



Artificial Intelligence-Powered Worker Engagement in Software Crowdsourcing

Junjie Wang, Ye Yang, and Qing Wang

From the Editor

This article presents an artificial intelligence-powered worker engagement framework in the context of crowdsourced testing and summarizes the implications and challenges when optimizing participation.—*Tim Menzies*

CROWDSOURCED SOFTWARE ENGINEERING (CSE) evolved from outsourcing and open source development. It has created a fundamental shift; there are now many open-call software minitasks that are advertised and performed through popular CSE platforms. For instance, TopCoder currently has more than 1.5 million registered designers, developers, and quality engineers; uTest has 400,000-plus testing specialists with diverse expertise to validate various aspects of digital quality. Evidence of the benefits of uTest crowd-testing includes:

- The average testing capacity grew 200%.

- The number of releases per year rose 150%.
- The number of critical fixes declined by an average of 50%.

Identifying and optimizing open participation from online developers is essential to the success of crowdsourced software tasks. Without appropriate decision support, online developers as well as task managers often make ad hoc decisions. Existing studies lack effective ways to address the uncertainty associated with online developers, the black-box nature of the crowdsourcing model, and the constantly changing context information of ongoing task progress.¹

Contextualization is important to enable the characterizing, learning, and uncovering of the underlying patterns of the software development

process. In our research,^{2,3} we seek to develop artificial intelligence (AI)-based automated methods to characterize in-process task progress and the crowdworker situation at a finer granularity to better support crowdsourcing decision making. In this article, we summarize the motivating empirical observations that led to our recent efforts and highlight an AI-based worker engagement framework for characterizing crowdworker skills and supporting both in-process recommendation and early termination detection for crowdtesting tasks.

Empirical Observations

Crowdworkers are shared resources. Open-call tasks frequently lead to ad hoc worker behaviors. In some cases, workers may choose tasks they are not good at and end up with

poor submissions; they may even drop out. In other cases, particularly in crowdtesting tasks, many workers with similar experience may submit duplicate bugs and result in a waste of effort. Our previous work explored various patterns among diversified workers.³ These empirical observations, as summarized in the following, foster a better understanding toward the worker behavior patterns as well as impacts on task outcomes.

Large Variation of Worker Activeness, Preferences, and Expertise

Figure 1(a) and (b) illustrates that crowdworkers have greatly diversified

characteristics. The x -axis shows example crowdworkers, and the y -axis displays the distribution of the workers' activeness and expertise. This reflects significant uncertainty associated with workers' dynamic availability and choices of tasks and, consequently, the quality of their submissions.

Insufficient Participation and Collaboration Lead to Poor Task Efficiency

As shown in Figure 1(c), low efficiency in the crowdtesting process is observed from the prevalence of long, flat, nonyielding windows in the bug arrival curve. This corresponds to a set of consecutive test reports that contain either

no bugs or duplicate bugs reported by early crowdworkers. It suggests a potential opportunity for accelerating the testing process by recommending a diverse set of crowdworkers in a dynamic manner, with the objective to shorten these nonyielding windows.

Excessive Worker Engagement Leads to Wasteful Duplicated Effort

Figure 1(d) presents the bug detection curves from four example crowdtesting tasks. These curves increase sharply during early stages and then exhibit a plateau effect, after which (i.e., the red points) reports find no new bugs. This indicates that the crowdtesting effort toward the end of

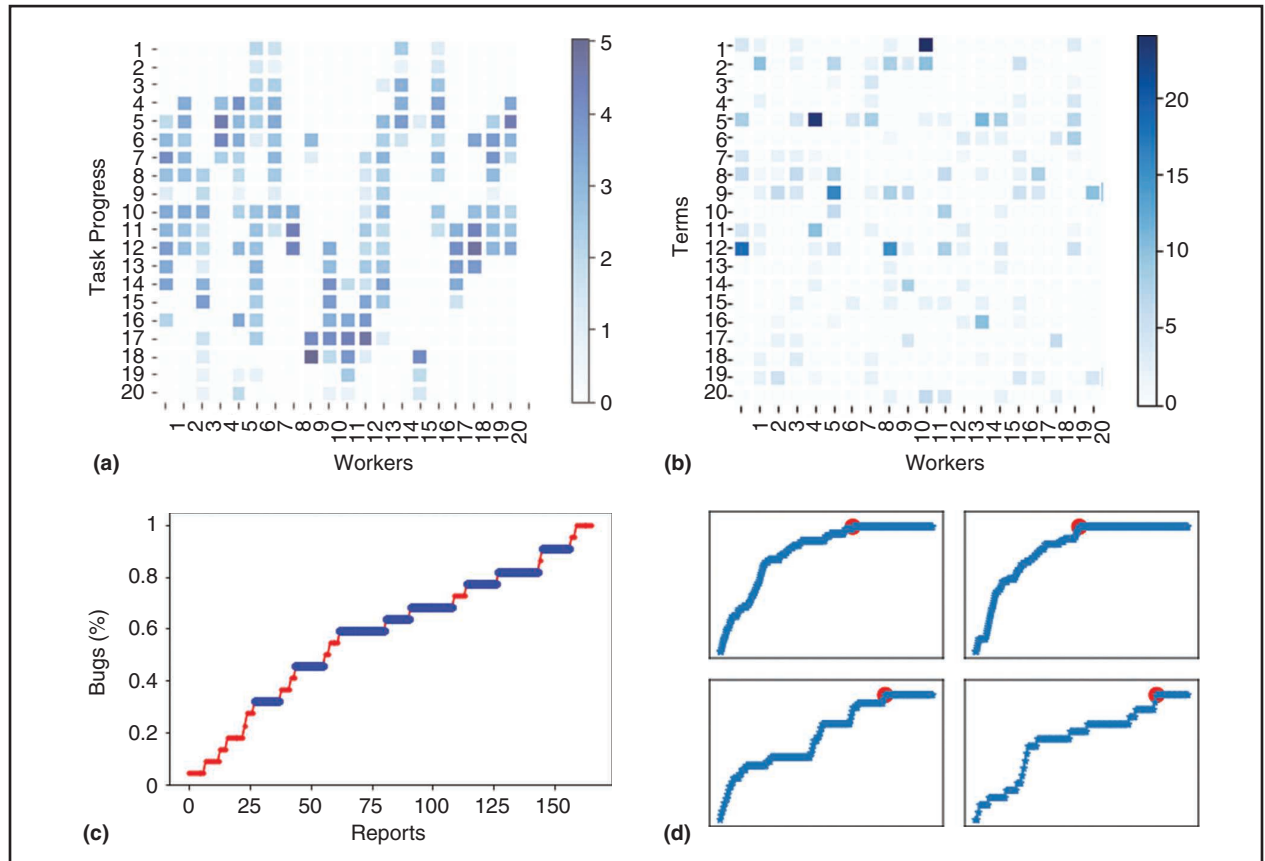


FIGURE 1. Empirical observations of crowdworkers (activeness, experience, and performance). (a) Crowdworkers' activeness. (b) Crowdworkers' expertise. (c) Nonyielding windows. (d) Decreasing bug detection.

the process is likely wasteful due to excessive worker engagement, and it suggests an opportunity and a practical need for intelligent task monitoring and early termination support.

AI-Based Crowdworker Engagement Framework

This framework contains three main components: the crowdsourcing context model, in-process recommendation, and early termination detection, as shown in Figure 2. Next, we introduce the design of each component.

Crowdsourcing Context Model

The crowdsourcing context model is constructed from two perspectives, i.e., the process context and the resource context, to capture in-process, progress-oriented information and crowdworkers' characteristics, respectively. For the process context, we use the notion of task adequacy to measure progress regarding the degree to which each descriptive term of the requirements has been covered. The resource context reflects the availability and capability factors concerning

competing crowdworker resources on the crowdsourcing platform.

Motivated by the observations in the previous section, the activeness, preference, and expertise of crowdworkers are integrated to model the resource context. Activeness measures the degree of availability of crowdworkers, preference assesses the extent to which a potential crowdworker might be interested in a candidate task, and expertise gauges a crowdworker's capability for detecting bugs. They can be dynamically modeled with AI-related techniques as a probabilistic representation based on a crowdworker's historical submitted reports and detected bugs. With this context model, crowdworkers can be better managed. More details can be found in Wang et al.³

In-Process Recommendation

The in-process crowdworker recommendation component dynamically learns the contextual information at a specific point during the crowdtesting process and recommends a diverse set of capable workers to

shorten the nonyielding window and improve bug detection efficiency. It consists of two main steps: learning-based ranking and diversity-based reranking. First, a set of features is defined and extracted from both the process context and the resource context; based on these features, the learning-based ranking computes the probability of crowdworkers being able to detect bugs within a specific context. Second, the diversity-based reranking adjusts the ranked list of recommended workers, based on the dynamic diversity measurement, to potentially reduce duplicate bugs. An evaluation of 636 crowdtesting tasks from one of the largest crowdtesting platforms in China shows that the process can shorten the nonyielding window by 50–58% and potentially reduce testing costs by 8–12%. More details can be found in Wang et al.³

Early Termination Detection

The early termination detection component leverages dynamical bug arrival data associated with crowdtesting reports and predicts, at a certain point in time, whether a task has obtained a satisfactory bug detection level; i.e., the bug can be closed. In essence, the component applies an incremental sampling technique to process arriving crowdtesting reports in chronological order. It organizes the reports into fixed-size groups as dynamic inputs and integrates data mining techniques as a capture–recapture (CRC) model and an autoregressive integrated moving average (ARIMA) model to raise progress awareness. It predicts two early termination indicators in an incremental manner, including: 1) the total number of bugs predicted by the CRC model and 2) the required test cost for achieving certain objectives that were predicted by the ARIMA model. Such estimates

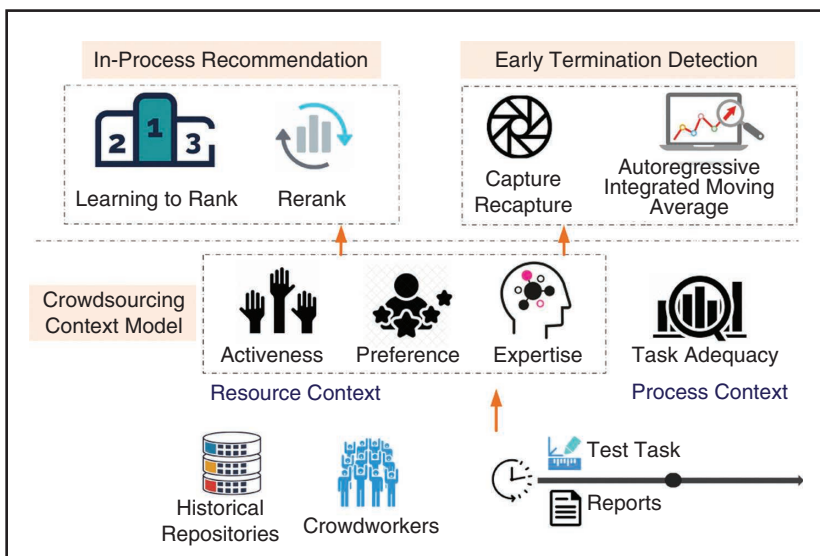
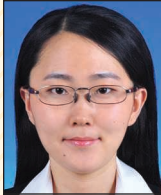
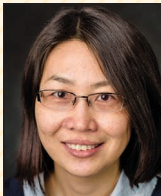


FIGURE 2. AI-based crowdworker engagement.

ABOUT THE AUTHORS



JUNJIE WANG is an associate researcher at the Institute of Software, Chinese Academy of Sciences. Further information about her can be found at <http://people.ucas.edu.cn/~0058217>. Contact her at junjie@iscas.ac.cn.



YE YANG is an associate professor at the School of Systems and Enterprises, Stevens Institute of Technology. Further information about her can be found at <https://web.stevens.edu/facultyprofile/?id=2042>. Contact her at yyang4@stevens.edu.



QING WANG is a researcher at the Institute of Software, Chinese Academy of Sciences. Contact her at wq@iscas.ac.cn.

are applied to support two typical crowdtesting decision scenarios, i.e., automating the early termination assessment and the semiautomation of the termination tradeoff analysis. An evaluation of the data set from China demonstrates that a median of 100% of the bugs can be detected at a 30% lower cost by using the early termination decision. More details can be found in Wang et al.²

Discussion

In theory, software tasks are more difficult to crowdsource than general tasks, such as the annotation of audio/language/image data. First, software has to be decomposed into many small undertakings that are suitable for crowdsourcing, which leads to significant management overhead.

Second, existing software engineering prediction models and methods are poorly suited to the highly dynamic yet largely “black-box” nature of crowdsourcing processes. Third, compared to in-house developers, the behaviors of external workers are less predictable and trustworthy. Such limitations and challenges produce a great risk to the project visibility and drive our interest in seeking AI-related techniques to learn and recommend practical insights to automate CSE management decisions and improve CSE effectiveness.

We argue that, although the in-process crowdworker recommendation is motivated and designed for addressing crowdworkers’ participation in a crowdtesting paradigm, it actually has great potential to be applied to other

crowdsourcing environments, e.g., software development. The recommendation is based on automatically characterizing crowdworkers through natural language processing and machine learning techniques, which are independent of different crowdsourcing types. Besides, more sophisticated skill sets that reflect these specialty crowdsourcing types can be implicitly represented by corresponding knowledge learned from historical records.

Need for More Actionable Analytics

As the underlying motivation for this research, the uncertainties associated with crowdsourcing management primarily stem from two aspects: the unpredictability of crowdworkers’ performance and the lack of visibility into crowdsourcing progress. Experience from this research demonstrated the effectiveness, potential, and necessity for more actionable, time-sensitive, case-sensitive analytics produced through AI techniques to better support CSE in-process decision making. Such time-sensitive analytics can provide powerful visibility into the task progress and insights for effective task management, which significantly facilitate CSE adoption at large.


Opportunities for Human-Centered Research

Actionable, in-process CSE decision making is frequently a complicated, systematic, human-centered problem. For example, in the crowdtesting context, the nonyielding windows may scatter widely across processes. The true effect of a recommendation system also depends on communication delays resulting from interactions between testing managers, the platform, and recommended workers. The longer the delays are, the less benefit there is. There is a critical need

for more human-centered research to explore how to best streamline the workflows of recommendation generation, communication, and confirmation functions to minimize potential delays in linking the best workers to the tasks under test.

Objectivity Versus Fairness in Recommendation

The recommendation in this work lies in the personalized characterization of workers that is learned from the crowdtesting platform's historical data. This facilitates the generation of an objective recommendation that is not influenced by community bias and value commitments. However, as with other history-based AI techniques, the proposed approach also suffers from the

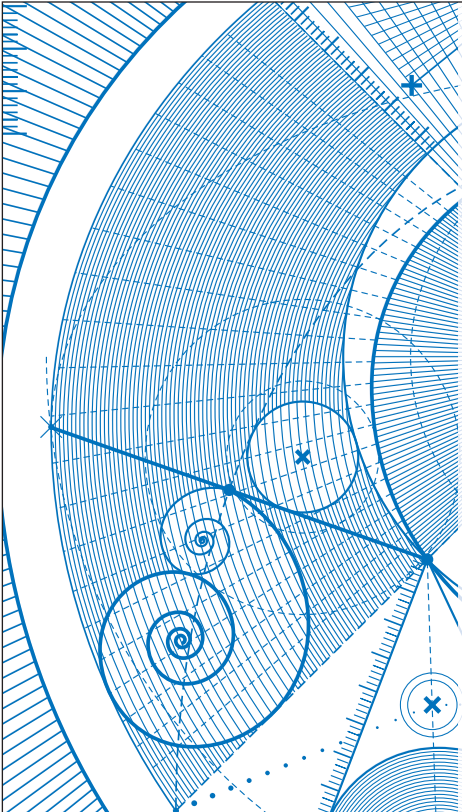
cold-start problem; i.e., it is unable to accommodate newcomers that have an empty history. Therefore, the objectivity in recommendation may, on the other hand, lead to a loss of fairness to the newcomers that are critical for a thriving crowdsourcing market.⁴ To mitigate this drawback, one possible strategy may be to use a historically calibrated characterization for newcomers and conduct human subject experiments to develop further insights into refining recommendations. 

Acknowledgment

This work is supported by the National Key R&D Program of China, under grant 2018YFB1403400.

References

1. K. Mao, L. Capra, M. Harman, and Y. Jia, "A survey of the use of crowdsourcing in software engineering," *J. Syst. Softw.*, vol. 126, pp. 57–84, Apr. 2017. doi: 10.1016/j.jss.2016.09.015.
2. J. Wang, Y. Yang, R. Krishna, T. Menzies, and Q. Wang, "iSENSE: Completion-aware crowdtesting management," in *Proc. IEEE/ACM 41st Int. Conf. Software Engineering (ICSE)*, May 2019, pp. 912–923. doi: 10.1109/ICSE.2019.00097.
3. J. Wang, Y. Yang, S. Wang, Y. Hu, D. Wang, and Q. Wang, "Context-aware in-process crowdworker recommendation," in *Proc. 42nd Int. Conf. Software Engineering (ICSE)*, 2020.
4. A. Chouldechova and A. Roth, "A snapshot of the frontiers of fairness in machine learning," *Commun. ACM*, vol. 63, no. 5, pp. 82–89, 2020. doi: 10.1145/3376898.



IEEE
Annals
of the History of Computing

From the analytical engine to the supercomputer, from Pascal to von Neumann, from punched cards to CD-ROMs—*IEEE Annals of the History of Computing* covers the breadth of computer history. The quarterly publication is an active center for the collection and dissemination of information on historical projects and organizations, oral history activities, and international conferences.

www.computer.org/annals

Digital Object Identifier 10.1109/MS.2020.3027399