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

1558-1225/21/$31.00 ©2021 IEEE
DOI 10.1109/ICSE43902.2021.00014

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