2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)

Replication Can Improve Prior Results: A GitHub Study of Pull Request Acceptance

Di Chen, Kathryn T. Stolee, Tim Menzies
Computer Science, NC State, USA,
dchen20@ncsu.edu; ktstolee@ncsu.edu; timm@ieee.org

***Abstract*—Crowdsourcing and data mining can be used to effectively reduce the effort associated with the partial replication and enhancement of qualitative studies.**

For example, in a primary study, other researchers explored factors influencing the fate of GitHub pull requests using an extensive qualitative analysis of 20 pull requests. Guided by their findings, we mapped some of their qualitative insights onto quantitative questions. To determine how well their findings generalize, we collected much more data (170 additional pull requests from 142 GitHub projects). Using crowdsourcing, that data was augmented with subjective qualitative human opinions about how pull requests extended the original issue. The crowd's answers were then combined with quantitative features and, using data mining, used to build a predictor for whether code would be merged. That predictor was far more accurate than the one built from the primary study's qualitative factors (F1=90 vs 68%), illustrating the value of a mixed-methods approach and replication to improve prior results.

To test the generality of this approach, the next step in future work is to conduct other studies that extend qualitative studies with crowdsourcing and data mining.

I. Introduction

Our ability to generate models from software engineering data has out-paced our abilities to reflect on those models. Studies can use thousands of projects, millions of lines of code, or tens of thousands of programmers [1]. However, when insights from human experts are overlooked, the conclusions from the automatically generated models can be both wrong and misleading [2]. After observing case studies where data mining in software engineering led to spectacularly wrong results, Basili and Shull [3] recommend qualitative analysis to collect and use insights from subject matter experts who understand software engineering.

The general problem we explore is how partial replication studies can scale and deepen the insights gained from primary qualitative studies. That is, after collecting qualitative insights from an in-depth analysis of a small sample, a partial replication study is conducted using a subset of the insights as a guide, but targeting a larger sample and using a different empirical methodology. To show that a given result is robust, the ideal case is for a completely independent set of researchers to replicate a published study using their own experimental design [4]. In this work, we explore a mixed-methods approach using a crowdsourced evaluation and data mining to build on a primary qualitative study from prior work [5], aimed at the goal of understanding the factors that govern pull request acceptance.

Crowdsourcing brings advantages of a lower cost compared to professional experts. Micro-task crowdsourcing using established platforms such as Amazon's Mechanical Turk (MTurk) [6] also provides a large worker pool with great diversity and fast completion times. The main issue with crowdsourcing is the low quality; an uncontrolled experimental context often leads to less credible results; however, some results suggest that crowdsourced workers perform similarly to student populations [7]. Data mining, on the other hand, is good at predicting future patterns based on the past. It is an inexpensive and fast tool to analyze quantitative data from crowdsourcing results, especially when data is large. However, data mining is limited in that it looks narrowly at the data. The starting point of this investigation was a conjecture that combining crowdsourcing and data mining would lead to better results that using either separately.

To test this conjecture, we used Tsay, Dabbish, Herbsleb (hereafter, TDH) [5]. That study explored how GitHub-based teams debate what new code gets merged through the lens of pull requests. To do this, they used a labor-intensive qualitative interview-process of 47 users of GitHub, as well as in-depth case studies of 20 pull-requests. They found that the submitter's level of prior interaction on the project changed how core and audience members interacted with the submitter during discussions around contributions. The results provide many insights into the factors and features that govern pull request acceptance. The TDH authors were clear about their methodology and results, making it a good candidate for partial replication and extension.

This paper extends that primary qualitative study of pull requests with an independent partial replication study using a crowdsourced evaluation and data mining. To perform this independent, partial replication and extension, using the insights from the original study, we design questions that can be answered by a crowdsourced worker pool and serve to confirm some of the original findings. The crowd is able to handle a larger pool of artifacts than the original study, which tests the external validity of the findings. In addition to the original 20 pull requests, the crowd in our study analyzes an additional 170 pull requests. Next, data mining was applied to the crowd's responses, resulting in accurate predictors for pull request acceptance. The predictors based on crowd data were compared to predictors built using quantitative methods from the literature (i.e., traditional data mining without crowdsourcing and without insights from the primary study).

978-1-7281-1519-1/19/$31.00 ©2019 IEEE
DOI 10.1109/ICPC.2019.00037

Our results show that the crowd can replicate the TDH results, and that for those factors studied, most results are stable when scaled to larger data sets. After using data mining to develop predictors for pull request acceptance, we found that the predictors based on quantitative factors from the literature were more accurate than the predictor based on the TDH features we studied. Even though the predictors for pull request acceptance based on data mining were more accurate than predictors based on crowd-generated data alone, it would be extremely premature to use this one result to make generalizations about the relative merits of the different empirical approaches. The primary study revealed insights that we were not able to scale up; for example, the original study found that evaluation outcomes of pull request were sometimes more complex than acceptance or rejection. That is, a rejected pull request may be followed up with another pull request from the core team fulfilling the goals of the rejected pull request [5]. Such insights would be difficult, if not impossible, to expose through data mining alone. That said, other insights can be verified through scaled replication, such as the impact of different features of a pull request discussion on that pull request's acceptance, which we explore in this work.

This paper makes three specific contributions:

- A cost-effective, independent, partial replication and extension of a primary study of pull request acceptance factors using a scaled sample of artifacts (RQ1).
- Analysis of the external validity of findings from the original study, demonstrating stability in most of the results. This has implications for which questions warrant further analysis (RQ2).
- Comparison of qualitative and quantitative factors that impact pull request acceptance from related work (RQ3).

To assist other researchers, a reproduction package with all our scripts and data is available[1] and in archival form (with a DOI)[2] to simplify all future citations to this material.

II. Background

With over 14 million users and 35 million repositories as of April 2016, GitHub has become the largest and most influential open source projects hosting sites. Numerous qualitative [5], [8]–[14], quantitative [1], [15]–[22] and mixed methods studies [23], [24] have been published about GitHub.

Pull requests are created when contributors want their changes to be merged to the main repository. After core members get pull requests, they inspect the changes and decide whether to accept or reject them. This process usually involves code inspection, discussion, and inline comments between contributors and repository owners. Note that core members have the ability to close the pull requests by either accepting the code and merging the contribution with the master branch, or rejecting the pull requests. Core members could also ignore the pull requests and leave them in an open state.

[1]github.com/dichen001/IST_17
[2]doi.org/10.5281/zenodo.802698

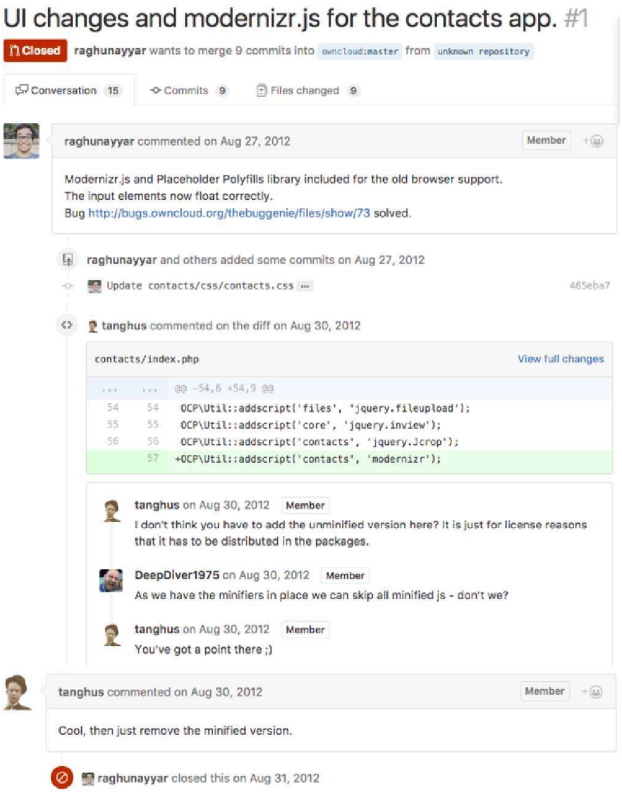

Fig. 1. An example GitHub pull request

Figure 1 shows an example of pull requests with reduced discussion (No.16 of the 20 pull requests TDH studied.[3]), inline code comments and final result (closed). This pull request, on the topic of *UI changes and moderizr.js for the contacts app*, was submitted by user *raghunayyar* on August 27, 2012. The intention of the pull request was to address a bug, which is linked within the submitter's first comment. User *tanghus* commented on the diff for file `contacts/index.php` on August 30, 2012. A discussion ensued between user *tanghus* and user *DeepDiver1975*. After one more comment from *tanghus*, the pull request is closed by the submitter on August 31, 2012. All three people involved with the pull request are *Member*s, meaning they are developers. From the contributors' page of this repository, we and the crowd could find the user *tanghus* and user *DeepDiver1975* are core members, while the submitter, user *raghunayyar*, is an external developer. The pull request was closed without being merged. The crowd worker who was assigned to analyze this pull requests finds:

1) There are core developers supporting this pull requests.
2) There are alternates proposed by the core members.
3) There are people disapproving the proposed solution .
4) No one disapproves the problem being solved here.
5) The pull requests are rejected but the core team implemented their own solution to the problem in the contribution.

[3]www.jsntsay.com/work/FSE2014.html

Note that findings 3 and 4 conflict with the results from TDH. TDH finds there are disapproving comments for the problems being proposed due to project appropriateness (but not for the solutions itself being inconsistent, as found by the crowd). A small number of such inconsistencies are not unexpected in qualitative work, and replication can help identify where ambiguities may be present.

In this paper, crowd workers analyze pull request features and the conversations within the pull request, as just illustrated, and answer quantitative questions about the pull request conversations and outcomes.

III. Research Questions

We evaluate the following research questions:

RQ1: *Can the crowd reproduce prior results with high quality?* One challenge with crowdsourcing is quality control; we employ several strategies to encourage high-quality responses from the crowd to determine if the crowd can partially replicate the results from the TDH paper.

RQ2: *Does crowdsourcing identify which conclusions from the primary study are stable?* This question is important for the external validity of the original findings used in the extension part of the experiment. We collected 170 additional pull requests using similar sampling criteria to the primary study, and added these to the original 20. We then tested if the crowd reaches the same or different conclusions using the original 20 and using the extended data set.

RQ3: *Can the pull request features identified in the primary study accurately predict pull request acceptance?* Given the larger data set collected and evaluated in this work, there is now an opportunity to evaluate the performance of prediction models based on (1) the subset of features identified as important in the primary study and included in our partial replication (collected in RQ1 and RQ2), and (2) features identified as important in previous data mining-only studies (identified from related work).

IV. Methodology

To leverage the advantages of crowdsourcing, we perform a five-step process from exploring prior studies to running the replication studies using crowdsourcing and then data mining.
Step 1: *Related work exploration on GitHub pull requests studies; extract data, insights, features and results from the existing work. Quantitative features should also be extracted from existing work (to answer RQ3).*
Step 2: *Map insights from qualitative work into questions that could be answered by crowd workers in micro-tasks. Map existing quantitative features into questions with known answers, which are "gold" questions used for quality control.*

TABLE I
A sample of related qualitative and quantitative work. Here, by "quantitative", we mean using data mining with little to no interaction with project personnel.

Year	Source	Method	Data	Title
2012 [8]	CSCW	Qualitative	Interview 24 GitHub Users. Pull requests case study 10.	Social coding in GitHub: transparency and collaboration in an open software repository
2013 [11]	CSCW	Qualitative	Interview 18 GitHub users. Pull requests case study 10.	Impression Formation in Online Peer Production: Activity Traces and Personal Profiles in GitHub
2014 [5]	FSE	Qualitative	Interview 47 GitHub users. Pull requests case study 20.	**Let's Talk About It: Evaluating Contributions through Discussion in GitHub (TDH)**
2015 [9]	ICSE	Qualitative	Online survey 749 integrators.	Work Practices and Challenges in Pull-Based Development: The Integrators Perspective
2016 [10]	ICSE	Qualitative	Online survey 645 contributors.	Work Practices and Challenges in Pull-Based Development: The Contributors Perspective
2014 [25]	ICSE	Quantitative	GHTorrent, 166,884 pull requests	An Exploratory Study of the Pull-Based Software Development Model
2014 [15]	ICSE	Quantitative	GitHub API, GitHub Archive. 659,501 pull requests	Influence of Social and Technical Factors for Evaluating Contribution in GitHub.
2014 [26]	ICSME	Quantitative	GHTorrent, 1,000 pull requests.	Reviewer Recommender of Pull-Requests in GitHub
2014 [27]	ICSME	Quantitative	GHTorrent	Continuous Integration in a SocialCoding World Empirical Evidence from GitHub
2014 [28]	APSEC	Quantitative	GHTorrent, 1,000 pull requests.	Who Should Review This Pull-Request: Reviewer Recommendation to Expedite Crowd Collaboration
2014 [19]	CrowdSoft	Quantitative	GHTorrent, GitHubArchive.	Investigating Social Media in GitHubs Pull-Requests: A Case Study on Ruby on Rails
2014 [29]	MSR	Quantitative	GHTorrent	A Dataset for Pull-Based Development Research
2014 [20]	MSR	Quantitative	GHTorrent, 78,955 pull requests.	An Insight into the Pull Requests of GitHub
2014 [30]	MSR	Quantitative	GHTorrent, 54,892 pull requests.	Security and emotion sentiment analysis of security discussions on GitHub
2014 [31]	MSR	Quantitative	GHTorrent	Do developers discuss design
2014 [32]	MSR	Quantitative	GHTorrent, 75,526 pull requests.	A study of external community contribution to opensource projects on GitHub
2015 [33]	MSR	Quantitative	GHTorrent	Automatically Prioritizing Pull Requests
2015 [18]	MSR	Quantitative	GHTorrent, 103,284 pull requests.	Wait For It: Determinants of Pull Request Evaluation Latency on GitHub
2014 [23]	MSR	Quantitative & Qualitative	Quant. : GHTorrent Qual. : 240 Survey, 434 projects.	The promises and perils of mining GitHub

Step 3: *Collect more artifacts using similar sampling processes to the primary study. Apply the mapped questions from Step 2 to the additional data.*
Step 4: *Using the original data as "gold" queries for quality control in crowdsourcing, run the crowdsourced study.*
Step 5: *Extract and analyze features defined in Step 2 from the crowd answers. Compare those with the findings from Step 1 to discover new insights.*

Next, we apply these methods to the primary TDH study [5].

A. Step 1: Literature Overview and Data Extraction

We first identified TDH as our primary study after finding its data source is publicly available and some of its insights about GitHub pull requests could be mapped into quantitative questions for the crowd to answer. Next, we explored prior work related to GitHub pull requests in the literature.

For the literature exploration, we searched for keywords 'pull', 'request' and 'GitHub' on Google Scholar from 2008 to 2016 and also obtained a dataset from 16 top software engineering conferences, 1992 to 2016 [34], filtering out the work unrelated to GitHub pull requests. Table I lists the remaining research papers that have studied pull requests in GitHub using either qualitative or quantitative methods. Here, we distinguish qualitative and quantitative methods by whether or not there is human involvement during the data collection process. Qualitative studies have human involvement and include interviews, controlled human experiments, and surveys. We observe that all previous studies on pull request in GitHub, except for one, use either qualitative or quantitative methods. The remaining study combines both with a very time consuming manual analysis for the qualitative part [23], which starts from the very beginning with no previous knowledge. This is quite different from ours; we leverage prior work and apply crowdsourcing directly on the results extracted from primary qualitative studies.

Table II summarizes the features found to be relevant in determining pull request acceptance. This includes all quantitative papers from Table I that use features to predict the outcomes of pull requests, and the features explored in at least one of those papers. Fewer papers are listed here some predicted for other thing such as sentiment or best reviewers for pull request. In Table II:

- White boxes □ denote that a paper examined that feature;
- Black boxes ■ denote when that paper concluded that feature was important;

TABLE II
FEATURES USED IN RELATED WORK. □ INDICATES THAT A FEATURE IS USED; ■ INDICATED THE FEATURE IS FOUND TO BE HEAVILY RELATED TO THE RESULTS OF PULL REQUESTS IN THE CORRESPONDING PAPER.

Category	Fetures	Description	[29]	[25]	[15]	[18]	[19]	[20]	Ours
Pull Request	lifetime_minites	Minutes between opening and closing	□						
Pull Request	mergetime_minutes	Minutes between opening and merging	□						
Pull Request	num_commits	Number of commits	□	□	□	■			□
Pull Request	src_churn	Number of lines changed (added + deleted)	□	■		■			□
Pull Request	test_churn	Number of test lines changed	□	□					
Pull Request	files_added	Number of files added	□						
Pull Request	files_deleted	Number of files deleted	□						
Pull Request	files_modified	Number of files modified	□						
Pull Request	files_changed	Number of files touched (sum of the above)	□	□	□				
Pull Request	src_files	Number of source code files touched by the pull request	□						
Pull Request	doc_files	Number of documentation (markup) files touched	□						
Pull Request	other_files	Number of non-source, non-documentation files touched	□						
Pull Request	num_commit_comments	Total number of code review comments	□						
Pull Request	num_issue_comments	Total number of discussion comments	□						
Pull Request	num_comments	Total number of discussion and code review	□	□	■	■			□
Pull Request	num_participants	Number of participants in the discussion	□	□					
Pull Request	test_inclusion	Whether or not the pull request included test cases			□	□			
Pull Request	prior_interaction	Number of events the submitter has participated previously			□				
Pull Request	social_distance	Whether the submitter follows the user who closes the PR			■	□			□
Pull Request	strength of social connection	Fraction of members interacted with the submitter in T_0 (the last 3 months prior to creation)				□			
Pull Request	description complexity	Total number of words in the pull request title and description				□			
Pull Request	first human response	Interval from PR creation to first response by reviewers				■			□
Pull Request	total CI latency:	Interval from PR creation to the last commit tested by CI				■			□
Pull Request	CI result:	Presence of errors and test failures while running Travis-CI				■			□
Pull Request	mention-@	Weather there exist an @-mention in the comments					□		
Repository	sloc	Executable lines of code at creation time.	□	■					□
Repository	team_size	Number of active cores in T_0	□	■	□	□		■	□
Repository	perc_external_contribs	Ratio of commits from externals over cores in T_0	□	■					□
Repository	commits_on_files_touched	Number of total commits on files touched by the PR in T_0	□	■		□			■
Repository	test_lines_per_kloc	Executable lines of test code per 1,000 lines of source code	□	■					□
Repository	test_cases_per_kloc	Number of test cases per 1,000 lines of source code	□						
Repository	asserts_per_kloc	Number of assert statements per 1,000 lines of source code	□						
Repository	watchers	Project watchers (stars) at creation	□		■				□
Repository	repo_age	Life of a project since the time of data collection			□	□			
Repository	workload	Total number of PRs still open at current PR creation time				□			
Repository	integrator availability	Minimal hours until either of the top 2 integrators are active				□			
Repository	project maturity	Number of forked projects as an estimate of project maturity						■	□
Developer	prev_pullreqs	Number of PRs previously submitted by the submitter	□	■					■
Developer	requester_succ_rate	Percentage of the submitters PRs got merged previously.	□	■		□			■
Developer	followers	Followers to the developer at creation	□		□	□			
Developer	collaborator_status	The user's collaborator status within the project		□	■	□			□
Developer	experience	Developers working experience with the project						■	□
Other	Friday effect	True if the pull request arrives Friday				□			

The last column in Table II shows what lessons we took from these prior studies for the data mining analysis (RQ3). If any other column marked a feature as important, then we added it into the set of features we examined. Such features are denoted with a white box □ in the last column. If, in RQ3, we determine the feature is informative for pull request acceptance, it is marked with a black box ■.

B. Step 2: Map Insights into Questions and Features

The tasks performed by the crowd were designed to collect quantitative information about the pull requests, which could be checked against a ground truth extracted programmatically (e.g., was the pull request accepted?), and also collect information related to the pull request discussion, which cannot be easily extracted programmatically, described next.

The primary study [5] concluded, among other things, that:

> *Methods to affect the decision making process for pull requests are mainly by offering support (Q1) from either external developers or core members.*
>
> *Issues raised around code contributions are mostly disapproval for the problems being solved (Q4), disapproval for the solutions (Q3) and suggestion for alternate solutions (Q2).*

These are the insights we use to derive quantitative questions for the crowd, which are mapped to the question in Table III (including Q5 regarding pull request acceptance).

In order to use crowdsourcing to do a case study for pull requests, our tasks contained questions related to the four concepts underlined above and shown explicitly in Table III in the *Concepts* column. This is followed by the *Questions* related to each concept. For example, in Q2, the worker would answer *Yes* or *No* depending on whether alternate solutions were proposed at all (*Q2_alternate_solution*), were proposed by core members (*Q2_alt_soln_core*), or were propose by other developers (*Q2_alt_soln_other*). These four concepts reference important findings from TDH's work and were selected because they could be easily converted into micro questions for crowd workers to answer, though we note that not all the TDH findings were converted into questions for the crowd. The full version of our questions are available on-line (tiny.cc/mt_questions).

Per Step 2 in our methods (Section IV), we use quantitative questions over the original pull requests from the TDH study as gold standard *tasks*. After extracting answers from the TDH results, we compare the crowd's performance on those pull requests to ensure the crowd is qualified to perform the tasks.

To further ensure response quality in a crowdsourced environment, for all pull requests, we also added three preliminary qualification questions that require crowd workers to identify the submitter, core members and external developers for each pull request; these are gold standard *questions*. These extra questions let a crowd worker grow familiar with analyzing pull request discussions, and let us reject answers from unqualified crowd workers since we could programmatically extract the ground truth from the pull request for comparison. Details on our quality control used during the study are in Section IV-D.

C. Step 3: Data Collection and Expansion

To make sure the pull requests are statistically similar to those of TDH's work [5], we applied similar selection rules on 612,207 pull requests that were opened as new in January 2016 from GHTorrent [35], which is a scalable, searchable, offline mirror of data offered through the GitHub Application Programmer Interface (API). The selection criteria are stated as follows. The main difference between our selection and TDH's selection is the time requirements for when the pull requests are created or last updated.

1) Pull requests should be closed.
2) Pull requests should have comments.
3) Pull request comment number should be above eight.
4) Exclude pull requests whose repository is a fork to avoid counting the same contribution multiple times.
5) Exclude pull requests whose last update is not later than January, 2016, so that we can make sure the project is still active.

TABLE III
QUESTIONS FOR EACH PULL REQUEST IN SECONDARY STUDY

Concepts	Questions	Response	Identifier
Q1: Is there a comment showing support for this pull request, and from which party?	Support showed	Yes/No	Q1_support
	Support from core members	Yes/No	Q1_spt_core
	Support from other developers	Yes/No	Q1_spt_other
Q2: Is there a comment proposing alternate solutions, and from which party?	Alternate solutions proposed	Yes/No	Q2_alternate_solution
	Alternate solution proposed by core members	Yes/No	Q2_alt_soln_core
	Alternate solution proposed by other developers	Yes/No	Q2_alt_soln_other
Q3: Did anyone disapprove the proposed solution in this pull request, and for what reason?	Disapproval for the solution proposed	Yes/No	Q3_dis_solution
	Disapproval due to bug	Yes/No	Q3_dis_soln_bug
	Disapproval because code could be improved	Yes/No	Q3_dis_soln_improve
	Disapproval due to consistency issues	Yes/No	Q3_dis_soln_consistency
Q4: Did anyone disapprove the problems being solved? E.g., question the value or appropriateness of this pull request for its repository.	Disapproval for the problem being solved	Yes/No	Q4_dis_probelm
	Disapproval due to no value for solving this problem	Yes/No	Q4_dis_prob_no_value
	Disapproval because the problem being solved does not fit the project well	Yes/No	Q4_dis_prob_no_fit
Q5: Does this pull request get merged/accepted?	Pull request got merged into the project	Yes/No	Q5_merged

6) Retain only pull requests with at least three participants and where the repository has at least ten forks and ten stars.

There are 565 pull requests left after applying the selection criteria stated above. From these pull requests, we sampled 170 such that half were ultimately merged and the other half were rejected.

The 170 additional pull requests were published on MTurk for analyzing in two rounds, together with the 20 carefully studied pull requests from TDH [5] inserted for each round as "gold" standard tasks. The 1st round has 100 (80 new + 20 TDH) pull requests in total, while 2nd round has 110 (90 new + 20 TDH) pull requests in total. So 210 total pull request were published on for the crowd to complete.

D. Step 4: Crowdsourcing Study

Mechanical Turk is a crowdsourcing platform that connects workers (i.e., the people performing the tasks) with requesters (i.e., the people creating the tasks) [6]. MTurk has been used extensively in software engineering applications [36] and for the evaluation of software engineering research (e.g., [7], [37], [38]). It manages recruitment and payment of participants and hosts the tasks posted by requesters.

1) Tasks: The tasks performed by participants are called Human Intelligence Tasks (HITs). These represent individual activities that crowd workers perform and submit. The scope of a HIT for our study included a single pull request, the questions outlined in Table III, as well as the following questions about the pull request that could be checked programmatically for quality control:

1) What is the submitter GitHub ID?
2) From the contributor page of this repository, who are the core members of this repository?
3) Who are the external developers among the participants of this pull request?
4) Does this pull request get merged/accepted? (Q5)

2) Participants: Workers for our study were required to have demonstrated high quality in prior tasks on the MTurk platform and answer simple questions about the pull request correctly. Quality and experience filters were applied to screen potential participants; only workers with HIT approval rate above 90%, and who had completed at least 100 approved HITs could participate. To make sure the crowd participants are qualified to analyze the pull requests in our study, we also require them to be GitHub users.

3) Cost: We want to make sure the cost is low but also provide a fair payment for the participants. According to several recent surveys on MTurk [39]–[42], the average hourly wage is $1.66 and MTurk workers are willing to work at $1.40/hour. We estimated about 10 minutes needed for each HIT, and first launched our task with $0.25 per HIT but only received 1 invalid feedback after 2 days. So we doubled our payment to $0.50 for each HIT, which requires to analyze one single pull request. Each round of tasks was completed in one week. Our final results show that 17 minutes are spent for each HIT on average, which means $1.76 per hour. In total, 27 workers participated in our tasks, and 77 hours of crowd time were spent to get all the pull requests studied[4].

4) Quality Control: A major issue in crowdsourcing is how to reduce the noise inherent in data collection. This section describes the three approaches we used to increase data quality:

- Qualification questions;
- Redundant question formats [44]; and
- "Gold" standard tasks [45], [46].

a) Qualification Questions: A domain-specific screening process was applied as participants were required to answer preliminary qualification questions related to identifying the pull request key players and pull request acceptance *on every pull request analyzed*. These are questions for which we can systematically extract values from the pull requests (i.e., those in Section IV-D1); if these objective questions are answered incorrectly, the task was rejected and made available to other crowd workers.

This is unlike the *qualification test* which is available on the MTurk platform to screen participants once for eligibility to participate in any of our tasks. A qualification test is administered once. Once participants pass, they can perform our HITs. Our *qualification questions*, on the other hand, were used for every HIT we published. This ensured quality responses for each and every HIT.

b) Redundant Question Formats: For each question in the task related to the pull request comment discussion, we require workers to answer a yes/no question and then copy the comments supporting their answers from the pull request into the text area under each question, as suggested in prior work [44]. Take question 1 for example (*Is there a comment proposing alternate solutions?*): if they choose *"Yes, from core members"*, then they need to copy the comments within the pull requests to the text area we provided.

c) "Gold" Standard Tasks: This study was run in two phases. In each phase, the original 20 pull requests were added to the group as "gold" standard tasks. The tasks were randomly assigned to crowd workers. For those crowd workers who got one of the 20 previously studied pull requests, we checked their answers against the ground truth [5]; inaccurate responses were rejected and those workers were blocked. This acted as a random quality control mechanism.

We used multiple quality control mechanisms in keeping with prior work [47], and the result quality was satisfactory. However, evaluating the effectiveness of each quality control mechanism we employed is left for future work.

E. Step 5: Data Mining Analysis

In this study, we have two groups of features: (1) all the quantitative features found important in previous works (see Table II), and (2) the qualitative features extracted from the results of studying pull requests in detail by the qualified

[4]Since the time of this writing, there has been a campaign of "fair wages" for crowd workers. In future studies, we would set our minimum costs to the minimum wage [43]

crowd (see Section IV-D). For each group of features, we run the CFS feature selector [48] to reduce the features to use for our decision tree classifier. To collect the quantitative features, we used the GitHub API to extract the features marked in the *ours* column of Table II.

CFS evaluates and ranks feature subsets. One reason to use CFS over, say, correlation, is that CFS returns *sets* of useful features while simpler feature selectors do not understand the interaction between features. CFS also assumes that a "good" set of features contains features that are highly connected with the target class, but weakly connected to each other. To implement this heuristic, each feature subset is scored as follows according to Hall, et al. [48]:

$$merits = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}$$

The $merits$ value is some subset s containing k features; r_{cf} is a score describing the connection of that feature set to the class; and r_{ff} is the mean score of the feature to feature connection between the items in s. Note that for this fraction to be maximal, r_{cf} must be large and r_{ff} must be small, which means features have to correlate more to the class than each other.

This equation is used to guide a best-first search with a horizon of five to select most informative set of features. Such a search proceeds as follows. The initial frontier is all sets containing one different feature. The frontier of size n, which initialized with 1, is sorted according to *merit* and the best item is grown to all sets of size *n+1* containing the best item from the last frontier. The search stops when no improvement have been seen in last five frontiers in $merit$. Return the best subset seen so far when stop.

Our experiments assessed three groups of features:

1) After CFS feature selector, the selected *quantitative features* are *commits_on_files_touched, requester_succ_rate, prev_pullreqs*, which are quite intuitive. (The *commits_on_files_touched* feature indicates the popularity of the modified files is highly relating to the final acceptance of the pull request while *requester_succ_rate* and *prev_pullreqs* show the requester's history interaction with the project also play an important role for the final acceptance.)
2) Using the same CFS feature selector, the selected *crowdsourced features* are *Q3_dis_solution, Q4_dis_prob_no_value*. These two selected features show that the final acceptance has a strong relation to the opposing voice in the pull requests discussion, especially for comments saying this pull request has no value for the project.
3) Combining all *quantitative features* and *crowdsourced features* into *combined features* and feed that into the same CFS feature selector, the following features were selected: *Q3_dis_solution, commits_on_files_touched, requester_succ_rate,* and *prev_pullreqs*, which all appeared using the CFS selector on the quantitative and qualitative features independently.

TABLE IV
QUALITY FOR CROWDSOURCING RESULTS FROM AMAZON MECHANICAL TURK (RQ1).

Questions	Precision	Recall	F1-Score
Q1	0.769	0.769	0.770
Q2	0.818	0.750	0.783
Q3	0.727	0.667	0.696
Q4	0.778	0.700	0.737
Q5	0.833	0.714	0.770
Total	**0.801**	**0.742**	**0.770**

For each of these three sets of features, we ran a 10x5 cross validation for supervised learning with the three different groups of features. These generate three models that predicted if a pull request would get merged/accepted or not. A decision tree learner was used as our supervised learning algorithm. This was selected after our initial studies with several other learners that proved to be less effective in this domain (Naive Bayes and SVM).

Using the MTurk micro-task crowdsourcing platform, we collect data for 1) the original 20 pull requests from the primary study (twice), and 2) 170 additional, independent pull requests. Of the 190 unique pull requests, 176 pull requests were evaluated by the crowd with high quality. Of those, 156 are new pull requests (dropping 14 from the sample of 170) and 34 of them are from the original TDH study (covering each of the 20 original pull requests at least once). The unqualified responses were a result of the redundant question format quality control approach, but an operational error led us to approve the tasks despite the poor comment quality, leaving us with a smaller data set for further analysis.

The crowd data includes qualitative information about the pull request discussion, such as whether there is a comment showing support, proposing an alternate solution, disapproving of the solution, and disapproving of the problem being solved. We refer to the data collected via the crowd as the *qualitative* pull request features.

V. RESULTS

The crowd data is substantially larger than the original data in terms of pull requests analyzed. The benefits of the larger sample size is two-fold. First, by using similar selection criteria in the replication study compared to the primary study, we are able to check the stability and external validity of the findings in the primary study using a much larger sample (RQ2). Second, in terms of informativeness, we can extract features from the crowd's answers, which is qualitative, and build models to predict pull request acceptance results. This allows us to compare the performance of models built with (a) some features identified as important in the primary study and (b) the features from related, quantitative works (RQ3).

A. RQ1: Can the crowd reproduce prior results with high quality?

RQ1 checks if our mapping in Section IV-B correctly captures the essence of the TDH study. Here, we used the results from the original 20 pull requests from TDH.

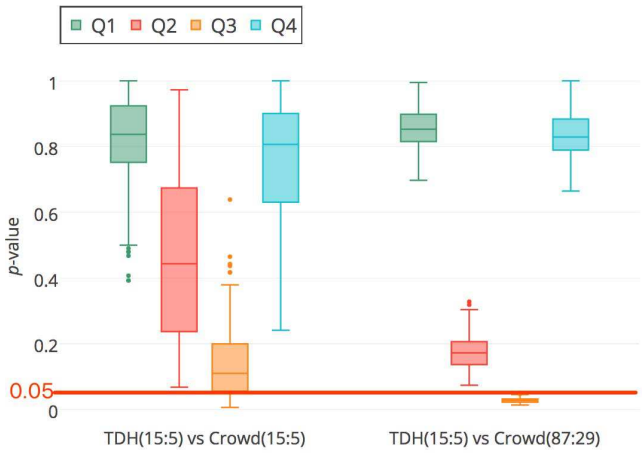

Fig. 2. Stability Checking: *p*-values for comparing unscaled and scaled crowd answers with answers from TDH. *p*-values computed using scipy.stats.chi2_contingency.

Table IV shows the $precision$, $recall$ and F_1 scores of the crowd working on the original tasks, using the prior TDH study as the ground truth. The *Questions* map to Table III. As seen in Table IV, the precision and recall of the crowd on the 20 original tasks is 80% and 74% respectively (so $F_1 \approx 77\%$). Based on prior work with data mining from software engineering data [49], [50], we find that these values represent a close correspondence between the TDH results and those from the crowd, hence indicating that the crowd can indeed perform the tasks with high quality.

We further manually examined the cases where the crowd disagreed with TDH, finding that sometimes TDH appears correct, and sometimes the crowd appears correct. For example, TDH classifies the 17th pull request they studied as no support, while the crowd found the comment from the user `drohthlis` saying *"This is great news!"*, which is an apparent indicator for the supporting this pull request after our examination. Another two cases are the 16th and 20th pull requests they studied. Crowd workers found clear suggestions for alternative solutions (i.e., *"What might be better is to ..."*, *"No, I think you can just push -f after squashing."*), which TDH does not find. While we did throw away some data due to quality, largely the crowd was able to replicate, and extend, some of the original TDH study.

As to the issues of speed and cost, TDH report that they required about 47 hours to collect interview data on 47 users within which, they investigated the practices about pull requests. TDH does not report the subsequent analysis time. By way of comparison, we spent $200, to buy 77 hours of crowd time. In that time, 190 (156 new ones and 34 from the primary study, which covers all 20 original pull requests) out of 210 published pull requests were validly analyzed.

In summary, we answer RQ1 in the affirmative.

B. RQ2: Are the primary study's results stable?

One motivation for this work was checking if crowdsourcing can scale and confirm the external validity of qualitative conclusions. This issue is of particular concern for crowdsourcing studies due to the subjective nature of the opinions from the crowd. If those opinions *increased* the variance of the collected data, then the more data we collect, the *less* reliable the conclusions.

To test for this concern, we compare the pull requests studied by the crowd (excluding the 20 gold tasks) with the 20 pull request studied by TDH. Among the original 20, 15 were merged and 5 were rejected. Given the 156 new pull requests, we first randomly select 15 merged and 5 rejected pull requests 100 times, so that we can compare these 2 independent samples at the same scale and with the same distribution of pull request acceptance outcomes. Then we run another 100 iteration for randomly selecting 87 merged and 29 rejected pull requests studied by the crowd, which still has the same distribution but at nearly six times larger scale. *p*-values are collected for each sample comparison in the 2 runs. We analyzed each of the four concepts from Table III separately (Q1-Q4).

Figure 2 shows the results of comparing pull requests from TDH and an independent sample with 2 different scales. As shown, Questions 1, 2, 4 are quite stable for both scales (i.e., p-values greater than 0.05 indicate no statistically significant difference with $\alpha = 0.05$). Moreover, Question 1 and 4 are becoming more stable when the scale becomes larger, while Question 2 becomes less stable at a larger scale. For Question 3, all of the p-values are lower than 0.05 at the large scale, though the median of its p-value is higher than 0.05 at the same scale as TDH. This may indicate that TDH did not cover enough pull requests to achieve a representative sample for the finding, which is mapped into Question 3 about disapproving comments. This may also indicate that TDH treated disapproving comments in GitHub pull requests differently than the crowd. As we have shown when answering RQ1, we have found several cases where the crowd and TDH disagree and the crowd is correct, though the crowd did have some wrong answers while TDH are correct. Note that the results are not to fault TDH, but serves as the evidence why we need to replicate and scale empirical studies.

Accordingly, we answer RQ2 in the affirmative. The results for Q1, Q2, and Q4 do not differ significantly between TDH and independent samples of the same size or of a larger size. The exception is Q3, for which the results differ significantly when scaling to a larger data set.

C. RQ3: How well can the qualitative and quantitative features predict PR acceptance?

The results are shown in Figure 3, expressed in terms of precision, recall, and the F_1 score; i.e. the harmonic mean of precision and recall, for each of three feature sets: quantitative, crowdsourced, and a combination. Note that the performances of the predictor using crowdsourced features are not as high

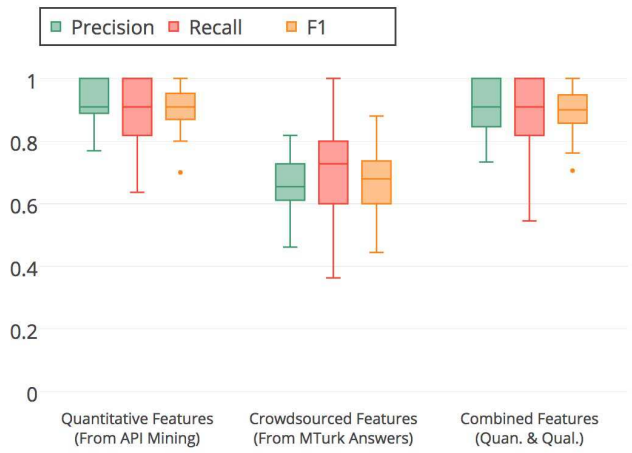

Fig. 3. Performance Comparison for Using different feature set to predict whether a pull requests will be accepted.

or as stable as the one built with quantitative features. We can see that:

- The selected quantitative features achieved F_1 score at 90% with a range of 20%;
- The selected crowdsourced features achieved lower F_1 score at 68% with a larger range.
- The combined selected features did better than just using qualitative; but performed no better than just using only the quantitative features.

At first glance, the models learned from crowdsourced features performed worse than using quantitative features extracted from numerous prior data mining studies, but this warrants further discussion. While Figure 3 shows the crowdsourced features generated from TDH results perform well on predicting the final acceptance with F_1 at 68%, it might be that considering more of the features from the original study could lead to even better results. For our study, we performed an independent partial replication and selected features that could be easily mapped to micro-questions, as stated in Section IV-B. That said, the instability observed for Q3 as the sample size increases (Figure 2) may be present for other features from the original study; further study is needed to expand the number of questions asked of the crowd to tease out this phenomenon.

In summary, the quantitative features extracted from dozens of recent papers on pull requests outperform the selected qualitative features we studied.

VI. Threats to Validity and Limitations

As with any empirical study, biases and limitations can affect the final results. Therefore, any conclusions made from this work must be considered with the following issues in mind.

Sampling bias: This threatens the external validity of any classification experiment. For example, the pull requests used here are selected using the rules described in IV-C. Only 170 additional highly discussed pull requests from active projects are sampled and analyzed, so our results may not reflect the patterns for all the pull requests. That said, we note that one reason to endorse crowdsourcing is that its sample size can be much larger than using just qualitative methods. For example, TDH reported results from just 20 pull requests.

Learner bias: For building the acceptance predictors in this study, we elected to use a decision tree classifier. We chose decision trees because it suits for small data samples and its results were comparable to the more complicated algorithms like Random Forest and SVM. Classification is a large and active field and any single study can only use a small subset of the known classification algorithms. Future work should repeat this study using other learners.

Evaluation bias: This paper uses *precision*, *recall* and F_1 *score* measures of predictor's performance. Other performance measures used in software engineering include accuracy and entropy. Future work should repeat this study using different evaluation biases.

Order bias: For the performance evaluation part, the order that the data trained and predicted affects the results. To mitigate this order bias, we run the 5-bin cross validation 10 times randomly changing the order of the pull requests each time.

VII. Implications

This paper presents one example in which crowdsourcing and data mining were used for partial replication and scaling of an empirical study. In our work, we use the experts to identify a set of interesting questions, and the crowd to answer those questions regarding a large data set. Then, data mining was used to compare the qualitative responses to quantitative information about the same data set. It is possible this workflow is indicative of a more general framework for scaling and replicating qualitative studies using the approach we used.

In the original qualitative TDH study, the authors found that 1) *supports*, 2) *alternate solutions*, 3) *disapproval for the proposed solutions*, and 4) *disapproval for the problems being solved* were important factors that guard pull requests' acceptance. In this scaled, crowdsourced replication, we found that factors 1, 2 and 4 still hold, but 3 was unstable. As shown in section V-B, we cannot tell whether the crowd are correct or TDH are correct, because we have observed situations where one is correct while the other is not for both sides. However, the implication here is that the combination of empirical methods allow us to pinpoint more precisely results that are steadfast against tests of external validity and the results that need further investigation.

In the end, the replication via quantitative study would have been impossible without the primary qualitative work, and we should make the best use of time-consuming qualitative work, instead of stopping after we get results from qualitative results (and vice versa). We find qualitative studies can inspire quantitative studies by carefully mapping out areas of concern. Primary qualitative studies can also provide the data needed to direct replications via quantitative crowdsourcing studies.

We also find a single primary qualitative study can direct the work of many quantitative studies, and our work is just one example of a partial replication study after TDH's qualitative work.

VIII. Related Work

There are many ways to categorize empirical studies in software engineering. Sjoberg, et al. [51] identify two general groups, primary research and secondary research. The most common primary research usually involves the collection and analysis of original data, utilizing methods such as experimentation, surveys, case studies, and action research. Secondary studies involve synthesis and aggregation of primary studies. According to Cohen [52], secondary studies can identify crucial areas and questions that have not been addressed adequately with previous empirical research.

Our paper falls into primary research that uses data from previously published studies. Using independent, partial replication, we check the external validity of some of the TDH results. That is, we explore whether claims for generality are justified, and whether our study yields the same results if we partially replicate it [53]. Our data collection applies the coding method, which is commonly used to extract values for quantitative variables from qualitative data in order to perform some type of quantitative or statistical analysis, with crowdsourcing to manage participant recruitment. Most of our data analysis falls into the category of replication for theory confirmation as we independently replicate parts of the results from TDH. However, in the data mining study, we introduce additional factors not studied in the original TDH study.

In software engineering, crowdsourcing is being used for many tasks traditionally performed by developers [54] including program synthesis [55], program verification [56], LTL specification creation [47], and testing [57], [58]. According to Mao, et al. [54], our work falls into the evaluation of SE research using crowdsourcing as the crowd is not performing software engineering tasks, but rather is used to evaluate software artifacts.

There are many reasons to use crowdsourcing in software engineering, and this space has been studied extensively [54]. Two important issues with crowdsourcing are *quality control* and *cost control*. While the crowd is capable of producing quality results, systems need to be in place to monitor result quality. Cost has a dependent relationship on the quality control measures; a lack of quality control can lead to results that must be thrown out and repeated, hence increasing the cost.

Quality control in crowdsourcing has been studied extensively within and beyond software engineering applications (e.g., [44]–[46], [59]–[61]). One way to address quality control is to use a golden set of questions [45], [46] for which the answer is known. Workers that perform poorly on the golden set are eliminated, which is also one of the strategies we take in this paper. Alternatively, tasks can be assigned to multiple workers and their results aggregated (e.g., in the TURKIT system [62], one task is performed iteratively, and each worker is asked to improve on the answer of the former, or in AutoMan [63], each task is performed by crowd members until statistical consensus is reached). The tradeoffs are in cost and volatility. The gold questions can be checked automatically, yielding low cost, but there may be high volatility in the results from a single individual. When using multiple people and aggregating the results, volatility is lower as the crowd is likely to converge [64], but the cost will be higher due to the required replication per question. Other quality control techniques include redundant question formats [44], notifications to workers about the value of their work [59], answer conflict detectors [60], and random click detection [61].

Cost control is important as commercial crowdsourcing platforms are not free. Economic incentives for crowd workers strongly effect crowd response quality [65]–[72]. To keep the quality high, payments need to be high enough to entice participation [73] but low enough to keep study costs reasonable.

IX. Conclusion

Crowdsourcing and data mining can be used to effectively reduce the effort associated with the partial replication and enhancement of qualitative studies.

For example, in this paper:

- We presented an empirical study on scaling and extending a primary study on pull request acceptance. Using the insights learned from the primary study, we designed a scaled study with different empirical methodology.
- To scale and extend the results from the original study, we used crowdsourcing.
- Then, we applied data mining to determine if the qualitative features fared better or worse than quantitative features over the same artifacts. We show that, with results and data from TDH's primary study, it is possible to map some of their insights into micro-questions for crowd workers and expand the data they studied to a larger scale.
- Further, in that second study, we could build better predictors that seen before.

We conjecture that this case study is representative of an underlying methodology for scaling and extending primary qualitative studies that require expert opinions. The long-term goal of this work is to encourage more studies that replicate parts of primary qualitative studies and use the gained insights to design subsequent studies. It is through independent replication with different methodologies that we can move closer to building software engineering theories. For that task, collaborators would be welcomed.

Acknowledgements

The work is partially funded by NSF awards #1506586, #1302169, #1749936, and #1645136.

REFERENCES

[1] B. Ray, D. Posnett, V. Filkov, and P. Devanbu, "A large scale study of programming languages and code quality in github," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2014, pp. 155–165.

[2] C. O'Neil, *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Crown Publishing Group (NY), 2016.

[3] F. Shull, V. Basili, J. Carver, J. C. Maldonado, G. H. Travassos, M. Mendonça, and S. Fabbri, "Replicating Software Engineering Experiments: Addressing the Tacit Knowledge Problem," in *ISESE '02: Proceedings of the 2002 International Symposium on Empirical Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2002, p. 7.

[4] F. J. Shull, J. C. Carver, S. Vegas, and N. Juristo, "The role of replications in empirical software engineering," *Empirical Softw. Engg.*, vol. 13, no. 2, pp. 211–218, Apr. 2008.

[5] J. Tsay, L. Dabbish, and J. Herbsleb, "Let's talk about it: evaluating contributions through discussion in github," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, 2014, pp. 144–154.

[6] "Amazon Mechanical Turk," www.mturk.com/mturk/welcome, June 2010. [Online]. Available: www.mturk.com/mturk/welcome

[7] K. T. Stolee and S. Elbaum, "On the use of input/output queries for code search," in *International Symposium. on Empirical Soft. Eng. and Measurement*, October 2013.

[8] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in github: transparency and collaboration in an open software repository," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*. ACM, 2012, pp. 1277–1286.

[9] G. Gousios, A. Zaidman, M.-A. Storey, and A. Van Deursen, "Work practices and challenges in pull-based development: the integrator's perspective," in *Proceedings of the 37th International Conference on Software Engineering-Volume 1*. IEEE Press, 2015, pp. 358–368.

[10] G. Gousios, M.-A. Storey, and A. Bacchelli, "Work practices and challenges in pull-based development: the contributor's perspective," in *Proceedings of the 38th International Conference on Software Engineering*. ACM, 2016, pp. 285–296.

[11] J. Marlow, L. Dabbish, and J. Herbsleb, "Impression formation in online peer production: activity traces and personal profiles in github," in *Proceedings of the 2013 conference on Computer supported cooperative work*. ACM, 2013, pp. 117–128.

[12] A. Begel, J. Bosch, and M.-A. Storey, "Social networking meets software development: Perspectives from github, msdn, stack exchange, and topcoder," *IEEE Software*, vol. 30, no. 1, pp. 52–66, 2013.

[13] N. McDonald and S. Goggins, "Performance and participation in open source software on github," in *CHI'13 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2013, pp. 139–144.

[14] R. Pham, L. Singer, O. Liskin, F. Figueira Filho, and K. Schneider, "Creating a shared understanding of testing culture on a social coding site," in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 2013, pp. 112–121.

[15] J. Tsay, L. Dabbish, and J. Herbsleb, "Influence of social and technical factors for evaluating contribution in github," in *Proceedings of the 36th international conference on Software engineering*. ACM, 2014, pp. 356–366.

[16] G. Gousios, M. Pinzger, and A. van Deursen, "An exploration of the pull-based software development model." ICSE, 2013.

[17] J. T. Tsay, L. Dabbish, and J. Herbsleb, "Social media and success in open source projects," in *Proceedings of the ACM 2012 conference on computer supported cooperative work companion*. ACM, 2012, pp. 223–226.

[18] Y. Yu, H. Wang, V. Filkov, P. Devanbu, and B. Vasilescu, "Wait for it: Determinants of pull request evaluation latency on github," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. IEEE, 2015, pp. 367–371.

[19] Y. Zhang, G. Yin, Y. Yu, and H. Wang, "Investigating social media in github's pull-requests: a case study on ruby on rails," in *Proceedings of the 1st International Workshop on Crowd-based Software Development Methods and Technologies*. ACM, 2014, pp. 37–41.

[20] M. M. Rahman and C. K. Roy, "An insight into the pull requests of github," in *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 2014, pp. 364–367.

[21] Y. Takhteyev and A. Hilts, "Investigating the geography of open source software through github," 2010.

[22] F. Thung, T. F. Bissyande, D. Lo, and L. Jiang, "Network structure of social coding in github," in *Software maintenance and reengineering (csmr), 2013 17th european conference on*. IEEE, 2013, pp. 323–326.

[23] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining github," in *Proceedings of the 11th working conference on mining software repositories*. ACM, 2014, pp. 92–101.

[24] K. Blincoe, J. Sheoran, S. Goggins, E. Petakovic, and D. Damian, "Understanding the popular users: Following, affiliation influence and leadership on github," *Information and Software Technology*, vol. 70, pp. 30–39, 2016.

[25] G. Gousios, M. Pinzger, and A. v. Deursen, "An exploratory study of the pull-based software development model," in *Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 345–355.

[26] Y. Yu, H. Wang, G. Yin, and C. X. Ling, "Reviewer recommender of pull-requests in github." *ICSME*, vol. 14, pp. 610–613, 2014.

[27] B. Vasilescu, S. Van Schuylenburg, J. Wulms, A. Serebrenik, and M. G. van den Brand, "Continuous integration in a social-coding world: Empirical evidence from github," in *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*. IEEE, 2014, pp. 401–405.

[28] Y. Yu, H. Wang, G. Yin, and C. X. Ling, "Who should review this pull-request: Reviewer recommendation to expedite crowd collaboration," in *2014 21st Asia-Pacific Software Engineering Conference*, vol. 1. IEEE, 2014, pp. 335–342.

[29] G. Gousios and A. Zaidman, "A dataset for pull-based development research," in *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 2014, pp. 368–371.

[30] D. Pletea, B. Vasilescu, and A. Serebrenik, "Security and emotion: sentiment analysis of security discussions on github," in *Proceedings of the 11th working conference on mining software repositories*. ACM, 2014, pp. 348–351.

[31] J. Brunet, G. C. Murphy, R. Terra, J. Figueiredo, and D. Serey, "Do developers discuss design?" in *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 2014, pp. 340–343.

[32] R. Padhye, S. Mani, and V. S. Sinha, "A study of external community contribution to open-source projects on github," in *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 2014, pp. 332–335.

[33] E. Van Der Veen, G. Gousios, and A. Zaidman, "Automatically prioritizing pull requests," in *Proceedings of the 12th Working Conference on Mining Software Repositories*. IEEE Press, 2015, pp. 357–361.

[34] G. Mathew, A. Agrawal, and T. Menzies, "Trends in topics in software engineering conferences, 1992 to 2016," Submitted to IST. Available from tiny.cc/citemap.

[35] G. Gousios, "The ghtorrent dataset and tool suite," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, ser. MSR '13, 2013, pp. 233–236.

[36] K. Mao, L. Capra, M. Harman, and Y. Jia, "A survey of the use of crowdsourcing in software engineering," Tech. Rep., 2015, cS Technical Report RN/15/01, University College London.

[37] K. T. Stolee and S. Elbaum, "Exploring the use of crowdsourcing to support empirical studies in software engineering," in *International Symposium on Empirical Software Engineering and Measurement*, 2010.

[38] Z. P. Fry, B. Landau, and W. Weimer, "A human study of patch maintainability," in *Proceedings of the 2012 International Symposium on Software Testing and Analysis*, ser. ISSTA 2012. New York, NY, USA: ACM, 2012, pp. 177–187. [Online]. Available: http://doi.acm.org/10.1145/2338965.2336775

[39] M. Buhrmester, T. Kwang, and S. D. Gosling, "Amazon's mechanical turk a new source of inexpensive, yet high-quality, data?" *Perspectives on psychological science*, vol. 6, no. 1, pp. 3–5, 2011.

[40] A. J. Berinsky, G. A. Huber, and G. S. Lenz, "Evaluating online labor markets for experimental research: Amazon. com's mechanical turk," *Political Analysis*, vol. 20, no. 3, pp. 351–368, 2012.

[41] G. Paolacci, J. Chandler, and P. G. Ipeirotis, "Running experiments on amazon mechanical turk," *Judgment and Decision making*, vol. 5, no. 5, pp. 411–419, 2010.

[42] P. G. Ipeirotis, "Demographics of mechanical turk," 2010.

[43] M. S. Silberman, B. Tomlinson, R. LaPlante, J. Ross, L. Irani, and A. Zaldivar, "Responsible research with crowds: Pay crowdworkers at least minimum wage," *Commun. ACM*, vol. 61, no. 3, pp. 39–41, Feb. 2018.

[44] K. T. Stolee, J. Saylor, and T. Lund, "Exploring the benefits of using redundant responses in crowdsourced evaluations," in *Proceedings of the Second International Workshop on CrowdSourcing in Software Engineering*. IEEE Press, 2015, pp. 38–44.

[45] O. Alonso and R. Baeza-Yates, "Design and implementation of relevance assessments using crowdsourcing," in *European Conference on Information Retrieval*. Springer, 2011, pp. 153–164.

[46] C. Sarasua, E. Simperl, and N. F. Noy, "Crowdmap: Crowdsourcing ontology alignment with microtasks," in *International Semantic Web Conference*. Springer, 2012, pp. 525–541.

[47] P. Sun, C. Brown, K. T. Stolee, , and I. Beschastnikh, "Back to the future: specification mining using crowd intelligence," in *International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2019.

[48] M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, The University of Waikato, 1999.

[49] C. Bird, T. Menzies, and T. Zimmermann, *The Art and Science of Analyzing Software Data*. Elsevier, 2015.

[50] W. Fu, T. Menzies, and X. Shen, "Tuning for software analytics: Is it really necessary?" *Information & Software Technology*, vol. 76, pp. 135–146, 2016.

[51] D. I. Sjoberg, T. Dyba, and M. Jorgensen, "The future of empirical methods in software engineering research," in *Future of Software Engineering, 2007. FOSE'07*. IEEE, 2007, pp. 358–378.

[52] B. P. Cohen, *Developing sociological knowledge: Theory and method*. Wadsworth Pub Co, 1989.

[53] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting empirical methods for software engineering research," in *Guide to advanced empirical software engineering*. Springer, 2008, pp. 285–311.

[54] K. Mao, L. Capra, M. Harman, and Y. Jia, "A survey of the use of crowdsourcing in software engineering," *RN*, vol. 15, no. 01, 2015.

[55] R. A. Cochran, L. D'Antoni, B. Livshits, D. Molnar, and M. Veanes, "Program boosting: Program synthesis via crowd-sourcing," in *Proceedings of the 42Nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ser. POPL '15, 2015, pp. 677–688.

[56] T. W. Schiller and M. D. Ernst, "Reducing the barriers to writing verified specifications," *SIGPLAN Not.*, vol. 47, no. 10, pp. 95–112, Oct. 2012.

[57] E. Dolstra, R. Vliegendhart, and J. Pouwelse, "Crowdsourcing gui tests," in *Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on*, March 2013, pp. 332–341.

[58] 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*, ser. EICS '13. New York, NY, USA: ACM, 2013, pp. 255–264. [Online]. Available: doi.acm.org/10.1145/2494603.2480303

[59] J. A. Krosnick, "Response strategies for coping with the cognitive demands of attitude measures in surveys," *Applied cognitive psychology*, vol. 5, no. 3, pp. 213–236, 1991.

[60] A. Shiel, "Conflict crowdsourcing: Harnessing the power of crowdsourcing for organizations working in conflict," 2013.

[61] S.-H. Kim, H. Yun, and J. S. Yi, "How to filter out random clickers in a crowdsourcing-based study?" in *Proceedings of the 2012 BELIV Workshop: Beyond Time and Errors - Novel Evaluation Methods for Visualization*, ser. BELIV '12, 2012, pp. 15:1–15:7.

[62] G. Little, "Turkit: Tools for iterative tasks on mechanical turk," in *Visual Languages and Human-Centric Computing, 2009. VL/HCC 2009. IEEE Symposium on*, Sept 2009, pp. 252–253.

[63] D. W. Barowy, C. Curtsinger, E. D. Berger, and A. McGregor, "Automan: A platform for integrating human-based and digital computation," *SIGPLAN Not.*, vol. 47, no. 10, pp. 639–654, Oct. 2012.

[64] T. Menzies, E. Kocagüneli, L. Minku, F. Peters, and B. Turhan, "Chapter 20 - Ensembles of Learning Machines," in *Sharing Data and Models in Software Engineering*, 2015, pp. 239–265. [Online]. Available: www.sciencedirect.com/science/article/pii/B9780124172951000205

[65] W. Mason and D. J. Watts, "Financial incentives and the performance of crowds," *ACM SigKDD Explorations Newsletter*, vol. 11, no. 2, pp. 100–108, 2010.

[66] G. Goel, A. Nikzad, and A. Singla, "Mechanism design for crowdsourcing markets with heterogeneous tasks," in *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.

[67] A. Mao, E. Kamar, Y. Chen, E. Horvitz, M. E. Schwamb, C. J. Lintott, and A. M. Smith, "Volunteering vs. work for pay: incentives and tradeoffs in crowdsourcing," in *Conf. on Human Computation*, 2013.

[68] A. Kittur, E. H. Chi, and B. Suh, "Crowdsourcing user studies with mechanical turk," in *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 2008, pp. 453–456.

[69] W. A. Mason and S. Suri, "How to use mechanical turk for cognitive science research," in *Proceedings of the 33rd annual conference of the cognitive science society*, 2011, pp. 66–67.

[70] K. Mao, Y. Yang, M. Li, and M. Harman, "Pricing crowdsourcing-based software development tasks," in *Proceedings of the 2013 International Conference on Software Engineering*, ser. ICSE '13, 2013, pp. 1205–1208.

[71] M. Yin, Y. Chen, and Y.-A. Sun, "Monetary interventions in crowdsourcing task switching," in *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.

[72] J. Wang, P. G. Ipeirotis, and F. Provost, "Quality-based pricing for crowdsourced workers," 2013.

[73] R. Vinayak and B. Hassibi, "Clustering by comparison: Stochastic block model for inference in crowdsourcing," in *Workshop Machine Learning and Crowdsourcing*, 2016.