A Review of Trending Crowdsourcing Topics in Software Engineering Highlighting Mobile Crowdsourcing and AI Utilization

Mohammed Alghasham, Mousa Alzakan, Mohammed Al-Hagery
Department of Computer Science, College of Computer, Qassim University, Buraydah 51452, Saudi Arabia

***Abstract*—Today's modern technologies and requirements make the utilization of crowdsourcing more viable and applicable. It is one of the problem-solving models that can be used in various domains to reduce costs and time. It is also an excellent way to find new and different ideas and solutions. This paper studies the use of crowdsourcing in software engineering and reveals adequate details to highlight its significance. A few recent literature reviews have been published to address specific topics or study general attributes of papers in crowdsourced software engineering. This paper, however, explores all recent publications related to software and crowdsourcing to find the trends and highlight mobile and AI usage in software crowdsourcing. The findings of this paper show that most research papers are in the areas of software management and software verification and validation. The results also reveal that machine learning and data mining techniques are predominant in software management crowdsourcing and software verification and validation. Furthermore, this study shows that the methods and techniques used in general crowdsourcing apply to mobile crowdsourcing except in mobile testing, where there is a need for clustering and prioritization of test reports.**

Keywords*—Software engineering; crowdsourcing; mobile crowdsourcing; software management; software verification and validation

I. Introduction

The word engineering in the field of software was inspired by the field of architecture engineering, where the design and building go through defined steps, even though the software has different characteristics. For example, a step in designing a building, for example, should take many considerations, such as budget, before sketching the design of the building. Similarly, when building software, the first step should not be coding the software, especially in large software and systems. The globalization of the current world forced both fields to adopt and use outsourcing, dispatch part of the software or building steps/processes to another company, to compete and evolve, especially when they lack time, workers, expertise, or other reasons. Consequently, the availability of the Internet to a tremendous number of users with various backgrounds and expertise adds more opportunities and challenges to the current working process, which leads to the use of heterogeneous users of the Internet in the working process for both fields. This new methodology was later called crowdsourced and defined by [1] in 2006, and also, different terminologies could refer to the same methodology as provided by [2].

In the crowdsourcing era, in its early days, it has been preliminarily defined as a problem-solving model [3], and recently, it has been reviewed in a wide range of domains [4]; software engineering, in between, has gained a considerable share in this emerging field [5], [6]. The notion of a problem-solving model for crowdsourcing is suggested by [3] for various applications while providing model examples and denoting contentious points such as crowds diversity, crowds exploitations, and intellectual labor, as well as other topics. A recent comprehensive literature review of all various fields where crowdsourcing has been utilized is authored by [4], which is the first literature to investigate all possible domains related to crowdsourcing. As crowdsourcing has been adopted in various domains, it has become an important topic. One of these domains is software engineering, which started with two publications in 2008 and a total of 509 publications at the end of 2020 [6]. For example, one publication [5] shows the possibility of engaging in empirical studies with crowdsourcing and presents the lesson learned to others for a successful one. Nevertheless, crowdsourcing is a promising technique employed in uncountable areas of SE regardless of other fields, and it is still ongoing research.

A number of recent literature reviews have been published that address crowdsourcing in software engineering. The study of the relationships for both co-authors and citations through social network analysis without considering the contextual content of papers is presented by [6]. Despite that, the research provides cohesive and extensive relationships and connectivity regarding basic publication attributes, such as authors' locations. The paper researches and reviews all of the publications until 2020. Another review [7] provides a baseline understanding of microtasks and explores previous research within crowdsourcing while listing and verifying microtask activities and their categories, which could be helpful for researchers and platforms. For integrating agile development methodology with crowdsourcing, a literature review is carried out to specify challenges and summarize them into five categories [8]. An overview of key processes and platforms, as well as other matters, to facilitate the adaptation of CSSD by organizations and highlight obstacles that make organizations reluctant to recognize CSSD is conducted by [9].

This paper is structured as follows: in this section, an introduction and motivation to crowdsourced software engineering is provided. Section II discusses the research methodology, research questions, and how the research is conducted. Then, Section III describes the literature outcomes, which are divided into areas related to software engineering. In Section IV, the research questions are answered by listing and discussing the findings. The conclusion of this literature review is in Section

V. Finally, suggestions for future work are in Section VI.

II. Research Methodology

The research methodology is an essential and well-defined step in literature review papers. Hence, this section presents the research questions and discusses the process for searching and selecting research papers and extracting and approaching the data.

A. Research Questions

This section provides answers to the following three listed questions:

- **RQ1: What are the directions and trends in crowdsourced software engineering?** The aim is to investigate and analyze the recent topics in crowdsourced software engineering.
- **RQ2: Did the papers focus on mobile crowdsourcing? Can general crowdsourced software engineering methods be used in mobile crowdsourcing?** The aim is to examine mobile crowdsourcing and conventional methods within mobile software engineering crowdsourcing.
- **RQ3: Did the papers in the review use AI? What type of AI did the papers use? In what areas did the papers use AI?** The aim is to explore the algorithms that the literature uses and in which areas of software engineering they are employed.

B. Conducting the Research

The first step in each research topic is to choose relevant keywords to find all relevant papers. Unrelated keywords could lead to small numbers of papers without any further hope of obtaining additional suitable research papers. Subsequently, gradually, more keywords are added, and the list of all keywords that are used within advanced search queries is as follows:

- ("Crowdsourcing" OR "crowdsourced" OR "crowd" OR "crowdsource") as (CrowdKeywords)

The first query, (CrowdKeywords), is used in conjunction with the following ones:

- (CrowdKeywords) AND ("Software Engineering")
- (CrowdKeywords) AND ("Software Development")
- (CrowdKeywords) AND ("Software Design")
- (CrowdKeywords) AND ("Requirements" or "crowdRE")
- (CrowdKeywords) AND ("Software" or "testing" or "test" or "defect")

As shown in Fig. 1, the search is first established through scholarly search engines online, without engaging in manual search activities, such as printed journals, and excluding books and thesis. There are a large number of databases. The papers are collected from IEEE, ACM, Science Direct, SpringerLink, Wiley Online Library, MDPI, AIMSciences, and Airiti. Additionally, this review focuses solely on recent papers and contributions, so any research paper published before 2022 is filtered out. After the preliminary collection of more than 100 publications, stored in a reference manager software, by reading just titles and keywords, all the collection papers are validated against the following criteria:

1) Papers must be very recent, and there are no duplications.
2) For quality control, remove preprinted or publications that are not peer-reviewed.
3) To ensure the relevance of collected publications, at least two authors review each paper's abstract to check whether the paper is related to the scope of the research or not.

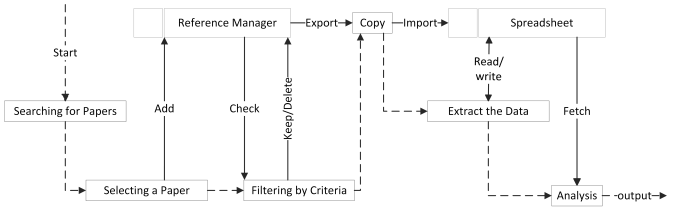

Fig. 1. Research methodology.

Next, the remaining 41 papers are exported from the reference manager to a spreadsheet. The spreadsheet contains the main attributes of each research paper and other details related to this research, such as the paper title, type, library, authors, date, the software engineering area, subarea, keywords, aim, objectives, main points, and summary. When the paper is selected to be included in the spreadsheet file, all details need to be obtained and filled out in the spreadsheet. With the help of the reference manager software, the keywords are also exported instead of manually filling them out. Still, several papers' keywords are manually extracted, which indicates inconsistent standardization between databases. These keywords are processed differently, and a program is developed that is executed inside the spreadsheet to handle the data. The program simply fetched all keywords with their corresponding paper and then clustered them based on each keyword.

III. Literature Outcomes

In this section, the papers selected for review are discussed. This literature review is divided into five subsections based on the problems the research papers solve. The subsections are software management, software specification, software development, software verification and validation, and software evolution. Fig. 2 depicts these subsections. Furthermore, two subsections: software management and software verification and validation, are further divided into subsubsections.

A. Software Management

Since software management is an extensive topic, its topics are further divided into subsections according to what is discussed in the selected papers. These subsections are productivity and motivation, task and crowd worker recommendation, trust issues, task pricing, and project documentation. Fig. 3 shows the subsections of software management.

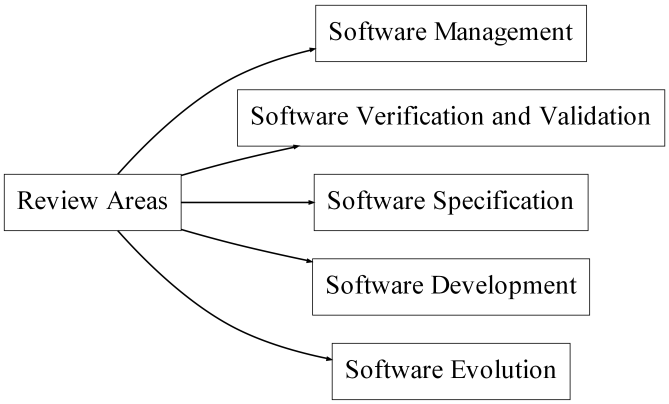

Fig. 2. Literature outcome main areas.

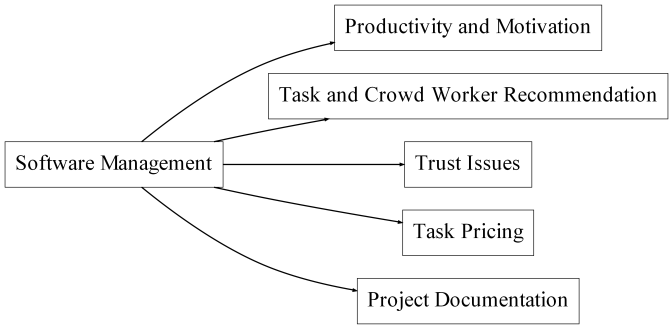

Fig. 3. Software management subareas.

1) Productivity and motivation: The authors of [10] investigate the collaboration of crowd workers in transient teams versus solo developers using a data-driven empirical study. The results show that the experience of teamwork affects the teams' performance in both the long and short term. In particular, the results show that individuals in each team can learn in the short term by getting support and sharing ideas with other members. Furthermore, in the long term, the members learn from working with other experienced team members. The paper contributes strategies for collaborative contest designers, platform operators, and crowd workers. These strategies include increasing the complexity of the contests, emphasizing virtual teams in contests, or reducing the number of total medals.

While [10] examines the effects of working solo as a developer or in teams on performance and learning, the authors of [11] study the effects of game elements that exist in several crowdsourcing platforms on individual contributors. Multiple crowdsourcing platforms use game elements such as contests, leaderboards, and rankings with the intent to motivate crowd workers. The results show that there is a different effect on the performance and effort of workers depending on their abilities. Data from TopCoder shows that feedback has a positive effect on high-scoring workers and a negative effect on low-scoring individuals.

Unlike [10], [11], the writers of [12] define, explain, and justify the area and use of the microtasks programming approach, regarding functions programming, on the factors such as team size, the time needed for new developers, and the velocity of the whole project. Moreover, they have conducted an experiment to study the positive and negative sides without focusing on areas related to the design and maintenance of microtasks. The experiment has shown several advantages of adopting microtasks, especially for the short project schedule. Overall, this study establishes a path for software corporations and encourages them to use microtasks, which could lead to the involvement of external developers, crowdsourcing in particular.

2) Task and crowd worker recommendation: The authors of [13] state that crowd workers often choose their testing tasks from whatever is immediately available. This abrupt choice could lead to wasted time and effort for both the tester and the requester. Furthermore, it could lead the tester to the inability to discover and find the bugs in the assigned task. Thus, the authors of this paper propose a context-aware personalized crowdsourced software testing method for task recommendation named PTRec. This method uses contextual, historical data and the preferences of the tester to recommend the appropriate task. PTRec consists of two models that are able to extract 60 features automatically to help the tester choose a suitable task. The goal is to reduce wasted efforts and the number of unpaid tasks. PTRec also uses the random forest learner technique to find the proper testing task which matches the workers' interests and expertise. Tests on 2404 crowd workers and 636 tasks reveal that this approach has 82% precision and saves the efforts of exploring tasks by 81%.

In addition to [13], another research [14] has proposed a new recommendation model which considers users' preferences. This method is a capability-corrected long- and short-term attention network (CLEAN). It outperforms the existing traditional models. The model considers the gradual interest changes of workers and the constraints and skills needed to perform the tasks while incorporating the contextual data of tasks besides common attributes.

According to the authors of [15], existing studies use one-time recommendations based on the knowledge of the worker at the start of a new task. Furthermore, they assert that this recommendation has a popularity bias. In other words, the methods in these studies recommend almost all of the tasks to users with the highest experience. To remedy these issues: this study proposes iRec2.0, which is a context- and fairness-aware in-process crowd worker recommendation method. iRec2.0 achieves its stated goals by modeling the dynamic testing context, using a learning-based technique, and applying a multi-objective optimization component. The outcomes of the evaluations show that this method has the potential to divide the tasks fairly among the users, decrease the testing process time, and save costs.

In [16], the authors assert that the quantity and quality of the completed tasks are directly affected by task recommendation. The authors also state that previous task recommendation modules focus on one user only per task. Unlike [13], [15], the authors of [16] present a method to recommend tasks and coworkers for these tasks. This method operates on the user's performance history on previous tasks in combination with a social network component for the coworker recommendation. This method checks and informs users if it is best to complete this task independently or if they should seek help from a

friend in their social network. This method factorizes a user-task rating matrix to get the latent matrix. Then it applies a greedy method to select tasks for each user. The method calculates the intimacy data and the extroversion data of users with their friends in the social network to recommend coworkers. The results by the authors show that it outperforms the current existing algorithms.

Crowdsource contribution is open to every qualified person. This openness could pose a problem of malicious workers and malicious task submissions. To tackle these problems, the authors of [17] propose a method named Outlier Detection for Streaming Task Assignment. Its goal is to detect malicious crowd workers. This technique uses an evolving time series to model the arrival of workers and their task submissions. This framework has a novel method based on Generative Adversarial Network (GAN), which is socially aware and can work with time series. The authors also propose a novel method to train the loss functions of GANs with social awareness. It also has the capability to assign tasks to users similar to [13]–[16]. This method uses a greedy algorithm to improve the efficiency of the process of task assignment.

Whereas [13]–[17] focus on task assignment, [18] focuses on the reliability of recruits. In [18], the authors investigate four different crowdsourcing platforms and a computer science (CS) mailing list to determine the reliability of their recruits for empirical software development studies. The crowdsourcing platforms are Prolific, Appen, Clickworker, and Mechanical Turk. The authors' criteria of reliability are programming skills, privacy, and security attitudes. For the university CS students, the authors also consider self-efficacy. The results show that while university CS students rated themselves lower than other crowdsourcing participants in secure development and self-efficacy, 89% of them answered all programming skills correctly. Furthermore, the study shows that university CS students are the most cost-effective recruits.

Out of all the studies on task management in this paper, only one study highlights the need for coordination in the platform for doing tasks related to software design between designers and clients. Therefore, it identifies all potential coordination limitations encountered in the process and promotes coordination propositions with the help of a questionnaire. The feedback from participations through the questionnaire verifies the limitations and welcomes the purpose solution to alleviate the lack of coordination in platforms [19]. Moreover, another study proposes four steps to task flow from the beginning of constructing the task until the aggregation of the results, and this solution is evaluated in two ways. Also, the crowdsourcing platforms have the advantage of incorporating and implementing proposed approaches [20].

3) Task pricing: Is giving crowd workers the ability to choose their preferred incentive will result in better performance and solution quality? Is one type of incentive for all participants an optimal motivator? In [21], the authors of this paper empirically investigate the effects of giving the choice of reward to participants on the quality of the solution submitted. The results of this work show that when participants can choose their preferred incentive, they will spend more time on their assigned tasks and produce better-quality solutions in contrast to participants who offered one type of incentive. The results show the importance of having a flexible reward structure and allowing participants to select what matches their motives.

Both [21], [22] state that personalized pricing can yield better resulting tasks than common pricing, which is pricing with no personalization. The authors of [22], though, remark that personalized pricing is arduous to incorporate into some systems due to its complexity. Therefore, [22] investigate two schemes: personalized pricing per worker and common pricing with bonus payments after task completion to explore their impacts. The results show that with the proper bonuses, common pricing is close to an approximation of optimal personalized pricing.

The authors of [23] argue that instead of having a fixed price for tasks, there should be a dynamic system for pricing tasks to incentivize crowd workers. Accordingly, The authors of this paper propose a system called CrowdPricer, which gives, in addition to the base payment for the accomplished tasks, bonuses for completing tasks which is the recommended method by [22]. In addition, CrowdPricer increases the utility expectation of the requester to guarantee profits. To achieve its stated goals, CrowdPricer learns the effect of bonuses on the quality of the delivered tasks using deep time sequence modeling. The authors' experimentations using a crowdsourcing platform and simulations show that using CrowdPricer results in higher-quality task solutions and maximizes the utility of the requester.

4) Trust issues: Several critical and ordinary trust issues are raised in the crowdsourcing implementation. The authors [24] present these issues by conducting a survey with practitioners, listing nine critical issues as "deficient assistance to best practices", "malicious code", "lack of licensed software utilization", "loss of data", "network security risks", "quality of workers", "social attacks", "crowd legal action", and "loss of intellectual property". Additionally, the study results are validated via a focus group of four experts in academia. One of the issues of intellectual property is investigated in the context of testing reports by one paper [25]. It proposes a system for intellectual rights confirmation with the integration of blockchain and the implementation of other methods. This system would overcome problems, such as code plagiarism, and prevent unwanted modifications of data with the help of the blockchain decentralized methodology [25]. Another paper [26] indicates the need for implementing blockchain to diminish unwelcome behavior raised by centralized systems. However, it showed that the quality of previous works regarding keeping traceability or increasing privacy is less than required. Therefore, it proposes STPChain based on blockchain, which better preserves traceability and improves privacy, including preventing wrong actions.

5) Project documentation: The paper [27] studies the effect of using various types of media as a documentation type and focuses solely on instructional screencast documentation. As the popularity of this type has recently increased, especially for crowd-based content and new developers, the paper suggests a platform for this content. In addition, the platform could have essential functionalities such as making the content searchable and linking its content to other artifacts.

B. Software Specification

As there is a vague overlap between crowd-based requirements engineering (CrowdRE) and market research (MR) primarily caused by the incremental use of automation, one study explains and identifies them. Furthermore, after providing various scenarios and equivalent implementation of both CrowdRE and MR, the study implies the overall benefits of CrowdRE, which could be sufficient [28].

Specialized public web forums and general user stories format are a great way to elicit crowd opinions and experiences on many topics. The authors of [29] use Reddit forums to collect requirement engineering data by analyzing the discussion in the forums. In particular, the authors propose crowdsourcing requirement engineering by valuation argumentation (CrowdRE-VArg) to identify and prioritize issues, design changes, and new features and decide on the appropriate requirements. CrowdRE-Varg uses machine learning and natural language processing to analyze end-users supporting and attacking arguments in discussions from users' posts on Reddit. Their results show the validity of the approach for using Reddit as a platform for rational mining and the eliciting of opinions. Another paper [30] analyzes user stories and proposes CREUS, Crowd-based Requirements Elicitation with User Stories, as an iterative process practical design for conducting pull feedback after engaging in three case studies. It provides qualitative analysis of user stories or feedback as the main contribution besides quantitative results as usual case studies.

Software requirement engineering (RE) is a challenging process, and it requires the constant availability of the stakeholders, which is not guaranteed. To handle the problems of RE, the author of [31] proposes a conceptual framework that combines the crowdsource software development (CSSD) approach with the SCRUM software development approach. The framework collects the data from the crowd at large, which increases efficiency and reduces costs. This framework consists of four main layers designed to use the features of both approaches. The layers deal with document preparation, prioritization of tasks, planning, design, and retrospective meeting. On the other hand, [32] present specific challenges regarding requirements for a specific area. The authors show and discuss the needs of older well-being adults for intelligent assistance systems through crowd-based requirements engineering (crowd-RE). Also, it demonstrates the crowd-RE process and some challenges in this area.

One research [33] studies prototype validation. It develops a platform that uses the crowd to obtain feedback and validate the prototype iteratively before actual development. First, it conducts a design science study to address the vague of applying crowd-workers and prototype validation in platforms. Then, through the formed knowledge and implementation, it develops a platform that tackles the difficulties. Moreover, the study is valuable for building new or enhancing current mechanisms with the crowd-validation process.

C. Software Development

In [34], the authors aim to identify the percentages of vulnerabilities in code submitted by participants in code in competitive programming (CP) platforms. This paper focuses on data, 6.1 million submissions to be exact, from the CodeChef CP platform. The results show that 34.2% of submissions have software vulnerabilities. The authors did not find conclusive evidence to correlate the number of vulnerabilities with the leaderboard position of the participant. Furthermore, the study shows that participants do not follow secure coding practices, and even when a participant with perfect scores reattempts the task, the study shows there are no security improvements in the new submission. One way to mitigate these issues is to use crowdsourcing. The authors of [20] suggest that instead of using automated or manual anti-pattern detection methods, which consume time and lack certainty, the use of crowdsourcing and propose four steps for task flow.

D. Software Verification and Validation

On the subject of software verification and validation, the selected papers solve problems in software testing and usability, test report clustering and prioritization, and quality of defects reports. Fig. 4 shows the subareas of software verification and validation.

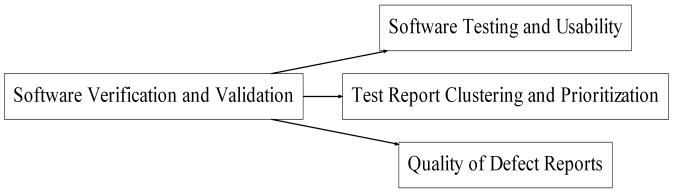

Fig. 4. Software verification and validation subareas.

1) Software testing and usability: After analyzing the problems that developers face in crowdsourced software development (CSD), the authors of [35] concluded that crowdsourcing is more fitting for software testing than software development. The authors state that the main advantages of crowdsourced software testing (CST) are reducing the time and cost of software testing. Furthermore, the authors express that better tools make developers work efficiently. Hence, the paper proposes a new testing program that incorporates crowdsourcing and open-source sharing techniques. An example of using crowdsourcing in testing is [36]. The authors of [36] did a comprehensive study to compare the cost and time of using novice crowd and expert heuristics in usability inspection. A single expert's heuristic usability inspection leads the novice crowd. The results show that, on average, both methods detect the same usability issues. However, the novice crowd method takes less time to identify the problems and costs less than the expert heuristic usability inspection.

To overcome the challenges of testing human-AI interactions and collaborations, the authors of [37] propose a Human-AI INtergarion Testing framework (HINT). HINT is a crowd-based framework that uses a humans-in-the-loop workflow to test AI-based experiences. This framework aims to solve the drawbacks of existing methods by simulating AI experiences that evolve over time, allowing rapid testing, providing early feedback during the development phase, and evaluating crowd workers and AI in offline testing. In addition, to overcome problems that exist in current testing implementation for IoTs, another research [38] develops a new crowdsourcing test system oriented toward the Internet of things with the integration of blockchain technologies as a potential solution. The system

consists of two modes, online and offline testing. The device or devices in the online mode, which are the main focus of interest, are real devices. In this case, testers engage and test the devices online with multi-thread technology implemented to allow concurrent testers, which is helpful, especially for limited hardware resources. The offline mode is to test the devices physically. Besides all security that the system provides, it contributes to a dependable online testing system for IoTs, especially with the lack of online testing for IoTs.

Regarding fault localization and exploiting the power of crowdsourcing, only one research [39] has investigated this area. It has a distinguished and unfamiliar approach for automated fault localization (AFL) in crowdsourcing software engineering by exploiting the solutions of all works and making them one set of referenced solutions. The main point is that when encountering fault statements, each statement in buggy programs is referenced and evaluated with an equivalent statement from other solutions.

2) Test report clustering and prioritization: The authors of [40] assert that an issue of the previous research papers dealing with the clustering of test reports is that the papers do not take into account the semantic connection between the screenshots and text in the analyzed test reports, which results in suboptimal results, especially in the deduplication of test reports. Therefore, This paper proposes a method using semi-supervised clustering using deep image understanding to analyze crowdsourced mobile application test reports. This method is SemCluster. SemCluster creates semantic binding rules from the semantic connection between screenshots and text descriptions in test reports. The results of this paper show that SemCluster outperforms the state-of-the-art method in six metrics of clustering results. Another approach [41] uses a fused features approach after obtaining text and screenshots features and then using common classification algorithms.

Liu et al. in [42] state that in addition to the large number of test reports produced by crowdsourcing that needs inspecting, one specific issue of mobile application test reports is that they have more screenshots than text descriptions of the tests. In addition to this paper, [40], [41], [43], [44] deal with these issues as well. To solve the issues of the number of reports and screenshots, the authors of [42] propose a novel method to understand text and images to cluster test reports. This method uses natural language processing to calculate the distance between reports. It also uses Spatial Pyramid Matching (SPM) to compute the similarity of the screenshots in the reports. The authors tested the method on 1400 screenshots and more than 1600 test reports from six industrial crowdsourced projects. Tests show that this method results in up to 37% improvement over the baseline in the average percentage of faults detected (APFD). Moreover, only the following paper [41] points out that existing automatic test report classification techniques are incompatible with crowdsourced mobile test reports as they contain incomplete texts, as well as the previously mentioned screenshots.

In [43], the authors propose and evaluate a method adapted from the prioritization of test reports in regression testing. This method sorts test reports in two phases. First, to process the text of the reports, this method uses natural language processing and word segmentation. Second, to prioritize the test reports, the paper uses a combination of a genetic algorithm, two greedy algorithms, and an adaptive random test case prioritization algorithm. It aims to make it easy and efficient for developers to check reports according to their priority. The results show that this method has promising performance in prioritizing the test reports, with an average percentage of faults detected (APFD) of more than 0.8.

The authors of [44] devise a new method called DivClass to prioritize test reports. DivClass combines diversity and classification strategies to order the reports for inspection. A feature of this method is that it handles duplicate test reports similar to the method proposed in [40], [42]–[44] in which they use natural language processing to analyze the test reports in one of their method steps. The next step in [44] is to build a similarity matrix using an asymmetric computation strategy. The final step consists of the previous two steps to prioritize the reports. The authors state that it reduces the number of tests to inspect, reducing the inspection cost. It also improves DivRisk, the state-of-the-art method, by 14.12% on average and has 0.8887 APFD.

3) Quality of defect reports: Three studies [45]–[47] provide ways to enhance the quality of the defect reports generated by usually non-expert crowds. The first research [45] studies the reports, which contain a good and bad description of bugs, submitted by crowdsourcing, usually non-professional testers. After it shows possible quality indicators, the paper proposes CTRQS as a framework to qualify test reports using analytical indicators based on dependency parsing. Therefore, in the end, just the reports that describe the defect better should be processed for localization and fixing. The second paper [46] attempts to generate more promising defect reports results by adding more than one tester participating together to find and report defects instead of one tester working alone. The result shows an enhancement regarding the quality of the final report by decreasing the invalid reported defects and increasing the report of difficulty defects. On the other hand, the third research [47] states the need for improving the crowd-workers instead of only establishing techniques for the quality of crowdsourced testing reports. Also, it indicates that enhancing the knowledge and abilities of testers from the beginning will provide better-quality reports. Therefore, this study proposes an assistant approach to suggest, guide, and educate crowds by exploiting Android's automated testing results.

In a different area of software quality, one research [48] suggests using crowdsourced workers, who meet minimum quality requirements, as a third party to evaluate and certify software meant for public users based on the available software documents and templates. Then, a quantitative quality score is assigned to the software-specific version, and the evaluation process is repeated after each considerable change in the current software version. Another area of research [49] claims that almost all previous techniques study the duplicated defects without false defects. Hence, this paper contributes by using duplicated defects to build a model which can provide a high estimation accuracy for valid defects. Furthermore, the model accuracy increases when the approach is applied to crowdsourced testing.

E. Software Evolution

In [50], Reis et al. use supervised machine learning methods with crowdsourced data collected over three years to identify code smells. The authors focus on Java code and three types of code smell, which are long methods, god classes, and feature envy. The data is collected from about a hundred teams, each team with an average of three members. The results of the papers prove the feasibility of crowdsemlling using supervised machine learning techniques applied to data collected from software developers (wisdom of the crowd). However, the authors state that further studies are needed to cover other types of code smells. The authors are currently developing an Ecplise IDE plugin which should simplify the crowdsourcing process. This plugin collects data about the code, identifies code smells, and gets the developer's opinion regarding the data detected by the plugin.

IV. Results and Discussion

RQ1: What are the directions and trends in crowdsourced software engineering? After collecting and analyzing the 41 papers published in 2022, This research reveals that the collected papers discuss the following five areas: software management, software specification, software development, software verification and validation, and software evolution. The most discussed area in crowdsourced software engineering is the area of software management. Out of the 41 papers, 17 papers deal with various software management problems. This review also determines that the least papers, one paper, are in the area of software evolution, even though it is an essential and costly activity in software engineering. Table I shows the areas, the count of the papers, and the paper selected in these areas.

TABLE I. Areas of the Selected Papers

Area	Selected	Papers
Software Management	17	[10], [11], [13]–[27]
Software Specification	6	[28]–[33]
Software Design & Implementation	2	[20], [34]
Software Verification & Validation	15	[35]–[49]
Software Evolution	1	[50]

Continuing the discussion on the directions and trends of the research in crowdsourced software engineering, in this review, both software management and software verification and validation are divided into subareas based on the collected papers. Table II shows the subareas of software management, the collected and selected papers, and the number of papers in each of the subareas. Software management has five subareas shown in Table II. A look at the areas of software management shows that most research papers in this literature review, which are 7 out of 17, are on task and crowd worker recommendations. There are three papers in each of the following fields productivity and motivation, task pricing, and trust issues in software. The least number of papers which is one study is on project documentation.

In the subarea of software verification and validation, there are 15 papers spanning three subareas. There exist five studies in each one of the three subareas. Table III shows the details of the subareas and papers of software verification and validation.

TABLE II. Subareas of Software Management

Subareas	Selected	Papers
Productivity & Motivation	3	[10]–[12]
Task & Worker Recommendation	7	[13]–[19]
Task Pricing	3	[21]–[23]
Trust Issues	3	[24]–[26]
Project Documentation	1	[27]

TABLE III. Subareas of Software Verification and Validation

Subareas	Selected	Papers
Testing & Usability	5	[35]–[39]
Test Report Clustering & Prioritization	5	[40]–[44]
Quality of Defect Reports	5	[45]–[49]

The results of the Tables I, II, and III lead us to conclude that the most trending topics in crowdsourced software engineering are two. First, the topics that deal with software management especially managing tasks, workers, and motivation. Second, all the subareas of software verification and validation. These subareas are software testing and usability, test report clustering and prioritization, and quality of defect reports.

Furthermore, an analysis of all keywords in the papers was conducted to investigate the trending topics from different viewpoints. The keywords are also used to see how diverse the papers are in the context of subtopics. As a result, the number of different keywords is more than 180 words from 41 papers. Table IV shows the keywords that appear in three papers or more, only the top eight keywords are listed, and the common keywords to the primary research topic are excluded, such as crowdsourcing, software engineering, and software. The results agree with the previous conclusion that software management and testing are the most dominant topics. Moreover, the selected papers in this review are more diverse, covering various subtopics, as there are 162 keywords that are unique and used only in one paper.

TABLE IV. Most Common Keywords that Appear in Papers

Keywords	Selected	Papers
Crowdsourced testing	9	[13], [15], [25], [41], [42], [44], [45], [47], [49]
Task Analysis	7	[13], [19], [42]–[44], [47], [49]
Testing	5	[13], [37], [42], [45], [47]
Computer Bugs	5	[13], [42], [44], [45], [47]
Software Testing	4	[43], [44], [46], [49]
Requirements Engineering	3	[28], [31], [32]
Mobile Applications	3	[42], [45], [47]
Software Quality	3	[20], [26], [50]

RQ2: Did the papers focus on mobile crowdsourcing? Can general crowdsourced software engineering methods be used in mobile crowdsourcing? Mobile development comes with its own set of problems, especially in software testing [40]–[44] describe these issues. In this review, there are ten papers that focus exclusively on mobile crowdsourcing, specifically mobile testing. The remaining papers, 31 papers,

focus on general software engineering crowdsourcing, which is applicable to mobile crowdsourcing as well. Table V lists the papers with their focus area. These results lead us to deduce that the crowdsourcing methods in general software engineering crowdsourcing are suitable for mobile crowdsourcing, except for mobile testing methods highlighted by [40]–[44].

TABLE V. PAPERS FOCUS

Focus	Selected	Papers
Mobile Crowdsourcing	10	[13], [15], [16], [40]–[45], [47]
General Crowdsourcing	31	[10]–[12], [14], [17]–[39], [46], [48]–[50]

RQ3: Did the papers in the review use AI? What type of AI did the papers use? In what areas did the papers use AI?

TABLE VI. AI IN PAPERS

AI Methods	Selected	Papers
ML & DM	14	[13]–[17], [23], [29], [30], [40]–[44], [50]
None	27	[10]–[12], [18]–[22], [24]–[28], [31]–[39], [45]–[49]

Yes, 14 papers use AI techniques, while 27 do not. Specifically, the papers use either Machine Learning (ML) or Data Mining (DM) methods. Table VI list the papers that use the AI methods. However, not all AI is applied to all the areas of crowdsourced software engineering. Table VII shows the areas and papers in which AI methods are employed. In Table VII, the results show that test report clustering and prioritization and task and worker recommendation use AI the most, with five papers each. Two papers in software specification use AI. Both task pricing and software evolution have one paper each that uses AI techniques.

TABLE VII. AI AREAS IN THE SELECTED PAPERS

Area	Selected	Papers
Test Report Clustering & Prioritization	5	[40]–[44]
Task & Worker Recommendation	5	[13]–[17]
Software Specification	2	[29], [30]
Task Pricing	1	[23]
Software Evolution	1	[50]

V. CONCLUSION

This literature review examines and studies the latest papers in crowdsourced software engineering to find the current trends and directions of the research literature. This paper focuses exclusively on all the publications of 2022 to get a clear and accurate picture of the crowdsourcing landscape. This review also answers a number of relevant and current questions. It answers whether the papers focus solely on mobile crowdsourcing and whether general crowdsourced software engineering methods apply to mobile crowdsourcing. Furthermore, it discusses the question of AI usage in the papers. In particular, this research checks whether the selected papers incorporate machine learning or data mining techniques into their proposed crowdsourcing solutions. The results of this literature review show that the largest number of contemporary research focuses on software management and software verification and validation. In mobile crowdsourcing, the results show that while general crowdsourcing methods work for most mobile crowdsourcing activities, mobile testing requires specific techniques to deal with the large number and the nature of tests. The results also show the papers that use machine learning and data mining methods to tackle specific crowdsourced software engineering areas.

VI. FUTURE WORK

One of the least discussed topics in software engineering crowdsourcing is secure software development. Crowd workers come from diverse places and have different programming and security backgrounds. Therefore, they will have various goals to achieve and different experiences. In this research, paper [34] shows the percentage of vulnerabilities in competitive programming platform submissions. The study did not find a connection between the leaderboard position of the participants and the number of vulnerabilities in their submitted code. Furthermore, even the resubmissions of full-scoring tasks did not have security improvements. These results beg the following research questions: Is it possible to have a crowdsourcing platform for secure software development? How to incentivize crowd workers to submit secure code? Can the incentivization techniques in [21]–[23] be used to encourage secure coding practices? Will it affect software verification and validation? These are all questions that need further research.

There are many possible areas of improvement in the quality of crowdsourcing that can be investigated. One of them is the minimum number of crowd workers engaging in one task. Each type of task requires a different number of workers, and in this way, the power and quality of crowd wisdom can be exploited in a cost-effective manner. Moreover, the crowds must be certified for the required type of tasks before joining crowdsourcing platforms. Certification will ensure higher quality workers. Alternatively, to encourage worker certification, certified workers can be paid more than non-certified ones. There could be several types of certifications depending on the type of platform or task. In addition, to our knowledge, there are no established crowdsourcing standards of best practices that ensure continuity and portability for both tasks and processes.

ACKNOWLEDGMENT

The researchers would like to thank the Deanship of Scientific Research, Qassim University, for funding the publication of this project.

REFERENCES

[1] J. Howe, "The rise of crowdsourcing," *Wired magazine*, vol. 14, no. 6, pp. 1–4, 2006, number: 6.

[2] A. Doan, R. Ramakrishnan, and A. Y. Halevy, "Crowdsourcing systems on the World-Wide Web," *Communications of the ACM*, vol. 54, no. 4, pp. 86–96, Apr. 2011. [Online]. Available: https://doi.org/10.1145/1924421.1924442

[3] D. C. Brabham, "Crowdsourcing as a Model for Problem Solving: An Introduction and Cases," *Convergence*, vol. 14, no. 1, pp. 75–90, Feb. 2008, publisher: SAGE Publications Ltd. [Online]. Available: https://doi.org/10.1177/1354856507084420

[4] N. Kasturi, S. G. Totad, and G. Ghosh, "Analysis on Potential Use of Crowdsourcing in Different Domain Using Metasynthesis," in *Emerging Technologies in Data Mining and Information Security*, ser. Lecture Notes in Networks and Systems, P. Dutta, S. Chakrabarti, A. Bhattacharya, S. Dutta, and V. Piuri, Eds. Singapore: Springer Nature, 2023, pp. 747–756.

[5] K. T. Stolee and S. Elbaum, "Exploring the use of crowdsourcing to support empirical studies in software engineering," in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '10. New York, NY, USA: Association for Computing Machinery, Sep. 2010, pp. 1–4. [Online]. Available: https://doi.org/10.1145/1852786.1852832

[6] A. Alabduljabbar and S. Alyahya, "Leveraging Social Network Analysis for Crowdsourced Software Engineering Research," *Applied Sciences*, vol. 12, no. 3, p. 1715, Jan. 2022, number: 3 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2076-3417/12/3/1715

[7] M. Zulfiqar, M. N. Malik, and H. H. Khan, "Microtasking Activities in Crowdsourced Software Development: A Systematic Literature Review," *IEEE Access*, vol. 10, pp. 24 721–24 737, 2022, conference Name: IEEE Access.

[8] S. Qayyum, S. Imtiaz, and H. H. Khan, "Challenges of Agile–Crowd Software Development: A Systematic Literature Review," *Journal of Circuits, Systems and Computers*, p. 2330001, 2022, publisher: World Scientific.

[9] D. d. C. Candria and R. M. d. Araujo, "Crowdsourcing Software Development - a possible path?" in *XVIII Brazilian Symposium on Information Systems*, ser. SBSI. New York, NY, USA: Association for Computing Machinery, Jun. 2022, pp. 1–8. [Online]. Available: https://doi.org/10.1145/3535511.3535532

[10] K. Huang, J. Zhou, and S. Chen, "Being a Solo Endeavor or Team Worker in Crowdsourcing Contests? It is a Long-term Decision You Need to Make," *Proceedings of the ACM on Human-Computer Interaction*, vol. 6, no. CSCW2, pp. 494:1–494:32, Nov. 2022, number: CSCW2. [Online]. Available: http://doi.org/10.1145/3555595

[11] M. Tsvetkova, S. Müller, O. Vuculescu, H. Ham, and R. A. Sergeev, "Relative Feedback Increases Disparities in Effort and Performance in Crowdsourcing Contests: Evidence from a Quasi-Experiment on Topcoder," *Proceedings of the ACM on Human-Computer Interaction*, vol. 6, no. CSCW2, pp. 536:1–536:27, Nov. 2022, number: CSCW2. [Online]. Available: http://doi.org/10.1145/3555649

[12] E. Aghayi and T. D. LaToza, "A controlled experiment on the impact of microtasking on programming," *Empirical Software Engineering*, vol. 28, no. 1, p. 10, Nov. 2022. [Online]. Available: https://doi.org/10.1007/s10664-022-10226-2

[13] J. Wang, Y. Yang, S. Wang, C. Chen, D. Wang, and Q. Wang, "Context-Aware Personalized Crowdtesting Task Recommendation," *IEEE Transactions on Software Engineering*, vol. 48, no. 8, pp. 3131–3144, Aug. 2022, number: 8 Conference Name: IEEE Transactions on Software Engineering.

[14] Z. Peng, D. Wan, A. Wang, X. Lu, and P. M. Pardalos, "Deep learning-based recommendation method for top-K tasks in software crowdsourcing systems," *Journal of Industrial and Management Optimization*, pp. 0–0, Nov. 2022, publisher: Journal of Industrial and Management Optimization. [Online]. Available: https://www.aimsciences.org/en/article/doi/10.3934/jimo.2022223

[15] J. Wang, Y. Yang, S. Wang, J. Hu, and Q. Wang, "Context- and Fairness-Aware In-Process Crowdworker Recommendation," *ACM Transactions on Software Engineering and Methodology*, vol. 31, no. 3, pp. 35:1–35:31, Mar. 2022, number: 3. [Online]. Available: http://doi.org/10.1145/3487571

[16] S. Chen, X. Zhao, J. Liu, G. Gao, and Y. Du, "Social-Network-Assisted Task Recommendation Algorithm in Mobile Crowd Sensing," in *Proceedings of the 7th International Conference on Information and Education Innovations*, ser. ICIEI '22. New York, NY, USA: Association for Computing Machinery, Sep. 2022, pp. 136–142. [Online]. Available: http://doi.org/10.1145/3535735.3535751

[17] Y. Zhao, X. Chen, L. Deng, T. Kieu, C. Guo, B. Yang, K. Zheng, and C. S. Jensen, "Outlier Detection for Streaming Task Assignment in Crowdsourcing," in *Proceedings of the ACM Web Conference 2022*, ser. WWW '22. New York, NY, USA: Association for Computing Machinery, Apr. 2022, pp. 1933–1943. [Online]. Available: http://doi.org/10.1145/3485447.3512067

[18] M. Tahaei and K. Vaniea, "Recruiting Participants With Programming Skills: A Comparison of Four Crowdsourcing Platforms and a CS Student Mailing List," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. CHI '22. New York, NY, USA: Association for Computing Machinery, Apr. 2022, pp. 1–15. [Online]. Available: http://doi.org/10.1145/3491102.3501957

[19] O. A. Haqbani and S. Alyahya, "Supporting Coordination among Participants in Crowdsourcing Software Design," in *2022 IEEE/ACIS 20th International Conference on Software Engineering Research, Management and Applications (SERA)*, May 2022, pp. 132–139, iSSN: 2770-8209.

[20] R. Esmaeilyfard, "Improving detection of web service antipatterns using crowdsourcing," *The Journal of Supercomputing*, vol. 78, no. 5, pp. 6340–6370, Apr. 2022, number: 5. [Online]. Available: https://doi.org/10.1007/s11227-021-04134-3

[21] E. N. Moghaddam, A. Aliahmadi, M. Bagherzadeh, S. Markovic, M. Micevski, and F. Saghafi, "Let me choose what I want: The influence of incentive choice flexibility on the quality of crowdsourcing solutions to innovation problems," *Technovation*, p. 102679, Dec. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166497222002309

[22] S. Shin, H. Choi, Y. Yi, and J. Ok, "Power of Bonus in Pricing for Crowdsourcing," in *Abstract Proceedings of the 2022 ACM SIGMETRICS/IFIP PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS/PERFORMANCE '22. New York, NY, USA: Association for Computing Machinery, Jun. 2022, pp. 43–44. [Online]. Available: http://doi.org/10.1145/3489048.3522633

[23] X. Miao, H. Peng, Y. Gao, Z. Zhang, and J. Yin, "On Dynamically Pricing Crowdsourcing Tasks," *ACM Transactions on Knowledge Discovery from Data*, Jun. 2022, just Accepted. [Online]. Available: http://doi.org/10.1145/3544018

[24] H. H. Khan, M. N. Malik, and Y. Alotaibi, "Trust Issues in Crowdsourced Software Engineering: An Empirical Study." *Journal of Information Science & Engineering*, vol. 38, no. 4, 2022, number: 4 ISBN: 1016-2364.

[25] S. Huang, Z. Yang, C. Zheng, Y. Wang, J. Du, Y. Ding, and J. Wan, "Intellectual Property Right Confirmation System Oriented to Crowdsourced Testing Services," in *2022 International Conference on Blockchain Technology and Information Security (ICBCTIS)*, Jul. 2022, pp. 64–68.

[26] M. Li, L. Yang, Q. Xia, M. Fang, G. Liang, and C. Zuo, "STPChain: a Crowdsourced Software Engineering Method for Software Traceability and Fine-grained Privacy Based on Blockchain," in *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, Jun. 2022, pp. 849–859, iSSN: 0730-3157.

[27] P. Moslehi, J. Rilling, and B. Adams, "A user survey on the adoption of crowd-based software engineering instructional screencasts by the new generation of software developers," *Journal of Systems and Software*, vol. 185, p. 111144, Mar. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121221002405

[28] E. C. Groen, "Where Does Crowd-based Requirements Engineering End and Market Research Begin?" in *2022 IEEE 30th International Requirements Engineering Conference Workshops (REW)*, Aug. 2022, pp. 136–138, iSSN: 2770-6834.

[29] J. A. Khan, A. Yasin, R. Fatima, D. Vasan, A. A. Khan, and A. W. Khan, "Valuating requirements arguments in the online user's forum for requirements decision-making: The CrowdRE-VArg framework," *Software: Practice and Experience*, vol. 52, no. 12, pp. 2537–2573, 2022, number: 12 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.3137. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.3137

[30] J. Wouters, A. Menkveld, S. Brinkkemper, and F. Dalpiaz, "Crowd-based requirements elicitation via pull feedback: method and case studies," *Requirements Engineering*, vol. 27, no. 4, pp. 429–455, Dec. 2022, number: 4. [Online]. Available: https://doi.org/10.1007/s00766-022-00384-6

[31] M. N. Alatawi, "A conceptual framework for crowdsourcing requirements engineering in SCRUM-based environment," *IET Software*, 2022, publisher: Wiley Online Library.

[32] L. Radeck, B. Paech, F. Kramer-Gmeiner, M. Wettstein, H.-W. Wahl, A.-L. Schubert, and U. Sperling, "Understanding IT-related Well-being, Aging and Health Needs of Older Adults with Crowd-Requirements Engineering," in *2022 IEEE 30th International Requirements Engineering Conference Workshops (REW)*, Aug. 2022, pp. 57–64, iSSN: 2770-6834.

[33] S. Gottschalk, S. Parvez, E. Yigitbas, and G. Engels, "Designing Platforms for Crowd-Based Software Prototype Validation: A Design Science Study," in *Product-Focused Software Process Improvement*, ser. Lecture Notes in Computer Science, D. Taibi, M. Kuhrmann, T. Mikkonen, J. Klünder, and P. Abrahamsson, Eds. Cham: Springer International Publishing, 2022, pp. 334–350.

[34] D. Das, N. S. Mathews, and S. Chimalakonda, "Exploring Security Vulnerabilities in Competitive Programming: An Empirical Study," in *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022*, ser. EASE '22. New York, NY, USA: Association for Computing Machinery, Jun. 2022, pp. 110–119. [Online]. Available: http://doi.org/10.1145/3530019.3530031

[35] W.-T. Tsai, L. Zhang, and S. Hu, "From Crowdsourced Software Development to Crowdtesting," in *5th International Conference on Crowd Science and Engineering*, ser. ICCSE '21. New York, NY, USA: Association for Computing Machinery, Mar. 2022, pp. 18–23. [Online]. Available: http://doi.org/10.1145/3503181.3503185

[36] M. Nasir, N. Ikram, and Z. Jalil, "Usability inspection: Novice crowd inspectors versus expert," *Journal of Systems and Software*, vol. 183, p. 111122, Jan. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121221002193

[37] Q. Z. Chen, T. Schnabel, B. Nushi, and S. Amershi, "HINT: Integration Testing for AI-based features with Humans in the Loop," in *27th International Conference on Intelligent User Interfaces*, ser. IUI '22. New York, NY, USA: Association for Computing Machinery, Mar. 2022, pp. 549–565. [Online]. Available: http://doi.org/10.1145/3490099.3511141

[38] Y. Lin, Z. Li, W. Yue, and J. Wen, "CrowdIoT: The Crowd-Sourcing Test System for IoT Devices Based on Blockchain," *Advances in Internet of Things*, vol. 12, no. 2, pp. 19–34, 2022, number: 2 Publisher: Scientific Research Publishing.

[39] LI Le-Ping, ZHANG Yu-Xia, and LIU Hui, "Crowdsourcing Software Development Oriented Fault Localization," *Journal of Software*, pp. 1–18, Nov. 2022. [Online]. Available: https://www.jos.org.cn/josen/article/abstract/6498

[40] M. Du, S. Yu, C. Fang, T. Li, H. Zhang, and Z. Chen, "SemCluster: a semi-supervised clustering tool for crowdsourced test reports with deep image understanding," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2022. New York, NY, USA: Association for Computing Machinery, Nov. 2022, pp. 1756–1759. [Online]. Available: http://doi.org/10.1145/3540250.3558933

[41] Y. Li, Y. Feng, R. Hao, D. Liu, C. Fang, Z. Chen, and B. Xu, "Classifying crowdsourced mobile test reports with image features: An empirical study," *Journal of Systems and Software*, vol. 184, p. 111121, Feb. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121221002181

[42] D. Liu, Y. Feng, X. Zhang, J. A. Jones, and Z. Chen, "Clustering Crowdsourced Test Reports of Mobile Applications Using Image Understanding," *IEEE Transactions on Software Engineering*, vol. 48, no. 4, pp. 1290–1308, Apr. 2022, number: 4 Conference Name: IEEE Transactions on Software Engineering.

[43] P. Zhu, Y. Li, T. Li, H. Ren, and X. Sun, "Advanced Crowdsourced Test Report Prioritization Based on Adaptive Strategy," *IEEE Access*, vol. 10, pp. 53 522–53 532, 2022, conference Name: IEEE Access.

[44] Y. Yang and X. Chen, "Crowdsourced Test Report Prioritization Based on Text Classification," *IEEE Access*, vol. 10, pp. 92 692–92 705, 2022, conference Name: IEEE Access.

[45] H. Zhang, Y. Zhao, S. Yu, and Z. Chen, "Automated Quality Assessment for Crowdsourced Test Reports Based on Dependency Parsing," in *2022 9th International Conference on Dependable Systems and Their Applications (DSA)*, Aug. 2022, pp. 34–41, iSSN: 2767-6684.

[46] S. Alyahya, "Collaborative Crowdsourced Software Testing," *Electronics*, vol. 11, no. 20, p. 3340, Jan. 2022, number: 20 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2079-9292/11/20/3340

[47] X. Ge, S. Yu, C. Fang, Q. Zhu, and Z. Zhao, "Leveraging Android Automated Testing to Assist Crowdsourced Testing," *IEEE Transactions on Software Engineering*, pp. 1–18, 2022, conference Name: IEEE Transactions on Software Engineering.

[48] R. Nandakumar, "Quantitative Quality Score for Software," in *15th Innovations in Software Engineering Conference*, ser. ISEC 2022. New York, NY, USA: Association for Computing Machinery, Feb. 2022, pp. 1–5. [Online]. Available: https://doi.org/10.1145/3511430.3511457

[49] K. Wu, S. Huang, Y. Shi, J. Zhu, and S. Tang, "Estimate the Precision of Defects Based on Reports Duplication in Crowdsourced Testing," *IEEE Access*, vol. 10, pp. 130 415–130 423, 2022, conference Name: IEEE Access.

[50] J. P. d. Reis, F. B. e. Abreu, and G. d. F. Carneiro, "Crowdsmelling: A preliminary study on using collective knowledge in code smells detection," *Empirical Software Engineering*, vol. 27, no. 3, p. 69, Mar. 2022, number: 3. [Online]. Available: https://doi.org/10.1007/s10664-021-10110-5