Playing Planning Poker in Crowds: Human Computation of Software Effort Estimates

Mohammed Alhamed
School of Computing Science
University of Glasgow
Glasgow, United Kingdom
Mohammed.Alhamed@glasgow.ac.uk

Tim Storer
School of Computing Science
University of Glasgow
Glasgow, United Kingdom
Timothy.Storer@glasgow.ac.uk

Abstract—**Reliable cost effective effort estimation remains a considerable challenge for software projects. Recent work has demonstrated that the popular Planning Poker practice can produce reliable estimates when undertaken within a software team of knowledgeable domain experts. However, the process depends on the availability of experts and can be time-consuming to perform, making it impractical for large scale or open source projects that may curate many thousands of outstanding tasks. This paper reports on a full study to investigate the feasibility of using crowd workers supplied with limited information about a task to provide comparably accurate estimates using Planning Poker. We describe the design of a Crowd Planning Poker (CPP) process implemented on Amazon Mechanical Turk and the results of a substantial set of trials, involving more than 5000 crowd workers and 39 diverse software tasks. Our results show that a carefully organised and selected crowd of workers can produce effort estimates that are of similar accuracy to those of a single expert.**

I. Introduction

Reliable software task estimation remains a critical factor in the success of software projects. The most recent edition of the Standish Group's CHAOS survey of software industry practitioners to be made publicly available states that around half of all software projects are delivered later than intended [1]. In older work, Emam and Koru [2] found that around 17% of cancelled projects were over schedule and 28% were over budget. Even worse, underestimation of effort may cause developers to rush, potentially resulting in unreliable software and increased error proneness [3].

Several authors have argued that reliable software task estimation is just as important for open source software (OSS) projects [4, 5]. For example, Koch [4] argues that established OSS projects share many of the characteristics of commercial developments that benefit from reliable task estimations, such as project roadmaps, release schedules and contributor onboarding activities. Separately, Asundi [6] notes that OSS projects are often not developed in isolation from commercial activities. Many organisations will release part or all of their code base as OSS, or depend on a community maintained OSS system. Obtaining reliable estimates for tasks in these projects helps dependents to understand when new features might be available or bugs fixed.

The problem: As described by more than one study, effort estimation is a challenging task in controlled and centrally managed commercial software development houses [1, 2, 7].

Such difficulties originate from an interplay of factors including software novelty, rapid changes in development technologies, team competences and the need for creativity in software development [6]. Generally, the amount of subjectivity and creativity make predicting the final form of the software product difficult, and thus, complicate software development planning and estimation.

This difficulty is exacerbated in OSS. Such projects typically adopt a 'Bazaar' development model, in which a large number of different contributors may complete tasks at different points in time [8], joining and leave a project as their interests change. Lee et al. [9] found that a substantial number of OSS contributors on Github only submitted a single pull request in the lifetime of a project. This fluidity and churn makes the development of the necessary project stability and expertise for reliable estimates difficult. For example, only 6.79% of the issues reported in Apache Foundation issue tracker[1] has been annotated with estimated or actual efforts. Without such information it becomes difficult to plan software development work in these contexts [4, 10].

Describing the wider problem of task triage, Hooimeijer and Weimer [11] note that the number of bug reports for popular projects typically outstrip the time available from domain experts to triage them. Reviewing the state (2019) of some popular open source projects illustrates the scale of this challenge. The Linux Kernel, Firefox Web Browser and JBoss projects respectively have 6456, 11751 and 17032 new issues to be investigated.

Our motivation: Software effort estimation (SEE) is at the core of software planning, yet existing methods depend on the availability of a stable group of project experts and are not well suited to the dimensions of scale of open source projects.

Planning Poker [12] is an expert-based effort estimation method used in Agile development methods, such as Scrum [13], that has become a popular approach to effort estimation for software tasks. The most recently published *State of Agile Report* states that the practice is used by 61% of the software

[1]https://issues.apache.org/jira/browse

teams surveyed [14]. Several studies in the literature have also reported that planning poker can provide reliable estimates [15, 16, 17] as compared with single expert estimation.

Similar to other 'Delphi' methods [18], conventional Planning Poker depends upon the availability of a team of experts to produce estimates in an iterative consensus building process. Running an expert estimation method such as planning poker is therefore costly. It requires a meeting of a team of relevant experts to be coordinated, which limit Planning Poker scalability and efficiency. Cohn's [19] description of Planning Poker process suggests that a team should expect to spend between five and ten minutes per task, allowing the team to estimate up to ten stories over a one hour session. Applying this guideline to the Linux Kernel backlog would require a Planning Poker session of approximately half a year. Even if only a percentage of these issues are prioritised for resolution, the dependency on expert availability imposes significant scaling restrictions.

In a review of Surowiecki's [20] keynote talk on the *Wisdom of the Crowds* at Agile 2008, Grenning wrote:

> "I was wondering how Wisdom of Crowds would relate to people on agile teams doing estimation and planning. I was specifically interested in how his research applied to Planning Poker, a practice used throughout the world on agile teams" [21]

Surowiecki's thesis is that large numbers of relatively inexpert workers can perform as well as small groups of experts, if the group activity is appropriately structured. Further, crowdsourcing platforms, such as Mechanical Turk [22] have significantly eased the task of recruiting large numbers of workers at relatively low cost.

Several studies have demonstrated that crowds can perform effectively on a variety of different types of software project task, including requirements management [23], software development [24], and testing [25]. This paper therefore explores Grenning's question to determine whether an inexpert crowd (either hired as in crowdsourcing markets or volunteered as in OSS projects) can produce reliable software task effort estimates and address the scalability of expert based Planning Poker. Since access to OSS community is limited, we opted to used crowdsourcing market as a similar OSS environment. We hypothesize that by designing a process that applies human computation[26] to an expert estimation method we can achieve effort estimation that is of comparable accuracy to that of small-group Planning Poker, but at scale.

Proposed method: Given our focus on applying human computation, we address the following research question:

> *RQ: Given a software task that required between half a day and two weeks effort, are crowdsourced effort estimates ofcomparable accuracy to those of experts?*

To address this question, we have developed and evaluated *Crowd Planning Poker* (CPP), an estimation practice that can be performed by inexpert crowd workers. We implement the orchestration of worker activity on the Mechanical Turk platform. Each crowd worker is presented with initial information about a task to be estimated and then asked to supply a categorical effort estimate and a justification. Once sufficient estimates are received, the consensus amongst the crowd is calculated. If consensus has been achieved, the process ends. Otherwise, a further round of estimation is undertaken, with additional information concerning the range of responses and justifications from the previous round provided to the workers. The iterative process continues until the crowd reaches consensus, or a limit on rounds is reached. CPP also incorporates a mechanism for filtering low quality estimates based on an evaluation of the behaviour of crowd workers and the justifications that they supply for their estimates.

Contribution: As far as we are aware, Grenning never followed up on his inquiry and we are unaware of any other attempt in the literature, except for our own pilot study [27]. Our work is therefore a continuation of the work reported there to investigate the application of crowdsourcing to software development task estimation using Planning Poker. Unlike the pilot study, the work includes full evaluation of the approach on a diverse range of thirty-nine (39) issues, comprising both feature requests and bug fixes. The issues are selected from three different open source projects, JBoss, Spring and Apache. In total, 80 Crowd Planning Poker rounds were executed and 807 estimates were received. Actual effort for task completion report in the issue repositories ranged from half a day through to two weeks. The results of the evaluation demonstrates that crowd workers can produce estimates of comparable accuracy to those of experts. A replication pack is available for inspection containing all the results generated for the experiment[2].

This paper is structured as follows. Section II reviews the present research in the existing literature. Section III presents the experimental design for investigating the efficacy of CPP. The results of the experiments are then presented in Section IV. Finally, Section V summarises our conclusions from the study and our reflections on the next steps in the research.

II. Related Work

Planning Poker has been the subject of a number of empirical studies in the academic literature. Moløkken-Østvold et al. [16] found that using Planning Poker to combine estimates produced better results in comparison to unstructured or mechanical methods. More recently, Mahnic and Hovelja [15] found a similar result in a study of 13 teams of students in a software engineering course. In both cases, the studies found that diverse groups of estimators result in more accurate estimates. Gandomani et al. [28] also conducted an empirical study of Planning Poker, comparing it to Wideband Delphi and expert estimation, and concluded that Planning Poker performed marginally better. Gandomani et al. also noted that both estimation methods were useful in reducing underestimation.

Several studies have attempted to apply automated, statistical or machine learning techniques to software task effort estimation. Such methods are potentially attractive, since they

[2]https://github.com/crowd-planning-poker/cpp-30-issue-replication-pack

could substantially reduce the cost of producing estimates, compared with expert based approaches. Early approaches such as Function Point analysis [29] and COCOMO [30, 31] developed cost models based on empirical analysis of large data sets of software projects. However, both these approaches require considerable expert judgement in producing estimates.

More recently, several attempts have been made to apply machine learning methods [32]. However, a comparison by Usman et al. [17] suggested that expert estimation techniques still outperform machine learning based approaches. Similarly, Wen et al. [32] found only two studies [33, 34] that reported that machine learning based approached outperform experts and neither study compared machine learning techniques to Planning Poker or other Delphi-like methods. Thus to date, a scalable method for producing reliable software task estimates remains elusive.

Apart from the pilot study [27], no previous research has investigated the application of human computation to Planning Poker. Human computation could present a solution to the issue of scale, as several studies have demonstrated that the technique can be applied to a wide variety of other software engineering practices [35], including requirements elicitation [23], source code implementation [24] and usability testing [25].

A particular concern in the application of crowdsourcing, however, is to ensure the quality of work produced by crowd workers. Numerous methods have been investigated, including using a Gold Standard [36], machine learning classifiers [37], another crowd to assess the quality of the work [38], and associated data such as worker behaviour [39]. None of these approaches are entirely satisfactory in the context of software task estimation. In general, there isn't an oracle available for generating a gold standard estimate for a software task. In addition, crowd workers may not return to perform repeated estimation tasks, making a quality analysis based on previous performance more difficult. Finally, employing a further crowd of workers to assess the quality of a submission while feasible, increases the cost of an estimate.

Alternative approaches seek proxy information concerning the quality of a submission, rather than the submission itself. Of particular relevance, McDonnell et al. [40] and Kutlu et al. [41] proposed that crowd workers be asked to supply a rationale for their decision alongside the supplied value. This work shows that obtaining rationales improves the quality of judgements without a substantial increase in task time. In addition, Dumitrache et al. [42] showed that rationales are useful for uncovering the reasons for subjective disagreement amongst crowd workers and reaching a subsequent consensus.

Separately, Rzeszotarski and Kittur [43] and Kazai and Zitouni [44] report the analysis of logs of crowd worker interactions with the task user interface in order to model their behaviour and engagement with the task. In these approaches, worker interactions with the computer interface are captured. A machine learning classifier is trained to predict the quality of a worker's submission based on the captured behaviour.

III. Experimental Design

In this section, we present our experimental design to compare the performance of a crowd in producing software task estimates with those produced by project expert estimation.

We explain the metric used to compare accuracy of estimates between experts and crowds;

describe the software tasks that formed the experimental objects of our study; rehearse the in-person Planning Poker practice and describe our Crowd Planning Poker (CPP) adaptation; and our technique for filtering estimates provided by the crowd workers based on the quality of an associated justification and their behaviour. We also briefly describe the outcome of an initial pilot study [27] that assisted in the design of the CPP process.

A. Baseline and Ground Truth

The purpose of the experiment is to determine whether the CPP practice performed by crowd workers can produce estimates comparable to those of experts. Therefore, it was necessary to obtain a set of software tasks that had been annotated with both an expert estimated and actual cost effort, providing an experimental baseline and ground truth, respectively. Three open source projects JBoss, Apache and Spring Integration were found to satisfy these criteria.

After searching the project's issue tracker systems 419 issues were found to be annotated by an expert time estimate and an actual spent time in person-hours. Although these communities have published their issue reporting documentation, the method for producing either the estimate or calculating the actual time cost are not stated. The researchers attempted to contact several members of the communities to determine the exact estimation process. As explained by those who responded, they rely on their experience to predict the issues estimates. For example, one response was:

> "We tried to experiment, but always fallen back into "guts feeling" based on experience and only sometimes challenge each other. Me as a tester usually take into account time for: (a) learning about issue and deployment needed * (b) deploy manually first (c) reproduce issue/fix (d) automate deployment, create tests (e) whatever else is needed (f) extra time buffer to mitigate unexpected problems (g) holidays or people time-off should be taken into account as well -> might be covered by f) already, but you have to think about it
>
> * This might take a long time, depending on difficulty of environment (Setup LDAP, Kerberos, DNS, Whatever service needs to support given product/tool)"

Moreover, the issues history log confirms that costs have been determined by one or more of the issue assignees. Therefore, the estimated time cost reported by the development team on the issue is referred to as an expert estimate in this study.

Instead of using a literal person-hours, Planning Poker is often used with a *relative* unit for cost estimation such as story

TABLE I
ADOPTED TIME-BASED CATEGORIES AND THEIR BOUNDARIES FOR A SOFTWARE TASK EFFORT ESTIMATE.

Category	Low	Middle	High
One hour	0	1	1
Half a day	2	4	5
A day	6	8	10
Half a week	11	20	30
A week	31	40	60
Two weeks	61	80	120

TABLE II
COMPARISON OF ESTIMATION ERROR METRICS OF THE WHOLE POPULATION (419 ISSUES), FILTERED SET OF ISSUES (126) AND SELECTED SAMPLE (39).

	#	Mean Absolute Error (hours)	Median Magnitude of Relative Error	Mean Magnitude of Relative Error
All	419	±29.3	128%	2475%
Filtered	126	±12.0	100%	773%
Sample	39	±10.5	90%	440%

points [12]. Cohn [19] states that approximate person-effort categories are more appropriate because it is often unrealistic to expect person-hour precision estimates to be accurate for software tasks. Further, Cohn [19] argues that teams eventually develop a tacit interpretation of the relationship between the relative categorical estimate and actual person-time costs, as the completed tasks are compared to the team's available person-hour budget over a number of sprints. In addition, estimate categories often adopt a metaphor that suggests increasing uncertainty with estimate magnitude. For example, Grenning [12] suggests using a Fibonacci sequence to indicate the margin of error between estimate sizes increases with magnitude.

To map this approach to CPP it was necessary to employ categorical units of which the crowd workers were likely to have a shared understanding without prior communication. Therefore, person-hour costs reported on the issues were translated into approximate person-day and person-week categories, labelled as one hour, half a day, one day, half a week, one week, two weeks and more than two weeks. The translation followed the same scheme as in the community issue tracker system (JIRA), where a working day is equal to 8 hours and a working week equal to 40 hours. This enabled comparison between CPP estimates and the person-hour costs reported on the issues (Table I). To draw boundaries between the scale categories a relative mid point between the two categories was selected. Table I illustrates the low, middle, and high possible person-hour for each category.

Before commencing with the experiment the set of issues was filtered to avoid issues that:

- required less than 30 minutes or more than two weeks to complete (119 issues);
- contained less than 20 words in the description (112 issues); or
- had received no comments and so were assumed to not be of interest to the community (203 issues).

Some issues were removed due to more than one filter. After the filtering step there were 126 issues left as candidates. Thirty-nine (39) issues were randomly selected from the filtered data set for use in the experiment. To ensure that a diverse range of effort magnitudes were included, issues were first organised into effort categories ranging from one hour through to two weeks, as described above. Issues were then selected randomly from these categories for inclusion in the sample. URLs for the selected issues are included in the replication pack. Issues selected were found to comprise a mixture of bugs, feature requests and enhancements.

B. Accuracy Measurements

Following [32] and Usman et al. [17], the Magnitude of Relative Error (MRE) and its mean (MMRE) and median (MdMRE) are used to express the accuracy of the estimates in the experiments. MRE is calculated as:

$$MRE = |\frac{e - e'}{e}| \qquad (1)$$

where, e is the person-hour actual effort as recorded on the issue tracker; and e' is the person-hour effort estimate either as recorded by the expert on the issue tracker or produced by the crowd during the experiment.

MMRE and MdMRE are used to represent a summary of the error for either crowd or expert estimates in the result tables. However, before proceeding, it was necessary to check whether the expert estimates for the selected issues were representative of the whole data set. The selected sample might represent an artificially low baseline if the estimates they contain are less accurate than those for the population of issues as a whole.

To do this, the mean absolute error, MMRE and MdMRE were calculated for the three sets of issues (all estimated issues, filtered issues, random sample), as shown in Table II. For expert estimates the MRE was calculated directly from the effort estimates reported in the respective project's issue tracker. For crowd workers, the categorical estimates from individual estimates were translated back to person-hours at the mid point for the category as shown in Table I.

The results show that the average estimation performance by experts in the sample is slightly better than for the whole or filtered set of issues. This assessment demonstrates that the selected baseline (expert estimates in the sample) is suitable for use in the study.

C. Crowd Planning Poker Workflow

Planning Poker [12] is an expert estimation technique, similar to older methods such as Delphi [45] and popularly associated with the Scrum software development process [13]. The objective of the method is to achieve consensus amongst a group of experts, whilst minimising bias that might arise from individual estimates and ensuring that conflicting opinions are discussed and resolved.

Each member of the development team has a set of cards, each labelled with a possible cost estimate. Different units of costs can be used, including t-shirt size, person-hours or story points. Cohn [19] proposes using story points as a means of comparing relative user story complexity, rather than producing an absolute estimate. In Cohn's approach, story point cards are labelled 0, 1, 2, 3, 5, 8, 13. A final card labelled with infinity can be used to signal that the task under consideration is too complex to be reliably estimated.

Team members start estimating the effort for a task issue individually, and pick the card with an appropriate label to make an estimate. Then, the team members reveal their cards simultaneously and check for estimation consistency. If there is no consensus between the estimates, then the team members explain their views to each other. In particular, the estimators with the lowest and highest estimates are asked to explain their reasoning. Additional rounds of estimation and discussion are then performed until they reach a consistent estimation about the issue. Cohn [19] recommends that if consensus hasn't been reached after three rounds of estimation then the team should revisit the issue separately.

The adaptation of Planning Poker to crowdsourcing in our CPP design is shown in the diagram in Figure 1. First, initial information about the issue (title and description) is presented to the crowd worker on a web based user interface. Additional contextual information can be revealed by clicking on a corresponding button. This information includes contextual project details, such as definitions of ambiguous terms and abbreviations, or more information about project specific terms, such as the name of a software component that appears in the description. The crowd worker can also access comments that were posted on the issue. Further information can be searched by the developer using a search dialog provided on the user interface.

Next, the worker is asked to select a category for their estimate and provide an accompanying justification. The CPP application collects this information, along with a log of the worker's behaviour on the CPP user interface. A quality evaluation is then performed on the submitted information according to the procedure summarized in Section III-E. Submissions that are classified as low quality are eliminated from any further use.

Once a sufficient number of estimates are received, the consensus of the crowd worker estimates is calculated (Fleiss' Kappa %) to determine if another round is required. If a further round is required, the crowd workers are offered a summary of the low, median and high, estimates from the previous round, along with the justifications provided. The provision of this supplementary information mimics the design of in-person Planning Poker. We refer to the summary of previous round as a seed answer that is fed back to the crowd workers, in a similar way to the discussion that takes place in in-person Planning Poker.

Crowd workers (the study subjects) were recruited from the Amazon Mechanical Turk platform [22]. Only workers with a self-declared experience of at least two years of software engineering were permitted to participate. Each estimation session employed a group of between 5 and 15 workers.

A custom web application has been developed to implement the experiment, using the Mechanical Turk API to interact with the market platform. All data, including worker interactions with the user interface, the estimate and the justification are captured using this application.

D. Pilot Study Summary

Having no prior literature to rely on, it was decided to implement a small pilot study of CPP, prior to proceeding to the full study [27]. The objective of the pilot study was to test the design of the CPP workflow. It was also desirable to test the selected size of the crowd to determine if the number of crowd workers was sufficient to provide a reliable estimate and to explore the quality of estimates and justifications provided by workers.

The selected nine issues for the pilot experiment has an average of 10 comments by different developers for each issue. The Flesch-Kincaid readability scale grade for the description averaged at 12. Overall, the design of the pilot study was similar to the workflow described above. However, several adjustments were made to simplify the conduct of the pilot study. In particular, estimation took place over a fixed number of three rounds for all trials. The fixed number of trials enabled observation of the consensus forming process.

An early observation from the pilot study was that a significant proportion (approximately 80%) of the submissions were of low quality in terms of worker engagement with the task and the justification supplied alongside an estimate. The pilot also confirmed that estimation accuracy was correlated with low submission quality.

To address this issue within the pilot we developed a quality model for worker behaviour and used it to manually filter out low quality estimates. The model was able to distinguish between spammers, non-expert workers, and domain experts using the quality model. In particular, we discovered that crowd workers who provided good justifications for their estimates wouldn't necessarily spend much time reviewing contextual information. We speculated that such workers had reviewed similar issues before and therefore had less need for additional information.

Table III shows the results of the pilot study once the low quality estimates had been removed. The results were encouraging. 301 estimates were received, of which 121 were accepted for use in the estimation process. As a consequence, crowd workers were able to predict the same estimate as of the expert for seven issues out of nine. Crowd workers also gave some insightful justifications for their estimates. On the basis of the pilot study results, we proceeded to further develop the CPP practice by incorporating an automated quality assessment of worker behaviour, as summarized in Section III-E below.

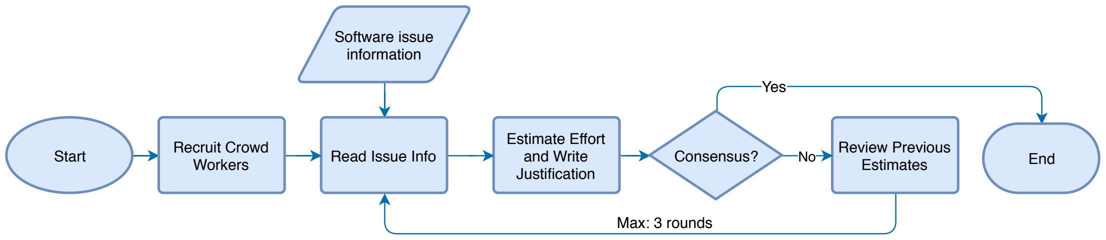
Software issue information
Start
Recruit Crowd Workers
Read Issue Info
Estimate Effort and Write Justification
Consensus?
Yes
No
Review Previous Estimates
End
Max: 3 rounds

Fig. 1. General model of the crowdsourcing planning poker task

TABLE III
SUMMARY OF PILOT RESULTS, INCLUDING NUMBER OF ESTIMATES RECEIVED, ACCEPTED, AMBIGUOUS AND REJECTED, OUTCOME FOR EACH TRIAL AND LEVEL OF AGREEMENT ACHIEVED WITHIN THE CROWD.

	Estimates				Actual Effort	Expert Estimate		Crowd Estimate	
Trial	All	Accepted	Ambiguous	Rejected	Category	Category	MRE	Category	MRE
1	35	18	12	5	Half-day	**Half-week**	**300%**	One week	1150%
2	32	17	6	9	Half-day	**Half-day**	**0%**	One week	1233%
3	22	6	7	9	Half-day	One hour	**67%**	**Half-day**	90%
4	33	11	3	19	Half-week	Two weeks	200%	**Half-week**	**40%**
5	34	11	4	19	Half-week	Half-week	**4%**	Half-Week	16%
6	40	16	9	15	Half-week	Half-day	80%	**Half-Week**	**33%**
7	37	17	9	11	Two Weeks	>Two weeks	113%	**Half-Week**	**77%**
8	31	13	7	11	Two Weeks	One week	17%	**Two Weeks**	**3%**
9	37	12	13	12	Two Weeks	>Two weeks	67%	**One Weeks**	**58%**
Total	**301**	**121**	**70**	**110**					

E. An Overview of Monitoring Quality Of Crowd Work On Subjective Tasks

Based on the results of the pilot study [27] described above, a multi-component quality model was developed for the quality of the workers' submissions. The model combines a quality assessment of the workers' justification for their estimate [40] with an evaluation of workers behaviour whilst working on the task, utilising a log trace of worker interactions with the CPP user interface [43, 44]. While previous research take all the traced events into consideration, our model used selected events from the trace of worker actions with pre-defined weights, as summarised in in Table IV. Some of the actions are binary, such as whether the worker clicked on the button to reveal addition information or not. Other actions are associated with particular properties, such as how long the worker spent in total on the task. The values for these events weights were based on observation of crowd behaviour during the pilot study [27].

The justification provided by the crowd worker was evaluated for the presence of four components: a task breakdown, a time assignment for each working block, a general discussion about the task topic, and an explanation of the estimation process applied.

All the submissions from the pilot study [27] were processed according to the model. The pilot submissions were then manually labelled according to the following definitions.

- Accepted estimates: the crowd worker's rationale contains at least two justification components, e.g. a breakdown of the required work with time specification for each block. The weighted behaviour of the crowd worker behind the submission is >75%.
- Ambiguous estimates: the crowd worker's rationale contain at least one justification component, and there is a relationship to the estimation process, e.g. mentioning the task in place. The weighted behaviour of the crowd worker behind the submission is between 30% - 75%.
- Rejected estimates: the crowd worker's rationale is completely unrelated to the task and issue topic. The weighted behaviour of the crowd worker behind the submission is <35%.

TABLE IV
SUMMARY OF ACTIONS USED FOR SCORING BEHAVIOUR

Event	Target	Properties	Weight
Type	Experience Field	15 words	10
Click	Extra Info Btn.	-	20
Click	Issue Comment Btn.	-	10
Click	Terms Definition Btn.	-	10
Click	Project Info Btn.	-	10
Spend	Extra info stage	25 second	10
Click	Google Search Btn.	-	10
Type	Justification field	24 words	10
Spend	Browser Window	3 min	10

The labelling of pilot submission was undertaken by a team of four researchers who first completed the task separately, before reviewing each submission collectively to reach consensus. The authors role was limited to explaining labelling instructions and facilitating the labelling meetings. The model

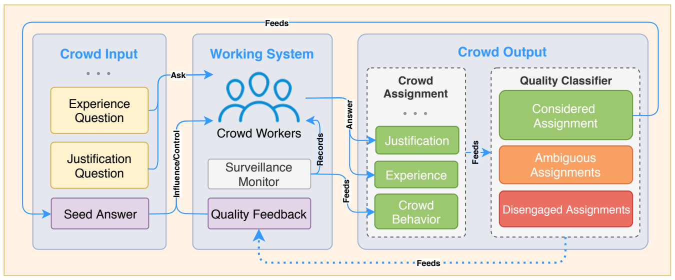

Fig. 2. Crowd planning poker quality model

values and labelling of pilot submissions was then used to train a random forest classifier from the scikit-learn library [46] to automatically accept or reject submissions. Testing the classifier resulted in an F-score of 0.935.

Figure 2 illustrates how the quality model was used within the overall CPP process. During estimation, the surveillance monitor implemented within the CPP user interface gathers information about the worker behaviour, which is then submitted for the task along with the justification. The quality classifier is then used to filter the received estimates. Accepted estimates were forwarded to a further round of estimation if required, whereas ambiguous and rejected estimates were removed by this process. Both accepted and ambiguous estimates were paid during the study, since both were considered to have engaged with the assignment presented in Mechanical Turk. Workers who were classified as ambiguous or disengaged automatically received notifications that their submissions might be rejected, giving the worker an opportunity to improve it by making changes. Thus, the worker was engaged in an instant feedback loop until reaching their submission reached the required quality level or they decided to withdraw from the task. In both cases, the burden of checking the quality of crowd submissions was eliminated. However, as the classifier is not a 100% accurate, rejected crowd workers were given an option to appeal to ensure that crowd workers were treated fairly for the purposes of the research.

IV. Results and Evaluation

Thirty trials were conducted (one per selected issue), as summarised in Table V. According to Munoz and Bangdiwala [47] interpretation of Fleiss' Kappa [48], all trials proceeded until an 'Almost Perfect' level (>0.75) [49] of agreement had been reached amongst the crowd workers, measured using Fleiss' Kappa [48]. The crowd workers reached a consensus within three rounds in all trials, with nine trials ending after a single round, ten trials ending after two rounds and eleven trials requiring three rounds of estimation.

Each round of CPP received between 10 and 5 estimates, with an average of 8 estimates received in each round, resulting in a total of between 5 and 30 estimates for each trial. Each round was kept open until a minimum of five estimates of sufficient quality had been received. Unlike the earlier pilot study [27], the proportion of rejected estimates was much lower, averaging 39% across all trials and reaching 50% in Trial 11 and Trial 18. The reduction in low quality submission is likely due to the automatic quality assessment and feedback process summarized in Section III-E.

Table V also shows a comparison between the final aggregate estimate produced by the crowd for each trial, the expert (baseline) estimate and actual effort (ground truth) for the issue as reported in the source project's issue tracker. The category (one hour, half day, etc.) of the final estimate and actual effort are reported in all cases. Further, the MRE and Mean MRE (MMRE) are shown for both the crowd and expert estimates relative to the actual effort. The next two sections reviews these results with respect to the original research questions.

A. Crowd Performance Compared with Experts

The research question for this work concerns the ability of the crowds of in-expert workers to produce estimates of a similar accuracy to those produced by single experts. The results of the 30 trials conducted are reported in Table V. The table reports the total number of rounds for each trial, along with the number of accepted and rejected submissions. The table also shows the actual categorical effort required for the task concerned and the expert's estimate, both as reported on the issue tracker and converted to a category, as well as the estimate produced by the crowd. Estimates that are in bold indicate the estimate that was closest to the reported effort (either expert or crowd, or both if the error was equal). Estimates are underlined if the correct category was also estimated.

As can be seen from the table, the crowd workers correctly predicted the effort category for 7 of the 30 trials (14, 18, 25, 30, 31, 32, 36), as compared with fourteen of the issues by the expert estimator (12, 13, 15, 16, 18, 21, 22, 23, 24, 30, 34, 36, 38, 39). By this comparison, expert estimators out perform crowds. However, an alternative analysis would be to consider which prediction (crowd or expert) was most accurate for each of the issues. Here, the crowd workers produced the same estimates as experts in 8 trials, crowd workers were more accurate estimates in 10 trials, and experts more accurate in 12 trials. This suggests that the overall estimation performance of crowd and experts were similar.

We further checked this comparison, by investigating whether a statistically significant difference existed between the distributions of MREs for both crowd and expert produced estimates. First, we applied the Shapiro-Wilk test to both MRE distributions to determine if either were normal. The result of the test for crowds (W=0.60984, p=9.645e-08) and experts (W=0.57812, p=3.987e-08) indicate that both were non-normal. We therefore applied the Mann-Whitney U Test, since both distributions are assumed to be independent. Applying this test to the two distributions resulted in being unable to reject the null hypothesis (W=497, p=0.4861), indicating that there is no evidence of a statistically significant difference between the MRE distributions and thus that the two effort estimation techniques have similar accuracy.

In a final analysis, we compared the Mean MRE (MMRE) of crowd estimates (239.83%) to the MMRE of expert estimates

TABLE V
SUMMARY OF TRIAL RESULTS, INCLUDING NUMBER OF ESTIMATES RECEIVED, ACCEPTED AND REJECTED, OUTCOME FOR EACH ROUND AND OVERALL TRIAL AND LEVEL OF AGREEMENT ACHIEVED WITHIN THE CROWD. APRA AND SUA ABBREVIATIONS IN THE AGREEMENT COLUMN STANDS FOR: ALMOST PERFECT AGREEMENT AND SUBSTANTIAL AGREEMENT ACCORDING TO MUNOZ AND BANGDIWALA [49].

		Estimates			Crowd Agreement		Actual Effort	Expert Estimate		Crowd Estimate	
Trial	Number of Round	All	Accepted	Rejected	(Fleiss' Kappa %)		Category	Category	MRE	Category	MRE
10	1	6	5	1	APrA	76.19%	A day	Half a week	100%	**Half a day**	50%
11	2	20	10	10	APrA	76.19%	One Hour	**A day**	500%	**A day**	700%
12	3	26	15	11	APrA	76.19%	One Hour	**One Hour**	0%	A day	700%
13	2	12	10	2	APrA	80.95%	Half a day	**Half a day**	0%	A day	100%
14	1	10	5	5	APrA	79.17%	One Hour	Half a day	100%	**One Hour**	0%
15	2	17	10	7	APrA	76.19%	One Hour	**One Hour**	0%	Half a day	300%
16	1	7	5	2	APrA	79.17%	Half a week	**Half a week**	20%	Half a day	80%
17	1	7	5	2	APrA	79.17%	One Hour	A day	700%	**Half a day**	300%
18	3	30	15	15	SuA	66.67%	Half a day	**Half a day**	25%	**Half a day**	0%
19	1	7	5	2	APrA	76.19%	Half a day	**A day**	50%	**A day**	100%
20	3	26	15	11	SuA	66.67%	One Hour	Half a week	2300%	**Half a day**	300%
21	1	8	5	3	APrA	79.17%	One Hour	**One Hour**	0%	Half a day	300%
22	2	16	10	6	APrA	83.33%	Half a week	**Half a week**	20%	A day	60%
23	2	18	10	8	APrA	79.17%	Half a day	**Half a day**	25%	A day	100%
24	2	16	10	6	APrA	83.33%	Half a week	**Half a week**	20%	Half a day	80%
25	2	22	10	12	APrA	79.17%	One Hour	Half a week	2300%	**One Hour**	0%
26	2	24	10	14	APrA	76.19%	Half a day	One week	1100%	**Half a week**	400%
27	2	14	10	4	APrA	83.33%	Half a day	Two weeks	1700%	**A day**	100%
28	2	17	10	7	APrA	75.0%	Half a day	Half a week	300%	**A day**	100%
29	3	21	15	6	APrA	79.17%	One Hour	**Half a day**	300%	Half a week	1900%
30	3	19	15	4	SuA	70.83%	Half a day	**Half a day**	0%	**Half a day**	0%
31	3	24	15	9	APrA	76.19%	One Hour	Half a day	200%	**One Hour**	0%
32	1	9	5	4	APrA	83.33%	Half a day	A day	100%	**Half a day**	0%
33	1	5	5	0	APrA	79.17%	A day	**Half a week**	200%	**Half a week**	150%
34	3	26	15	11	SuA	71.43%	One Hour	**One Hour**	0%	Half a day	300%
35	3	17	15	2	APrA	80.95%	A day	**Half a week**	100%	**Half a week**	150%
36	3	23	15	8	SuA	66.67%	Half a day	**Half a day**	0%	**Half a day**	0%
37	3	22	15	7	SuA	52.38%	A day	**Half a week**	100%	**Half a week**	150%
38	3	27	15	12	SuA	66.67%	One Hour	**One Hour**	0%	A day	700%
39	1	10	5	5	APrA	83.33%	Half a day	**Half a day**	0%	One Hour	75%
Total	**62**	**506**	**310**	**196**							

(342%) across all the issues and found that crowd workers error is less than the experts by 102.2%. This suggests that crowd workers are more likely to *under-estimate by a category* as compared to experts who are more likely to *over-estimate using person hours*. More research is required to investigate this phenomenon. Overall, the results also demonstrate that the CPP process can effectively discriminate between tasks of different orders of magnitude, ranging from half a day through to two weeks.

B. CPP Efficiency

In this section a brief discussion of the CPP running costs. Whether employed in a commercial or open source setting, measuring the costs in terms of worker time is necessary to assess scalability. The total amount of time that crowd workers took to produce an estimate through CPP ranged from 17 and 76 minutes, including idle time. Unsurprisingly, the number of rounds in a trial had a significant influenced on the time taken, with Trial 17 requiring just a single round and lasting just four minutes, for example. These results suggest that producing an estimate from a crowd takes some additional time, compared with a Planning Poker process conducted by a group of experts as described in Section I. Expert estimation may also be considerably faster when the expert group already has a good understanding of the task to be estimated and can rapidly achieve consensus without the need for discussion. Nevertheless, the results demonstrate that crowds can produce estimates relatively quickly and on demand. In addition, the work demonstrates that CPP can estimate multiple tasks in parallel, as compared with in-person Planning Poker, where only one issue can be considered at a time.

Conducting the experiment trials results in an average cost of $1.99 to produce a final estimate for one issue. (again, this figure is influenced by the number of rounds taken in a trial). This cost would appear to compare very favourably with the cost of running a Planning Poker session within a software team. Assuming a team of five developer with an average hourly salary of $40 (excluding other costs) can estimate 10 tasks in hour, then the average cost per estimate would be \$20. Thus, the results of the trials demonstrate the potential for a significant cost saving in the context of playing CPP in a Commerical software development houses.

C. Beyond Estimates - Crowd Insights

An additional benefit of requesting a rationale from crowd workers when they supply their estimate is that further insight and analysis of the task to be estimated can be obtained. Many of the workers provided useful information about how to approach the task. Such advice and guidance was often very detailed, for example, on a task concerning the creation of a preview mode for sites using the Apache Maven site plugin (MSITE-68), a crowd worker wrote:

> "This seems like a good case for building at the DOM level, to "implement" the changes in parallel for the previews. If that is in fact the case, it would probably take about a day to get a working prototype. If not...then a day would also probably be enough to know that this simply cannot be done."

The crowd worker provides a suggestion that the resolution of the issue can be done by monitoring a page's DOM for changes to create a preview. They also include a suggestion that a prototype should be created first to determine whether the feature is feasible.

For another issue, concerning the implementation of a new indexing mechanism for a JBoss workspace, the crowd worker provides a detailed breakdown of the work to be done:

> "1. how to determine, and what is the most efficient and accurate query for nodes and necessary information?
>
> 2. Initial testing for viability of indexing nodes (no lost data, consistency, etc)
>
> 3. Deeper testing incl. stress testing at higher node counts, ensure all threads are deleted, etc."

In particular, the crowd worker emphasises the importance of different types of testing, noting that non-functional testing should be treated separately from the design and functional testing of the feature.

These examples were intriguing, as we hadn't anticipated that crowd workers would provide insights with significant *domain specific* knowledge. These suggestions and explanations have the potential to be of significant assistance to a team during the wider triage process for a software task that occurs alongside estimation. Further work is needed to understand the extent to which this expertise can be leveraged and focused.

D. Threats to Validity

A limitation of the study is employing issues created for open source projects. This decision was necessary as the experiment required a source of software tasks that could be provided to anonymous crowd workers and that had been annotated with expert estimated and actual work cost. This meant there was a risk that the crowd workers could access the issue trackers themselves and simply supply the actual reported cost, creating a threat to validity of the reliability results.

This risk was mitigated in several ways. First, the issue identifiers were not supplied to the crowd workers and issues were selected from issue trackers that required user registration. This created an additional step to deter workers. Second, workers were asked for a categorical submission, rather than an absolute person-hour value, creating an additional step if the source issue was accessed. Finally, workers were encouraged to supply their estimate and it was clear that payment was not contingent on supplying the correct result. Consequently there is no evidence in the behaviour logs that the workers accessed project issue trackers, although this may have occurred outside the CPP user interface.

A second, external, threat to validity concerns the steps taken to filter the selected issues from the open source projects as described in Section III-A. Our results are therefore limited to assessing the estimation performance of crowd planning poker on issues that met our filtering criteria. Further, the analysis in II showed that the Mean and Median MRE for the expert estimates (the experimental baseline) were significantly lower in our filtered sample compared with the unfiltered sample. This suggests that experts perform worse when making estimates on issues that were 'lower' quality according to the filtering critiera. Therefore the baseline for evaluating crowd planning poker for these issues would be lower, compared with the filtered issues. However, we are unable to determine within the scope of the current work how each the filters might impact estimate accuracy for crowd workers. We briefly discuss future experimental work connected with this in the next section.

V. Conclusions

This paper has presented the first study of the application of crowdsourcing to Planning Poker for the production of software task estimates, answering Grenning's speculation more than a decade ago. The work demonstrates that crowd workers, organised in a Crowd Planning Poker process, can reliably produce software task estimates compared to experts; and at substantially reduced cost compared to small teams of domain experts. The crowd workers were able to discriminate between tasks of varying complexity and provide useful insights as to the resolution of the task.

These results therefore present several opportunities for future research directions. First, an observed benefit of CPP compared to in-person Planning Poker is the ability to obtain results on demand, rather than needing to wait for a team's regular planning session. In addition, we noted that the crowd workers often provided useful insights as to how the best approach to take to resolve the issue and the sub-tasks that this might involve. Therefore, CPP could be used by a software team to obtain an initial estimate for a task along with some initial guidance, prior to the task being triaged by a team member. Alternatively, CPP could be used to quickly flag issues that lack sufficient information for an accurate estimate to be made, either within a crowd or by experts. We plan further studies to understand how a software team could incorporate crowd estimates within existing triage workflows.

Related to this possibility, is the need to assess the extent to which software teams can publish software project tasks to a crowd for estimation when the issues may contain potentially sensitive or private information. To date, our research has focused on estimation of tasks as issues drawn from open source projects that are not affected by this concern, but do have large backlogs of unresolved issues. Conversely, small software teams working in a commercial setting may be reluctant to publish the full details of an issue to be resolved for fear of releasing commercially sensitive information, such as product directions.

As a further work in this direction, we intend to investigate the extent to which issues can be obfuscated to address this concern, without reducing the reliability of the estimate. Similarly, we are investigating the possibility of measuring the specificity of an issue, as issues that concern less project specific activities may be less sensitive for a project.

An alternative approach would be to consider other sources of recruitment of the crowd workers. Large software organisations may host multiple projects and employ many hundreds or thousands of developers. Similarly, successful OSS projects attract similar numbers of volunteers. Rather than employing crowd workers on sites such as Mechanical Turk, such development efforts might leverage the resources available within their own organisations. Such an approach might also further enhance the accuracy of estimates.

References

[1] The Standish Group, "The CHAOS report 2015," 2015.

[2] K. E. Emam and A. G. Koru, "A replicated survey of IT software project failures," *IEEE Software*, vol. 25, no. 5, pp. 84–90, September/October 2008.

[3] T. Furuyama, Y. Arai, and K. Iio, "Analysis of fault generation caused by stress during software development," in *Achieving Quality in Software*. Springer, 1996, pp. 14–28.

[4] S. Koch, "Effort modeling and programmer participation in open source software projects," *Information Economics and Policy*, vol. 20, no. 4, pp. 345–355, Dec. 2008. [Online]. Available: https://doi.org/10.1016/j.infoecopol.2008.06.004

[5] F. Qi, X.-Y. Jing, X. Zhu, X. Xie, B. Xu, and S. Ying, "Software effort estimation based on open source projects: Case study of Github," *Information and Software Technology*, vol. 92, pp. 145–157, Dec. 2017. [Online]. Available: https://doi.org/10.1016/j.infsof.2017.07.015

[6] J. Asundi, "The need for effort estimation models for open source software projects," in *Proceedings of the fifth workshop on Open source software engineering - 5-WOSSE*. ACM Press, 2005. [Online]. Available: https://doi.org/10.1145/1083258.1083260

[7] K. Moløkken and M. Jørgensen, "A review of surveys on software effort estimation," in *2003 International Symposium on Empirical Software Engineering (ISESE 2003), 30 September - 1 October 2003. Rome, Italy*. IEEE Computer Society, 2003, pp. 223–231. [Online]. Available: https://doi.org/10.1109/ISESE.2003.1237981

[8] E. Raymond, "The cathedral and the bazaar," *Knowledge, Technology & Policy*, vol. 12, no. 3, pp. 23–49, Sep. 1999. [Online]. Available: https://doi.org/10.1007/s12130-999-1026-0

[9] A. Lee, J. C. Carver, and A. Bosu, "Understanding the impressions, motivations, and barriers of one time code contributors to FLOSS projects: a survey," in *Proceedings of the 39th International Conference on Software Engineering, ICSE 2017, Buenos Aires, Argentina, May 20-28, 2017*, S. Uchitel, A. Orso, and M. P. Robillard, Eds. IEEE / ACM, 2017, pp. 187–197.

[10] G. Robles, J. M. González-Barahona, C. Cervigón, A. Capiluppi, and D. Izquierdo-Cortázar, "Estimating development effort in free/open source software projects by mining software repositories: a case study of OpenStack," in *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*. ACM Press, 2014. [Online]. Available: https://doi.org/10.1145/2597073.2597107

[11] P. Hooimeijer and W. Weimer, "Modeling bug report quality," in *22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007), November 5-9, 2007, Atlanta, Georgia, USA*, R. E. K. Stirewalt, A. Egyed, and B. Fischer, Eds. ACM, 2007, pp. 34–43. [Online]. Available: https://doi.org/10.1145/1321631.1321639

[12] J. Grenning, "Planning poker or how to avoid analysis paralysis while release planning," *Hawthorn Woods: Renaissance Software Consulting*, vol. 3, pp. 22–23, 2002.

[13] K. Schwaber and M. Beedle, *Agile software development with Scrum*. Prentice Hall Upper Saddle River, 2002, vol. 1.

[14] CollabNet VersionOne, "13th annual state of agile report," https://www.stateofagile.com, May 2019.

[15] V. Mahnic and T. Hovelja, "On using planning poker for estimating user stories," *Journal of Systems and Software*, vol. 85, no. 9, pp. 2086–2095, 2012. [Online]. Available: https://doi.org/10.1016/j.jss.2012.04.005

[16] K. Moløkken-Østvold, N. C. Haugen, and H. C. Benestad, "Using planning poker for combining expert estimates in software projects," *Journal of Systems and Software*, vol. 81, no. 12, pp. 2106–2117, 2008. [Online]. Available: https://doi.org/10.1016/j.jss.2008.03.058

[17] M. Usman, E. Mendes, F. Weidt, and R. Britto, "Effort estimation in agile software development: a systematic literature review," in *The 10th International Conference on Predictive Models in Software Engineering, PROMISE '14, Torino, Italy, September 17, 2014*, S. Wagner and M. D. Penta, Eds. ACM, 2014, pp. 82–91. [Online]. Available: https://doi.org/10.1145/2639490.2639503

[18] R. D. Stutzke, *Estimating Software-Intensive Systems:*

Projects, Products, and Processes, ser. SEI Series in Software Engineering. Addison-Wesley Professional, 2005.

[19] M. Cohn, *Agile estimating and planning*. Pearson Education, 2005.

[20] J. Surowiecki, *The Wisdom of Crowds: Why the Many Are Smarter Than the Few*. Abacus, March 2005.

[21] J. Grenning, "Agile 2008 - wisdom of crowds keynote and planning poker," https://blog.wingman-sw.com/archives/20, August 2008.

[22] Amazon Web Services, "Amazon mechanical turk developer guide," https://docs.aws.amazon.com/AWSMechTurk/latest/AWSMechanicalTurkRequester/, January 2017.

[23] S. L. Lim, D. E. Damian, and A. Finkelstein, "Stakesource2.0: using social networks of stakeholders to identify and prioritise requirements," in *Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, Waikiki, Honolulu , HI, USA, May 21-28, 2011*, R. N. Taylor, H. C. Gall, and N. Medvidovic, Eds. ACM, 2011, pp. 1022–1024. [Online]. Available: https://doi.org/10.1145/1985793.1985983

[24] T. D. LaToza and A. van der Hoek, "A vision of crowd development," in *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 2*, A. Bertolino, G. Canfora, and S. G. Elbaum, Eds. IEEE Computer Society, 2015, pp. 563–566. [Online]. Available: https://doi.org/10.1109/ICSE.2015.194

[25] C. Schneider and T. Cheung, "The power of the crowd: Performing usability testing using an on-demand workforce," in *Information Systems Development, Reflections, Challenges and New Directions [Proceedings of ISD 2011, Heriot-Watt University, Edinburgh, Scotland, UK, August 24 - 26, 2011]*, R. Pooley, J. Coady, C. Schneider, H. Linger, C. Barry, and M. Lang, Eds. Springer, 2011, pp. 551–560. [Online]. Available: https://doi.org/10.1007/978-1-4614-4951-5_44

[26] A. J. Quinn and B. B. Bederson, "Human computation: a survey and taxonomy of a growing field," in *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Vancouver, BC, Canada, May 7-12, 2011*, D. S. Tan, S. Amershi, B. Begole, W. A. Kellogg, and M. Tungare, Eds. ACM, 2011, pp. 1403–1412. [Online]. Available: https://doi.org/10.1145/1978942.1979148

[27] M. Alhamed and T. Storer, "Estimating software task effort in crowds," in *2019 IEEE International Conference on Software Maintenance and Evolution, ICSME 2019, Cleveland, OH, USA, September 29 - October 4, 2019*. IEEE, 2019, pp. 281–285. [Online]. Available: https://doi.org/10.1109/ICSME.2019.00042

[28] T. J. Gandomani, K. T. Wei, , and A. K. Binhamid, "A case study research on software cost estimation using experts' estimates, wideband delphi, and planning poker technique," *International Journal of Software Engineering and Its Applications*, vol. 8, no. 11, pp. 173–182, 2014.

[29] A. J. Albrecht, "Measuring application development productivity," in *Proceedings of the Joint Share Guide IBM Application Development Symposium*, Monterey, California, USA., October 1979, pp. 83–92.

[30] B. W. Boehm, B. Clark, E. Horowitz, J. C. Westland, R. J. Madachy, and R. W. Selby, "Cost models for future software life cycle processes: COCOMO 2.0," *Ann. Software Eng.*, vol. 1, pp. 57–94, 1995. [Online]. Available: https://doi.org/10.1007/BF02249046

[31] B. W. Boehm, *Software Engineering Economics*, ser. Advances in computing science & technology series. Upper Saddle River, NJ 07458, USA: Prentice-Hall, October 1981.

[32] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Information & Software Technology*, vol. 54, no. 1, pp. 41–59, 2012. [Online]. Available: https://doi.org/10.1016/j.infsof.2011.09.002

[33] S. G. MacDonell and M. J. Shepperd, "Combining techniques to optimize effort predictions in software project management," *Journal of Systems and Software*, vol. 66, no. 2, pp. 91–98, 2003. [Online]. Available: https://doi.org/10.1016/S0164-1212(02)00067-5

[34] H. Park and S. Baek, "An empirical validation of a neural network model for software effort estimation," *Expert Systems with Applications*, vol. 35, no. 3, pp. 929–937, 2008.

[35] K. Mao, L. Capra, M. Harman, and Y. Jia, "A survey of the use of crowdsourcing in software engineering," *Journal of Systems and Software*, vol. 126, pp. 57–84, 2017. [Online]. Available: https://doi.org/10.1016/j.jss.2016.09.015

[36] P. Donmez, J. G. Carbonell, and J. Schneider, "Efficiently learning the accuracy of labeling sources for selective sampling," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 259–268.

[37] S. Zhu, S. K. Kane, J. Feng, and A. Sears, "A crowdsourcing quality control model for tasks distributed in parallel," in *CHI Conference on Human Factors in Computing Systems, CHI '12, Extended Abstracts Volume, Austin, TX, USA, May 5-10, 2012*, J. A. Konstan, E. H. Chi, and K. Höök, Eds. ACM, 2012, pp. 2501–2506. [Online]. Available: https://doi.org/10.1145/2212776.2223826

[38] Y. Baba and H. Kashima, "Statistical quality estimation for general crowdsourcing tasks," in *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, I. S. Dhillon, Y. Koren, R. Ghani, T. E. Senator, P. Bradley, R. Parekh, J. He, R. L. Grossman, and R. Uthurusamy,

Eds. ACM, 2013, pp. 554–562. [Online]. Available: https://doi.org/10.1145/2487575.2487600

[39] T. Goyal, T. McDonnell, M. Kutlu, T. Elsayed, and M. Lease, "Your behavior signals your reliability: Modeling crowd behavioral traces to ensure quality relevance annotations," in *Proceedings of the Sixth AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2018, Zürich, Switzerland, July 5-8, 2018.*, Y. Chen and G. Kazai, Eds. AAAI Press, 2018, pp. 41–49. [Online]. Available: https://aaai.org/ocs/index.php/HCOMP/HCOMP18/paper/view/17924

[40] T. McDonnell, M. Lease, M. Kutlu, and T. Elsayed, "Why is that relevant? collecting annotator rationales for relevance judgments," in *Proceedings of the Fourth AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2016, 30 October - 3 November, 2016, Austin, Texas, USA.*, A. Ghosh and M. Lease, Eds. AAAI Press, 2016, pp. 139–148. [Online]. Available: http://aaai.org/ocs/index.php/HCOMP/HCOMP16/paper/view/14043

[41] M. Kutlu, T. McDonnell, Y. Barkallah, T. Elsayed, and M. Lease, "Crowd vs. expert: What can relevance judgment rationales teach us about assessor disagreement?" in *The 41st International ACM SIGIR Conference on Research &#38; Development in Information Retrieval*, ser. SIGIR '18. New York, NY, USA: ACM, 2018, pp. 805–814. [Online]. Available: http://doi.acm.org/10.1145/3209978.3210033

[42] A. Dumitrache, O. Inel, L. Aroyo, B. Timmermans, and C. Welty, "Crowdtruth 2.0: Quality metrics for crowdsourcing with disagreement (short paper)," in *Proceedings of the 1st Workshop on Subjectivity, Ambiguity and Disagreement in Crowdsourcing, and Short Paper Proceedings of the 1st Workshop on Disentangling the Relation Between Crowdsourcing and Bias Management (SAD 2018 and CrowdBias 2018) co-located the 6th AAAI Conference on Human Computation and Crowdsourcing (HCOMP 2018), Zürich, Switzerland, July 5, 2018.*, ser. CEUR Workshop Proceedings, L. Aroyo, A. Dumitrache, P. Paritosh, A. J. Quinn, C. Welty, A. Checco, G. Demartini, U. Gadiraju, and C. Sarasua, Eds., vol. 2276. CEUR-WS.org, 2018, pp. 11–18. [Online]. Available: http://ceur-ws.org/Vol-2276/paper2.pdf

[43] J. M. Rzeszotarski and A. Kittur, "Instrumenting the crowd: using implicit behavioral measures to predict task performance," in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, Santa Barbara, CA, USA, October 16-19, 2011*, J. S. Pierce, M. Agrawala, and S. R. Klemmer, Eds. ACM, 2011, pp. 13–22. [Online]. Available: https://doi.org/10.1145/2047196.2047199

[44] G. Kazai and I. Zitouni, "Quality management in crowdsourcing using gold judges behavior," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016*, P. N. Bennett, V. Josifovski, J. Neville, and F. Radlinski, Eds. ACM, 2016, pp. 267–276. [Online]. Available: https://doi.org/10.1145/2835776.2835835

[45] N. Dalkey and O. Helmer, "An experimental application of the delphi method to the use of experts," *Management science*, vol. 9, no. 3, pp. 458–467, 1963.

[46] scikit-learn, "scikit-learn random forest classifier," https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, December 2019.

[47] S. R. Munoz and S. I. Bangdiwala, "Interpretation of kappa and b statistics measures of agreement," *Journal of Applied Statistics*, vol. 24, no. 1, pp. 105–112, Feb. 1997. [Online]. Available: https://doi.org/10.1080/02664769723918

[48] J. L. Fleiss, "Measuring nominal scale agreement among many raters." *Psychological bulletin*, vol. 76, no. 5, p. 378, 1971.

[49] S. R. Munoz and S. I. Bangdiwala, "Interpretation of kappa and b statistics measures of agreement," *Journal of Applied Statistics*, vol. 24, no. 1, pp. 105–112, 1997. [Online]. Available: https://doi.org/10.1080/02664769723918