Leveraging Crowdsourcing For Team Elasticity: An Empirical Evaluation at TopCoder

Razieh L Saremi
School of Systems and Enterprises
Stevens Institute of Technology
Hoboken, NJ, 07030, USA
+1 (510) 260-3274
rlotfali@stevens.edu

Ye Yang
School of Systems and Enterprises
Stevens Institute of Technology
Hoboken, NJ, 07030, USA
+1 (201) 216-8560
ye.yang@stevens.edu

Guenther Ruhe
Software Engineering Decision Support Laboratory
University of Calgary
Alberta, Canada
+1 (403) 220 7692
ruhe@ucalgary.ca

David Messinger
Topcoder
201 S Capitol Ave #1100, Indianapolis, IN, 46225, USA
+1 (978) 590-3344
dmessinger@topcoder.com

***Abstract*—There is an emergent trend in software development projects that mini-tasks can be crowdsourced to achieve rapid development and delivery. For software managers requesting crowdsourcing services, it is beneficial to be able to evaluate and assure the availability and performance of trustable workers on their tasks. However, existing rating systems are facing challenges such as providing limited information regarding worker's abilities as well as potential threats from workers' gaming or cheating the systems. To develop better understanding of worker performance in software crowdsourcing, this paper reports an empirical study at TopCoder, one of the primary software crowdsourcing platforms.**

We aim at investigating the following questions: How diverse are crowd workers in terms of skill and experience? How fast do crowd workers respond to a task call? How reliable are crowd workers in submitting tasks? And how much does CSD benefit schedule reduction? The main results of this study showed that on average, (i) 59% of workers respond to a task call in the first 24 hours; (ii) 24% of the workers who registered early will make submissions to tasks, and 76% of them exceeding the acceptance criteria; and (iii) an overall average of 1.82 schedule acceleration rate is observed through organizing mass parallel development in 4 software crowdsourcing projects. Such empirical evidences are beneficial to help exploring resourcing options and improve team elasticity in adaptive software development.

Keywords- Agility, Elasticity, Crowdsourced software development, Worker performance, Worker availability, Velocity, Topcoder

I. INTRODUCTION

Software development is rarely done in an isolated environment; instead it increasingly depends on the collaboration among different groups of stakeholders [1]. Such collaborative development processes frequently face challenges from rapidly evolving requirements, conflicting stakeholder needs and constraints, as well as unanticipated human factors. Adaptive development methodologies have been widely adapted to address these challenges and create solutions quickly [2].

Greening defines *Agility* as the capability of an actor to adopt to the changes fast and provide solutions to achieve goals in such a changing environment, and *Elasticity* as the capability to respond to required change [2]. In practice, many adaptive methodologies rely heavily on individual and team's capability in identifying and adapting to changes. Several issues have been reported related to resources dependency in agile practices. For example, on the one hand, agile development teams typically face fixed or inflexible resource, because hiring developers with the required agile skills and mindsets is very pricy; On the other hand, not all members are self-motivated and able to manage their own tasks, therefore lack of individual commitments is an outstanding issue in agile teams [3].

As an emergent new paradigm, Crowdsourced Software Development (CSD) has been increasingly adopted in recent decades [4,5,6]. CSD is reported to be attributed with significant cost-savings, quicker task completion times, formation of expert communities, higher amount of creativity, and access to a pool of workers with different skill sets with availability of 24 hours per day [4,7]. Consequently, leveraging crowd workforce in CSD has great potential to not only increase worker availability, but also allow companies to recruit best talents on demand. Another benefit of applying CSD is reducing time to market, due to worker abundance. We believe the advantages of CSD would help to address the above resource issues and improve team elasticity in adaptive software development.

For adaptive teams to leverage CSD to increase team elasticity, it is critical to understand crowd worker's sensitivity and performance to the tasks and rate of task elasticity and success. However, there is a lack of empirical investigations on the impact of worker's response to new tasks and reliability of submissions in the crowd sourcing market and project scheduling as well as effect of crowd workers on project agility in CSD.

In this study, we aim at approaching these gaps by investigating the following questions:

(i) How diverse are crowd workers in terms of skill and experience?

(ii) How fast does crowd respond to a task call?

(iii) How reliable are the crowds in submitting tasks?

(iv)How does crowdsourced software development benefit schedule reduction?

We report the design and analysis results of an empirical study based on real-world data gathered at the TopCoder website. TopCoder is the largest software development crowdsourcing platform with an online community of over 1M crowd software workers [8]. As one of the most successful CSD platforms, TopCoder has over 1million registered workers from over 190 countries, averaging 30 thousand logins every 90 days, 7 thousand challenges hosted per year and 70 million dollars in challenge payouts [8]. This fact makes a great available pool of workers and mini-tasks to study team elasticity for crowd software projects.

The rest of the paper is organized as follows: Section II introduces the background and related work; Section III presents the design of the research conducted; Section IV reports the empirical results to answer the four stated research questions. Section V discusses the results and challenges in adopting CSD; and finally, Section VI gives a summary and outlook to future work.

II. BACKGROUND AND RELATED WORK

A. *Worker reputation system*

One of the main issues related to crowdsourcing is how much to trust unknown crowd workers [9]. Due to diversity of workers with different individual skill levels, it is not practical for requesters to evaluate all the worker's trustworthiness [9], and there is no single metric reflecting the overall record of worker interaction in the pool of workers [10].

In order to solve this problem, crowdsourcing platforms employ different reputation systems to manage crowd rating based on their participation history. For example, HITs rate is used in Amazon Mechanical Turk [10], and developer rating is used in TopCoder [8]. However, such rating only provides limited knowledge about worker's preference and performance, and could be unreliable if the workers try to cheat the platform. One example in Amazon Mechanical Turk is rank boosting [10, 11], where workers are mostly registering for easy tasks or fake tasks that they are uploading themselves in order to increase their rating. Another example is distorted pursuit [9], in which workers quickly submit a possibly correct answer in order increase their benefits instead of working on the task and submitting acceptable answer.

In CSD platforms, for example, TopCoder adopts a more complicated reputation based system which is a numeric worker rating system based on Elo rating algorithm [12], Elo rating is a method for calculating the relative skill levels of players in competitor-versus-competitor games. Based on this numeric rating and a 5-level rating scheme, TopCoder divides worker community into 5 groups. The 5 worker groups are defined into 5 belts of Red, Yellow, Blue, Green and Gray, which corresponds to the highest skillful workers to the lowest ones [8]. The numeric ratings are with respect to three different task categories including algorithm, marathon matches and development [12], following sophisticated calculation algorithm. In addition, TopCoder employs a Reliability metric to reflect crowd worker's reliability on successfully completing the tasks. It is measured based on the ratio of the number of successful submissions to tasks over the most recent 15 registered tasks of a crowd worker.

Table 1 shows the distribution of workers belonging to different rating belts in the TopCoder dataset used in this study. It is shown that, among the total of 5062 workers, more than 90% of the workers are in Gray belt, which is non-experienced group. The other 10% of workers are more experienced solid workers.

TABLE 1: SUMMARY OF DIFFERENT WORKER BELTS

Belt	Rating Range (X)	# Workers	% Workers
Gray	X<900	4557	90.02%
Green	900<X<1200	146	2.88%
Blue	1200<X<1500	273	5.39%
Yellow	1500<X<2200	78	1.54%
Red	X>2200	8	0.16%

B. *Worker behaviors analysis*

Some studies have looked at worker behaviors from qualitative perspectives. For example, Eickhof et. al [14] discussed that unreliable workers are not very interested in taking novel tasks which requires creativity and abstract thinking. Kittur et al. [15] argued that the best way of avoiding unreliable workers and assuring good amount of quality is that the task should be uploaded in a way that both trustable and unreliable workers take tasks at the same time. Sorokin et. al [16] demonstrated that more expensive tasks would result higher quality submissions by attracting reliable delicate workers.

Other studies focus on analytical-based worker evaluation system, i.e. analyzing worker's reliability with respect to specific task context. Two typical contextual factors that influence on worker behaviors are: the amount of award which reflects task complexity and task type which reflects the required skillset [17,18]. Ye et.al [9] proposed a trust model based on these two factors to optimize worker identification via an evolutionary algorithm. The result can distinguish honest workers and dishonest workers when both have high overall task taking rates.

C. *Challenges of CSD*

There are many aspects that make CSD different and more challenging in evaluating crowd worker trust than general crowdsourcing.

First, tasks in CSD are inherently more complex than general crowdsourcing tasks, require specific experience w.r.t. certain programming language(s), technologies and development and/or deployment platforms. In addition, on average, most tasks have a duration of half month from first day of registration to the submission deadline, which is much longer than general crowdsourcing tasks such as image

labeling or receipts transcribing. Therefore, it is critical to assess and attract workers with the desired experience and skillset to take the tasks [3].

Second, from software managers' point of view, utilizing external, unknown, uncontrollable, crowd workers would put their projects under greater uncertainty and risk compared with in-house development [4, 5, 7, 17]. In many cases, workers may register for more than one task at a time and drop the ones they can't complete [18] before the submission deadlines. A task with many unreliable workers is subject to high risk of failure or cancellation. Intuitively, it will be expected that higher rated workers would represent more experienced and reliable workers. However, there is no empirical evidence supporting that.

Third, existing studies have reported various other challenges regarding CSD adoption, including task decomposition, communication and coordination, quality assurance, and so on [4, 6, 24, 26]. However, there is a lack of study on empirical analysis of CSD adoption at the project level.

These above challenges drive our motivation towards further empirical investigation on comparison of various characteristics of different rating group workers in CSD projects, and the effect on team elasticity at project level.

III. RESEARCH DESIGN

A. Empirical Evaluation Framework

Driven by the resource related challenges in software development, we design four evaluation studies to provide empirical evidences on feasibility and benefits of CSD. The evaluation framework is illustrated in Fig 1.

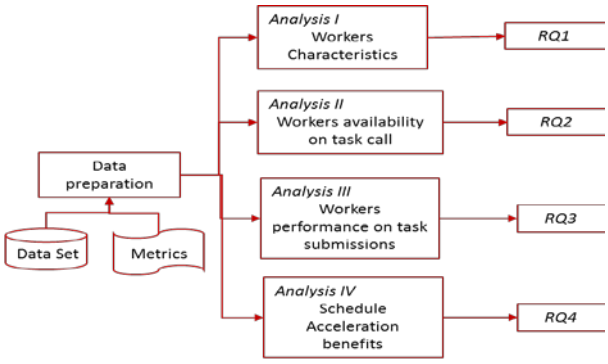

Figure. 1: Main flow of proposed framework and relationship to research questions

As Fig 1 shows, first we investigate workers' characteristics in the platform, the result led us to analyzing workers' availability to take a task and successfully return the result. At this point we need to understand workers' performance to understand team elasticity. And finally, we will discuss on benefits of applying CSD in terms of schedule acceleration. All these steps will be discussed in details in section III-D. The four research questions in this study are:

RQ1 (Worker's Characteristics): How diverse are crowd workers in terms of skills and experience?

This research question aims at providing general overview of worker characteristics in terms of membership age, reliability, skill ratings etc.

RQ2 (Worker's Availability): How fast do crowd workers respond to task calls?

The worker's availability in response to a task will be measured in the number of registrants for tasks as soon as a task is uploaded in platform. Also, reliability of task submissions can be measured by the number submissions received and the highest score of the submissions.

TABLE2: SUMMARY OF METRICS DEFINITION

Metric	**Definition**
Task attributes	
Duration (D)	Total available time from registration date to submissions deadline. Range: (0, ∞)
Task registration start date (TR)	Time when task is available on line for workers to register
Task submission end date (TS)	Deadline that all workers who registered for task must submit their final results
Award (P)	Monitory prize (Dollars) in task description. Range: (0, ∞)
Worker attributes	
# Registration (R)	Number of registrants that are willing to compete on total number of tasks in specific period of time. Range: (0, ∞)
# Submissions (S)	Number of submissions that a task receives by its submission deadline in specific period of time. Range: (0, #registrants]
Submissions Ratio (SR)	The ratio between the number of tasks a worker submitted and the total number of tasks that a worker registered.
Reliability (RE)	The percentage of successful task submissions in a worker's most recent 15 task registrations. Range: (0, 1)
Years of Membership	Year since a worker become a member. Range: (0, ∞)
Rating (R)	Platform reputation rating of a specific worker. Range: (0, ∞)
Worker-task pair level attributes	
Worker registration date (WR)	Date and time that a worker registered for a task.
Worker Submissions date (WS)	Date and time a worker submitted for a task.
Registration order (RO)	Rank of a worker's arrival time at a task by his registration date
Score (SO)	Point value of submission evaluated through peer review. Range: (0, 100]

RQ3 (Worker's Performance): How reliable are the crowd workers in submitting tasks?

The consistency of worker performance will be measured using relative velocity and submissions quality, indicating percentage of submissions duration usage by workers to submit a task and the quality of the submission for each task.

RQ4 (Schedule Acceleration): How does crowdsourcing benefit schedule reduction?

This question aims at deriving empirical evidence on comparing project schedule estimation of TopCoder projects.

B. Dataset and Metrics

The gathered dataset contains 403 individual projects including 4907 component development tasks and 8108 workers from Jan 2014 to Feb 2015, extracted from TopCoder website [8].

Tasks are uploaded as competitions in the platform, where Crowd software workers would register and complete the challenges. On average, most of the tasks have a life cycle of one and half months from first day of registration to the submissions deadline. When the workers submit the final files, it will be reviewed by experts to check the final results and grant the scores.

In order to analyze uploaded mini tasks in a project, we categorized the available data and defined the following metrics, as summarized in Table 2.

(i) Task attributes describe the basic quantitative characteristics of a task including total associated award and total number of uploaded tasks in a limited period of time;
(ii) Worker attributes measure workers' overall submission ratio, reliability, year's membership, and rating;
(iii) Worker-task pair level attributes include all worker' registration and submission dates, final scores, worker's registration order for each task.

C. Dataset preparation

The first step is to remove workers who never registered or submitted any task in their membership history. This reduced number of active workers to 5062 workers.

Second, because it is our interest to study worker's behaviors and performance in responding to task calls, we calculate, for each worker-task pair, the worker's registration order on that task. As shown in Table 2, this is derived from the rank of worker arrival at a task, i.e. his/her registration time. For example, if a worker has registered for 10 tasks in our dataset, then he/she will be characterized through 10 registration orders.

Third, workers are grouped to different rating belts per TopCoder's definition [8], and we calculate the rating belt level metrics as defined in Table 2.

We will use this data set generated from the third step in the analysis of RQ2 and RQ3. To drive answers to RQ4, we identify 4 large CSD projects in our data set, and use them as representative projects. Table 3 summarizes the overview information of the 4 projects.

D. Empirical Studies and design

Four analysis are designed based on the above metrics and proposed research questions. Specifically, we are interested in investigating the following analysis in CSD:

1) RQ1 (Worker's Characteristics)

To perform the empirical analysis after data cleaning, workers were sorted based on registration order per task. Then we clustered workers based on 5 different belts per Top coder's definition of worker rating, i.e. Red, Yellow, Blue, Green and Gray, which are representing the highest skillful worker to the lowest one [8]. Rating belt is showing the skill level and success rate of workers in platform. Crowd worker's reliability of competing on the tasks is measured based on last 15 competition workers register and submit for. For example, if a worker submitted 14 tasks out of 15 last registered tasks, his reliability is 93% (14/15). Top coder considers workers with a minimum reliability rate of 80% to be eligible to get a bonus (12 submissions out of 15 registrations).

In this part, we will be analyzing the impact of years of membership and reliability on worker's behavior per rating belt.

2) *RQ2 (Worker's Availability)*

We investigate two aspects of worker availability: 1) how fast they respond to task calls? 2) how reliable are they in completing tasks? Worker availability in response to a task call in CSD is derived using two measures: worker's registration order and average submissions rate per registration order in the platform.

Since the average number of registrants per task is 18 [31], we will analyze the availability of workers with registration order less than or equal to 20. Average response time (day) to new task call per registering order per belt as well as average submissions ratio per registering order per belt is a good measure to analyze worker's availability. To analyze worker's availability, we will use following definitions:

Def. 1: Response time, $RT_{i,k}$, measure the speed of a worker i arrival on task k, derived from the difference in the number of days between the task's registration starting date, i.e. TR_k and the worker's registration date on the task, i.e. $WR_{i,k}$:

$$RT_{i,k} = WR_{i,k} - TR_k \quad \text{Eq.-1}$$

Def. 2: Average Response Time of workers in the same registration order i, i.e. ART_i, is the average respond time of n worker with the same registration order i.

$$ART_i = \sum_{k=1}^{n} (c * RT_{i,k})/n \quad \text{Eq.-2}$$

Def. 3: Average Submissions Ratio, ASR, is the average submissions ratio of workers from the same rating group.

$$ROS(i) = \sum_{i=1}^{n} (SRi)/n \quad \text{Eq.-3}$$

3) RQ3 (Worker's Performance)

Workers Relative Velocity and Quality would be used to analyze worker performance and responsiveness in task taking and their reliability in submit tasks in the platform and task completion rate in this analysis. The related measures are defined at below.

Def. 4: Relative Velocity, i.e. $RV_{i,k}$, measures the ratio between a worker i's actual duration on completing a task k and the allowed task duration of task k.

$$RV_{i,k} = (WS_{i,k} - WR_{i,k})/(TS_k - TR_k) \quad \text{Eq. - 4}$$

Def. 5: Average Relative Velocity ARV(i) measures the average of RV ratio between actual duration and allowed task duration of workers from the same rating group.

$$ARV(i) = \left(\sum_{i=1}^{n} (WSi - WRi) / \sum_{i=1}^{n} (TSi - TRi)\right)/n \quad \text{Eq.-5}$$

Also, the quality of task submissions is the second important factor in analyzing worker's performance.

Def. 6: Quality Q(i) of worker i is defined as the average score SO(i) of worker i per registration order per belt.

$$Q(i) = \sum_{i=1}^{n} (SOi)/n \quad \text{Eq.-6}$$

Def. 7: Average Quality AQ(i) in the same registration order, average quality Q(i) of worker per registration order per belt.

$$AQ(i) = \sum_{i=1}^{n} (Qi)/n \quad \text{Eq.-7}$$

4) RQ4 (Schedule Acceleration)

In order to study the schedule acceleration effects of crowdsourcing, we select four large CSD projects (in terms of the total number of tasks, we assume projects with 100 uploaded tasks or more are big enough to relatively have long release time) in our dataset as subject and calculate the total CSD effort for these projects; then we choose three baseline software schedule estimation models to derive representative nominal schedule from the effort data of the 4 CSD projects. It is our assumption that the derived nominal schedule will reflect the corresponding in-house development schedule to complete a project with similar amounts of effort. Finally, we compare the results with the actual crowdsourcing schedule.

To calculate the total CSD effort, we use the combination of the effort from the top-2 workers with higher submissions scores to represent the effort for each task. More specifically, the effort for a CSD task is defined as:

Def 8: Effort spent on each task is measured by the weighted sum of the top-2 workers' effort spent on the task, with corresponding weights of 80% and 20%, respectively. Worker's effort is measured by the number of days between his registration date and his submission date. If there is only one active worker, the effort would be calculated based on that worker's actual effort only.

Next, we select three baseline schedule estimation models which are commonly used in industry, for deriving corresponding nominal schedules to be compared with CSD project duration. Most software schedule estimation methods adopt an underlying arithmetic model for predicting duration from development effort, which contains some model constants C and exponent D as follows:

$$Duration = C*(Effort)^{D}$$

The first is schedule estimation model in COCOMO II [19], which follows the basics model shown t bellow:

$$Duration\_I = 3.67*(Effort^{0.28})*(SCED\%)$$

Since we are deriving a nominal schedule, the SCED% is assumed to be 100%, meaning no schedule compression or stretch-out will be considered.

The second model is CORADMO [20], which is a sub-model in the COCOMO II family, specialized for agile software development projects. To reflect the schedule acceleration effects in agile development, CORADMO assumes a square root relationship between duration and effort, instead a cubic root relationship as that in the COCOMO II model:

$$Duration\_II = Effort^{0.5}$$

The third model is proposed by McConnel [21], similar to COCOMO II but with different model constant parameters:

$$Duration\_III = 3.0*(Effort^{0.33})$$

Finally, we analyze schedule acceleration rate per project in CSD, based on the following definition:

Def 9: Schedule Acceleration Rate, SARi, is the ratio of estimated nominal duration if following traditional in-house development, i.e. Duration_I, II, and III, and the actual CSD duration for each project.

The results are presented next.

IV. EMPERICAL RESULTS

A. *Overall Worker Characteristics (RQ1)*

At project level, a total of 403 projects were further decomposed into a total of 4907 mini tasks, across 14 different challenge types [18]. The average number of workers per task is about 18, the average task price is about $750, and the average task duration is about 16 days. As shown in Figure 2, 80% of the tasks have less or equal to 14 registrants, with 3 or less submissions. 13% of all tasks in our data base receive more than 20 registrants (650 tasks).

We further look at the worker distribution on the top 5 task types, including Code, First2Finish, Assembly, Content Creation, and UI Prototyping. Figure 3 shows the worker distribution results, in five different rating belts, for these top 5 task types, which accounts for 94% of all tasks. The left vertical axis shows the percentage of workers in each rating belt, and the right vertical axis reflects the unit price for tasks. The dotted line shows the average unit prices of the top 5 task types.

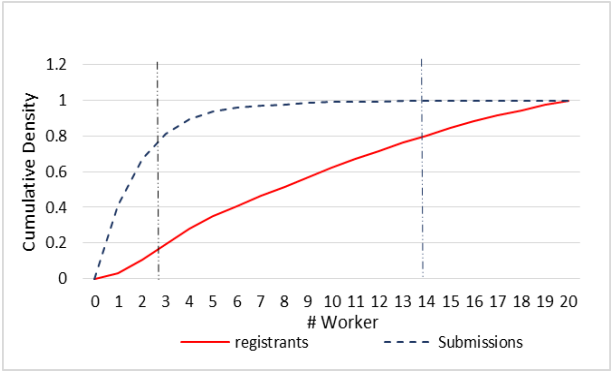

Figure 2. Cumulative density plot of #Submissions and #Registration per task

As illustrated in Figure 3, Gray workers, with the lowest rating among all groups, are mostly (i.e., 48%) interested in participating in coding, while Red workers as highest rated workers prefer to apply for first2finish challenges. Almost 35% of Red workers are concentrated in First2finish task

types, none of them applied for UI prototype challenges. Yellow workers are mostly interested in assembly tasks with 23% of workers and less interested in content creation. It is also observed that the Green and Blue categories have very similar task preference patterns.

Another important aspect in software crowdsourcing is worker's reliability. According to Figure 4, Green and Blue category workers contain more than 50% members with reliability above 0.5 (which means 50% of the time they successfully deliver jobs.). Additionally, there are about 20% of members with reliability above 0.8. Yellow and Red groups, though with relative higher rating, have only 20% of their members with reliability above 0.5 and 0.3, respectively.

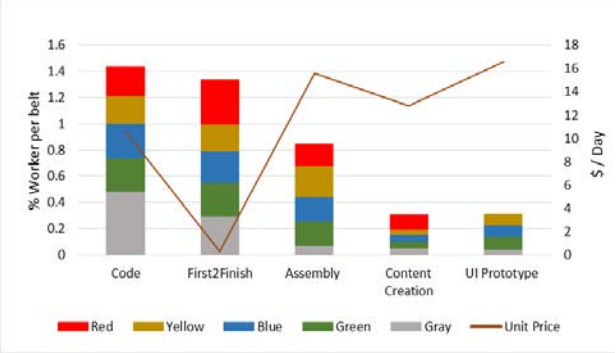

Figure 3. Workers belt contribution in top 5 task types

We run repeated measure one-way ANOVA test on the worker's reliability data from the five groups. Based on ANOVA test results, the worker's reliability is significantly different across the five groups (i.e. p-value is 0.005).

Finding 1.1: Workers from different rating groups (denoted in different color columns) have somewhat different preferences in task selection in terms of task types, prices, and durations.

Finding 1.2: Workers from different rating groups have absolutely different experience level and reliability distribution. Workers joining after year 2010 are mostly experienced workers with strong interest and motivations; however, workers with higher ratings are more skillful, not necessarily more reliable than workers with lower ratings.

B. Worker Availability: How quickly workers respond to task calls? (RQ2)

To derive empirical evidence on worker availability in CSD, we focus on the first 20 clusters, i.e. worker-task pairs with registration orders below 20. In each rating group, we calculated the average response time (in days) for each cluster. Figure 5 shows a line chart of average response time and registration order for all five rating groups. The results show that the average response time for the top-20 registrants is not exceeding 1 day (i.e., 23.57 hours).

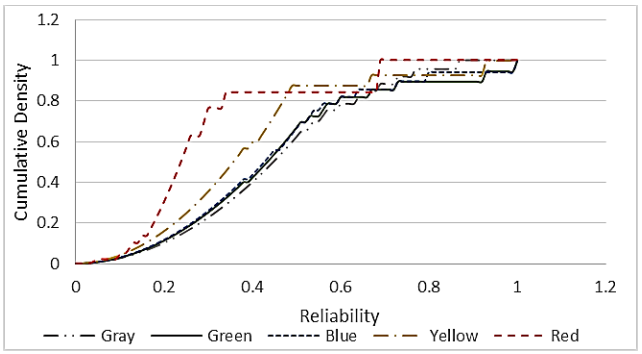

Figure 4. Cumulative density of worker's reliability across five rating groups

On average, 59% of workers respond to a task call within the first 24 hours after task being uploaded; 48% of the first 5 workers in Green, Blue and, yellow belts are registering within the first 12 hours. Gray category is relatively slow in responding to task calls, and 65% are responding in the first days, 18% are still registering in the second day. It is observed from Figure 6 that for workers from Green and Blue group, the response time is not very sensitive to registering orders, which means the workers are very motivated to register for new uploaded tasks, even though they are aware of potentially dozens of competitors. Another interesting observation is that, first and second registrants in the Red group tend to respond to tasks immediately after it is uploaded. The reason for the fluctuation in Red group is because of the small sample size of this group, which only contains 8 workers.

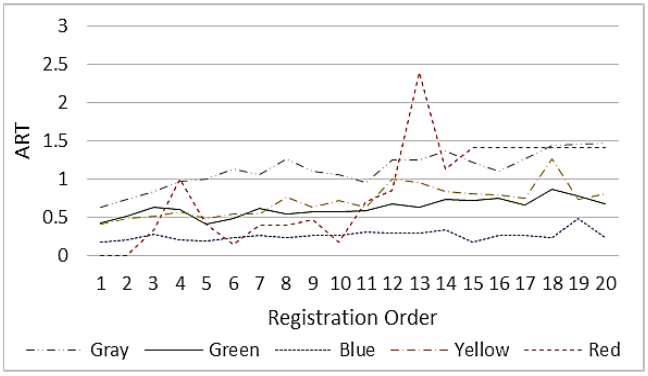

Figure 5. Average Response Time (ART) per day in dependence of the order of registration (up to 20 workers per task)

The result of the ANOVA test showed that worker registration order is significantly different across all 5 groups (i.e. p-value is 0.000).

Figure 6 shows the relationship between average submissions ratio w.r.t. registration orders. There is a general trend of decrease in submissions ratio as registration order goes up from 1 to 20. This is a clear evidence of the strong motivation in completing tasks and winning prizes among early registrants. On average, more than half of early registrant workers (first and second registrants) have submitted tasks. As it is illustrated in Figure 6 that workers in higher belt are not actively submitting for tasks where they are not among the early registrants, i.e. with a registration

order greater than 12. Workers in Gray, Green, Blue, and Yellow categories are following the same pattern; however, workers in Blue and Yellow categories are facing more fluctuated trend.

More specifically, first registrants in the Yellow category have an average submissions ratio of 60%, by going up in registering order the rate is decreased to 20% for the 20th registered worker. The submission ratio for the first register order in Blue category is about 39% and it gradually drops to 10% for the 20th registrant.

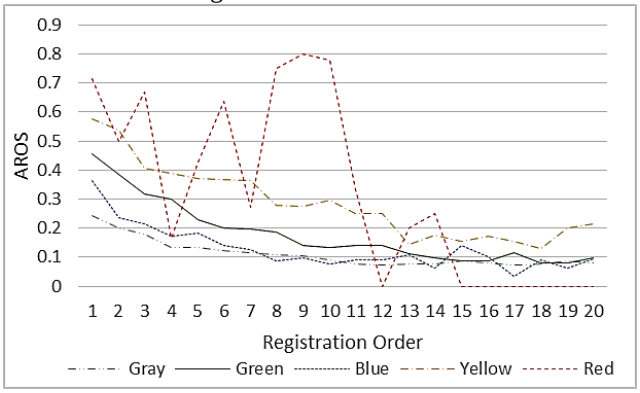

Figure 6. Average submission ratio per registration order (AROS) per belt

Among all the belts, Gray and Green members seem to have smoother patterns. First registration order for these clusters has 25% and 45% submissions ratio, respectively. Submissions ratio in both categories is decreasing to 10% for the 20th registrant. In the Red category, the average submissions ratio for top 3 registrants is 60% which drops to 20% for the 4th registrants. Statistical analysis supports that average submission ratios for the five group are significantly different (i.e. p-value at 0.000067).

Finding 2.1: The average response time for the top-20 registrants is not exceeding 1 day (i.e. 23.57 hours).

Finding 2.2: 59% of workers are registering for a new task within 24 hours of uploading; and 48% of the first 5 workers in Green, Blue and, yellow belts are registering within the first 12 hours.

Finding 2.3: There is a decreasing trend of submissions ratio in reverse to registration order.

C. Workers performance in task submissions (RQ3)

Figure 7 shows the average relative velocity of workers for different rating belts, which measures the percentage of acceleration in task completion. Interestingly, lower rated workers are using less time; while higher rated workers, required higher amount of time to submit. This can be due to different task choices, different worker motivation patterns, or individual skills.

On average 34% of the workers can complete the requested tasks within only 10% of allowed time for submissions, and 60% of workers are submitting the final files by using 70% of total submission duration. More specifically, 20% of Red group workers are using less than 40% of duration to submit, while this rises to around 40% of Blue and Green groups. It takes almost 35% of Yellow group workers and increase to 70% of Gray group workers.

However, results of ANOVA test do not support that rating clusters are significantly influencing workers' velocity, i.e. p-value is 0.14, which is greater than 0.05.

Moreover, the quality of worker's submissions is another important factor in task success. We can measure quality based on final score that workers granted per submissions.

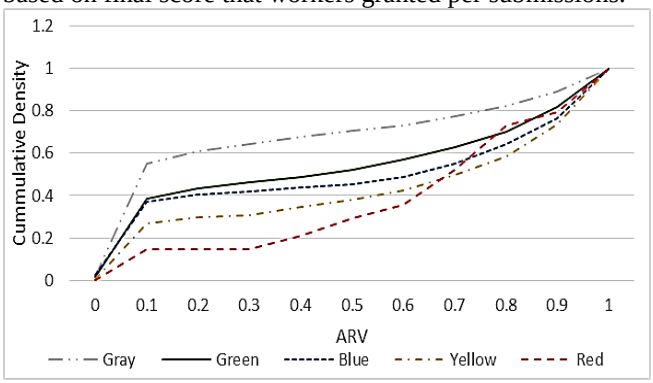

Figure 7. Average Relative Velocity (ARV) in different belt categories

Figure 8 shows the cumulative distribution of average score per belt for workers with registration order less than 20. As it is illustrated higher rated belts are granted higher scores. Almost 50% of Gray members are granted score of 0, which

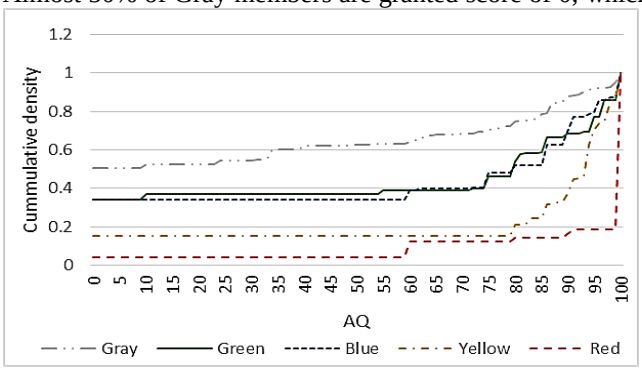

Figure 8. Evolution of Average Quality (AQ) in different belts

means they never have any submissions or they have granted 0 as their score, there is a slow increasing trend up to score of 35 for 65% of members, and only 25% of gray members granting score higher than 85. In Red category, around 18% of members are either receiving scores less than 69 or dropped the task out. The granted score increased to 90 for next 2% of the population and there is a sharp rise for net 80% of members with score of 100. In Yellow category, almost 20% of workers are granting scores less than 80, then scores gradually increase to 100 for next 80% of population. Almost the same patterns happen for members of Green and Blue workers, however, they experience a slower rising trend than the Yellow ones. ANOVA test showed that worker quality of submission is significant different across all 5 groups (i.e. p-value is 0.00).

Finding 3.1: Average relative velocity to complete tasks within 80% of allowed duration. 34% of the workers can

complete the requested tasks within only 10% of allowed time.

Finding 3.2: Higher rated workers are more reliable in task submissions quality, on average 86% of higher rated workers (Red and Yellow) successfully deliver products passing reviews (i.e. granted scores above 75 out of 100), while the number is less than 50% for the workers from the other lower rating groups.

D. Schedule Accelaration (RQ4)

For the purpose of our study, we conducted comparison analysis on the four biggest projects of this data set in different periods of time in terms of uploading the number of tasks.

Table 3 summarizes results from the analysis on schedule acceleration effect in four CSD projects (Part III-D-4). "The "Effort (worker-day)" row shows the effort aggregated from individual submitter's effort on all tasks within a CSD project. The "Effort (worker-month)" row converts the previous effort into a unit of worker-month by dividing 22 (i.e. usually 22 workdays per month as defined in COCOMO II [19]). The "Actual CSD Duration" row shows the duration from the beginning of the first task to the end of the last task within a project. The next three rows provide the estimated representative durations following the three baseline models as discussed in Section 3.4.4, assuming the same project being developed in traditional methodologies other than CSD. Then the next three rows are the calculated SAR by comparing the estimated duration with the actual CSD duration.

TABLE 3. DERIVED SCHEDULE AND SCHEDULE ACCELERATION RATIO

Project	**I**	**II**	**III**	**IV**
Effort (man-day)	6005.7	11037.2	8552.7	13841
Effort (man-month)	273	501.7	388.8	629.1
Actual Duration (months)	9	13	11	14
Duration_I	17.7	20.9	19.5	22.3
Duration_II	16.5	22.4	19.7	25.1
Duration_III	19.1	23.3	21.5	25.2
SAR_I	2	1.6	1.8	1.6
SAR_II	1.8	1.7	1.8	1.8
SAR_III	2.1	1.8	2	1.8
Average SAR	1.97	1.7	1.87	1.73

The last row shows the average SAR for each project, ranging from 1.7 to 1.97. This indicates an overall average SAR of 1.82 (the mean of the four-project level average SAR), with standard deviation of 0.1258. These results reflect that the duration of crowdsourcing software projects is significantly reduced, i.e. by a factor of 1.82 compared with the nominal schedule if following with traditional in-house development methodologies. This indicates a huge potential of schedule shortening in crowdsourced development if with properly design and governance.

Finding 4.1: Compared with traditional methodology, the duration of CSD projects are generally shortened, by a overall schedule acceleration factor of 1.82.

Moreover, team elasticity in CSD is highly flexible, Table 3. This indicates a viable option for teams facing resource shortage that their team elasticity could be improved if they can attract crowd workers who are responding in first couple of days.

V. DISCUSSION AND EVALUATION

A. Worker Availability and Performance

The empirical findings reported above provides insights for software managers, platform providers, and researchers that might help them to better understand worker performance and propose solutions to better governance in CSD.

Finding 1.1 and 1.2 confirm the intuition that workers from different rating groups have very different preferences in task selection and reliability distribution. Obviously, different rating groups are associated with different expertise and experience levels. As summarized in Finding 2.1, the average response time for the top-20 registrants is less than 1 day (i.e. 23.57 hours), and this implies the great potential for organizations to immediately tap into talents beyond traditional boundaries. The evidence in Finding 2.3 shows the strong motivation in completing tasks and winning prizes among early registrants, with more than 50% of submission rate among the first two registrants. This result does not support Top coder assumption of 90%-96% success rate with having more than 1 registrant [22]. However, it is still promising for organizations to take advantage of software expertise in the crowd in successfully delivering requested software solutions.

In terms of crowd worker performance, finding 3.1 provides additional support for the rapid delivery effect attributed to CSD. On average 34% of the workers can complete the requested tasks within only 10% of allowed time; and 80% of workers can complete within 70% of allowed time. When considering the quality of delivery from the crowd workers, as anticipated, higher rated workers are more guaranteed to make satisfactory submissions (Finding 3.2). Though, the data also show that workers in lower rated belts are more reliable in adopting to change and following a more stable submission pattern. Hence, it suggests additional insights for CSD tasks decomposition in a way that can better attract early registrants and leverage various patterns of different rating groups.

Understanding how workers with different experience respond to different types of tasks, different prices, and different durations will help managers in planning for crowdsourcing.

B. Team Elasticity

As shown in Finding 4.1 and 4.2, the SAR results were derived from comparison between the nominal duration of a typical, corresponding traditional in-house projects and the duration of the crowdsourced projects. The average SAR of 1.82, ranging from 1.7 and 1.97, shows the duration of CSD projects are generally shortened by 82%.

According to one of the author's experience with TopCoder projects, this number is reasonable yet conservative. In a recent report [23], it is reported with a total schedule acceleration rate of 3 through CSD, based on an interview study conducted in one TopCoder's client organization. One of the possible reason for this difference is that in our study, the corresponding management overhead within the requesting company is not included as we don't have access to such data. Considering the additional in-house management and coordination effort, the estimated schedule of the corresponding nominal in-house development would most likely be longer, and the resultant schedule acceleration ratio would be even greater.

Based on findings from RQ4, we are interested to investigate the relationship between team elasticity and schedule acceleration effects. We define the measure of team elasticity as follows:

Def. 10: Team Elasticity (TE) is the ratio of the maximum number of registrant for project tasks per week divided by the minimum number of registrants per week, across the total project duration.

Then we conduct additional analysis on team elasticity of the 4 projects used in RQ4, and the results are summarized in Table 4. Pearson correlation analysis result shows there is a 0.76 coefficient between team elasticity and average SAR, which indicates a high correlation between degree of team elasticity and schedule acceleration potential.

This indicates CSD may help organizations with integrating elastic using external human resource to reduce cost from internal employment and exploring the distributed production model to speed up the development process [21]. This can lead the project manager to a better flexible resource allocation rate in terms of diversity of skills, budget and schedule planning.

TABLE 4. TEAM ELASTICITY VS. SCHEDULE ACCELERATION RATIO

Project	**Project I**	**Project II**	**Project III**	**Project IV**
#tasks	156	306	177	277
# Reg / #Sub	2174 / 428	4136 / 807	4222 / 494	2412 / 867
Team Elasticity	187	14.42	8.5	33.3
Average SAR	1.97	1.7	1.87	1.73

C. Challenges in adopting CSD

Although CSD has been increasingly adopted in practice, there are some open issues blocking the scale-up of CSD, such as task decomposition, communication, process alignment, quality assurance, and legal IP issues [4, 24, 25]. Among these, the communication and process alignment concerns seem to be of high relevance in adaptive software development, considering the fact that coordinating workers is difficult among distributed global crowd.

In most cases, organizational coordination techniques can be applied to crowd work as well. CSD platforms generally provide various kinds of communication, collaboration, and coordination services and support among the requesters, providers, and platform vendors [24, 25]. In a recent qualitative study [25], Machado et. al. identified 36 collaboration barriers and 30 communication practices in CSD. They also concluded a mapping between these barriers and practices. Further investigation on the relationship between them is needed in order to provide facilitating methods and techniques to enhance CSD communications.

As to the challenge in aligning the process of in-house agile processes with CSD processes raised in [4], the result of an average task duration of 16 days seems slightly inconvenient to fit into agile cycles. However, in practice, CSD projects on TopCoder actually had a mix bag with agile processes. Some agile teams use TopCoder for extra velocity on their teams to go beyond their normal workload by offloading tasks to TopCoder if they are in a 2-week sprint.

In addition, since 2015 TopCoder employed a new type of tasks, First2Finish, which is to award the first acceptable submission only, and the task will be closed after that, possibly before its planed submission deadline. This new task type is increasingly adopted and will benefit to shrink CSD task timelines to fit into typical agile cycles.

D. Limitations

First, the study only focuses on competitive CSD tasks on the Topcoder platform. Many more platforms do exist, and even though the results achieved are based on a comprehensive set of about 5000 development tasks, the results cannot be claimed externally valid. There is no guarantee the same results would remain exactly the same in other CSD platforms.

Second, there are many different factors that may influence workers' decision in task selection and completion, e.g., motivation patterns, achievement level w.r.t particular problem domains, availability at the time, etc. These are not considered in our current study because of data shortage. What is reported in this study are overall characteristics on various worker rating belt category.

Third, the data we analyzed in schedule acceleration effect was limited to the four sample projects. The estimated schedule was based on a representative total effort by aggregating the crowd worker individual effort for submitting tasks. This assumption may not be valid because the team member relationship is competitive, but in-house team is collaborative.

Fourth, acceptance of results still needs to be proven. Most of the findings are observational ("as it was/is"). First steps towards using the observational insights for actual decision-making haven been made in [6].

Fifth, in analyzing SAR, we did not consider management overhead in the Effort. This factor will increase the task scheduling and consequently larger SAR for both traditional and CSD methods.

Last, as shown in Table 1, the numbers of members in each rating belt vary significantly. For example, the number of red workers is very few, and the number of gray workers which consists of almost 90% of the total members. While

this confirms the typical developer resource pyramid, the data scarcity in the red group may bring limitations in the results.

VI. CONCLUSIONS

Crowdsourced Software Development (CSD) is an emerging paradigm that has been increasingly adopted. Leveraging crowd workforce in CSD has great potential to increase team elasticity and rapid delivery. For adaptive teams to leverage CSD to increase team elasticity, it is critical to understand crowd worker's sensitivity and performance to the tasks and rate of task elasticity and success.

This paper reports an empirical study to address that end. Based on the available empirical data and related research, we developed a set of research questions about the impact of worker performance with different skill and experience level. The evaluation studies were conducted on a set of 4902 competitive crowdsourcing tasks extracted from Top coder platform. The main results of this study showed that on average, (i) 59% of workers respond to a task call in the first 24 hours; (ii) 24% of the workers who registered early will make submissions to tasks, and 76% of them exceeding the acceptance criteria; and (iii) an overall average of 1.82 schedule acceleration rate is observed through organizing mass parallel development in 4 software crowdsourcing projects. Such empirical evidences are beneficial to help exploring resourcing options and improve team elasticity in adaptive software development.

As ongoing and future work, we are considering 1) more data collection on crowd deliverables and develop better understanding about the quality of CSD products, e.g. defects and trustworthiness aspects; 2) investigate on task descriptions and mine task dependency patterns within CSD projects and their relationship to CSD success; 3) develop facilitating techniques to support decisions during task scheduling phase, e.g. CSD process simulator.

ACKNOWLEDGMENT

This research was partially supported by the Natural Sciences and Engineering Research Council of Canada, NSERC Discovery Grant 250343-12 (Third author).

REFERENCES

[1] A. Fuggetta, Software process: A Roadmap, in: The Future of Software Engineering, ACM Press New York, New York, USA, 2000, pp. 25-34.

[2] D. Greening A. Fuggetta, "Agile Base Patterns in the Agile Canon", 2016 49th Hawaii International Conference on System Sciences.

[3] R.Hoda, J.Noble, S.Marshall," Self-organizing roles on agile software development teams". IEEE Trans. Softw. Eng. 39 (3), 422–444,2013.

[4] K.-J. Stol and B. Fitzgerald, "Two's company, three's a crowd: A case study of crowdsourcing software development," in roceedings of the 36th International Conference on Software Engineering, 2014.

[5] K. R.Lakhani, L. B.,Jeppesen, P. A.Lohse, J. A.Panetta, " The value of openess in scientific problem solving". HBS Working Paper Number: 07-050, Harvard Business School, 2007.

[6] Y. Yang, M. R. Karim, R. Saremi and G. Ruhe," "Who Should Take This Task? – Dynamic Decision Support for Crowd Workers", to appear in: Proceedings ESEM 2016.

[7] Lakhani, K.R., Garvin, D.A. and Lonstein, E. 2010. TopCoder (A): Developing Software through Crowdsourcing, Harvard Business School 610-032, 2010.

[8] Topcoder website: http://www.topcoder.com

[9] B. Ye, Y. Wang, and L. Liu. CrowdTrust : A Context-Aware Trust Model for Workers Selection in Crowdsourcing Environments. IEEE ICWS 2015, 2015.

[10] C. Eickhoff and A.P. de Vries. Increasing Cheat Robustness of Crowdsourcing Tasks. Information Retrieval To appear, 2012

[11] P. Ipeirotis, "Be a top mechanical turk worker: You need $5 and 5 minutes," Blog: Behind Enemy Lines, 2010.

[12] https://www.topcoder.com/member-onboarding/understanding-your-topcoder-rating/

[13] N. Archak. Money, glory and cheap talk: analyzing strategic behavior of contestants in simultaneous crowdsourcing contests on TopCoder.com. In Proceedings of the 19th international conference on World Wide Web, WWW '10, pages 21–30, New York, NY, USA, 2010. ACM.

[14] C. Eickhoff and A.P. de Vries. Increasing Cheat Robustness of Crowdsourcing Tasks. Information Retrieval To appear, 2012

[15] A .Kittur, E.Chi, B.Suh,"Crowdsourcing user studies with mechanical turk: 26th annual SIGCHI conference on human factors in computing systems (pp. 453–456). 2008,ACM

[16] A.Sorokin, D.Forsyth," Utility data annotation with amazon mechanical turk",In Computer vision and pattern recognition workshops. CVPRW'08. IEEE computer society conference on IEEE (pp. 1–8) , 2008.

[17] H. Yu, C. Miao, Z. Shen, C. Leung, Y. Chen, and Q. Yang, "Efficient task sub-delegation for crowdsourcing," in AAAI, pp. 1305–1311, 2015.

[18] Y. Yang, R.L. Saremi, "Effects of Award on Worker Behaviors in Crowdsourcing ", Benjin,Chaina, ,Imperical software engineering and measures, ESEM 2015.

[19] B. Bohem, C.Abts, A.Chulani," software development cost approach-A survay", Annal od software engineering 10 , 2000

[20] D.Ingold, B.Boehm," A Model for Estimating Agile Project Schedule Acceleration", International Conference on Software and System Process. Pages 29-35. New York, NY, USA. 2013.

[21] S. Mcconnel "Software estimation: Demystifying the Black Art", 2006, Microsoft Press

[22] A. Begel, J. Bosch, M. A. Storey, "Social networking meets software development: perspectives from github, msdn, stack exchange, and topcoder." IEEE Software, vol. 30 (1), 2013, pp. 52-66.

[23] http://crowdsourcing.topcoder.com/tei

[24] X. Peng, M. Ali Babar and C. Ebert, "Collaborative Software Development Platforms for Crowdsourcing," in *IEEE Software*, vol. 31, no. 2, pp. 30-36, Mar.-Apr. 2014.

[25] M.Leticia, J.Kroll, S.Marczak, R.Pikladniki, "Breaking Collaboration Barriers through Communication Practices in Software Crowdsourcing." Global Software Engineering (ICGSE),11th International Conference on. IEEE, 2016.

[26] K.Mao, L.Capra, M.Harman,Y.Jia,"A Survey of the Use of Crowdsourcing in Software Engineering.",Technical Report RN/15/01, Department of Computer Science, University College London,2013.