Given Enough Eyeballs, all Bugs are Shallow – A Literature Review for the Use of Crowdsourcing in Software Testing

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

Abstract

Over the last years, the use of crowdsourcing has gained a lot of attention in the domain of software engineering. One key aspect of software development is the testing of software. Literature suggests that crowdsourced software testing (CST) is a reliable and feasible tool for manifold kinds of testing. Research in CST made great strides; however, it is mostly unstructured and not linked to traditional software testing practice and terminology. By conducting a literature review of traditional and crowdsourced software testing literature, this paper delivers two major contributions. First, it synthesizes the fields of crowdsourcing research and traditional software testing. Second, the paper gives a comprehensive overview over findings in CST-research and provides a classification into different software testing types.

1. Introduction

Today, many IT departments face an increasingly dynamic environment, shorter product lifecycles and cost pressure. The rapid development of new IT-enabled business models and a fast growing hardware market as well as its segmentation - smartphones, tablets, wearable technologies, or the Internet of Things - are making software testing increasingly complex. Given the increased complexity, the domain of software testing is about to develop manifold approaches to overcome this issue. One approach is test automation, that is the automated execution of pre-scripted tests via software [1]. However, since automated testing is still not applicable in many settings [2] and most tasks still require human intelligence in order to be performed, traditional approaches are becoming less applicable – both economically and practicably [3].

With the advent of digitization and the rise of advanced web technologies, more and more companies are using IT to connect with groups of individuals for resource [4] and value creation purposes [5, 6]. Using groups of individuals over the internet that voluntarily undertake tasks based on a flexible open call is known as crowdsourcing [7, 8] and recently found its application in software testing [9, 10].

In crowdsourced software testing (CST) or crowdtesting, a diverse pool of people test software in real environments using their own devices [11]. This form of quality assurance is adapted from open source code reviews, following the mantra: "Given enough eyeballs, all bugs are shallow" [12]. Research on CST made great strides over the last years. It was applied for various testing types (i.e., usability testing, validation testing, etc.) and in various research contexts (i.e., education, corporate context, and experimental settings) and has shown to be a feasible and reliable tool in software testing. Despite these merits, CST research is mostly unstructured and often lacks a proper "terminology" and classification into existing software testing practices. Thus, there is no systematization of CST research.

To address this issue, I have conducted a literature review [13, 14]. By doing so, the contribution of this paper is twofold. At first, I will portray the research landscape and provide a comprehensive overview and systematization. Thereby, I synthesize the fields of crowdsourcing research and traditional software testing to provide a comprehensive overview of the application of CST. Second, I will provide in-depth knowledge to the growing body of literature for crowdsourcing in software engineering. For practitioners, the paper illustrates when and where CST has successfully been applied and where it gives an indication to when CST might be a feasible mechanism compared to other testing techniques such as traditional manual testing or test automation.

The remainder of the paper is structured as follows. The next section provides the conceptual background of crowdsourced software testing, as well as a definition of the term software testing. The third section explains the review approach, providing

URI: http://hdl.handle.net/10125/50404
ISBN: 978-0-9981331-1-9

insights into the methodology. Following that, I present descriptive findings before providing a systematization of CST research and present specific issues addressed in the papers. Thus, I discuss the results and lay out interesting topics for future research [15]. Finally, I present the papers' contributions for theory and practice before closing it with respective limitations and a conclusion.

2. Conceptual Background

2.1. Software Testing

Software testing is an integral component of software development and arguably the least understood part of the software development process [16]. There are several definitions of the activity itself and the purpose of software testing. Often times, software testing is defined as the "*the process of executing a program with the intent of finding errors*" [17]. Another very common definition is: *"Software testing is the process of executing a software system to determine whether it matches its specification and executes in its intended environment."* [16]

Thereby, it is important to note that the fact that the system is being executed is a characteristic that distinguishes software testing from code reviews, in which uncompiled source code is analyzed (sometimes referred to as "static testing") [18].

While this definition offers a very important confinement to code review, it is quite mechanistic and does not take account for the increasing emergence of digital software products as well as increasing expectations of users towards software.

In this vein, software testing can be seen as *the verification process for the assessment of software quality and a process for achieving that quality by supporting the interests of all stakeholders of an application, that is, end-users, developers, software designers, and software testers* [19, 20]. To achieve that goal, it becomes clear that different types of testing by various stakeholders at different times (during or subsequent to development) need to be performed [18]. Subsequently, this definition implies that there are different types and methods of testing and, indeed, there are numerous classifications depending on one's standpoint. Most of the following typologies are not mutually exclusive and sometimes used interchangeably.

One approach to classify software testing is based on the software's environment, which it is modeled at. That is, *unit testing*, *integration testing*, *and system testing*. In unit testing, the tester is only concerned with individual software components or a collection of components. In integration testing, multiple components are tested. Here, the focus of integration lies on the communication between the components. System testing refers to the testing of all components as deliverable products [16].

Another classification, which is not mutually exclusive to the prior one, is based on the "visibility" of the code for the tester and it divides white-*box-testing* (also referred to as structural or glass-box testing) and *black-box-testing*. (also referred to as end-user testing) [18]. In white-box testing, the tester performs testing through examination and knowledge of source code. The goal is to ensure that internal operation performs according to the software specification and all internal components have been adequately exercised [21]. In contrast, in black-box-testing, the tester does not know the source code and only inputs and outputs are visible. This form of testing covers not only functional aspects; it may also cover non-functional aspects such as performance, reliability, and security of a software. Thus, the main concern is the production of the correct output given specific inputs [18]. In this vein, there is also often the distinction between *functional testing* and *non-functional testing*.

The last pair of terms often used in software testing is *verification testin*g and *validation testing*. Again, these terms are not mutually exclusive to black-box or white-box-testing and are used depending on the stakeholder.

Verification testing is the evaluation of products in a given development activity to determine both correctness and consistency with respect to the products and standards provided as input to that specific activity. To sum up, verification ensures that "you have built the software right". In contrast, validation testing (also referred to as *acceptance testing*) ensures that the application as provided will fulfill its intended use. Thus, validation testing ensures that "you built the right software" [22].

Last, thus not part of the "traditional" testing terminology, there are several forms of testing that are concerned with neither verification or validation testing nor functional/non-functional testing. These testing methods examine the usability or quality of experience (QoE) of a software; respectively the entire service provided by an application. *Usability testing* is defined as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use" [23]. Subsequently, usability testing is a software test with the goal of improving it [24]. Quality of experience refers to the "degree of delight or annoyance of the user of an

application or service. It results from the fulfilment of his or her expectations with respect to the utility and/or enjoyment of the application or service in the light of the users personality and current state" [25]. *Quality of experience testing* refers to four levels that are the context, the user, the system, and the content level [26].

For this paper, I use the following classification of testing practices:

(1) *Functional (black-box-testing) and verification testing*: All testing activities (on every software level) that are concerned with functionality and the verification that the software is "built the right way"
(2) *Non-functional testing* (e.g., performance testing or vulnerability testing): All testing activities referring to the verification and validation of non-functional aspects, such as reliability, performance, security, etc.
(3) *Validation and acceptance testing*: All testing activities conducted by potential end users (or similar groups) to ensure that the "right software" is built.
(4) *Usability testing*: All testing activities concerned with the testing of the effectiveness, efficiency, and satisfaction in a specified context of use.
(5) *Quality of experience testing*: All testing activities that examine the overall experience with the software product (i.e., context, user, system, and content level).

2.2. Crowdsourced Software Testing (CST)

Crowdsourced software testing (CST) or crowdtesting is a specific application of crowdsourcing in the domain of software development. It refers to the outsourcing of software testing activities to the crowd. It grants access to a diverse pool of people who voluntarily test software in real environments using their own devices [11]. Depending on the type of testing (e.g., functional testing, usability testing), these tasks as well as the targeted crowds can be very diverse [27]. Usually, crowds are engaged via an episodic or collaborative IT structure [4]. However, CST incorporates parts of both. While testers usually participate in a "competition"-like bug hunt, the result of a crowdtest – a list of bugs – is a collaborative effort of the entire crowd. Further, testers have an elaborated online profile to display their skills and experience. Thus, they act as crowd workers in a virtual labor market (VLM) [8]. In this vein, CST can be considered as both tournament crowdsourcing and a VLM [28].

CST can be applied in a number of different types of testing, but research, so far, usually applied CST in dynamic testing scenarios where a written code is executed and examined by the crowd. Further, the crowd is usually concerned with output given by specific inputs since they do not know or see the source code.

To a certain degree, these characteristics match to the definition of *beta testing*, as well. Beta testing is defined as "*operational testing by potential and/or existing users/customers…to determine whether or not a component or system satisfies the user/customer needs and fits within the business processes. Beta testing is often employed as a form of external acceptance testing for commercial off-the-shelf software in order to acquire feedback from the market"* [29].

Thus, CST and beta testing both use external resources to test software under real-world conditions. Subsequently, the terms are sometimes used interchangeable; however, there are four substantial differences between CST and traditional beta testing:

(1) *CST has an increased scope* compared to beta testing: With CST, it is possible to acquire testers without addressing the general public and let the crowd evaluate software mock-ups or designs, perform regression testing, perform even non-functional testing such as performance testing, and conduct verification and quality assurance tests right before the release. In contrast, beta testing usually functions as the final quality gate before software release.
(2) *CST is task-based*: Whereas beta testing usually has a strong explorative focus ("Use the software and report bugs"), crowdsourced software testing tasks are much more specific. Testers are asked to go on a user journey and test certain use cases or even conduct traditional test cases.
(3) *CST users are trained and have an incentive to report bugs*: In beta testing, testers usually do not receive a reward. In CST, testers frequently receive monetary rewards (often on a per bug basis). Sometimes, the payment is on a first-come, first-served basis, so that testers have a strong incentive to be fast and to train themselves to properly report bugs.
(4) *CST has tester-task matching:* In beta testing, the call for participation is rather open. With CST, companies are able to select testers from a large pool of individuals based on a variety of factors and characteristics (e.g., testing skills, language, devices, age, gender, etc.). Thus, the selection provides an efficient way to match tasks and testers, thereby ensuring that only suitable people test the software. [10]

3. Methodology

The literature review is an essential approach to conceptualize research areas and synthesize prior research [30]. Thus, there are very different approaches and goals to conduct a literature review ranging from purely qualitative (narrative literature review) to a meta-analysis where the goal is to provide support for a research topic by synthesizing and analyzing the quantitative results of numerous empirical studies [31].

The objective of this paper is to portray the landscape of CST as an emerging research area and provide a comprehensive overview over the work researchers have done so far. Further, the paper provides a classification into existing testing types on the one hand, while also providing insights regarding the research foci of the current body of literature on the other hand. Given the infancy of this research area, it is not the goal to examine any variables, correlations, or theories. Therefore, a descriptive review approach is the most suited for the current state of research in this area. A descriptive literature review focuses on revealing reproducible and quantifiable results. It offers quantification concerning publication time, research methodology, and research outcomes [31]. While this descriptive approach is more of a traditional narrative literature review regarding its purpose, I follow a protocol-based approach to conduct the review. Thus, I explicitly describe the steps and processes for searching, selecting, and validating studies and summarizing results – characteristics of a systematic literature review [14].

The literature on CST is at the interface between information systems research, software engineering, and computer science. Since the topic is relatively new, it is unlikely that there are many publications in top journals in the respective fields already. Hence, I started by searching literature in the major databases for these research streams: IEEE Xplore, ACM digital library, the electronic library of the AIS (AISeL), and EBSCO Host Business Source Premier (BSP). I searched in each of these databases using the search strings “crowd*” AND “software*”, “crowd*" AND "testing”, “crowd” AND “usability”, as well as “crowd” AND “user experience”. I looked for the strings in a paper’s title, keywords, and abstracts. Since the keyword search was very broad and various databases have different formats, adaptations to the specific search strings were made. Overall, the search revealed more than 2.000 hits in the respective databases. The search was conducted in March 2017.

In a screening and selection step, I examined the papers regarding their titles and abstracts based on three inclusion criteria. These criteria included a research focus on crowdsourcing, software testing or a related field (i.e., usability testing, user experience/quality of experience testing), a not purely descriptive purpose of the paper, as well as full-text online availability of the paper in one of the mentioned databases. Thirty papers matched these criteria. Backward and forward searches were performed to identify more relevant literature [30]. I applied the same inclusion criteria and quality standards. Five papers were identified through forward and backward search.

4. Findings

4.1. Descriptive Findings

The literature review identified 35 relevant papers that had been published in 2016 or earlier. Figure 1 depicts the number of publications per year. It indicates that CST research is a rather new research field. There are very few publications before 2012. The field of research gained track in 2013. Since then, the number of publications stays at about the same level.

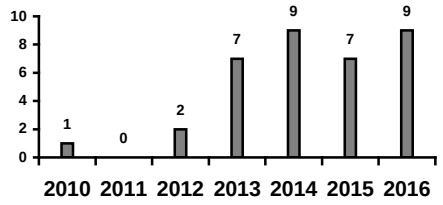

Figure 1: Number of Publications per Year

It is worth remarking that the papers were mainly presented at conferences (27 out of 35). Only six papers have been published in peer-reviewed journals (cf. figure 2). Further, only three contributions came from the field of information systems research. With 15 contributions, conferences affiliated to or organized by the Association for Computer Machinery (ACM) contribute the highest number of papers in the analysis. Eight publications are from the Institute of Electrical and Electronics Engineers (IEEE) conferences and two publications from joint IEEE/ACM conferences. To conclude, there is a tendency that research so far is mostly conducted in the field of software engineering and computer science.

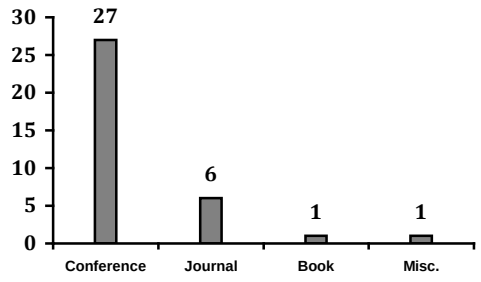

Figure 2: Number of Publications per Outlet

Given the fact that CST research is carried out mostly in computer science and software engineering, it is no surprise that different forms of experiments or the development of prototypes for a particular application purpose dominate the research methodologies used in literature. Experiments account for more than 50% (18 out of 35) of the applied research methods in the papers. Many experiments were conducted in the field using micro task platforms such as Amazon Mechanical Turk (e.g., [32]) or other platforms (e.g., [33], [34]) to confirm hypotheses. However, laboratory experiments are also amongst the research methods chosen (e.g., [35], [36]). Next to experiments, the development and evaluation of prototypes for different CST scenarios and purposes was applied in five papers (e.g., [37], [38]). Only five papers (e.g., [11], [39] conducted case studies related to CST in a real-world context. Last, certain aspects of CST were also examined with other research methods such as action research [40], focus group interviews [41], and quantitative survey research [42]. Figure 3 provides an overview over the research methods used in the identified papers.

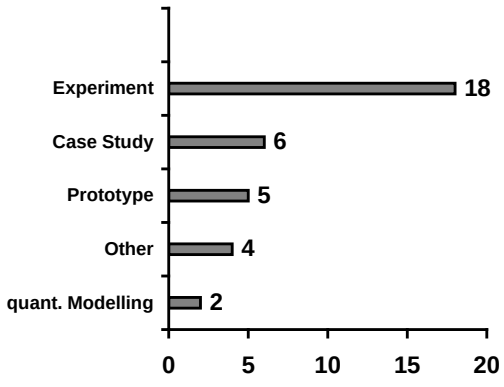

Figure 3: Research Methodology of Publications

4.2. Application of Crowdsourced Software Testing

The major research goal of this paper was to provide a comprehensive overview and classification of CST research in existing types of testing. Hence, the paper uses the definition of testing derived in section two including the classification of testing types. Then the identified literature according to the type of testing that was conducted or investigated in the study were grouped. However, there were papers investigating CST from a rather conceptual perspective, thus, they were not classified. Besides, a paper can be in more than one category if it investigates multiple test types. Figure 4 depicts the percentages of papers classified per testing type.

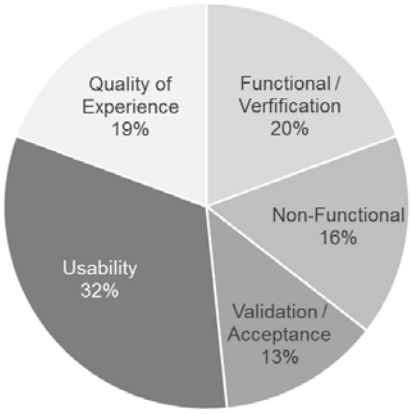

Figure 4: Classification per Testing Type

While there tends to be the perception (especially in the industry) that CST is solely used for usability and acceptance testing (end-user testing), research shows that this statement is not true, at least for research publications. Validation and usability testing account only for about 45% of the testing types applied in the identified papers. Hence, there is a lot of research conducted in other areas of testing, for instance functional testing (20%).

It is also noteworthy that crowdsourced software testing can be applied for the testing of non-functional software aspects such as vulnerability, privacy, or performance. Last, quality of experience testing is also an established field of research that discovered crowdsourcing mechanisms as an interesting approach to conduct studies compared to the traditional laboratory setting. Table 1 depicts the references identified per type of testing.

Table 1: Crowdsourced Software Testing Research Classification

Type of Testing	Articles
Functional and Verification Testing	[3], [43], [44], [45], [46], [47], [48]
Non-Functional Testing	[32], [43], [44], (performance); [49] (vulnerability); [50] (privacy)
Validation and Acceptance Testing	[11], [34], [35], [36],
Usability Testing/ User Experience	[35],[41], [51], [52], [53], [54], [55], [56], [57]
Quality of Experience	[26], [58], [59], [60], [61], [62]

Research in *functional and verification testing* demonstrated that even complex testing tasks such as the verification of cross-browser issues [46] or the reproduction of context-sensitive app crashes [45] are possible to be tested with the crowd. In this vein, also *non-functional testing* such as performance testing [32] is possible. Other research deals with the base condition of the test itself. It was found that time constraints actually improve test performance [47] and that explorative testing is more effective in terms of bug detection than traditional test case-based testing [63]. However, one issue identified relates to the number of test reports test managers receive when conducting CST [10]. Hence, researchers developed approaches to effectively prioritize [44] and classify [48] test reports and support automation techniques [50] to make CST more scalable. Although, this is particularly interesting for functional testing, it is transferable to all of the following types of crowdsourced testing.

As for *validation and acceptance testing*, two in-depth case studies that compare CST to traditional in-house testing provide deep insights. The cases unveil that CST delivers comparable quality while being more flexible and CST provides other valuable insights such as a very good documentation and additional suggestions to improve the software [11]. Other scholars looked at the task design and the expertise of the crowd testing the software. First results indicate that it is not always necessary to have "experts" testing the software, especially regarding validation and usability testing [34].

Usability testing is somewhat the natural habitat for CST. Research in this area has shown that usability testing with a crowd is feasible, produces reliable and high quality results [11, 56], is cost efficient [53, 56], and can help reduce critical usability testing obstacles such as resource constraints, limited understanding of the usability concept, and resistance to the adoption of usability practices [51]. In addition to that, scholars have developed several workflows and corresponding tools to use crowdsourced usability testing [35, 54, 55], including machine learning approaches to group similar responses and filter meaningless submissions to reduce workload for test managers [64].

For *quality of experience testing* the concept of crowdsourcing is a relatively new phenomenon. Thus, scholars predominantly focused on feasibility and best practices for crowdsourced quality of experience testing [26] to provide recommendations to conduct such tests [59]. Overall, the studies suggest that crowdsourcing is a reliable alternative to traditional QoE approaches. There are also first attempts to provide in-depth knowledge regarding conditions, such as test conditions [61] and the provision of an evaluation framework [62].

5. Discussion and Future Research

The research of crowdsourced software testing is at an early stage. This is not just reflected by the publication dates and the fact that most papers have been published at conferences. In fact, research is dominated by experimentation and the application for different scenarios und purposes, rather than attempting to conceptualize the topic. This experimentation, however, presents positive results for all described testing types. The main take away is that at least on an experimental and rather specific level, CST seems to be a promising solution – in terms of feasibility or quality and cost effectiveness. Thus, the results clearly substantiate the fact that "crowdsourced software testing is not beta testing". However, the lack of application in "real-world" scenarios becomes apparent; only six papers examined CST in a real-world setting. Accordingly, to gain more relevance and explore the topic further, qualitative case studies and a conceptualization in a real-world organizational context are necessary. First studies attempted to conduct case studies within a real-world context and under real-world conditions including corresponding restrictions and constraints [11, 53, 58].

Overall, the focus of the reviewed papers is of rather technical nature. Accordingly, I attempted to identify existing research foci. Thus, I detected four overarching topics within the papers' research (cf. Table 2).

Table 2: Summary of Research Foci

Research Focus	Description
Application	+ Application in different scenarios provides in-depth knowledge - Limited generalization and conceptualization of the topic
Performance Factors	+ identification of important performance factors (e.g., crowd composition [34], process guidance [36]) - No comprehensive overview or conceptualization
Prototypes	+ Optimization/improvement of process steps and workflows - Underlying design principles are not unveiled
Evaluation	+ increase comparability to other test approaches - lack of robustness of results (only few studies)

Primarily, the majority of the papers *(1) applied* CST in different scenarios. Those papers are valuable contributions. By providing in-depth descriptions and an analysis of the results, they contribute to knowledge, for instance by providing best practices [59] and demonstrate that CST, in fact, is applicable and feasible. However, the studies have oftentimes been conducted in specific contexts and only provide limited generalization and conceptualization of the topic. Thus, avenues for future research are to provide conceptualization and real-world context, especially for organizations that intend to use CST.

So far, little is known regarding the *(2) factors influencing the performance* of crowdsourced software testing. Scholars made great strides and this literature review identified various papers investigating potentially important factors in the process, such as crowd composition [34], guidance throughout the process [36], as well as time constraints [47]. However, there is no comprehensive overview or conceptualization regarding factors influencing the performance of CST.

Most papers are published in IEEE or ACM affiliated conferences, hence it is not of surprise that many of the papers *(3) develop prototypes* (e.g., [49], [37]) or attempt to optimize parts of the crowdsourcing process with algorithms (e.g., test report prioritization [48]) to increase the feasibility and cost efficiency of CST. However, these solutions are practical examples and the underlying design principles remain unknown. Scholars should attempt to identify the underlying process steps and design principles to provide a conceptualization of the process as well as design guidelines.

While there are first attempts to *(4) evaluate CST* and compare it to traditional testing approaches such as in-house testing with test experts [11] that identified favorable scenarios for the application of CST, there is a clear need to extend the evaluation. Much more work is needed to achieve a certain robustness of results and thus be able to compare crowdsourced software testing with testing by test experts, traditional lab usability testing, outsourced testing, or even test automation.

6. Contribution

This paper investigates the use of crowdsourced software testing by conducting a literature review and thereby delivering two main contributions.

First, the main objective of a descriptive literature review is the description of the existing state-of-the art. Hence, the paper provides a comprehensive overview of the fields of application in CST so far, clustered by the type of testing applied in the respective paper. While there is often the perception that CST is only applicable in end-user testing scenarios, the literature review revealed that this is not entirely true. There are many examples of other papers demonstrating that CST is, in fact, applicable even for complex testing tasks such as the functional testing of cross-browser issues [33] or even delicate testing such as vulnerability testing [49]. This in-depth knowledge and synthesis contributes to the growing body of literature of crowdsourcing for software engineering. This paper might serve as an example to conduct literature reviews in other areas of software engineering in a similar fashion to create knowledge from a bottom-up research approach across related disciplines as a first step to develop a theory or systematization of crowdsourcing in this particular area.

Second, by developing the classification and clustering the research papers in the according testing types, the paper synthesizes the research of crowdsourcing in software testing and traditional software testing research and practices. This will help scholars of both research fields to gain better understanding of the phenomenon of crowdsourcing on the one hand, and conduct research with CST to better position their research in the domain of software testing on the other hand.

On top of that, the literature review identified manifold papers that have effectively shown that very complex tasks, such as non-functional testing or quality of experience testing can effectively be crowdsourced. While the area of crowdsourcing

simple tasks is well explored, there is still a lot of research necessary to understand crowdsourcing of complex tasks.

For practitioners, this literature review illustrates the areas in which crowdsourced software testing has successfully been applied and gives an indication to when CST might be a feasible sourcing mechanism compared to other testing techniques such as traditional manual testing or test automation.

7. Limitations

As with every literature review, this review faces some limitations. These are the restricted scope of the literature review, the selection of the included papers, and the extraction of the contained information. Naturally, there is always a trade-of between completeness and practicality in a literature review [13].

Certainly, the choice of databases is one limitation. However, since the topic is at the interface between software engineering and information systems, the review did cover the most relevant databases. Second, the selection of the included literature is influenced by interpretation. That means the selection has not been standardized. However, I based the selection on objective criteria. Last, since the review process included the selection of the found papers only based on their title, keywords, and abstracts, the information were limited. Similar to the selection of databases, objectivity of the findings might be challenged by the extraction of the included information. To achieve robust results, the established categories for the data extraction and the results were discussed with other researchers from the respective fields.

8. Conclusion

In this paper, I examined the state-of-the-art in crowdsourced software testing research and presented a comprehensive overview of research in that area. While research in crowdsourced software testing made great strides in recent years, it is mostly unstructured and not linked to traditional software testing practice and terminology. On top of that, results are not integrated and often without a real-world context that could effectively determine the applicability in an organizational context. By conducting a literature review, this paper delivered two major contributions. First, the paper provides a comprehensive overview and synthetization of findings in CST-research and presents a classification into software testing types. Second, the paper synthesizes the research of scholars from crowdsourcing research in software testing and traditional software testing practices and thereby helps scholars to better explain and position their research.

9. References

[1] D. Huizinga and A. Kolawa, *Automated defect prevention: best practices in software management*: John Wiley & Sons, 2007.

[2] D. M. Rafi, K. R. K. Moses, K. Petersen, and M. V. Mäntylä, "Benefits and limitations of automated software testing: Systematic literature review and practitioner survey," in *Proceedings of the 7th International Workshop on Automation of Software Test*, 2012, pp. 36-42.

[3] E. Dolstra, R. Vliegendhart, and J. Pouwelse, "Crowdsourcing GUI Tests," presented at the International Conference on Software Testing, Verification and Validation (ICST 2013), 2013.

[4] J. Prpic and P. Shukla, "Crowd science: Measurements, models, and methods," in *Hawaii International Conference on System Sciences (HICSS 2016)*, 2016, pp. 4365-4374.

[5] A. Afuah and C. L. Tucci, "Crowdsourcing as a solution to distant search," *Academy of Management Review*, vol. 37, pp. 355-375, 2012.

[6] L. B. Jeppesen and K. R. Lakhani, "Marginality and Problem-Solving Effectiveness in Broadcast Search," *Organization Science*, vol. 21, pp. 1016-1033, 2010.

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

[8] D. Durward, I. Blohm, and J. M. Leimeister, "Crowd Work," *Business & Information Systems Engineering*, vol. 58, pp. 281–286, 2016.

[9] T. D. LaToza and A. van der Hoek, "Crowdsourcing in Software Engineering: Models, Motivations, and Challenges," *IEEE Software*, vol. 33, pp. 74-80, 2016.

[10] N. Leicht, I. Blohm, and J. M. Leimeister, "Leveraging the Power of the Crowd for Software Testing," *IEEE Software*, vol. 34, pp. 62 - 69, 2017.

[11] N. Leicht, N. Knop, I. Blohm, C. Müller-Bloch, and J. M. Leimeister, "When is Crowdsourcing advantageous? The Case of Crowdsourced Software Testing," presented at the European Conference on Information Systems (ECIS 2016), Istanbul, Turkey, 2016.

[12] E. Raymond, "The cathedral and the bazaar," *Knowledge, Technology & Policy*, vol. 12, pp. 23-49, 1999.

[13] J. vom Brocke, A. Simons, K. Riemer, B. Niehaves, R. Plattfaut, and A. Cleven, "Standing on the shoulders of giants: challenges and recommendations of literature search in

information systems research," *Communications of the Association for Information Systems*, vol. 37, pp. 205-224, 2015.
[14] S. K. Boell and D. Cecez-Kecmanovic, "On being 'systematic'in literature reviews in IS," *Journal of Information Technology*, vol. 30, pp. 161-173, 2015.
[15] F. Rowe, "What literature review is not: diversity, boundaries and recommendations," *European Journal of Information Systems*, vol. 23, pp. 241-255, 2014.
[16] J. A. Whittaker, "What is software testing? And why is it so hard?," *IEEE Software*, vol. 17, pp. 70-79, 2000.
[17] G. J. Myers, C. Sandler, and T. Badgett, *The art of software testing*: John Wiley & Sons, 2011.
[18] R. F. Roggio, J. S. Gordon, and J. R. Comer, "Taxonomy of Common Software Testing Terminology: Framework for Key Software Engineering Testing Concepts," *Journal of Information Systems Applied Research*, vol. 7, pp. 4-12, 2014.
[19] A. Bertolino, "Software testing research: Achievements, challenges, dreams," in *Future of Software Engineering (2007)*, 2007, pp. 85-103.
[20] K. Naik and P. Tripathy, *Software testing and quality assurance: theory and practice*: John Wiley & Sons, 2011.
[21] R. S. Pressman, *Software engineering: a practitioner's approach*: Palgrave Macmillan, 2005.
[22] R. D. Stutzke, *Estimating software-intensive systems: projects, products, and processes*: Pearson Education, 2005.
[23] ISO 9241-11, *Ergonomics Requirements for Office with Visual Display Terminals (VDTs)*. Geneva: International Organization for Standardization, 1998.
[24] J. S. Dumas and J. Redish, *A practical guide to usability testing*: Intellect Books, 1999.
[25] K. Brunnström, S. A. Beker, K. De Moor, A. Dooms, S. Egger, M.-N. Garcia, T. Hossfeld, S. Jumisko-Pyykkö, C. Keimel, and M.-C. Larabi, "Qualinet white paper on definitions of quality of experience," ed, 2013.
[26] T. Hoßfeld, C. Keimel, M. Hirth, B. Gardlo, J. Habigt, K. Diepold, and P. Tran-Gia, "Best practices for QoE crowdtesting: QoE assessment with crowdsourcing," *IEEE Transactions on Multimedia*, vol. 16, pp. 541-558, 2014.
[27] K.-J. Stol and B. Fitzgerald, "Two's company, three's a crowd: a case study of crowdsourcing software development," presented at the International Conference on Software Engineering (ICSE 2014), 2014.
[28] J. Prpić, A. Taeihagh, and J. Melton, "The fundamentals of policy crowdsourcing," *Policy & Internet*, vol. 7, pp. 340-361, 2015.
[29] *Standard Glossary of Terms Used in Software Testing* vol. v.3.1: International Software Testing Qualifications Board (IQSTB).
[30] J. Webster and R. T. Watson, "Analyzing the past to prepare for the future: Writing a literature review," *MIS Quarterly*, vol. 26, p. 3, 2002.
[31] W. R. King and J. He, "Understanding the role and methods of meta-analysis in IS research," *Communications of the Association for Information Systems*, vol. 16, p. 32, 2005.
[32] B. Taylor, A. K. Dey, D. Siewiorek, and A. Smailagic, "Using Crowd Sourcing to Measure the Effects of System Response Delays on User Engagement," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 2016, pp. 4413-4422.
[33] A. J. Ko and P. K. Chilana, "How power users help and hinder open bug reporting," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2010, pp. 1665-1674.
[34] N. Leicht, M. Rhyn, and G. Hansbauer, "Can Laymen Outperform Experts? The Effects of User Expertise and Task Design in Crowdsourced Software Testing," presented at the European Conference on Information Systems (ECIS 2016), Istanbul, Turkey, 2016.
[35] M. Gordon, "Web accessibility evaluation with the crowd: using glance to rapidly code user testing video," in *Proceedings of the 16th International ACM SIGACCESS Conference on Computers & Accessibility*, 2014, pp. 339-340.
[36] Y.-H. Tung and S.-S. Tseng, "A novel approach to collaborative testing in a crowdsourcing environment," *Journal of Systems and Software*, vol. 86, pp. 2143-2153, 2013.
[37] M. Yan, H. Sun, and X. Liu, "iTest: testing software with mobile crowdsourcing," in *Proceedings of the 1st International Workshop on Crowd-based Software Development Methods and Technologies*, 2014, pp. 19-24.
[38] X. Zhang, Z. Chen, C. Fang, and Z. Liu, "Guiding the crowds for Android testing," in *Proceedings of the 38th International Conference on Software Engineering Companion*, 2016, pp. 752-753.
[39] S. Zogaj, U. Bretschneider, and J. M. Leimeister, "Managing crowdsourced software testing: a case study based insight on the challenges of a crowdsourcing intermediary," *Journal of Business Economics*, vol. 84, pp. 375-405, 2014.
[40] N. Leicht, I. Blohm, and J. M. Leimeister, "How to Systematically Conduct Crowdsourced Software Testing? Insights from an Action Research Project," presented at the International Conference on Information Systems (ICIS 2016), Dublin, Ireland, 2016.
[41] N. Sherief, N. Jiang, M. Hosseini, K. Phalp, and R. Ali, "Crowdsourcing software evaluation," presented at the International Conference on Evaluation and Assessment in Software Engineering, 2014.
[42] F. Guaiani and H. Muccini, "Crowd and Laboratory Testing, Can They Co-exist? An

Exploratory Study," in *International Workshop on CrowdSourcing in Software Engineering (CSI-SE 2015)*, 2015, pp. 32-37.

[43] Z. Chen and B. Luo, "Quasi-crowdsourcing testing for educational projects," presented at the International Conference on Software Engineering (ICSE 2014), 2014.

[44] Y. Feng, Z. Chen, J. A. Jones, C. Fang, and B. Xu, "Test report prioritization to assist crowdsourced testing," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 2015, pp. 225-236.

[45] M. Gómez, R. Rouvoy, B. Adams, and L. Seinturier, "Reproducing context-sensitive crashes of mobile apps using crowdsourced monitoring," in *Proceedings of the International Workshop on Mobile Software Engineering and Systems*, 2016, pp. 88-99.

[46] M. He, H. Tang, G. Wu, J. Wei, and H. Zhong, "A Crowdsourcing framework for Detecting Cross-Browser Issues in Web Application," in *Proceedings of the 7th Asia-Pacific Symposium on Internetware*, 2015, pp. 239-242.

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

[48] J. Wang, Q. Cui, Q. Wang, and S. Wang, "Towards effectively test report classification to assist crowdsourced testing," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2016.

[49] H.-J. Su and J.-Y. Pan, "Crowdsourcing platform for collaboration management in vulnerability verification," presented at the Asia-Pacific Network Operations and Management Symposium (APNOMS 2016), 2016.

[50] S. Amini, J. Lin, J. Hong, J. Lindqvist, and J. Zhang, "Towards scalable evaluation of mobile applications through crowdsourcing and automation," presented at the CMU-CyLab-12-006, Carnegie Mellon University, 2012.

[51] A. Bruun and J. Stage, "New approaches to usability evaluation in software development: Barefoot and crowdsourcing," *Journal of Systems and Software*, vol. 105, pp. 40-53, 2015.

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

[53] V. H. Gomide, P. A. Valle, J. O. Ferreira, J. R. Barbosa, A. F. da Rocha, and T. Barbosa, "Affective crowdsourcing applied to usability testing," *International Journal of Computer Scienceand Information Technologies*, vol. 5, pp. 575-579, 2014.

[54] H. He, Z. Ma, H. Chen, and W. Shao, "How the crowd impacts commercial applications: A user-oriented approach," presented at the International Workshop on Crowd-based Software Development Methods and Technologies, 2014.

[55] M. Nebeling, M. Speicher, and M. C. Norrie, "Crowdstudy: General toolkit for crowdsourced evaluation of web interfaces," in *Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems*, 2013, pp. 255-264.

[56] C. Schneider and T. Cheung, "The Power of the Crowd: Performing Usability Testing Using an On-Demand Workforce," in *Information Systems Development*, ed: Springer, 2013, pp. 551-560.

[57] R. Vliegendhart, E. Dolstra, and J. Pouwelse, "Crowdsourced user interface testing for multimedia applications," in *Proceedings of the ACM multimedia 2012 workshop on Crowdsourcing for multimedia*, 2012, pp. 21-22.

[58] B. Gardlo, S. Egger, M. Seufert, and R. Schatz, "Crowdsourcing 2.0: Enhancing execution speed and reliability of web-based QoE testing," in *IEEE International Conference on Communications (ICC 2014)*, 2014, pp. 1070-1075.

[59] T. Hoßfeld and J. Redi, "Journey through the crowd: Best practices and recommendations for crowdsourced QoE," presented at the Seventh International Workshop on Quality of Multimedia Experience (QoMEX), 2015.

[60] R. K. Mok, W. Li, and R. K. Chang, "Detecting low-quality crowdtesting workers," in *International Symposium on Quality of Service (IWQoS 2015)*, 2015, pp. 201-206.

[61] M. Seufert, O. Zach, T. Hoßfeld, M. Slanina, and P. Tran-Gia, "Impact of test condition selection in adaptive crowdsourcing studies on subjective quality," presented at the International Conference on Quality of Multimedia Experience (QoMEX 2016), 2016.

[62] Q. Xu, J. Xiong, Q. Huang, and Y. Yao, "Robust evaluation for quality of experience in crowdsourcing," in *Proceedings of the 21st ACM international conference on Multimedia*, 2013, pp. 43-52.

[63] W. Afzal, A. N. Ghazi, J. Itkonen, R. Torkar, A. Andrews, and K. Bhatti, "An experiment on the effectiveness and efficiency of exploratory testing," *Empirical Software Engineering*, vol. 20, pp. 844-878, 2015.

[64] M. Rhyn and I. Blohm, "A Machine Learning Approach for Classifying Textual Data in Crowdsourcing," presented at the International Conference on Wirtschaftsinformatik (WI2017), St.Gallen, Switzerland, 2017.