Study on Patterns and Effect of Task Diversity in Software Crowdsourcing

Denisse Martinez Mejorado, Razieh Saremi, Ye Yang, and Jose E. Ramirez-Marquez
Stevens Institute of Technology, School of Systems and Enterprises
Hoboken, NJ, United States
{dmartin2,rsaremi,yyang4,jmarquez}@stevens.edu

Abstract—Context: The success of software crowdsourcing depends on steady tasks supply and active worker pool. Existing analysis reveals an average task failure ratio of 15.7% in software crowdsourcing market.

Goal: The objective of this study is to empirically investigate patterns and effect of task diversity in software crowdsourcing platform in order to improve the success and efficiency of software crowdsourcing.

Method: We propose a conceptual task diversity model, and develop an approach to measuring and analyzing task diversity. More specifically, this includes grouping similar tasks, ranking them based on their competition level and identifying the dominant attributes that distinguish among these levels, and then studying the impact of task diversity on task success and worker performance in crowdsourcing platform. The empirical study is conducted on more than one years real-world data from TopCoder, the leading software crowdsourcing platform.

Results: We identified that monetary prize and task complexity are the dominant attributes that differentiate among different competition levels. Based on these dominant attributes, we found three task diversity patterns (configurations) from workers behavior perspective: responsive-to-prize, responsive-to-prize-and-complexity and over-responsive-to-prize. This study supports that 1) responsive-to-prize configuration provides highest level of task density and workers’ reliability in a platform; 2) responsive-to-prize-and-complexity configuration leads to attracting high level of trustworthy workers; 3) over-responsive-to-prize configuration results in highest task stability and the lowest failure ratio in the platform for not high similar tasks.

Conclusions: The findings are helpful to task project owners in a more efficient task planning and improve tasks success level in the platform.

Index Terms—software crowdsourcing, task diversity, worker performance, task success

I. INTRODUCTION

Crowdsourced software development (CSD) has gained increased popularity in recent years for its extraordinary benefits. Examples of such benefits include cost reduction, faster time-to-market, higher quality, creativity and open innovation [1], attributed to its direct access to infinite, diverse talent pool [2], [3].

Generally, decisions regarding to crowdsourcing a software task usually involves organizational and technical factors such as characteristics of the software problem, knowledge and expertise required for deriving viable solutions, and the size of the crowd pool [4], etc. Due to the lack of transparency between task requesters and crowd workers, this emergent CSD paradigm has introduced inherent risks in planning and managing crowdsourced projects [5] [6] [2] [7].

From a task requesters perspective, top risks associated with crowdsourced tasks include uncertainty on both the number and trustworthiness of registrants and quality of the received submissions from the unknown workers [7] [8] [9]. Existing studies reveal a 82.9% task-dropping rate in software crowdsourcing marketplace, which leads to a 15.7% of crowdsourcing failure [10].

From crowd workers’ perspective, they usually choose to register, work and submit for tasks based on some personal utility algorithm, their skills and some unknown factors [11]. It is also reported that they prefer to continue working on similar context tasks in terms of task type, required technology and platform, and their previous experience [12] [13] [14].

Therefore, understanding, attracting, and improving worker engagement is one of the key challenges in successful software crowdsourcing work [15]. For example, higher density of tasks, i.e. a large number of similar tasks during a given time period, in a crowdsourcing market may offer different task opportunities to crowd workers and increase task competition intensity, and consequently lead to higher chance of task starvation or cancellation. It is essential to develop better understanding of the sensitivity in worker performance [7] [16] [17] and task outcome under different competition circumstances, i.e. with respect to different levels of task diversity. To the best of our knowledge, there is no investigation on impact of the different dynamic of tasks supply in the platform as task diversity on tasks success.

The objective of this study is to empirically investigate patterns and impact of task diversity in software crowdsourcing platform in order to improve the success and efficiency of software crowdsourcing. In this study, we first present a motivational example to shed light on the identification of three different patterns of task diversity in software crowdsourcing platform. Then, we propose a conceptual task diversity model, and develop an approach to characterizing and analyzing patterns and effect of task diversity. More specifically, this includes grouping similar tasks, ranking them based on their competition level and identifying the dominant attributes that distinguish among these levels, and then studying the impact of task diversity on task success and worker performance. The empirical study is conducted on more than one years real-world data from TopCoder, the leading software crowdsourc-
ing platform with an online community of over 1.5M workers and 55k mini-tasks (website [https://www.topcoder.com]). The evaluation results show that: 1) monetary prize and task complexity are the dominant attributes of task diversity; 2) exists three task diversity patterns (configurations) based on workers behavior: responsive-to-prize, responsive-to-prize-and-complexity and over-responsive-to-prize; 3) responsive-to-prize configuration provides highest level of task density while a over-responsive-to-prize configuration results in highest task stability and the lowest failure ratio in the platform for not high similar tasks, and 4) responsive-to-prize configuration provides highest level of workers’ reliability in a platform and a responsive-to-prize-and-complexity configuration leads to attracting high level of trustworthy workers.

The remainder of this paper is structured as follows. Section II introduces a motivational example that inspires this study. Section III presents related work. Section IV outlines our research design. Section V presents the empirical results. Section VI discusses the key findings of our study. Section VII presents the conclusion and outlines a number of directions for future work.

II. A Motivational Example

We suppose that task diversity in the platform is a relevant factor in workers decision making process registrations and submissions for a task. To investigate dynamic patterns of crowdsourced software tasks in terms of task supply and submissions for a task. To investigate dynamic patterns of crowdsourced software tasks in terms of task supply and submissions for a task. We visualize for each month the distribution of tasks’ clusters for one month, in the x-axis is the task prize and in the y-axis the task complexity, where each color represents a cluster of tasks. On the right is the legend for the competition level for each cluster, where cl1 groups the cluster with highest competition level (number of registrations), cl2 the second best, cl3 the third and cl4 the fourth. The plot illustrates in blue, the cluster of tasks with higher competition level (number of registrations), 22 in average, followed by 2nd ranked cluster yellow with 21, a 3rd ranked cluster in green, with 16 and the 4th in red with 12. In this time-period, we observe 1) competition level follows a prize order, where we can expect that if a task pays a higher prize it can expect a higher number of registrations; 2) the difference in the average of registrations per cluster does not vary much.

Responsive-to-prize-and-complexity in Figure 1 on the left side, presents the distribution of tasks’ clusters for one month, in the x-axis is the task prize and in the y-axis the task complexity, where each color represents a cluster of tasks. We visualize for each month the distribution of tasks’ clusters for one month, in the x-axis is the task prize and in the y-axis the task complexity, where each color represents a cluster of tasks. On the right is the legend for the competition level for each cluster, where cl1 groups the cluster with highest competition level (number of registrations), cl2 the second best, cl3 the third and cl4 the fourth. The plot illustrates in blue, the cluster of tasks with higher competition level (number of registrations), 22 in average, followed by 2nd ranked cluster yellow with 21, a 3rd ranked cluster in green, with 16 and the 4th in red with 12. In this time-period, we observe 1) competition level follows a prize order, where we can expect that if a task pays a higher prize it can expect a higher number of registrations; 2) the difference in the average of registrations per cluster does not vary much.

Responsive-to-prize-and-complexity in Figure 1 on the middle, shows the case of a time-period, where not only prize plays a role in the attraction of workers, but also task complexity. Observe that the top ranked cluster, colored in blue, groups tasks with high prize but low complexity, with 18 registrations in average. While the 2nd ranked, colored in yellow, has a high prize and complexity, with 17 registrations. In this time-period we observe that workers attraction follows a task prize, followed by task complexity.

Over-responsive-to-prize in Figure 1 on the right side, illustrates the case of a month where we have outliers of tasks in terms of prize. The top ranked cluster in blue, had an average of registrations of 40, while the 2nd ranked (yellow) just 26, the 3rd ranked (green) 20, and 4th ranked (red) only 8. In this month, we observe that when tasks with high prize, compared to the average prize in that time-period, arrives, it creates a disruption and modifies the behavior of workers. In this case the difference in registrations between the 1st and 2nd ranked clusters is 14 registrations, which is more higher than what we observe in responsive-to-prize and responsive-to-prize-and-complexity patterns.

After analyzing these patterns, we performed an analysis of the diversity patterns on task success and worker performance, and found that 1) responsive-to-prize configuration provides highest level of task density and workers’ reliability in a platform; 2) responsive-to-prize-and-complexity configuration leads to attracting high level of trustworthy workers; 3) over-responsive-to-prize configuration results in highest task stability and the lowest failure ratio in the platform for not high similar tasks.

III. Related Work

A. Task Similarity in CSD

Generally, workers tend to optimize their personal utility factor to register for a task. It is reported that workers
are more interested in working in similar tasks in terms of monetary prize [8], context, technology [12], and complexity level. However, workers usually try to register for a greater number of tasks than they can complete [10]. It is reported that high task similarity level negatively impacts task competition level and team elasticity [18]. Combination of these observations lead to receiving task failure due to 1) receiving zero registration for task based on low degree of similar tasks and lack of available skillful worker [8], and 2) receiving non-qualified submissions or zero submissions based on lack of time to work on all the registered task by the worker [19].

B. Workers Performance in CSD

Software workers arrival to the platform and the pattern of taking tasks to completion are factors that shape the worker dynamic in a crowdsourcing platform, however, the reliability in returning the qualified tasks creates the dynamic of the platform. Generally, not only would the award associated with the task influence the workers interests in competitions [14], the number of registrants for the task, the number of submissions by individual workers, and certainly the workers historical score rate would directly affect their final performance [2] [20]. For newcomers or beginners, there is a time window required to improve and to develop into an active worker [11]. Therefore, it is typical that the workers need to communicate with the task owner in order to better understand the problems to be solved [15]. Existing studies show that over time, registrants gain more experience, exhibit better performance, and consequently gain higher scores [11] [21] [15]. Still, there are workers who manage not only to win but also to raise their submission-to-win ratio [12]. This motivate workers to develop behavioral strategies in Topcoder [19] [21]. Moreover, the ranking mechanism used by TopCoder contributes to the efficiency of online competition and provides more freedom of choice for higher rate workers in terms of controlling competition level [19].

C. Modeling Competition in CSD

A few studies have considered game theory in competitive crowdsourcing. For example, Wu et. al. proposed the min-max nature of software competition contributes to the quality and creativity of delivered software [22]. Not only the associated monetary prize with the task would influence competition level, but also the number of registrants for the task, number of submissions by individual worker, and of workers historical score rate would directly affect on the final competition [23]. Such a mutually destructive contest, often called the Chicken Game or Hawk-Dove [24], in which less aggressive competitors (chicken or dove) will yield to aggressive competitors. In such crowdsourced environments, task duration can be used to identify careless workers [23]. Moreover, according to competitive exclusion principle [25], sometimes referred as Gause’s law [26], in competing for limited resources, competitors with the slightest advantage over others frequently win and eventually dominate in the system.

In a CSD platform, a competitive environment not only influences the decisions of contestants regarding which tasks to register and submit but also how these react to their peers. Hu and Wu modeled single-round matches through a game theory framework where participants find bugs in a program [27]. The authors found that the major factors for contestants decisions are effort cost and probability of making a successful challenge, reinforcing the finding in [28], and that contestants with a high rating are more likely to launch challenges against lower ones.

IV. RESEARCH DESIGN

Driven by task diversity patterns introduced in Section II, this study aims at further investigating the dominant influencing attributes and effect of task diversity patterns on crowdsourcing task success. This section presents a conceptual task diversity model, and an approach to characterizing and analyzing patterns and effect of task diversity.

A. Conceptual Task Diversity Model

To better understanding the patterns in the tasks supply during a time-period, we propose a conceptual task diversity model to characterize the basic relationship between task attributes, task diversity pattern, task success and worker performance, as illustrates in Figure 2. For the purpose of the study, we will use the following working definitions.

1) Task Dominant Attributes: Task dominant attributes are the subset of task dynamic attributes that influences task diversity pattern. A list of typical task attributes is summarized in Table I.
2) Task Competition Level: Task competition level can be understood as the level of attraction of workers to a task [20], [8], [19], which can be measured by the number of registrations. More registrations leads to higher competition level per task.

3) Task Diversity Patterns: Task diversity pattern is defined as the configuration of tasks supply in a time-period with respect to different competition levels in a CSD platform.

4) Task Similarity Level: Task similarity level is defined as the degree of task similarity between a set of simultaneously open tasks. To analyze task similarity, we calculated the tasks local distance from each other to analyzed task similarity factor.

Def. 1: Task local distance \(D_i S_j\), is a tuple of all tasks attributes in the dataset. In respect to introduce variables in Table I, task local distance is defined as:

\[
D_i S_j = (P, D, Type, Tech, PL)
\]

Def. 2: Task Similarity Factor \(TS_{ij}\) is dot product and magnitude of local distance of two tasks:

\[
TS_{ij} = \frac{\sum_{i,j=0}^{n} D_i S_j(T_j, T_i)}{\sum_{i=0}^{n} \sqrt{(D_i S_j(T_j))} * \sum_{j=0}^{n} \sqrt{(D_i S_j(T_j))}}
\]

5) Task Success: In this study, task success is defined as a tuple of task density \((TD_i)\), task stability \((TSL_i)\), and task failure \((TF_i)\). Task density is a good measure to determine the probability of attracting workers in a dynamic competitive market base on resource share in the market, while task stability is the probability of receiving a submission by attracted workers. To measure the impact of task diversity pattern on task success, we consider task similarity factor among tasks in same configuration to compare performance metrics among similar tasks.

\[
TaskSuccess_i = (TD_i, TSL_i, TF_i)
\]

for a given time-period \(j\), similarity level \(i\) and task diversity pattern \(k\).

Task density, task stability, and task failure are defined as below:

Def. 1: Task Density \(TD_j\) is the ratio of similar arrival tasks per time-period \(AT_i\) in the platform by total number of arrival tasks \(CAT_i\) per task configuration in the platform,

\[
TD_j = \frac{\sum_{i=0}^{n} AT_i}{\sum_{i=0}^{n} CAT_i}
\]

Average submissions for task \(i\) in the same group of similar tasks per task configuration will illustrates task stability to be completed in the platform.

Def. 2: Task Stability Level, \(TSL_j\), measures average submissions of arrival similar tasks per task configuration.

\[
TSL_j = \frac{\sum_{i=0}^{n} S_i}{\sum_{i=0}^{n} CAT_i}
\]

Average failed similar tasks per task configuration in the same month represents task failure ratio.

Def. 3: Task Failure, \(TF_j\), per similarity level per task configuration is number of failed tasks \((ft_i)\) in the platform, which arrived at the same calendar month per configuration, \(CF_i\):

\[
TF_j = \frac{\sum_{i=0}^{n} ft_i}{\sum_{i=0}^{n} CF_i}
\]

6) Worker Performance: Worker performance is defined as tuple of workers’ reliability in making a submission when registered for a task \((RL_k)\) and workers’ trustworthiness in making a qualified submission for the registered task \((TL_k)\).

Note that each worker is a tuple of number of registration \((WR_k)\), number of submissions \((WS_k)\), and number of valid submission \((WV S_k)\). Similarly, to measure the impact of task diversity pattern on worker performance, we consider task similarity factor among tasks in same configuration to compare performance metrics among similar tasks.

\[
WorkerPerformance_k = (RL_k, TL_k)
\]

for a given time-period \(j\), similarity level \(i\) and task diversity pattern \(k\).

Worker reliability and worker trustworthiness are defined as follow:

Def. 6: Workers Reliability Level, \(RL_k\), measures average reliability for worker \(k\) to register for similar tasks in the same task configuration, \(CR_i\):

\[
RL_k = \frac{\sum_{i=0}^{n} S_i}{\sum_{i=0}^{n} CR_i}
\]

Tasks owners trust on receiving qualified submissions after receiving a registration by a reliable worker. This is defined as the ratio of receiving a valid submission by workers per similarity level in a task configuration to make a submission for the task.

Fig. 2: Conceptual Task Diversity Model
Def. 7: Workers Trust Level, $TL_k$, measures average valid submission, $VS_i$, for worker $k$ to register for similar tasks in the same task configuration, $CR_i$.

\[ TL_k = \frac{\sum_{i=0}^{n} VS_i}{\sum_{i=0}^{n} CR_i} \]

B. Research Questions

Three research questions are formulated as following:

- **RQ1 (Distribution of Task Diversity Patterns):** How do different task diversity patterns (configurations) distribute in a competitive CSD?
  
  This research question aims at providing general overview of task diversity based on the dominant tasks attributes and competition levels per time-period (monthly) in the CSD platform;

- **RQ2 (Task Success):** How does different task diversity patterns impact tasks success?
  
  Understanding task density, task stability and task failure ratio per group of task type can be good measure to indicate tasks success;

- **RQ3 (Worker Performance):** How does different task diversity patterns impact workers performance?
  
  The ratio of attracting reliable workers to compete on the tasks per task type and trust-able to return valid submissions by the workers represent workers consistency to work on tasks.

C. Dataset

The dataset from TopCoder contains 403 individual projects including 4,908 component development tasks and 8,108 workers from January 2014 to February 2015 (14 months). 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 results and grant the scores.

The dataset contains tasks attributes such as technology, platform, task description, monetary prize, days to submit, registration date, submission date, and the time-period (month) on which the task was launched in the platform. In this step, task’s (workers) performance metrics are not included. Then, we create attributes, such as days for registration (RD), days from registration to submission (RTSD), task complexity (TC), which is proxy by the number of words in the task description, number of links in the task description (LC), number of platforms (#PLT), number of technologies (#Tech) and the creation of binary variables for each technology (Tech) and platform (PLT) required in each task, where:

\[ f(x, s) = \begin{cases} 
1 & \text{task x requires technology or platform s} \\
0 & \text{otherwise} 
\end{cases} \]

We create a dataset of task attributes for each time-period (month). The tasks attributes used in the analysis are presented in top section of Table I.

D. Empirical Study Design

In this work we followed a similar task content similarity clustering approach as [29], which includes task features such as task duration, award, technology and platform and applied a K-Means algorithm. As Figure [3] illustrates, our approach contains the following steps.

Fig. 3: Main flow of the proposed framework and relationship to research questions.

1) Clustering Analysis: We perform a clustering analysis to identify the configurations of groups of tasks for each time-period (month). For each dataset we perform the following: first, an unsupervised clustering for each time-period where each task is a vector of attributes and these were normalized transforming the original values to values between 0 and 1. The clustering method used is K-Means. To identify the number of optimal clusters we use the “elbow” criteria. This method observes the variance explained as the number of clusters increases. We select the “optimal” number of clusters, when the marginal gain of adding one more cluster drops, creating an “elbow” in the plot that illustrates the inverse of variance explained per number of clusters. The number of optimal clusters for each month varies.

Second, we rank each cluster by the competition level, which is the mean of registrants per cluster. In this step, we have every observation (task) assigned to a cluster where cluster 1 is the top ranked, cluster 2 the second best ranked, successively.

Third, we identify the dominant attributes that differentiate ranked clusters to one another. We performed a decision tree classification where the input are the attributes created and the classification attribute is the ranked cluster from previous step. The dataset is divided in two samples: train (75%) and test (25%) and different decision trees were run, looking for the least number of attributes that generate the highest accuracy in the classification. After performing these three steps for each
The task diversity patterns (configurations) were defined as given based on the response of workers in each configuration. These names were responsive-to-prize, responsive-to-prize-and-complexity and over-reactive-to-prize as shown in Figure 1. These names were ranked cluster label of each observation for each time-period. In our analysis the frequent dominant attributes for each time-period were monetary prize (x-axis) and complexity (y-axis). Each observation is colored based on the cluster's rank. For example, tasks in cluster 1 (top ranked) are colored in blue, cluster 2 (2nd best) in yellow, cluster 3 in green and cluster 4 in red. This allows to visualize the distribution of clusters considering the dominant attributes identified previously by visualizing the dominant attributes in Table II for each time-period. As it is observed, the frequent dominant task attributes that differentiate among the ranked clusters are prize and complexity.

2) Task Diversity Analysis: We analyze each time-period by visualizing the dominant attributes identified previously and included the ranked cluster label of each observation for each time-period. In our analysis the frequent dominant attributes for each time-period were monetary prize (x-axis) and complexity (y-axis). Each observation is colored based on the cluster's rank. For example, tasks in cluster 1 (top ranked) are colored in blue, cluster 2 (2nd best) in yellow, cluster 3 in green and cluster 4 in red. This allows to visualize the distribution of clusters considering the dominant attributes (prize and complexity) and how well the clusters performed against each other (competition level).

After analyzing the plots for all time-periods, we found three task diversity patterns (configurations): responsive-to-prize, responsive-to-prize-and-complexity and over-reactive-to-prize as shown in Figure 1. These names were given based on the response of workers in each configuration. The task diversity patterns (configurations) were defined as follows:

- **Responsive-to-prize**: time-period where competition level follows prize order. In this configuration we expect that tasks with higher monetary prize attract higher competition level, while the average competition level per task cluster does not vary much;
- **Responsive-to-prize-and-complexity**: time-period where competition level follows both prize and task complexity order. In this configuration we expect that tasks attract competition level not only based on monetary prize but also how complex a task is;
- **Over-reactive-to-prize**: time-period where there are few tasks with monetary prize above the average, as an outlier, which are attracting the highest competition level. In this configuration a considerable difference in task competition level between the 1st and 2nd ranked clusters is expected.

3) Task Success Analysis: We study the impact of the different task diversity patterns (configurations) on tasks success by looking into task density, task stability and task failure ratio. To do so, the number of similar arrival task as well as similar open task per task configuration was analyzed. Then, task density, task stability and task failure per task similarity level and task configuration was analyzed.

4) Worker Performance Analysis: We investigate the impact of the different task diversity patterns (configurations) on workers performance by looking into workers’ reliability and trust. The probability of a worker make a submission after registered for a task will be reported as worker reliability, and the probability of a submission passes the peer review and labeled as valid submission provides workers trust. In TopCoder, crowd workers reliability of competing on the tasks is measured based on last 15 competitions workers registered and submitted. For example, if a worker submitted 14 tasks out of 15 last registered tasks, his reliability is 93% (14/15).

This analysis is relevant since each task configuration may impact the submissions of workers, for example, in the pattern over-reactive-to-prize we observed that tasks with high prize above the average, attract more competition level, however how many of these registrants will actually submit their task.

V. EMPIRICAL RESULTS

1) Distribution of Task diversity patterns (RQ1): We perform a longitudinal analysis of task diversity patterns analyzing 14 months, January 2014 to February 2015.

The dominant task attributes weights are summarized on Table III for each time-period. As it is observed, the frequent most dominant is monetary prize, followed by complexity. These attributes are key to differentiate among different competition levels and used to identify task diversity patterns (configurations).

Table III summarizes the tasks’ configurations for each month. As it can be observed, 5 months had a

### TABLE I: Summary of Metrics Definition

| Type               | Metrics                        | Definition                                           |
|--------------------|--------------------------------|------------------------------------------------------|
| Days for Registration (RD) | Number of days to for the task. |
| Days from Reg. to Sub.(RTSD) | Number of days to submit the task once registration is closed. |
| Duration (D)     | Total available days from registration date to submissions deadline. |
| Monetary Prize (P) | Monetary prize (Dollars) in task description. |
| Task Complexity (TC) | Number of words in task description. |
| Link count (LC)  | Number of links in the task description. |
| Technology (Tech) | Required programming language to perform the task. |
| Platform (#PLT)  | Associate platform that a task is performing. |
| Technologies (#Tech) | Number of technologies used in task. |
| Task Type (Type)  | Type of challenge depends on development phase. |
| Task Status       | Completed or failed task. |
| # Registration (R) | Number of registrants that are willing to compete on total number of tasks in specific period of time. |
| # Submissions (S) | Number of submissions that a task receives by its submission deadline in specific period of time. |
| # Valid Submissions (VS) | Number of submissions that a task receives by its submission deadline and passed the peer review in specific period of time. |
TABLE II: Top dominant attributes weights

| Attribute | Jan’14 | Feb’14 | Mar’14 | Apr’14 | May’14 | Jun’14 | Jul’14 | Aug’14 | Sep’14 | Oct’14 | Nov’14 | Dec’14 | Jan’15 | Feb’15 |
|-----------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| Prize     | 0.599  | 0.951  | 0.775  | 0.994  | 0.967  | 1      | 0.956  | 0.976  | 1      | 0.886  | 0.988  | 0.852  | 1      | 0.486  |
| Complexity| 0.401  | 0.032  | 0.185  | 0.007  | 0.039  | 0.024  | 0.106  | 0.004  | 0.143  | 0.514  |

responsive-to-prize configuration, 3 were responsive-to-prize-and-complexity and 6 were over-responsive-to-prize.

In the 1st half of 2014, we had responsive-to-prize months, which means that we observed an expected behavior: higher attraction of workers (competition level) as the prize is higher.

In the 2nd half of the 2014, we observed over-responsive-to-prize pattern, where we had tasks with high prize, above the average, creating a disruption and attracting an unusual number of workers to register to these.

The responsive-to-prize-and-complexity months were less frequent. These periods, the workers decision to register is not only determined by the prize, but also the complexity of the tasks. In this case, a task manager should consider this dynamic if they want to have success when launching high-complex tasks given this dynamic of tasks supply.

Finding 1.1: The task diversity patterns identified were responsive-to-prize, which was present in the 1st half of 2014, over-responsive-to-prize frequent in the 2nd half of 2014, while responsive-to-prize-and-complexity was the least frequent, found only in 3 months.

2) Task Success (RQ2): In order to have a better understanding of tasks success, we studied distribution of task density in the different tasks similarity levels per task diversity pattern (configuration). By considering the task similarity among tasks, we can compare similar tasks under different task configurations. Figure 4 illustrates the task density ratio per similarity level per task configuration.

As it is clear in Figure 4, all tasks configurations have increasing trend of task density as task similarity increases. While responsive-to-prize and responsive-to-prize-and-complexity configurations seem to follow a similar shape, over-responsive-to-prize configuration started with lowest task density among tasks with 60% similarity with 0.05% task density. This pattern raised, containing the highest tasks density among all three configurations when tasks with 90% similarity arrive in the platform with 76% task density. Responsive-to-prize configuration follows a steady increasing pattern in providing task density with increasing task similarity in among open tasks in the platform. While it provides around 16% and 20% density for task similarity of 70% and 80% respectively, the density level increased to 52% for tasks with similarity level of 90%. On the other hand, tasks density dropped for the responsive-to-prize-and-complexity configuration when task similarity increased from 60% to 70% from 14% to 11%, and again increased to 30% for tasks with 80% similarity, and ended to 44% task similarity among tasks with similarity of 90%.

Another element that impacts on task robustness is task stability. We investigate the level of task stability per similarity level per task configuration. As Figure 5 shows, responsive-to-prize-and-complexity configuration is providing lowest task stability in 60% similarity with almost 10% stability, while over-responsive-to-prize is the highest with 19% stability level and balance configuration provides 16% task stability for the group of tasks with 60% similarity. Interestingly all tasks configurations are providing almost 20% of task stability when we have a task arrival with 70% similarity. While task stability increased to 21% for responsive-to-prize and over-responsive-to-prize configurations among tasks with 80% similarity, task stability dropped to 12% for responsive-to-prize-and-complexity configuration. All three configurations follow the same pattern with task stability of 23% for responsive-to-prize and over-responsive-to-prize, and 14% for responsive-to-prize-and-complexity configuration.

After understanding different patterns of task density and task stability for each configuration, we investigate the task failure ratio per task similarity level per configuration. Figure 6 illustrates the distribution of task failure ratio per similarity level per task configuration.
TABLE III: Task diversity pattern per month.

| Cont. | Jan'14 | Feb'14 | Mar'14 | Apr'14 | May'14 | Jun'14 | Jul'14 | Aug'14 | Sep'14 | Oct'14 | Nov'14 | Dec'14 | Jan'15 | Feb'15 |
|-------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| RP    | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      |
| RPC   | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      |
| ORP   | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      | ✔      |

RP = responsive-to-prize, RPC = responsive-to-prize-and-complexity and ORP = over-responsive-to-prize.

level per configuration. Interestingly, over-responsive-to-prize provides on average lower task failure ratio in the platform. Tasks with similarity level of 60% face almost 3% failure under over-responsive-to-prize configuration. It increased to 19% failure ratio for tasks with 70% similarity and 29% for similarity of 80% and above. While responsive-to-prize configuration faced the failure ratio of 23% for task similarity of 60% and 26% for task similarity of 70%. It drops to 23% and 21% for tasks with similarity of 80% and 90% respectively. Responsive-to-prize-and-complexity configuration started with 14% task failure for 60% task similarity, decreasing to 10% task failure for tasks with 70% similarity, and 7% for tasks with 80% similarity. However, all the tasks with 90% similarity failed in the responsive-to-prize-and-complexity configuration.

Finding 2.1: For tasks with similarity level less than 90%, over-responsive-to-prize configuration provides the lower task density, possibly attracting more workers. For tasks similarity of 90% responsive-to-prize-and-complexity configuration provides lowest task density.

Finding 2.2: Over-responsive-to-prize configuration provides highest level of task stability for all different task similarity level.

Finding 2.3: For tasks with similarity level of 60%, over-responsive-to-prize configuration provides the lowest failure ratio, tasks between 70% - 80% similarity responsive-to-prize-and-complexity configuration has lowest failure and with high similarity of 90% balance configuration lead to lowest task failure ratio.

3) Worker Performance (RQ3): Workers performance can be analyzed based on workers reliability and trust in returning a valid submission. Therefore, we analyzed attracting reliable workers and receiving valid submissions per task based on the open tasks in different task configuration. Figure 7 illustrates the average workers reliability per task configuration. As it is shown in Figure 7, responsive-to-prize configuration attracts the most reliable workers for all different similarity level. Task in responsive-to-prize configuration attracted workers with 12% reliability for 60% and 70% task similarity and 14% reliability for tasks with 80% similarity and more. However, over-responsive-to-prize configuration attracts workers with reliability of 11% for task similarity of 60%, it followed almost the same pattern as responsive-to-prize configuration for rest of the task similarity levels. Responsive-to-prize-and-complexity configuration attracts the lowest reliable workers among task configurations. The highest reliability happens among 70% similar tasks which is lower than the lower reliability among responsive-to-prize and shewed. Task similarity with 60%, 80% and 90% attracted workers with reliability of 7%, 8% and 8% respectively for responsive-to-prize-and-complexity configuration.

Besides attracting reliable workers to make a submission, it is important to trust on the workers submissions. Hence, we investigate workers trust ratio in returning valid submission. Figure 8 represents the distribution of workers trust ratio among different task similarity per configuration. Among tasks with similarity of 60%, over-responsive-to-prize configuration received the highest trust ratio of 19% followed by responsive-to-prize for 13% and responsive-to-prize-and-complexity for 8% trust. Interestingly in task similarity of 70%, workers trust in responsive-to-prize-and-complexity configuration raised up to 18% while for both responsive-to-prize and over-responsive-to-prize trust is about 16%. Workers trust ratio increased to 17% for task similarity of 80% in responsive-to-prize configuration and 16% for over-responsive-to-prize configuration.
ration, while it decreased to 11% for responsive-to-prize-and-complexity configuration. And finally, for task similarity of 90% task trust increased to 20%, 22%, 14% for responsive-to-prize, over-responsive-to-prize and responsive-to-prize-and-complexity, respectively. Interestingly while responsive-to-prize configuration is following a linear pattern among different task configurations, over-responsive-to-prize is providing a U-shape pattern.

![Figure 8: Average Workers’ Trust per Task Similarity and Diversity Configuration.](image)

Finding 3.1: Responsive-to-prize configuration attracts the highest reliable works among all different task configurations.

Finding 3.2: Over-responsive-to-prize configuration attracts highest trustworthy workers.

VI. DISCUSSION

A. Task Diversity Patterns

The dominant task attributes that differentiate among one competition level (number of registrations) and another, are monetary prize and complexity, reinforcing findings on previous research [2], [4], [11], [15], [19].

These dominant attributes were input for the detection of task diversity patterns. We identified three configurations: responsive-to-prize, responsive-to-prize-and-complexity and over-responsive-to-prize. During January 2014 and February 2015, we identified that the 1st half of 2014 was frequent a responsive-to-prize configuration, while in the 2nd half presented an over-responsive-to-prize pattern. Only in three months was observed a responsive-to-prize-and-complexity, finding 1.1.

B. Task Success

To successfully crowdsource a software project in a CSD platform, not only, it is important to fully understand task density in the platform at any point of time. But also, it is vital to know the task stability and failure ratio in the platform. This research investigated these factors based on task similarity level per identified task configuration. It is reported that task similarity impacts highly on task competition level, task stability level and consequently project success [18]. As it is shown in Figure 4 responsive-to-prize configuration provides a smoother raise in task density per similarity level finding 2.1. Higher task density lead to a more task taking choice for workers. Generally, it seems responsive-to-prize-and-complexity configuration provides higher level of task density among lower task similarity.

Accord to finding 2.2, over-responsive-to-prize configuration provides higher level of stability for all task similarity levels. However, it is reported that increasing degree of task similarity in the platform would ease switching context for workers, and lead to lots of failure tasks due to starvation or zero submissions [30] [12]. responsive-to-prize configuration receives the lowest failure ratio among the highest task similarity level. While over-responsive-to-prize configuration provides the lowest failure ratio for tasks with the lowest similarity level (i.e 60%) and then increased to the same level of failure as responsive-to-prize configuration. Interestingly responsive-to-prize-and-complexity configuration lead to failure ratio lower than 10% for task similarity lower than 90%.

Generally, as it is reported in finding 2.3, for task with similarity lower than 90%, competing in the responsive-to-prize-and-complexity configuration is the best strategy in terms of task failure and task density, while this configuration does not provide a good task stability.

C. Worker Performance

To assure of having a successful project in CSD, beside the importance of attracting reliable workers whom make a submission, it is crucial to make sure to at least attract one trustworthy worker who not only makes a submission but also makes a valid submission. The result of investigating workers performance under different task diversity patterns (configurations) presented that responsive-to-prize configuration of tasks attract highest number of reliable workers for all different similar groups of tasks, finding 3.1.

Moreover, finding 3.2 indicates that, although over-responsive-to-prize configuration of tasks is runner up in attracting reliable workers, it successfully attracted the most trustworthy group of workers among all different configurations. Workers trust is following a U-shape and decreasing for task similarity of 70% and 80%, yet on average it provides the highest trust ratio among all three identified tasks configurations.

D. Threats to Validity

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 5,000 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 tasks similarity, task diversity patterns and workers decision in task selection and completion. Our similarity algorithm and task diversity patterns approach are based on known task attributes in TopCoder. Different similarity algorithm and task diversity patterns approaches may lead us to different but almost similar results.
workers performance.

VII. CONCLUSION AND FUTURE WORK

To understand the probability of a task's success in a crowdsourcing platform, not only one should understand the task diversity patterns but also, they need to understand the tasks' success per point of time and workers' performance in taking a task and returning a valid submission. This research investigated task diversity by applying clustering method based on the dominant task attributes: prize and complexity and observed the competition level along these attributes. Then analyzed both tasks' success and workers' performance per configuration based on task similarity level.

This research shows that attracting competition following complexity order negatively impacts on attracting trustworthy workers and consequently tasks' success in the platform, while attracting reliable workers in returning submissions. On the other hand, following task monetary prize leads to lowest level of task failure for middle-level similar tasks.

This study supports that 1) responsive-to-prize configuration provides highest level of task density and workers' reliability in a platform; 2) responsive-to-prize-and-complexity configuration leads to attracting high level of trustworthy workers; 3) over-responsive-to-prize configuration results in highest task stability and the lowest failure ratio in the platform for not high-similarity tasks.

In future, we would like to focus on the similar crowd worker behavior and performance based on task similarity level and analyze workers decision making based on task-worker performance network to report more decision elements according to task size and scheduling date, task utilization and workers performance.

REFERENCES

[1] K.-J. Stol and B. Fitzgerald, “Two’s company, three’s a crowd: a case study of crowdsourcing software development,” in Proceedings of the 36th International Conference on Software Engineering, 2014, pp. 187–198.
[2] K. R. Lakhani, D. A. Garvin, and E. Lonstein, “Topcoder (a): Developing software through crowdsourcing,” Harvard Business School General Management Unit Case, no. 610-032, 2010.
[3] A. L. Zanatta, L. S. Machado, G. B. Pereira, R. Prikладник, and E. Carmel, “Software crowdsourcing platforms,” IEEE Software, vol. 33, no. 6, pp. 112–116, 2016.
[4] N. H. Thuan, P. Antunes, and D. Johnstone, “Factors influencing the decision to crowdsource: A systematic literature review,” Information Systems Frontiers, vol. 18, no. 1, pp. 47–68, 2016.
[5] K.-J. Stol and B. Fitzgerald, “Researching crowdsourcing software development: perspectives and concerns,” in Proceedings of the 1st International Workshop on Crowdsourcing in Software Engineering, 2014, pp. 7–10.
[6] K. R. Lakhani, L. B. Jeppesen, P. A. Lobse, J. A. Panetta et al., “The value of openness in scientific problem solving,” 2007.
[7] H. Yu, C. Miao, Z. Shen, C. Leung, Y. Chen, and Q. Yang, “Efficient task sub-delegation for crowdsourcing,” in Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.
[8] Y. Yang and R. Saremi, “Award vs. worker behaviors in competitive crowdsourcing tasks,” in 2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). IEEE, 2015, pp. 1–10.
[9] C. Eickhoff and A. P. de Vries, “Increasing cheat robustness of crowdsourcing tasks,” Information retrieval, vol. 16, no. 2, pp. 121–137, 2013.
[10] Y. Yang, M. R. Karim, R. Saremi, and G. Ruhe, “Who should take this task? dynamic decision support for crowd workers,” in Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, 2016, pp. 1–10.
[11] S. Faradani, B. Hartmann, and P. G. Ipeirotis, “Whats the right price? pricing tasks for finishing on time,” in Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence, 2011.
[12] D. E. Difallah, G. Demartini, and P. Cadre-Mauroux, “Scheduling human intelligence tasks in multi-tenant crowd-powered systems,” in Proceedings of the 25th international conference on World Wide Web, 2016, pp. 855–865.
[13] M. J. Crump, J. V. McDonnell, and T. M. Gureckis, “Evaluating amazon’s mechanical turk as a tool for experimental behavioral research,” PloS one, vol. 8, no. 3, 2013.
[14] M. Yin, Y. Chen, and Y.-A. Sun, “Monetary interventions in crowdsourcing task switching,” in Second AAAI Conference on Human Computation and Crowdsourcing, 2014.
[15] A. Kittur, J. V. Nickerson, M. Bernstein, E. Gerber, A. Shaw, J. Zimmerman, M. Lease, and J. Horton, “The future of crowd work,” in Proceedings of the 2013 conference on Computer supported cooperative work, 2013, pp. 1301–1318.
[16] A. Kittur, E. H. Chi, and B. Suh, “Crowdsourcing user studies with mechanical turk,” in Proceedings of the SIGCHI conference on human factors in computing systems, 2008, pp. 453–456.
[17] A. Sorokin and D. Forsyth, “Utility data annotation with amazon mechanical turk,” in 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops. IEEE, 2008, pp. 1–8.
[18] R. Saremi, M. L. Saremi, P. Desai, and R. Anzalone, “Is this the right time to post my task? an empirical analysis on a task similarity arrival in topcoder,” arXiv preprint arXiv:2004.12501, 2020.
[19] 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, 2010, pp. 21–30.
[20] R. L. Saremi, Y. Yang, G. Ruhe, and D. Messinger, “Leveraging crowdsourcing for team elasticity: an empirical evaluation at topcoder,” in 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP). IEEE, 2017, pp. 103–112.
[21] N. Archak and A. Ghose, “Learning-by-doing and project choice: a dynamic structural model of crowdsourcing,” LEARNING, vol. 1, pp. 1–2010, 2010.
[22] W. Wu, W.-T. Tsai, and W. Li, “Creative software crowdsourcing: from components and algorithm development to project concept formations,” International Journal of Creative Computing, vol. 1, no. 1, pp. 57–91, 2013.
[23] Y. Moshfeghi, A. F. Huertas-Rosero, and J. M. Jose, “Identifying careless workers in crowdsourcing platforms: a game theory approach,” in Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, 2016, pp. 857–860.
[24] A. Rapoport and A. M. Chammah, “The game of chicken,” American Behavioral Scientist, vol. 10, no. 3, pp. 10–28, 1966.
[25] G. Hardin, “The competitive exclusion principle,” science, vol. 131, no. 3409, pp. 1292–1297, 1960.
[26] A. Pocheville, “The ecological niche: history and recent controversies,” in Handbook of evolutionary thinking in the sciences. Springer, 2015, pp. 547–586.
[27] Z. Hu and W. Wu, “Game theoretic analysis for offense-defense challenges of algorithm contests on topcoder,” in 2015 IEEE Symposium on Service-Oriented System Engineering. IEEE, 2015, pp. 339–346.
[28] ——, “A game theoretic model of software crowdsourcing,” in 2014 IEEE 8th International Symposium on Service Oriented System Engineering. IEEE, 2014, pp. 446–453.
[29] Y. Fu, H. Sun, and L. Ye, “Competition-aware task routing for contest based crowdsourced software development,” in 2017 6th International Workshop on Software Mining (SoftwareMining). IEEE, 2017, pp. 32–39.
[30] R. Saremi, “A hybrid simulation model for crowdsourced software development,” in Proceedings of the 5th International Workshop on Crowd Sourcing in Software Engineering, 2018, pp. 28–29.