for this study include semi-structured, in-depth interviews, project documentation such as meeting minutes and monthly status updates, as well as platform data including real-time access to bug reports and raw data submitted by the crowdsourcees. For our interviews, we created a roughly structured guideline with questions regarding topics such as project cost drivers and quality of crowdsourcee work. Interviews were conducted before and after the tests to elucidate expectations, as well as after the project finished, in order to review the course of the project. Overall, we conducted twelve interviews, six for each case with duration of about 30-60 minutes per interview. Interviews were conducted between February and July 2015. In addition, we conducted two workshops with the companies´ test managers to validate bugs and to define according severity levels. The duration of the workshops was about 90-120 minutes. All interviews were recorded and subsequently transcribed. In addition, detailed notes were taken during the interviews. Table 2 depicts the data sources used in both case studies.

(1) Pre-Interviews	
Interviewee(s)	*Content/Subject*
Group interview (Case A:1; Case B:1) (all interviewees together)	• Determination of project requirements (time, cost, quality) • Determination of specific testing specifications
Project manager (Case A:1; Case B:1)	• Inquiry of expectations of quality • Inquiry of perceived project progression
Test manager(s) (Case A: 2; Case B:1)	
(2) Post-Interviews	
Group interview (Case A:1; Case B:1) (all interviewees together)	• Inquiry of perceived crowdsourced software testing project success and satisfaction with progress, quality, and results • Comparison with in-house testing
Project manager (Case A: 1; Case B:0)	
Test manager(s) (Case A: 1; Case B:1)	
(3) Project Documentation	• Meeting minutes; Reports provided by intermediary
(4) Platform Data	• Crowdsourcee reports and bug reports; observation of crowdtests

Table 2. Data sources

For data analysis, we conducted a content analysis using category-based coding (Miles and Huberman, 1994). However, we also considered extant theory and used the data to refine and advance existing knowledge (Auerbach and Silverstein, 2003), thus enhancing confidence in our findings (Dubé and Paré, 2003).

5 Case Studies

In this section, we will elaborate on our case studies and its findings to employ them to identify advantageous instances. By focusing on the case of crowdsourced software testing, we uncover the main determinants of the relative advantage of crowdsourced software testing.

5.1 Case A: The Case of the Swiss Bank

For our first case, we chose a Swiss bank with a large centralized test department. The scope of the testing project was the testing of the mobile banking application provided by the bank for its customers. To overall project followed the traditional waterfall approach. To gain better insights regarding the reliability of results, the project was divided in two testing iterations: Focus of the first iteration was the functional testing of the public areas in the app, i.e., all parts where users do not need log in. Since the company was also interested in user feedback, a combined usability/functional test was conducted two weeks after the first test run had been executed. The overall time frame from project initiation – i.e., kick-off meeting between the bank and the intermediary – to project closure comprised eight weeks. The test execution time, i.e., the length of the actual crowdtest, in both iterations was three days. The bank decided to make use of the managed service as an overarching goal was to determine whether crowdsourced software testing is able to reduce load peaks during the release cycles. The managed service includes support throughout the whole process by the intermediary, from test preparation through the testing phase to the evaluation of the submitted bugs.

Relative Effectiveness of Crowdsourcing

For the first test iteration, 26 crowdsourcees (81% German, 19% Swiss; 69% male) with 21 device/operating system (OS) combinations, e.g., iPhone6 with iOS 9.0, Samsung Galaxy S4 with Android 4.4.2, conducted the test. For the second test, 30 crowdsourcees (60% German, 40% Swiss; 77% male) tested with 29 different device/OS combinations. This diversity allowed the bank to conduct a multitude of compatibility tests which they could have not performed in-house due to the limited variety of devices they keep in stock. Most of the crowdsources had a professional testing background (55%), whereas 24% were leisure time testers and 21% crowdsources had basic testing experience. Furthermore, the bank specified that a certain amount of the testers should be customers in order to receive valuable usability feedback in the second iteration by real end users. This heterogeneity among the crowdsourcees was perceived as great asset, especially in regards of the coverage of devices to test the compatibility of the app and the unbiased view of the crowdsourcees compared to in-house testers.

"Heterogeneity of the crowd is very important for me. They not only perform the same activities on the same devices over and over again like our in-house testers. They really are a heterogeneous group that is able to find defects which would have never come to our minds otherwise." (Swiss bank test manager 1)

In the first test iteration a total of 55 bugs were found. The intermediary filtered non-reproducible or poor documented bugs and reported 39 bugs to the bank. The test managers finally accepted 31 out of these 39 bugs for further consideration. In the next step, all non-functional bugs were cut, which left a total of 10 functional bugs. Out of these 10 bugs, one bug was excluded due to low severity. Four bugs were also reported by in-house testers. Ultimately, the crowdsourcees found five functional bugs that in-house testers did not find compared to four bugs by in-house testing that the crowdsourcees did not detect. These results indicate that the crowdsourcees performed as well as the in-house testing team in terms of test quality. This assumption is also supported by qualitative data.

"We were happy with the work of the crowdsourcees. I was quite surprised by the quality of their work. What some testers found was impressive." (Swiss bank test manager 2)

An important aspect in quality is the reliability of the testing performance and the corresponding results. To examine the reliability of crowdsourced software testing, the bank decided to design the second test cycle as a regression test to validate that the modified software has no new errors and ensure that modifications were correct (Graves et al., 2001). Subsequently, the bugs found in the first test cycle were fixed and a new build was delivered and tested. The crowd proved to be sufficiently reliable since the amount of bugs found in the second test iteration was significantly lower. Moreover, there were several bugs which were not fixed due to time pressure and low severity and were thus reported again in the second cycle.

Further, the bank captured time-related benefits. The testing cycles were each conducted and evaluated in about one week. After a kick-off meeting between the intermediary and the bank to define the scope and crowd-related specifications such as socio-demographics, testing experience, and hardware configurations (device/OS combination), the intermediary invited the crowdsourcees based on those specifications. Just 24 hours later the testing phase started. The test duration was three days in which the 26 crowdsourcees tested the application one hour on average. Furthermore, the test was conducted over the weekend and most activity by the crowdsourcees occurred in the evening as most of them participated in addition to holding down a regular job. This short set-up time and the possibility to utilize the weekend as labor time allowed the bank to apply crowdsourced software testing on short notice, which was perceived as one of the greatest benefits in this project.

"One thing amazed me in particular – it was the short time in which we were able to set up and execute the test. (...) It gave us a lot of flexibility and it is possible to apply it very quick if necessary." (Swiss bank project manager)

Relative Costs of Crowdsourced Software Testing

The main cost driver for crowdsourcing in this case is a fix sum the bank paid to the intermediary for its services. That fee included the remuneration of the crowdsourcees as well as consulting services, i.e., the development of the test cases (i.e., the tasks performed by the crowd), crowd support throughout the test, filtering bugs, and the preparation of a final bug report. The consulting services provided by the intermediary helped to reduce the extra costs in terms of time the bank employees had to spend on the crowdtest. Furthermore, we found that conducting the software tests in-house would require three to four working days, which would result in costs that are roughly equivalent to the costs incurred when using crowdsourcing. Hence, we contend that direct costs of crowdsourced software testing may well be similar to the costs of conducting software tests in-house.

However, there were still activities that had to be performed by the internal test manager, creating extra costs for the bank. The test manager reported that he spent about 1.2 days for the first test phase. There are three main cost drivers: First, coordination costs occurred, i.e., the preparation of the test, including the selection of crowd criteria (geographic and demographic characteristics as well as device/OS combinations) and definition of the test scope. Second, control costs occurred, i.e., evaluation and analysis of the bugs that were identified. Third, knowledge transfer costs occurred, i.e., the translation of these bugs into requests for the developers. Nevertheless, we found that by the second test cycle, the test manager was able to significantly reduce the time by almost 40% due to the fact that many configurations stayed the same. Furthermore, training curve effects emerged during the process. This shows that one approach to increase the effectiveness of crowdsourcing is repetition. Repetition facilitates the reduction of the amount of extra costs for crowdsourcing.

5.2 Case B: The Case of the Swiss Industrial Enterprise

The second case was conducted with a large Swiss industrial enterprise with a rather small test department. The software development process is agile and testing takes part after every sprint. The company chose the service of the same crowdtesting intermediary as in the previous case. The scope of the project was the testing of the company's public application, where customers can get information about the products of the company and even order spare parts. The application is usually used by plumbers or retail experts with special knowledge of the spare parts. The setting did focus only on functional testing and was not interested in usability feedback in the first instance. The crowdsourced software test was supposed to include 20 testers, who tested the app in twelve different country versions. The overall time frame from the project was comparable with the first case, about eight weeks. The test execution time was three days starting Friday evening.

Relative Effectiveness of Crowdsourcing

The company chose a setting which led to a highly diverse crowd. That was perceived as a valuable asset. In total, 22 testers participated in the test, two more than expected. 19 testers covered 10 different countries; three additional testers did not specify where they tested. The specification of the company regarding countries and devices were mostly fulfilled. In addition to that the testers covered a broad range of device/operating system configurations. In total, 22 combinations were covered, which was perceived as an asset. Therefore the company could conduct a broad compatibility test, which was not possible in-house and was perceived as an advantage from crowdsourced software testing compared to traditional in-house testing. Furthermore, the test was conducted by laymen. The test should demonstrate that an application which is usually used by experts can be tested by novices and still produces satisfactory results.

Most of the bugs were found in the first 36 hours after the test has started. In total, 49 bugs were submitted by the crowd, 19 of them were accepted by the company. Six of the 19 accepted bugs were fixed, seven bugs were already known internally and the last seven bugs were not further considered due to low severity. On one hand the crowdsourcees found 6 bugs, the internal testing did not find. On the other hand the internal testing found 5 bugs the crowdtester did not detect. These results indicate that quality-wise, the crowdsourcees performed as well as the in-house testing team. The test manger rated both results quality wise as comparable.

Thirty bugs were not accepted due to their usability nature, them being duplicates or intended behavior of the application, i.e., user reported perceived malfunctions of the app that are within their specification. However, a few usability bugs were accepted by the company even though they were out of scope, because these usability bugs had a high severity and could not be ignored.

"We knew the usability bug before, but for us it was a bug with a medium severity, at most. So we thought about changing it. When we saw that more than half of the crowd testers submitted that very same usability bug, even though it was out of scope, we knew it was really problematic. We changed the severity and will fix it." (Swiss industrial enterprise test manager)

Finally, six new tickets were the result of the crowd test. The test manager perceived the bugs of the crowd comparable to the internal testing in terms of quality and quantity in tickets. Further, the case study revealed that the company captured time-related benefits. The test itself took three days. The major part of the test was on the weekend. Hence, the company could work shortly after the weekend with the result and perceived an additional gain of two working days over the weekend.

"The bugs that we will fix internally were almost all found by the crowd; therefore it is not identical but comparable. So the overall quality is pretty good" (Swiss industrial enterprise test manager).

Relative Costs of Crowdsourced Software Testing

In regards to the costs of the testing project, the company found that internal and external testing through the crowd had almost the same costs. Regarding the crowd test only, the direct costs, i.e., direct payment to the intermediary, and extra costs were equally high. The company ordered a so called "Self Service". That means, the intermediary supplied the platform and supported the company until the test but there was no support regarding verification of bugs. The intermediary set up the test, provided the testers and started the test. During the test and afterwards the company used the platform and managed the crowd, as the results independently. The company revised all bugs itself and communicated directly with the crowd during and after the test. Due to the self-service the direct costs them-self were 30% lower compared to costs for the in-house testers.

However, there were still activities that had to be performed by the internal test manager, creating extra costs for the company. The test manager reported that he spent about two days for the crowd test. The comparison of the extra costs in terms of preparation and the post processing, such as communicating with the crowd etc., of the crowd test and the internal costs for executing a test process by the test manager showed that the crowdsourced software test was 50% more expensive. There are three main cost drivers of the crowd test: First, coordination costs occurred, i.e., the preparation of the test, including the selection of crowd criteria (geographic and demographic characteristics as well as device/OS combinations) and definition of the test scope. Second, control costs occurred, i.e., evaluation and analysis of the bugs that were identified. Third, knowledge transfer costs occurred, i.e., the translation of these bugs into requests for the developers. Especially the control costs were significantly higher than internally. In particular, the evaluation of the bugs by the test manager was much more extensive than with an internal test. The amount of bugs submitted by the crowd was higher than by internal testers. All 49 bugs had to be reviewed, understood and reproduced. Internally fewer bugs were submitted with a higher acceptance rate, leading to lower control costs.

"The next time I will use the Full Managed Service, and then I have less work. But overall I am satisfied; we covered almost all countries and devices" (Swiss industrial enterprise test manager)

5.3 Cross-Case Analysis

The two case studies found common ground in the results. In the following, we will shed light on these factors, which also determine the relative advantage of crowdsourced software testing. We compare how the three main factors of the relative advantage of crowdsourcing, i.e., the effectiveness of crowdsourcing, direct costs, and extra costs, manifest themselves in both the cases. First, the factor time or speed was perceived as advantage in both cases. The tests were set up fast and executed in only three days. During the test the crowdsourcees submitted most of their bugs in the first 36 hours of each test. Furthermore, it was possible to access the bugs in real time, allowing the test manager to synchronize activities. In both of our cases the tests were conducted over the weekend, and the results were evaluated the beginning of the following week. This gave the bank and the industrial company two perceived additional work days.

Second, the quality of the accepted bugs of the crowdsourced tests were rated equal compared to the quality of the replicated internal tests in both cases. In addition to that, the crowd submitted usability bugs and suggestions, even though the crowd was not supposed to do so. Nevertheless, in both cases the submission of usability bugs was valued as an advantage, due to the fact that these usability bugs were useful by directing the focus of the companies on problems they had not perceived as important or not existent, indicating a certain operational blindness. The amount of bugs submitted by the crowd were similar in both cases, as well as the amount of new bugs rated by the companies (cf. figure 1).

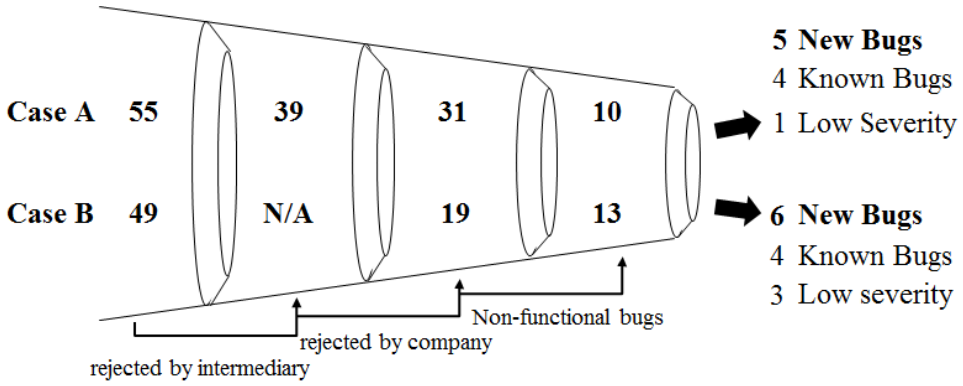

Figure 1. Bug selection and verification

Third, the heterogeneity of crowdtesting was experienced as an asset in both cases. The companies could define their specifications of their tests individually according to the requirements of the tests, such as devices, demography of the testers or countries. These specifications were mostly met by the crowd. Finally, the total costs of the crowd tests were comparable to similar in-house tests. Yet, the balance between direct costs and extra costs weighs differently between the two cases. The main reason for this is the service level chosen by the companies. The bank chose a full managed service and the industrial company selected a self-service. The main difference lies in the service frame; the test with the self-service was supported by the intermediary until the test start. The intermediary helped setting up the test by organizing the right testers according to the specification of the company, providing them with the instructions and the application to be tested. After the organizational part of the test was completed, the intermediary let the industrial company take over at the start of the test, the operational part. The control costs, especially the evaluation of the bugs submitted was responsible for the increased part of the extra costs in the case of the industrial company. The bank ordered a managed service; therefore a first analysis of the bugs submitted by the crowd was conducted by the intermediary, reducing the control costs for the bank. This is the reason why in the figure 1 the second number in Case B is not available. The intermediary did not support the industrial company by reviewing bugs as a first instance. In addition to that, the industrial company rejected much more bugs compared to the Case A due to a narrow out of scope definition.

6 Discussion

The case studies and cross-case analysis unfold the main drivers and conditions for a relative advantage of crowdsourced software testing compared to in-house production and proved to be applicable in both software development paradigms. Given the circumstances that the costs for crowdsourcing are approximately equal to in-house production as the case in our two case studies, crowdsourcing can be advantageous under many conditions (cf. Table 3).

First, the test object influences the relative advantage of crowdsourcing. The case companies tested public mobile applications for customers, thus these applications are subsequently not complex in its nature because they are built and designed for a specific, but public audience. Due to this fact, test managers are confronted with the almost impossible assignment to ensure the compatibility on all publicly available device/OS configurations while crowdsourcing can easily cope with that issue and produce relief in that matter. Also, through crowdsourced software testing, companies can address specific target groups (customers of the bank in Case A; people from very different countries for Case B) which further increases the relative advantage.

Instances in which crowdtesting is advantageous	*Instances in which in-house testing is advantageous*
Test object is for a broad/public user group and for multiple platforms/devices	Test object is very specific and/or rather small test scope
Specific target group of testers (e.g. specific demographics)	Testers require functional expert knowledge
Usability feedback welcomed	Specific tests, i.e. security or vulnerability testing or special hardware required
Time pressure	Low maturity of application
Lack of internal test resources	High security standards of application (authentication)
Standardized testing (i.e. regression testing)	

Table 3. Instances for relative advantage of crowdsourced software testing

Our cross-case analysis further revealed time-related benefits. Crowdsourced software testing can be ramped up very quickly and becomes more effective if the testing unit has to deal with time or resource constraints, i.e. crowdsourcing can be used to reduce internal load peaks through mobilizing workforce on weekends and evenings. Hence, test managers usually have real-time access to submitted bugs and thus can synchronize the process of testing and bug verification and validation. Accordingly, we posit the relative advantage of crowdsourced software testing increases under time and resource constraints. Moreover, companies applying crowdtesting can benefit from valuable usability feedback which might not be regarded as severe or not recognized due to organizational blindness.

However, crowdsourced software might not always be advantageous. This is especially the case when the testers require a lot functional (not testing) expert knowledge to test the application or to conduct very specific testing tasks such as security testing. The case of the Swiss bank revealed that there are also a lot of data security issues and concerns. In general, one can say as a rule of thumb: The higher the security standards (i.e. two-factor authentication with special hardware for a banking app), the less advantageous crowdsourcing becomes. Crowdsourcing costs, especially extra costs can become a relative disadvantage in small testing departments where internal communication between test manager and testers is easy and very fast.

7 Implications

7.1 Theoretical Contributions

We make two important contributions to the crowdsourcing literature. First, the conceptualization explores the mechanisms of the relative advantage of crowdsourcing. Closing this breach is a crucial step towards explaining when firms should use crowdsourcing as a governance structure, instead of employing internal or outsourced problem solving. Furthermore, we believe that this model can be used as a basis to explore the concrete mechanisms of how crowdsourcing creates value for all crowdsourcing task types, not only for the specific case of crowdsourced software testing.

Second, this paper illustrates the relative advantage of crowdsourcing in the domain of software testing. Hence, we illustrate the specific factors that influence how crowdsourcers profit from using crowdtesting. In doing so, we shed light to the instances in which crowdtesting is an advantageous mode of value creation and in which it is not. Our empirical analysis reveals that the benefits of crowdsourcing for companies rather evolve via an increased effectiveness of solving the problems at hand than via relative cost benefits. In this vein, occurring extra-costs do offset invoked direct costs that make crowdsourcing frequently appear as cheap alternative. These results are particularly true for companies that start engaging in crowdsourcing. However, our results also imply that crowdsourcers may quickly capitalize on learning effects such that cost benefits may also incur in the long run. In doing so, we follow the call of various researchers (e.g.,Afuah and Tucci, 2012, Zhao and Zhu, 2014) in order to explore the value of crowdsourcing. We argue that this is an important step towards con-

ceptualizing the relative advantage of using crowdsourcing instead of other governance structures in the case of crowdsourced software testing.

7.2 Practical Implications

For practitioners, our paper contributes knowledge regarding when crowdsourcing should be used or not. Our results reveal the factors that determine whether or not crowdsourcing is more effective than in-house production. More specifically, our study is relevant for practitioners who are interested in crowdsourced software testing. By specifying effectiveness as well as direct and extra costs, we provide an overview of the factors test managers need to consider when assessing crowdsourcing as a governance structure. Hence, our results can help them assess whether crowdsourcing represents a superior sourcing approach in their specific case.

7.3 Limitations and Future Research

Our paper is not without shortcomings. This study operationalizes the factors that determine the relative advantage of crowdsourcing software testing tasks. This is a specific type of crowdsourcing that may come in many different notions. Literature illustrates that effectiveness and costs of crowdsourcing largely depend on the type of crowdsourcing task and its modularizability (Schenk and Guittard, 2011, Afuah and Tucci, 2012). Although we do not see any reason why our results should not generalize to other types of crowdsourcing, future research should address similar studies in other domains of crowdsourcing in order to increase generalizability of our results (Eisenhardt, 1989). Second, our study has to deal with limitations of qualitative research, although our study design, was supposed to produce generalizable results due to employing two comparable case study design and rigorous analysis of the obtained data. However, quantitative approaches such as survey research could further increase generalizability of our results. Third, our results indicated that crowdsourcer quickly capitalized on learning effects that we could not sufficiently cover in the frame of our research. However, how the effectiveness and associated costs of crowdsourcing evolve over time has not yet addressed by existing research. Thus, we need more longitudinal research about how evolving "crowdsourcing capabilities" do change the relative advantage of crowdsourcing over time.

8 Conclusion

In this paper, we present two case studies from the domain of crowdsourced software testing. Drawing on crowdsourcing literature and transaction action cost theory, we systematically analyze two organizations that applied crowdtesting to unveil the instances of the relative advantage of crowdsourcing compared to in-house production. We find that crowdtesting is comparable in terms of testing quality and costs, but provides large advantages in terms of speed, heterogeneity of testers and user feedback as added value.

Acknowledgements

The third author received support from the basic research fund of the University of St. Gallen.

References

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

Afuah, A. and Tucci, C. L. (2013), "Value capture and crowdsourcing", *Academy of Management Review*, 38 (3), pp. 457-460.

Auerbach, C. and Silverstein, L. B. (2003), *Qualitative data: An introduction to coding and analysis*, NYU press.

Bernstein, A., Klein, M. and Malone, T. W. (2012), "Programming the global brain", *Communications of the ACM*, 55 (5), pp. 41-43.

Blohm, I., Leimeister, J. M. and Krcmar, H. (2013), "Crowdsourcing: How to Benefit from (Too) Many Great Ideas", *MIS Quarterly Executive*, 4 (12), pp. 199-211.

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

Brabham, D. C. (2008), "Crowdsourcing as a Model of Problem Solving", *Convergence: The International Journal of Research into New Media Technologies*, 14 (1), pp. 75-90.

Byko, M. (2004), "SpaceShipOne, the Ansari X Prize, and the materials of the civilian space race", *JOM Journal of the Minerals, Metals and Materials Society*, 56 (11), pp. 24-28.

Carmel, E. and Nicholson, B. (2005), "Small firms and offshore software outsourcing: high transaction costs and their mitigation", *Journal of Global Information Management*, 13 (3), pp. 33-54.

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.

Chen, Z. and Luo, B. (2014), "Quasi-crowdsourcing testing for educational projects", in *Companion Proceedings of the 36th International Conference on Software Engineering*, pp. 272-275.

Dibbern, J., Winkler, J. and Heinzl, A. (2008), "Explaining variations in client extra costs between software projects offshored to india", *MIS quarterly*, 32 (2), pp. 333-366.

Dubé, L. and Paré, G. (2003), "Rigor in information systems positivist case research: current practices, trends, and recommendations", *MIS quarterly*, pp. 597-636.

Eisenhardt, K. M. (1989), "Building theories from case study research", *Academy of Management Review*, 14 (4), pp. 532-550.

Eisenhardt, K. M. and Graebner, M. E. (2007), "Theory building from cases: opportunities and challenges", *Academy of management journal*, 50 (1), pp. 25-32.

Estellés-Arolas, E. and González-Ladrón-de-Guevara, F. (2012), "Towards an integrated crowdsourcing definition", *Journal of Information science*, 38 (2), pp. 189-200.

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

Gonzalez, R., Gasco, J. and Llopis, J. (2005), "Information systems outsourcing risks: a study of large firms", *Industrial management & Data systems*, 105 (1), pp. 45-62.

Graves, T. L., Harrold, M. J., Kim, J.-M., Porter, A. and Rothermel, G. (2001), "An empirical study of regression test selection techniques", *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 10 (2), pp. 184-208.

Hippel, E. v. and Krogh, G. v. (2003), "Open source software and the "private-collective" innovation model: Issues for organization science", *Organization science*, 14 (2), pp. 209-223.

Howe, J. (2006), "Crowdsourcing: A definition", *Crowdsourcing: Tracking the rise of the amateur*.

Kleemann, F., Voß, G. G. and Rieder, K. (2008), "Un (der) paid innovators: The commercial utilization of consumer work through crowdsourcing", *Science, Technology & Innovation Studies*, 4 (1), pp. PP. 5-26.

Ko, D.-G., Kirsch, L. J. and King, W. R. (2005), "Antecedents of knowledge transfer from consultants to clients in enterprise system implementations", *MIS quarterly*, pp. 59-85.

Lakhani, K. R., Boudreau, K. J., Loh, P.-R., Backstrom, L., Baldwin, C., Lonstein, E., Lydon, M., MacCormack, A., Arnaout, R. A. and Guinan, E. C. (2013), "Prize-based contests can provide solutions to computational biology problems", *Nature biotechnology*, 31 (2), pp. 108-111.

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.

Leimeister, J. M. (2010), "Collective intelligence", *Business & Information Systems Engineering*, 2 (4), pp. 245-248.

Lepak, D. P., Smith, K. G. and Taylor, M. S. (2007), "Value creation and value capture: a multilevel perspective", *Academy of Management Review*, 32 (1), pp. 180-194.

Lerner, J. and Tirole, J. (2002), "Some simple Economics of Open Source", *The Journal of Industrial Economics*, 50 (2), pp. 0022-1821.

Liu, D., Bias, R. G., Lease, M. and Kuipers, R. (2012), "Crowdsourcing for usability testing", *Proceedings of the American Society for Information Science and Technology*, 49 (1), pp. 1-10.

Malone, T. W., Laubacher, R. and Dellarocas, C. (2010), "The collective intelligence genome", *IEEE Engineering Management Review*, 38 (3), p. 38.

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.

Meredith, J. (1998), "Building operations management theory through case and field research", *Journal of operations management*, 16 (4), pp. 441-454.

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

Pare, G. (2004), "Investigating information systems with positivist case research", *The Communications of the Association for Information Systems*, 13 (1), p. 57.

Poetz, M. K. and Schreier, M. (2012), "The value of crowdsourcing: can users really compete with professionals in generating new product ideas?", *Journal of Product Innovation Management*, 29 (2), pp. 245-256.

Sabherwal, R. and King, W. R. (1995), "An Empirical Taxonomy of the Decision-Making Processes concerning Strategic Applications of Information Systems", *Journal of Management Information Systems*, 11 (4), pp. 177-214.

Schenk, E. and Guittard, C. (2009), "Crowdsourcing: What can be Outsourced to the Crowd, and Why", in *Workshop on Open Source Innovation, Strasbourg, France*.

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

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.

Van de Ven, A. H., Delbecq, A. L. and Koenig Jr, R. (1976), "Determinants of coordination modes within organizations", *American sociological review*, pp. 322-338.

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

Williamson, O. E. (1979), "Transaction-cost economics: the governance of contractual relations", *Journal of law and economics*, pp. 233-261.

Williamson, O. E. (1981), "The economics of organization: The transaction cost approach", *American journal of sociology*, pp. 548-577.

Yin, R. K. (2013), *Case study research: Design and methods*, Sage publications.

Zaidan, O. F. and Callison-Burch, C. (2011), "Crowdsourcing translation: Professional quality from non-professionals", in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pp. 1220-1229.

Zhao, Y. and Zhu, Q. (2014), "Evaluation on crowdsourcing research: Current status and future direction", *Information Systems Frontiers*, 16 (3), pp. 417-434.

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.