A Hybrid Simulation Model for Failure Prediction in Crowdsourced Software Development

Razieh Saremi, Ye Yang
School of Systems and Enterprices
Stevens Institute of Technology
Hoboken, NJ, USA
{rsaremi, yyang4, gvesonde}@stevens.edu

Guenther Ruhe
Software Engineering Decision Support Lab
University of Calgary
Alberta, Canada
ruhe@ucalgary.ca

He Zhang
Software Institute
Nanjing University
Jiangsu, China
hezhang@nju.edu.cn

Abstract—A typical crowdsourcing software development (CSD) marketplace consists of a list of software tasks as service demands and a pool of freelancer developers as service suppliers. Highly dynamic and competitive CSD marketplaces may result in task failure due to unforeseen risks, such as increased competition over shared worker supply, or uncertainty associated with workers’ experience and skills, and so on. To ensure the success and effectiveness of CSD, it is essential to develop improved understanding and visibility into characteristics and risks associated with CSD processes. In this paper, we present a hybrid simulation model to forecast crowdsourcing task failure risk in competitive CSD platforms. The simulation model is composed of three layered components: the macro-level reflects the overall crowdsourcing platform based on system dynamics, the meso-level represents the task life cycle based on discrete event simulation, and the micro-level models the crowd workers’ decision-making processes based on agent-based simulation. This hybrid simulation model is evaluated through three CSD decision scenarios to demonstrate its effectiveness, using a real-world historical dataset. The proposed simulation model empowers crowdsourcing managers to explore potential crowdsourcing outcomes with respect to different task scheduling options.

Index Terms—Crowdsourcing, Task Scheduling, Task Similarity, Task Failure, Simulation, Agent Based, Discrete Event, Systems Dynamic, TopCoder

I. INTRODUCTION

As an emerging paradigm, Crowdsourced Software Development (CSD) has been increasingly adopted to develop software applications [1] [2]. A crowdsourced environment happens where there is a pool of undefined human resources with internet access engaged in an articulated market of online open call projects [3]. The CSD platform can be considered as a supply-and-demand market between tasks providers and workers [4]. In order to function effectively, it must address both the needs of task providers as demands and crowd workers as suppliers. From task owners perspective, a CSD platform needs to ensure their tasks receiving broad participation, and on-time, high-quality deliveries. From crowd workers’ perspective, a CSD platform needs to address a diverse set of worker motivations for participation [5] [6], such as winning task prizes, gaining community reputation, building technical skills, and so on.

In CSD practice, any mismatch between these needs may lead to task failure. Task failure happens due to lack of available skillful workers [7], or crowd workers who are not interested in taking the tasks based on some personal utility algorithm, their skills and some unknown factors [8]. Existing studies reveal a 82.9% task-dropping rate in software crowdsourcing marketplace, which leads to a 15.7% of crowdsourcing failure [9]. On the other hand, task owners do not know or have control over crowd workers, which raises concerns on the qualification and trustworthiness of the crowd workers [2], [6], [10]. On the other hand, crowd workers are shared resources frequently multi-tasking on different requesters tasks based on their individual goals and preference. These implicit factors frequently complicate the CSD processes and cause resource discrepancy, and consequently increase the likelihood of task failure.

Existing literature shows that workers are more interested in tasks matching their expertise and/or prior experience, and their preference on task selection [11] [5] [12] [13]. Generally speaking, task similarity is one of the most important factors in a worker’s decision-making process, which is based on a worker’s individual knowledge and skill set [14]. Higher task similarity provides more choices for crowd workers to choose tasks, and higher openness for platforms to work with different task providers. As a platform moved from low to moderate levels of openness, innovation among crowd workers increased [15] [16].

Task scheduling is a common method to prevent resource discrepancy, and decrease task failure [4]. Various task scheduling methods, such as HIT-Bundle [4] and QOS [17], have been introduced in the crowdsourcing research field to aid the process of researching, designing, developing and training system to reduce task failures. Existing scheduling methods primarily focus on static aspects of tasks, such as task ID, or task’s type, or task’s required technology and neglect dynamic aspects of tasks, such as workers’ registration on the qualification and trustworthiness of the crowd workers on last T days per similar tasks or workers’ average submissions rate on similar arrival tasks or number in the platform. In addition, existing scheduling methods fail to provide enough information to predict different task execution patterns based on workers’ decision making. To that end, this paper proposes...
a novel hybrid simulation model for task failure prediction in CSD. It integrates three different simulation techniques, i.e. system dynamics simulation (SDS) [18], discrete event simulation (DES) [19], and agent-based simulation (ABS) models [20], to provide insights on the effects of various compounding factors [21] [22] for attracting reliable crowd workers on the task failure risk in CSD. While SDS methods represent the overall view of the system and dynamic relation among the stakeholders [18], DES models capture the actual process-level details in development [19], and ABS models simulate the behavior among individual workers and social groups [20]. Moreover, simulation allows one to reenact a scenario in order to identify bottlenecks in the process and provide solutions [23].

The simulation model aims at answering the following question: **Given a competition context in a CSD marketplace, how to strategically schedule a new task to ensure task success and prevent task failure?** More specifically, this paper reports the impact of three different components of the proposed model in crowdsourcing task scheduling. In developing the hybrid simulation model, we leverage empirical findings including crowd worker behavior, task completion outcome, and crowd team performance from our previous works [5] [24] [25], and packaged knowledge on SDS, DES, and ABS simulation models in different layers as three different components of the proposed simulation model. SDS model simulation follows empirical finding on workers behavior based on task speciation [5], DES model simulates task completion flow based on task arrival patterns in CSD [24] and ABS model simulates workers performance following empirical findings of team performance [25]. To the best of our knowledge, there is no existing study using a hybrid simulation model of CSD based on a combination of all three different simulation methods of SDS, DES, and ABS.

The rest of the paper is organized as follows: Section 2 presents a motivational example in CSD task scheduling, Section 3 provides the overview of the proposed work, Section 4 presents the implementation of the hybrid simulation model, Section 5 illustrates its effectiveness through two case studies, Section 6 is related work, and Section 7 concludes the paper with future work.

II. BACKGROUND

A. A Motivational Example

Figure 1 illustrates a CSD project on the TopCoder platform, which consists of 19 tasks with a total duration of 110 days. The project experienced a 57% task failure ratio, meaning 11 of the 19 tasks were not successful. More specifically, 8 tasks (i.e. #1, 2, 4, 5, 8, 11, 13, and 17) failed due to receiving zero submissions, and other reasons for failure include client request and unclear requirements, etc. These tasks are basically repeated postings of three distinct tasks (i.e. tasks 2, 4, and 8) as illustrated in Figure 1-a, task 2 was cancelled and re-posted as tasks 5 and 7, which was finally completed on the third attempt. Similarly, task 4 was re-posted as task 6, and completed with no modification. More interestingly, task 8 was re-posted 6 times, as tasks 11, 13, 14, 15, 17, and 18 respectively. Each re-posting modified Task 8 in terms of the monetary prize, task type, and required technology. Finally, the task was successfully completed as Task 18.

Would it be possible that task similarity and worker reliability play role in task success? As shown in Figure 1-b, this seems to be true from the comparison of the number of simultaneously open, similar tasks as well as average worker reliability across 7 re-postings for task 8. More specifically, these 7 re-posted tasks were competing on average with 27 other tasks with similarity level as degree of task openness greater than 60%, while the successful re-posted tasks were only competing with five tasks with the same level of openness. Interestingly, the task with lowest number of competing open tasks attracted workers with higher average reliability factor of 0.19. Task 8’s journey in the motivation example supports the fact that the level of task openness directly impacts task success. Also, a lower level of task openness in the platform positively impacts attracting workers with a higher reliability factor; this relationship is demonstrated in Figure 1-b. Attracting workers with a higher reliability factor increases the chance of receiving qualified submissions and task success. Existing research reflects that crowd workers are more interested in working on tasks with similar skillset requirements [4]. Compared with general crowdsourcing platforms, CSD platforms are designed for crowdsourcing competitively paid, complex, and knowledge-intensive software development tasks. Therefore, it is assumed that there is a much stronger correlation between the degree of a task’s openness in terms of task similarity and resource diversity than in terms of skillset. However, observations from the motivational example indicate that it is difficult to draw inclusive conclusions regarding the causality relationships among task attributes, worker attributes, and task failure, due to the complicated, intertwining relationships among them. To the best of our knowledge, none of the available studies focused on the causal similarity among simultaneously open tasks on the platform and its impact on workers’ availability and task failure.

B. Previous Work

In general, new task postings and worker arrivals are discrete events, typically following exponential distribution [26]. In addition, the higher the level of diversity among active workers, the lower the degree of task failure [26]. Previously we reported three empirical studies which led to the understanding and development of empirically derived worker behavior patterns, open team performance patterns, and task completion patterns. The result of these empirical studies showed that:

Crowd Worker Behavior: Generally, the monetary prize associated with task, as an element of task openness, and the competition level per task have an inverse relation [5]. It appears that there is a direct relationship between the competition level per task and number of submissions; however, increasing the level of task attractiveness for workers leads to a decrease...
in the number of submissions in some cases. Interestingly, workers’ performance follows a variety of U-shape curves by changing task openness for similar tasks [5].

Crowd Team Performance: More than half of workers responded to a task call on the very first day of task’s publication and registered for them, while only 24% of registrants make a submission (i.e. 14% of available workers) [25]. Also 76% of the submitters (i.e. 10% of available workers) exceeded the acceptance criteria [25]. Higher experience among workers resulted in a higher chance of task submissions (i.e. 60%), while lower experience resulted in a lower level of task submissions (i.e. 25%); average experience resulted in a submissions level of around 57% [25]. Table I summarizes workers’ performance on the platform.

Task Completion Outcome: Parallelism is the strongest method of task arrival in CSD [24]; it leads to an 82% reduction in the schedule acceleration rate [25]. A higher degree of parallelism can lead to increased task competition and a shorter planning schedule to complete the project, thus providing consequently better resource allocation. Moreover, it is reported that about 75% of posted tasks would price under 67% of total project budget in parallel mass production [24].

III. THE PROPOSED HYBRID SIMULATION MODEL

A. Working Definition

In this work, we present a model for CSD involving four types of entities: platform, tasks, projects, and crowd-workers.

Platform: A platform is a marketplace where companies place open calls for specific software development tasks to be completed by a large pool of undefined groups of external software workers [19]. The general purpose of a crowdsourcing platform is to provide a market in which a company can: (1) advertise tasks to be crowdsourced; (2) specify task rewards paid to task winners; and (3) allow crowd workers as agents to participate in and complete tasks [20].

Therefore, at any given time $t(t_q)$, the platform ($Plat$) contains a set of $m$ projects ($P$), a set of $n$ active tasks ($T$), and a set of $r$ crowd workers ($A_r$). Note that one project might corresponds to a set of tasks, which is a subset of ($T$). Platform is defined as follows:

$$Plat = \{t_q, \{P_k\}, \{A_r\}, \{T_i\}\}$$
**Task**: Tasks are typically time-dependent and complex, with a high-level of interdependencies. In this research, a task \((T_i)\) contains a tuple of different characteristics including the task’s identification \((ID_i)\), arrival time \((A_{t_i})\), duration \((D_i)\), associated award \((AW_i)\), requirements \((req_i)\), type \((Tech_i)\), required technology \((Tech_i)\), as well as the the task’s status \((TS_i)\) (as a completed \((i)\) or failed task \((i))\). The formula to create a crowdsourced task \((T_i)\) in a CSD platform is as follows:

\[
T_i = (ID_i, D_i, At_i, AW_i, req_i, Tech_i, TD_i, TS_i)
\]

Where
\[
\begin{align*}
   k &= 1, 2, 3, \ldots, m \\
   z &= 1, 2, 3, \ldots, r \\
   i &= 1, 2, 3, \ldots, n \\
   q &= 1, 2, 3, \ldots, l
\end{align*}
\]

**Project**: The sequence of time dependent tasks that needs to be completed makes a project \((P_k)\). In this study, the time sequence of an arriving task is considered as a measurement of the series of parallel or sequential tasks that followed finish-to-start task dependency techniques. Task \((n)\), \(T_n\), is parallel with task \((n-1)\), \(T_{n-1}\), if it starts before \(T_{n-1}\). And \(T_n\) is a sequential task for \(T_{n-1}\) if it starts after \(T_{n}\) is completed. The project duration \((PD_K)\) is defined as the total duration of sequential tasks per project. Also, the ratio of failed-to-successful tasks per project is a measure of project success. Therefore, in this research, a project is defined as a tuple of different characteristics of decomposed tasks \({T_i}\), project duration \((PD_K)\), project failure \((PF_K)\), and the number of crowd workers assigned to the task \((A_r)\) using the following formula:

\[
P_k = \{\{T_i\}, PD_k, PF_k, \{A_r\}\}
\]

Where
\[
PD_k = \sum_{i=0}^{n} S_i
\]

And
\[
PF_k = \frac{\sum_{i=0}^{n} F_i}{\sum_{i=0}^{n} T_i}
\]

**Crowd Workers**: According to Howe [3], crowd workers \((AZ)\) are a large and undefined group of skilled workers who have access to a task via the internet. In this research, workers are a tuple of different characteristics of workers’ identification \((AID_r)\), reliability factor \((Re_r)\), rating \((Ra_r)\), skillset \((SK_r)\), score \((So_i)\), number of wins \((Wi_i)\), location \((L_r)\), and his/her membership tenure \((MA_r)\) at a given time \(t_i\). A crowd worker is defined with the following formula:

\[
A_r = (AID_r, Re_r, Ra_r, SK_r, So_i, Wi_i, L_r, MA_r)
\]

where \(Z= 1, 2, 3, \ldots, r\)

**B. Overview of the Hybrid Simulation Model**

Figure 2 illustrates the overview of the hybrid simulation model. In the hybrid model, the agent-based simulation (ABS) model is responsible for an individual agent’s behavior, while the discrete event simulation (DES) manages tasks sequentially, and the system dynamic simulation (SDS) displays the interactions among the system’s parameters and feedback within the platform. Once a task arrives in the platform, the agents’ decision-making process starts via the ABS model. An agent’s different decision-making strategy defines his or her performance in the platform. Agents’ performance directly impacts task failure rate and the market’s trust in the platform.

1) **Macro-level Competition Model**: At the top platform level, the SDS model, is applied to represent the overall relationship among task arrival, worker reliability, and task competition outcome by revealing a pattern of broader market behavior [27] [28]. As illustrated in macro level of Figure 2, it contains 14 variables including task, agent decision, workers’ performance, task similarity, worker profile, worker skillset, and different available crowdsourced markets. The SDS model represents the causal loops among different levels of the platform.

“Tasks arrival” in the platform is an event modeled by a Poisson distribution. Any new arrival task will trigger its execution process modelled through a DES, and joins the platform’s pool of open tasks. More details on the DES model is provided in the following subsection. Each new task is associated with a similarity factor upon arrival. According to empirical analysis, a higher degree of “task similarity” among the pool of open tasks in the platform leads to higher competition level. An average task similarity of 90% led to an average competition level of 55%. However, the degree of task similarity does not impact the level of submissions; failures increased when the task similarity rose in the pool of open tasks. For example, an average task similarity of greater than 80% provided a failure ratio of 20%, while an average task similarity of 60% provided an average failure ratio of 13%.

As Figure 2 illustrates, having other available CSD platforms in the market impacts “workers’ arrival” and a company’s decision to work with a specific platform. Also, “agents’ skillset” and task similarity are the main factors impacting agents’ decision-making processes to post a specific task on the platform. An agent’s decision to “register” and “submit” a task provides information for “worker’s performance”, and consequently updates the “worker’s profile”.

Workers’ performances on the platform lead to different task statuses of success or failure. If the task is successfully completed, it counts as “complete” and is reported to the “requestor company”. In this model, task completion ratio \((TCR_k)\) is calculated based on the ratio of passing review tasks to the registered tasks.

\[
TCR_k = \frac{\sum_{i=0}^{n} C_i}{\sum_{i=0}^{n} R_i}
\]
However, if the task is not qualified, it will count as a "failure". Task failure ratio \( TFR_k \) in this model is the ratio of tasks that are registered but are dropped or do not pass peer review. Failure ratio in this model calculates as follow.

\[
TFR_k = 1 - \frac{\sum_{i=0}^{n} C_i}{\sum_{i=0}^{n} R_i}
\]

Table II summarizes the structured attributes used in the SD model.

2) Task Completion: The task’s lifecycle is one of the most important factors in an agent’s decision-making process in crowdsourcing, since it represents a task’s priority and complexity. In this model, tasks are defined as a set of discrete events (DE) that have a start and end date. An arrival task \( T_j \) in this model is defined as a list of \( w \) number of tasks from the same project that arrive simultaneously in the platform.

\[
T_j = \{ T_i \}
\]

where \( D_j < D_j < D_{j+1} \) 
\[ j = 1, 2, 3, \ldots, w \]

Task arrival in this model, is an event modeled by the Poisson distribution of a population of tasks with a lambda equal to 87. The Messo level in Figure 2 illustrates the overview of the DSE model. When a task arrives, it becomes an “arrived” task in the platform, meaning that it can be “registered” by available agents to start the process. Empirical analysis revealed that arrival tasks will attract software workers at a rate of 70%. If the task does not attract any software workers and faces zero registration, it will be marked as “starved”.

Based on different decision-making scenarios by competing agents, the proposed model provides two different failure prediction ratios in different phases of the task lifecycle. TopCoder currently applies a heuristic-based color-coded (red, yellow, and green) predictor for a task outcome in the registration phase. The model involves three simple rules with respect to the sum of the reliability ratings of all software workers registered for a task [29]. If the sum of reliability ratings is greater than or equal to two, the predictor produces a green label to symbolize the task’s success to the requester. If it is less than one, the predictor produces a red color which represents a likely failed task. Yellow is the result of a reliability rating in-between one and two, which represents an uncertain task status. In this model, the heuristic model used by TopCoder is extended to predict the first failure ratio in the registration phase, \( FPR_k \), as illustrated below.

\[
FPR_k = \begin{cases} 
\frac{\sum_{i=0}^{n} Re_j \ast P_i}{\sum_{i=0}^{n} Re_i \ast P_i} & \text{if } \sum_{i=0}^{n} Re_j > 2 \\
\frac{\sum_{i=0}^{n} Re_j \ast P_i}{1} & \text{if } 1 < \sum_{i=0}^{n} Re_j < 2 \\
\frac{\sum_{i=0}^{n} Re_j \ast P_i}{\sum_{i=0}^{n} Re_j} & \text{if } \sum_{i=0}^{n} Re_j < 1 
\end{cases}
\]

In this equation, \( P_j \) is the probability of zero qualified submissions by the registered agents as defined in Table I. By the end of the task’s duration, new registered tasks may be "submitted" by agents. If registered tasks face zero submissions, they were “dropped”. In this model, as soon as the first submission happens, the task submissions ratio \( TSR_k \) is dynamically calculated based on the ratio of submitted to registered tasks in the platform \( TSR_k \), as displayed below.

\[
TSR_k = 1 - \frac{\sum_{i=0}^{n} S_i}{\sum_{i=0}^{n} R_i}
\]

Statistical correlation analysis on different drivers in a crowdsourced task indicated that task failure and task submissions ratio in the platform are directly related. Therefore, the second task failure prediction ratio \( FPS_k \) can be analyzed.
based on a linear regression model of dynamic task submission ratio.

\[ FPS_k = 0.0473(TSR_k) + 0.014 \]

The average CSD task duration is 16 days [5]. According to this study’s empirical analysis, task duration followed a triangular distribution with a maximum of 30 days, a minimum of one day, and a mode of 16 days.

Submitted tasks must pass the peer review phase to ensure the quality of submissions. In this phase, if the quality of task is greater than or equal to 75, it is complete; if not, it has failed. If the submitted task is qualified, it is recorded as “completed”; otherwise, the task is recorded as a “failure”. Moreover, all the starved and dropped tasks are reported as failures.

Table III summarizes the structured attributes used in the DES model.

### TABLE II
**Variable Used in Systems Dynamic Model**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
<th>Model Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Similarity</td>
<td>Tuple of monetary prize, task duration, task complexity, required technology and task type. Range [0,1]</td>
<td>tTSim(double)</td>
</tr>
<tr>
<td>Task Duration</td>
<td>Total available time from task registration start date to submission deadline. Range:(0, ∞)</td>
<td>duration(time)</td>
</tr>
<tr>
<td>Failure Ratio</td>
<td>Ratio of cancelled tasks per total tasks per platform. Range [0,1]</td>
<td>tFR(double)</td>
</tr>
<tr>
<td>Task Arrival</td>
<td>Task uploaded to the platform to be worked on. Range: (0, ∞)</td>
<td>tTAr(int)</td>
</tr>
<tr>
<td>Task Failure</td>
<td>Task that does not receiving any qualified submissions.</td>
<td>tTF(int)</td>
</tr>
<tr>
<td>Task Submissions Rate</td>
<td>Percentage of submitted tasks to the total number of registered tasks in any given time. Range [0,1]</td>
<td>tTSR(int)</td>
</tr>
<tr>
<td>Task Completion</td>
<td>Percentage of number of tasks that can pass the peer review to total number of registered tasks in any given time. Range [0,1]</td>
<td>tTCR(double)</td>
</tr>
<tr>
<td>Task Drop Ratio</td>
<td>Percentage of number of tasks that is not submitted while is registered to total number of registered tasks in any given time. Range [0,1]</td>
<td>tTDR(double)</td>
</tr>
<tr>
<td>Agent Decision</td>
<td>Agent decides to register for a task and/or make a submission for the same task.</td>
<td>tAD(Register, Submit)</td>
</tr>
<tr>
<td>Agents Performance</td>
<td>Number of qualified submissions that an agent makes Range: (0, i)</td>
<td>tAP(double)</td>
</tr>
<tr>
<td>Agent Skill Set</td>
<td>Workers’ list of skillsets in their profile.</td>
<td>tASS(Java, C)</td>
</tr>
</tbody>
</table>

### TABLE III
**Variable Used in Discrete Event Simulation Model**

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
<th>Model Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registered (R)</td>
<td>Number of registrants that are willing to compete on total number of tasks during a specific period of time. Range: (0, r]</td>
<td>register(int)</td>
</tr>
<tr>
<td>Task Duration</td>
<td>Total available time from task registration start date to submission deadline. Range:(0, ∞)</td>
<td>duration(time)</td>
</tr>
<tr>
<td>Submitted (S)</td>
<td>Number of submissions that a task receives by its submission deadline in specific period of time. Range: (0, # registrants)</td>
<td>tsubmit(int)</td>
</tr>
<tr>
<td>Peer Reviewed</td>
<td>Process of reviewing a submitted task to check the quality of submissions. Range: (0, # registrants)</td>
<td>tpeer(int)</td>
</tr>
<tr>
<td>Completed (C)</td>
<td>Qualified task that has successfully passed the peer review.</td>
<td>tcomp(int)</td>
</tr>
<tr>
<td>Re-worked ([RW])</td>
<td>Qualified task that needs some adjustment in order to pass the peer review.</td>
<td>trework(int)</td>
</tr>
<tr>
<td>Failed (F)</td>
<td>Non-qualified task that has not passed peer review.</td>
<td>tfail(int)</td>
</tr>
<tr>
<td>State</td>
<td>Each task has a state that declares the task situation during the submission process.</td>
<td>state (Arrived, Registered, Submitted, Reviewed, Complete)</td>
</tr>
</tbody>
</table>

### 3) Agent Model:
Crowdsourced projects integrate online and unknown workers’ elements into the design. It is reported that crowd workers often overestimate their productivity [30], and they register for more tasks than they can complete. Therefore, simulating crowd-workers with various characteristics, decision-making processes, and performance ratios is difficult. Applying an agent-based (ABS) method to simulate crowd-workers’ behavior individually provides the option of observing a diversity of attributes. Crowd-workers are represented as agents who have one or more of the following characteristics:

1) Identifiable with a set of rules that direct their behavior; an autonomous agent that can act independently in the environment and have control over their actions.

2) Situated workers that work in the same environment and interacting with each other.

3) Flexible agent that can adapt their behavior to be a better fit to the environment [31].

The agents’ arrival to the platform follows a non-homogenous Poisson distribution [8]. Agents are assigned unique IDs upon their entrance into the simulation. As is displayed in Figure 3, the agent-based model contains: i) the agent environment, ii) a set of agents’ attributes, and iii) and the agent decision-making process [18]. In any crowdsourcing platform, an individual
Agent has “agent’s knowledge”, which is based on his or her skillset, background, and the society s/he represents [32] [33]. The agent is also a member of the “pool of agents”. This membership allows the agent to interact and communicate with other agents within the pool. TopCoder divides the “pool of agents” into five groups.

It then subdivides each of these five agent groups into five belts that correspond to skill level in the following descending order: red (highest), yellow, blue, green, and gray (lowest) [9], as it is summarized in Table I. Agents update their “personal knowledge” based on the other agents’ performances in the platform. The combination of interaction and communication with other agents in the pool and an agent’s personal knowledge creates an agent’s “strategy” in the system.

To address the rest of the agents’ characteristics, agents’ strategy provides a utility factor [4] that defines their individual behaviors. Therefore, each agent has an internal “decision-making” process consisting of two components: “registering” for a task and “submitting” the task. An agent’s decision-making process is related to the information that the agent receives from the agent’s community and social environment, as well as other agent’s decision to compete on the task [25].

Once a task is generated in the system, agents decide to register for the task. An agent’s registration for a task categorizes him or her as an active agent. Agent registration in this model follows an event at a rate of one registration per day per agent and a random number greater than 0.8. Agents perform and eventually submit a task based on a variety of factors, including the number of open tasks in their task list, a task’s complexity and competition level, as well as the probability of winning over the competition. It seems that agents analyze their probability of winning based on the task’s competition level and the number of more highly ranked opponents [8].

It is reported that some agents use their history of victories as a policy of assuring to win the registered tasks [8]. As the result, an agent registers for a greater number of task than s/he can perform and eventually drops some of the tasks. An agent’s final decision to submit a task and his or her level of submissions impacts the agent’s profile. The agent may or may not decide to submit the registered task. A task’s submission decision is calculated at a rate of 0.51 per day per agent when the random number associated with the agent is greater than 0.86. When an agent submits a new task, an agent’s reliability factor will be updated.

According to previous research, 59% of agents in the active pool respond to the task call in the first day, while only 24% of the agents who initially responded to tasks call go on make a submission [25]. A total of 76% of the agents exceeded the submissions deadline [25]. 48% of registrants are among average-rated agents (i.e. those with a green or blue belt), while 86% of higher-rated agents (i.e. those with a yellow or red belt) submit qualified submissions. The personal knowledge in the proposed ABS model reflects the reported empirical analysis. Also, according to empirical data, reviewed tasks with scores greater than 75% are considered qualified and the submitting agent is reported as the winner or runner-up. Therefore, qualified submissions are determined by assigning a random number greater than 0.75 as quality score. When a submission passes peer review, an agent’s attribute rate is updated, and the new score is reported in the agent’s profile. Decisions from simulated agents determine task progress. The details of the model integration are explained in part III-4-b algorithm 1. Table IV summarizes the variables used in the agent-based model.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Definition</th>
<th>Model Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registering</td>
<td>Number of tasks a worker registered for in specific period of time. Range: (0, i)</td>
<td>a_registering (ArrayList&lt;Task&gt;)</td>
</tr>
<tr>
<td>Submitting</td>
<td>Number of tasks a worker submits in specific period of time. Range: (0, #registrants)</td>
<td>a_submitting (ArrayList&lt;Task&gt;)</td>
</tr>
<tr>
<td>Quality</td>
<td>The quality of submitted task based on associated score. Range: (0, 1)</td>
<td>tQ(int)</td>
</tr>
<tr>
<td>Reliability</td>
<td>The percentage of successful task submissions in a worker’s most recent 15 task registrations. Range: (0, 1)</td>
<td>tRe(int)</td>
</tr>
<tr>
<td>Trust-ability</td>
<td>Percentage of number of qualified submissions among total number of submissions that a worker makes</td>
<td>tT-A(int)</td>
</tr>
</tbody>
</table>
IV. IMPLEMENTATION OF THE PROPOSED HYBRID SIMULATION MODEL

In many cases, a simulation is a tool for decision-making that can help reduce risk on a tactical or operation level [19], while simulating task scheduling helps on the operational level. In the presented model, the proposed model in [26] is extended, and a hybrid simulation model is created based on SDS, DES, and ABS using Anylogic\(^1\). Driven by the resource-related challenges in software development, there are three steps in the hybrid model: the micro-, meso-, and macro-level.

a) Macro-level: Platform Competition

Tasks and agents created by daily events enter the existing pool of open tasks and available agents in the platform, also known as the dynamic crowdsourced marketplace. Task execution follows the project schedule provided by DES. Each task is associated with a similarity rate created by an event that follows a uniform distribution between 30% and 98% based on our previous empirical analysis [34]. In this model, agent arrivals follow the Poisson distribution. Arriving tasks in the platform impact an agent’s decision-making process and consequently, the agent’s experience. An agent’s experience in the model follows a beta distribution with a minimum of zero and a maximum of 3000 where \( \alpha = 1 \) and \( \beta = 5 \). The result of the SDS model provides the platform failure ratio.

b) Meso-level: Task Completion

Task arrival in the DES model is sequential and follows a specific project schedule based on task requirements. Each task must pass three states to be successfully completed upon arrival.

As Figure 4 illustrates, as soon as the task is executed, it is added to the state of arrived tasks. Once the first agent registers for the task, a message (1) is released as a trigger and the task’s state becomes “registered”. When a task is in this state, the model provides a failure prediction ratio based on the agents’ reliability and experience levels. A message (2) released to report the registration phase failure.

The registered task is associated with the duration following the triangular distribution reported in Part III-B-2. The registered task moves to a submitted state as soon as one of the registered agents makes a submission and releases a submission message. A submission message (3) worked as the second trigger and changed the task’s status to “submitted”. Once the task is moved to the submitted state, the failure prediction model switches to the submissions phase and continues the prediction based on the task submissions ratio in a time interval. In this moment, a new message (4) was sent out to report the submission phase failure.

When the submissions deadline is over, the task moves to the peer review phase. Another message (5) is released to change the task’s status to “peer review”. If the quality score is greater or equal to 75, the task is reported as completed; otherwise, if the score is less than 75, the task is marked as failed and will be re-posted. The completed task is reported to the requester company and its sequential task will post in the system to be performed.

c) Micro-level: Agent Model

Agent arrival in the AB model is an event that follows a Poisson distribution with \( \gamma = 800 \). Once the model runs, agents start arriving in the model. When an agent decides to register for a task, the agent’s personal profile gets updated with new registered tasks.

In this model, an agent’s registration decision follows an event with a rate of one registration per day. Also, none of the agents can work on more than five open tasks simultaneously. If the agent meets the above criteria and s/he has the required skillset to perform the task, s/he can decide to register for the task. In the registration decision phase, all agents that meet the above criteria are assigned to a random number in range of (0,1). If an agent’s random number is greater or equal to 0.8, s/he will register for the task. As soon as the agent registers for the task, the task’s status changes to registered. Moreover, according to the empirical findings, the average competition level per task is 18 [5]. Therefore, in this model, the probability of an agent registering for a task with more than 18 assigned registrants follows Bernoulli distribution (\( P = 0.3 \)), and the probability of an agent competing on a task with assigned registrants is less than or equal 18 is 1.

If an agent registers for a task, s/he can decide to make a submission for the same task. Agents’ submissions decisions for registered tasks follow an event with a rate of 0.51 submissions per day.

For deciding to make a submission, each agent is assigned to a random number in the range of (0,1). If the product of the random number and the agent’s registration probability, based on the agent experience level, is less than 0.051, the agent can make a submission for the task. Once the agent

\(^{1}\)https://www.anylogic.com/
makes a submission, the task’s state changes from registered to submitted.

After sending the submission message, the number of submissions updates in the agent’s profile and his or her reliability and rating factor is updated. The reliability of agents who make a submission follows a distribution with an average of 10%. In this step, if the random score assigned to the submitted task is greater than 75, the agent will be reported as the winner. Algorithm 1 presents the associated pseudo code with the AB model in the simulation.

V. EVALUATION

To investigate the impact of task openness and agent diversity, the following research questions were formulated and studied in this paper:

RQ1(Tasks’ Openness): How much openness in the pool of open tasks can minimize the task failure ratio?

The reliability of qualified task submissions is measured based on the degree of task similarity in the pool of open tasks.

RQ2(Agent Diversity): How diverse should attracted agents be per task to receive a higher chance of qualified submissions?

The consistency of agents’ availability in response to a task and the level of qualified performance is measured based on an agent’s experience level in the competition. The resulting hybrid simulation model is illustrated in Figure 2. This model addresses decision-making in terms of planning and understanding purposes [19].

A. Performance of Failure Prediction Simulation in the Platform

1) Platform Evaluation Design: The success of a CSD platform is based on the level of task failure in the platform. Since task failure is the result of agents’ performance in the platform, it is very important to understand agents’ utilization in the CSD platform. To address that end, the simulation was updated with the associated schedule of the example project and ran 30 times under the basic setting of simulation design. Each run lasted 60 days. During the simulation time, if a task failed due to a lack of qualified submissions or receiving zero submissions, it was reposted. The last step was to understand agents’ performance in the competition. The resulting hybrid simulation model is illustrated in Figure 2. This model addresses decision-making in terms of planning and understanding purposes [19].

Algorithm 1: Agent Decision Making Procedure

```plaintext
1 return (Agent Registering Status, Agent Submitting Status, Agent Winning Status)

input: An agent from the pool of crowd agents, A
result: Agent decision making process for: registering for a task, making a submission and passing peer review.

Agent registering for a task
Create Arrival Event (1, day)
for Task do
if ((Agent is Registering), (Agent Open List < 5), (Agent Expertise = Task Requirements), (Agent Rating > 0)) then
if (# Registrants for Task < 18) then
if (uniform distribution > 0.80) then
send {"Agent Registered"}
Task added to Agent Open List
Agent added to Task Registrants List
Task Rating = Agent Rating;
else
if (Bernoulli(0.3) = 1) then
send {"Agent Registered"}
Task added to Agent Open List
Agent added to Task Registrants List
Task Rating = Agent Rating

Agent submitting for a task
Create Arrival Event (0.51, day)
for Task in Agent Open List do
if ((Agent is Submitting), (Agent Open List > 1)) then
Generate a Random Number
P(qualified Sub) Based on Table I
if (P(qualified Sub) * Random) < 0.051 then
Send {"Agent Submitted"}
Agent Score Reported
Task added to Agent Submitted List
Agent added to Task Submitter List
for Task in Agent Submitting List do
if (Agent Score > Task Score) then
Task Score = Agent Score
Agent reported as Task Winner
Task added to Agent Winner List
Send {"Agent Won"}
```

The t-test was also applied to the prediction results in each state to confirm the models’ accuracy.
2) **Platform Performance**: As it is presented in Figure 5-a, the average success ratio in the platform while running the project was 71% on average. The average none-qualified submissions ratio was 19% and the average zero submission ratio was 7%. Therefore, the average task failure ratio was 13%. Also, the mean relative error (MRE) of the failure prediction models was only 1.1% in the registration phase and 2% in the submissions phase. The results of the t-test on 60 observations per failure prediction model revealed that the probability of error in failure prediction in the registration phase was almost 0 and the error in the submissions phase was almost 1% with zero hypothesized mean difference in both states. This result shows that the simulation is performing accurately. The Pearson correlation provides a incredibly close correlation between the actual failure for the project based on the data set and failure simulation results in the simulation. According to this test, the correlation in the registration phase was 0.21 and in the submission phase it was 0.42 with p-value less than 0.001.

Additionally, it was reported that the agents’ utilization of the platform impacted the platform success ratio. After a deeper examination of the agents’ utilization of the platform at the time of sample project, as displayed in Figure 5-b, it was concluded that the average agents’ utilization factor was 43% with a standard deviation of 0.15. This information can be used to create a control chart to study the agents’ utilization of the platform. Figure 8 displays the control chart of agent utilization in the platform. Unlike an industrial control chart where the goal is to keep the quality as close as possible to the average, in CSD, the goal is to have utilization as high as possible and closer to the upper level line. Therefore, a closer examination occurred for the days that had a utilization factor near to or exceeding the upper level.

Analyzing days with a higher level of agents’ utilization factor revealed that, on days with the highest utility factor, the average similarity of the open pool of task was less than 60%. Interestingly enough, in those days, the availability of agents in the middle-ranking clusters (i.e. green and blue belts) were higher and relatively close to each other, as displayed in Figure 5-c.

These results encouraged the team to design and test two different scenarios to answer the research questions that will be discussed in the following section.

**B. Evaluation Study**

The aim of the proposed hybrid simulation model (HSM) is to understand the optimal level of openness in the pool of available tasks and diversity in the competition level in order to minimize the task failure ratio in any CSD platform. In order to achieve the research goal, two different scenarios were proposed:

*Scenario 1 (Task Openness)*: To successfully crowdsource a software project in a crowdsourcing platform, not only is it important to fully understand the project’s task dependencies, but also, it is vital to know the impact of available open tasks on each other within the platform. Different task similarity levels may represent task difficulty, task size, and task priority [5]. Also, the degree of task similarity represents task openness in the pool of open tasks in the platform. Therefore, understanding workers’ behavior and performance based on the degree of task similarity in the pool of open tasks is very helpful to present more effective task planning.

Different researchers discuss that shorter task duration is associated with less complex work. This relationship may impact a worker’s decision-making processes in terms of taking tasks because he or she may want to achieve a higher number of tasks. Also, it is reported that there is a general negative correlation between a monetary prize as one of the elements of task similarity and worker behavior [5]; workers are more attracted to tasks with a certain range of associated monetary prizes (i.e. $750) [5]. Therefore, task similarity can be an important factor in task failure rate. Empirical studies demonstrate that, even though a higher similarity level among tasks leads to higher registration, it also results in lower submissions and a higher failure ratio. Scenario 1 aims to investigate the impact of task similarity level on agents’ performance in the CSD platform. In this scenario, four different testing policies based on previous empirical findings are tested. Each testing policy provides an arriving task with a limited similarity rate into the platform. The results of the four different testing policies were compared to each other.

*Scenario 2 (Agents’ Diversity)*: According to TopCoder, higher rating levels among agents represents a higher level of
Fig. 6. Scenario 1, Impact of Task Similarity on Task Failure

experience. Also, it is reported that a higher reliability factor among agents presents the highest chance of them submitting qualified tasks [29] [35]. In general, it is expected that more highly rated agents are associated with a higher reliability factor due to their experience. However, the reliability factor measure is based on the number of agents’ submissions and it is not related to a submission’s quality. It is reported that more highly experienced agents may take advantage of their reputation and apply cheap talk to the task [7]. Scenario 2 aims to investigate the impact of diversity among agents in terms of agents’ experience level on agents’ performance.

In this scenario, four different testing policies were reviewed based on different agents’ experience levels as measured by TopCoder2. Each testing policy attracted agents with a specific experience level to register for the chosen task, and agents’ behavior and performance level were reported. The results of the different testing policies were compared to each other. To apply evaluation scenarios, the motivation example was simulated. In the motivation example, task 8 was re-posted seven times until it was completed. This study aims to analyze this task under both scenarios.

1) Scenario 1, Task Openness: New arrival tasks in the platform joined the pool of open tasks and waited in the queue to be registered by agents. It is reported that agents are often more attracted to tasks with similar monetary prizes [5], task complexities, and context to each other [4] [17]. This fact creates a demand market with an associated similarity factor to each task. Workers use this factor as one of the inputs to their decision-making process. Therefore, it is expected that a higher number of available similar tasks directly impacts the level of competition per task. The impact of this relationship was examined in this scenario. According to empirical analysis, new arrival tasks compete with tasks with average similarity of 69% per week. In Scenario 1, it was assumed that the project manager can control the openness of the pool of open tasks in terms of task similarity at the time of their task posting. The simulation ran 30 times and the average of all the runs was used in this scenario. Four different testing policies based on degree of task similarity are discussed. The results of this study report the task failure prediction ratio in different states of a task. This scenario ran 120 times, more specifically 30 times under four different testing scenarios. Figure 6 illustrates the failure prediction per testing policies and Table V summarizes agents’ participation per testing policy in Scenario 1.

<table>
<thead>
<tr>
<th>TABLE V</th>
<th>SUMMARY OF SCENARIO 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>TP1</td>
</tr>
<tr>
<td>Task Status</td>
<td>Fail</td>
</tr>
<tr>
<td>Success</td>
<td>12</td>
</tr>
<tr>
<td>Failure Prediction</td>
<td>60%</td>
</tr>
</tbody>
</table>

Testing Policy 1: 60% Degree of Task Similarity
The project manager posted a task in the pool with degree of similarity equal to 60%. The result of this policy led to, on average, only 1% failure prediction at the early registration stage; by the 26th time that the first submission happened, failure prediction increased up to 35%. This fact demonstrates that, by passing time, the chance of receiving highly reliable agents to compete on a task is low; however, the average failure ratio in this state is 8%. At time 26, failure prediction shifted to the submission state and increased to 45%. Unfortunately, the task only received four submissions out of on average 48
registrations, which increased the failure prediction to 55% on the task submission deadline. The results of the simulation indicated that the task failed for 18 out of 30 runs, which provides an actual failure ratio of 60%.

Also, applying this policy attracted a wide diversity of agents. The results revealed that competing for tasks in a pool of other tasks with 60% similarity attracted on average 44% of all gray belt agents, 22% of all green belt agents, 21% or all blue belt agents, 11% of all yellow belt agents, and almost 6% of all red belt agents.

**Testing Policy 2: 70% Degree of Task Similarity**

Under testing policy 2, the project manager kept the similarity level of the pool of open tasks at 70%. At the first registration, the failure predictor predicted a 9% task failure. With increased time passed and higher competition levels, the task failure prediction increased to 33% in the registration phase. The average failure prediction in the registration state was 14%. At time 26, the first submission was made, and the failure prediction increased to 75%. Finally, at time 30, the task received its third and final submission with a failure prediction of 67%. This task attracted 36 registrants and received only three submissions. The simulation under this policy resulted in the task failing for 22 out of 30 runs. This result indicates a 73% task failure rate. Moreover, this task attracted competitors from the gray, green, blue and red ranking belts with a participation ratio of 37.5%, 25%, 25% and 13% respectively. Among all the participants, only gray and green agents made submissions. It seems that, under this policy, the task cannot attract high-reliability agents to make a submission; however, it successfully attracted more highly ranked agents to compete.

**Testing Policy 3: 80% Degree of Task Similarity**

Under this policy, the project manager created a pool of tasks with an 80% task similarity level. However, at time one, the task provided a competition environment with only 1% failure prediction; at time two, failure prediction rose to 22%. This policy attracted slow-arriving agents to the competition. Only six agents arrived in the first 26 days before the task received its first submissions. At time 12, the failure ratio increased to 36% from 26%, and during time 16, it rose to 41%. By the 18th run, the rate was at 47%. At time 26, the first submission occurred and pushed the failure prediction metrics to the submissions state. This task only received on average one submission for an average of 23 registrations. This low submission number increased the failure prediction to 63% at time 26, and finally, to a failure prediction rate of 78% on the 30th and final run. Also, the task failed for 25 out of 30 runs, which provides an 83% failure rate. Interestingly applying this policy resulted in a lower level of diversity among participants in terms of ranking belt. During this test, the task attracted on average, 50% of gray belt agents, 45% of green belt agents, and only 5% of blue agents. Also, tasks were only submitted by gray and green belt agents.

**Testing Policy 4: 90% Degree of Task Similarity**

When the project manager followed policy four, the task faced the challenge of very slow agent attraction. The failure prediction rate started at almost 0% and increased to 52% in day 24. The task faced two jumps in attracting agents. The first jump occurred on day 4, when the failure prediction rate rose to 25% from 14%. The second jump happened on day 16, when the failure prediction reached 43% due to attracting low reliable workers. This task received on average two submissions out of 32 registrations. The first submission happened on day 25 and the second one occurred on day 28. The task failure prediction in the submission state increased to an average of 78% on day 30. The simulation resulted in 23 failed tasks out of 30 runs, representing a 77% task failure rate. While applying a higher level of task similarity did not help with lower failure predictions, the increased task similarity attracted a better diversity of agents in terms of belt rankings. Simulation results indicated that 45% of participants were gray belt agents, 27% green belt agents, 9% blue belt agents, and 18% yellow belt agents. However, only the gray and green belt groups made submissions, each for 50%.

**Discussion and Findings**

It seems that the submissions ratio decreases when the level of task openness in the pool of open tasks decreases. In other words, the submission ratio falls when the degree of task similarity rises. Interestingly, tasks with a similarity level of 70% and 60% attracted more reliable agents in terms of submissions. Moreover, a higher similarity level among tasks resulted in a higher chance of task failure. This fact could be because the group of agents attracted to the task have similar skillsets. This result confirms the fact that agents are more interested in lower levels of switching concepts. Also, a higher level of similarity among available tasks in the pool of open tasks may provide a higher chance of cheap talk [7] per task due to a higher number of available options for active agents. Further investigation revealed that the lowest and highest degrees of similarity (i.e 60% and 90%) led to higher diversity in terms of agents’ rating levels, while middle-level degrees of task similarity attracted less diverse agents with lower level ratings. One reason for this grouping is that lower belt agents have a lower level of skillsets and experience, which makes them more interested in working on similar tasks [4].

Task 8 of the motivation example was tested with the findings of Scenario 1. On April 2nd, the day task 8 was posted in the platform, the average similarity among the pool of open tasks was 75%, as displayed in Figure 7. The average similarity of open tasks on April 1st, April 5th and April 6th was under 70%, while on April 4th it is under 60%. According to findings of Scenario 1, if task 8 was posted with two days delay on April 4th, the failure prediction would have dropped from 73% to 60% and shorten project schedule by 28 days. Interestingly, when the task was successfully completed as task 18, the average task similarity in the pool of open tasks was 54%.
**Finding:** A higher level of openness (60%) in the pool of tasks leads to a lower chance of task failure for the crowdsourced project.

2) **Scenario 2, Agents’ Diversity:** Reliability in receiving a submission for a task by a registered agent is one of the main factors affecting a task’s success. The reliability factor, based on the number of an agent’s submissions, is not related to the quality of the task submission. However, it is expected that higher-rated agents have higher reliability in making a submission.

Generally, agents do attempt to register for a higher number of tasks than they can complete [2], but even so, higher-rated agents may apply the cheap talk strategy to ease the competition level and guarantee their win [7]. In this scenario, the impact of the task’s restricting access availability was analyzed for agents with certain belt ratings to address that end. The simulation ran under four different availability policies and each policy ran 30 times. Figure 8 presents the failure prediction and submissions ratios under the different testing policies, and Table VI summarizes agents’ participation in the different testing policies in Scenario 2.

**Testing Policy 1: Red and Yellow**
Under this policy, the project manager gave agents with a belt rating of more than 1,500 points access to compete on the task. The first registration happened at time seven and had a failure prediction ratio around 1%; the ratio rose to 11% at time 14. At time 15 the first submission occurred, and the task failure prediction ratio was 18%. The task submissions deadline continued until time 27, and the failure prediction increased to 83%. On average, four agents were interested in this task and only one agent made a submission. Under this policy, the chance of receiving an unqualified submission was 80%. Yellow belt agents made up 82% of participants in this task while 18% were red belt agents. Submission levels for yellow and red belt agents were 80% and 20% respectively.

**Testing Policy 2: Red, Yellow and Blue**
Under this policy, all agents with a rating above 1200 points were permitted to compete on the task. During time four, the first agent arrived, and the task failure prediction ratio was 5%. It seems that the second arrived agent was more reliable than the first one as the failure prediction ratio reduced to 4%. The first submissions occurred during time five and reduced the task failure prediction to 3%. While registration still continued, after this point, only one more agent made a submission, which increased the failure prediction up to 40%. On average, the task received ten registrations and three submissions under this policy, which provides a 60% chance of task failure. Of the attracted agents, 62% were blue belts, 31% were yellow belts, and only 8% were red belt agents. Interestingly, red belt agents did not make any submissions, while yellow belt agents made 33% of submissions and the majority occurred by blue belt agents.

**Testing Policy 3: All but Gray**
In this policy, the project manager opened the task to all available agents with 900 rating points or above. The higher diversity of belt levels provided more competition starting from time three where the failure prediction level was 9%. The first submission attempt occurred on the time four and decreased the failure prediction ratio to 5%. While registration was continuing, the last submission for the task was received at time 12 and increased the failure prediction up to 40%. Under this policy, the task attracted 12 registrants and a maximum of four submissions. There was a 73% chance that the task received unqualified submissions. Under this policy, 53% of registrants were green belt agents, 26% were blue belt agents, 16% were yellow belt agents, and only 5% were red belt agents.

<table>
<thead>
<tr>
<th>Table VI Summary of Scenario 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Task Status</strong></td>
</tr>
<tr>
<td>Fail</td>
</tr>
<tr>
<td>Success</td>
</tr>
<tr>
<td>Failure Prediction</td>
</tr>
<tr>
<td>Gray Reg</td>
</tr>
<tr>
<td>sub</td>
</tr>
<tr>
<td>Green Reg</td>
</tr>
<tr>
<td>sub</td>
</tr>
<tr>
<td>Blue Reg</td>
</tr>
<tr>
<td>sub</td>
</tr>
<tr>
<td>Yellow Reg</td>
</tr>
<tr>
<td>sub</td>
</tr>
<tr>
<td>Red Reg</td>
</tr>
<tr>
<td>sub</td>
</tr>
</tbody>
</table>
agents. Against the expectation, more highly ranked agents did not make any submissions. While 50% of submissions was made by green belt agents, blue belt agents submitted the other half.

Testing Policy 4: All Welcome
In this policy, all available agents in the platform were welcome to compete on the posted task. Simulation results revealed that the first agents arrived at time three and had a task failure prediction of 2%. The second agent who arrived decreased the failure prediction to 1%; however, the third agent arrival increased the failure prediction level. The first submission was made on the sixth run. After that submission, no further submissions were made during registration. This fact increased the task failure prediction to 93%. On average, the task received 20 registrations and only one submission. The chance of receiving an unqualified submission under this policy was approximately 86%. This policy welcomed a wide diversity of available agents in the platform as competitors on this task. The competition level contained 44% gray belt agents, 29% green belt agents, 17% blue belt agents, 7% yellow belt agents, and 2% red belt agents. Interestingly, while gray belt agents were the highest level of participants, they only made 18% of the submissions. While green belt agents made 45% of submissions, blue belt agents submitted 36% of the time. Yellow and red belt agents did not submit for the tasks under this policy at all.

Discussion and Finding

It was expected that a higher level of diversity in agents’ rating and experience provided lowered task failures. However, the simulation results reported that a higher level of diversity among agents led to the highest level of failure prediction per task. Interestingly, the lowest level of diversity led to the same result, while attracting mid-level agents provided the lowest level of task failure prediction. The result of policies three and four in Scenario 2, not only provided higher task submission ratio of 30% and 27%, but they also resulted in lower failure prediction rates of 60% and 73% respectively. This result supports our empirical studies’ reports that middle-rated agents are more reliable in making a submission [25].

The findings of Scenario 2 were tested on task 8 from the motivation example. As displayed in Figure 9, the day that task 8 was posted in the platform, the availability of middle level agents (blue and green belts) was less than 5%. The availability of mid-level agents was considerably higher on April 3rd, April 4th, and 6th. Interestingly, according to empirical data on April 3rd and April 4th, the platform hosted the highest level of mid-level agents. It seems that if the task had been posted on either of those days, it might have had a higher chance of success. According to the result of Scenario 2, if the task had been posted on April 6th, the chances of failure would have decreased from 86% to 73% and the project schedule would have been 26 days shorter. Interestingly, based on available data, when the task was successfully completed as task 18, the average of mid-level agents' availability was 7%.

Finding: Attracting mid-level experienced agents (blue and green belts) to the task competition decreases the chance of
The outcome will be a set of requirements and specifications. These requirements are used as the input for the future architecture phase, while the application is decomposed into multiple components [36]. In CSD workflow [5] [37], the requester company divides the project into many small tasks, prepares task descriptions, and distributes tasks through the platform. Each task is tagged with a pre-specified prize as the award to winners and a required schedule deadline to complete. On average, most of the tasks have a life span of 2-4 weeks from the first day of registration.

Crowd software workers browse and register to work on selected tasks, and then submit the work products once completed. After workers submit the final submissions, the files will be evaluated by experts and experienced workers, through a peer review process, to check the code quality and/or document quality [2]. The number of submissions and the associated evaluated scores replicate the level of success in task success. In TopCoder, usually the award goes to the top 1 or 2 winners with the highest scores. If there are zero submissions, or no qualified submissions, the task will be treated as starved or cancelled.

B. Simulation Methods

Simulation techniques have been demonstrated to be a powerful method in modeling and understanding NP-hard problems in software engineering processes, such as system dynamics as well as discrete event and agent-based simulation models [21].

Discrete Event Simulation (DES) Methods are able to create a step by step simulation of the flow of the entities and their impact in the system. Any event happening will lead to a change in the state of the system. All the entities, attributes, events and relationships among stakeholders should be defined in the model of the system [38]. The main advantage of DES is capturing the actual process level details and representing each work product as a unique individual in the development process [39] [40].

Agent based simulation (ABS) techniques have been employed to simulate individuals and social groups’ behaviors in crowdsourcing, as well as the interaction between requestors and software workers [22] [32] [33]. ABS helps to present more details of development process. While in ABS, the number of stakeholders (agents) is constant. In a systems dynamic (SD) simulation the number of stakeholders will vary by passing time continuously. Such difference makes it possible to predict the number of active software workers and their arrival distribution [41] [42]. In addition, systems dynamic simulation methods provide the possibility of changing one or several factors (attributes) while the remaining ones are unchanged, thus supporting all possible scenarios to be checked in order to make decisions based on managerial policies [39] [43]. Also, it is difficult to identify a complex sequence of activities in the process in SD models [43].

In practice, a mixture simulation model merging systems dynamic and discrete event models has been used to support project planning and improvement of global software development [42]. Also, a hybrid simulation model based on a combination of systems dynamic, discrete event and agent-based simulation techniques was presented for a Kanban scheduling system [41] [44]. Nevertheless, crowdsourced platforms have been simulated via systems dynamic modeling [26].

It is of our interest to study the hybrid simulation approach in CSD, in order to expand and improve understanding of
crowdsourcing decision making. More specifically, impact of three different component in crowdsourcing task scheduling.

C. Task Scheduling in Software Engineering

Task scheduling in software engineering is a sequence of time dependent functions/tasks that make a project [45]. Generally, scheduling is subject to minimizing cost and time [46], while maximizing quality, resource allocation and task priority [47]. Dependencies among tasks and resources makes task scheduling an NP-hard problem. In the traditional software development process, task scheduling is the next step after defining requirement development and deciding on task decomposition [48]. Since more than half of software requirements are interdependent and only a few of them are independent, scheduling is complex [49].

Traditional scheduling techniques are not considering human resource allocation, or if they do, they are not covering resources with various skills. Since humans are the main resource in software projects, scheduling can be more flexible compared to other industries [50]. To overcome this issue, project managers are using global software development techniques to reduce cost and shorten release time. Although, different time zone and resource location may add to the scheduling challenges [51]. CSD is one of the methods that help reducing software development challenges.

D. Task Scheduling in CSD

Batching tasks is an effective scheduling method in CSD to reduce the complexity of tasks and will dramatically reduce cost [52]. Batching crowdsourcing tasks would lead to a faster result than approaches, which keeps workers separate and is also faster than the average of the fastest individual worker [53]. There is a theoretical minimum batch size for every project as one of the principles of product development flow [54] To some extent, the success of software crowdsourcing is associated with reduced batch size in small tasks. Tasks can be batched based on their similarity in skill requirements, associate price, priority and development phase etc.

Another useful scheduling method in CSD is parallelism. This method will create the chance to utilize a greater pool of workers [55], help workers to specialize and complete the task in less time, drive solutions, help the requester to clearly understand how workers decide to compete on a task, and analyze the crowd workers’ performance [8]. Shorter schedule planning can be one of the biggest advantages of using parallelism for managers [7].

Moreover, complex projects cannot be performed based on simple available parallel approaches. Complex projects have more dependencies and multiple changing requirements [53], that requires different workers with different levels of expertise. Therefore, this is one of the main challenges in applying an effective method to schedule decomposed projects [56]. Considering the fact that coordinating workers is difficult among a distributed global crowd, in most cases, organizational coordination techniques can be applied to crowd work as well [57], such as programming and feedback as general coordination methods [58].

E. Available Scheduling Methods in CSD

Delay scheduling [46] is the first presented scheduling method for crowdsourced projects to maximize the probability of a worker receiving tasks from the same batch of tasks they were working on. Extension of this idea introduced a new method called fair sharing schedule [47]. In this method, heterogeneous resources would be shared among all tasks with different demands, which ensures that all tasks would receive the same amount of resources to be fair. For example, this method was used in Hadoop Yarn. Later on, weighted fair sharing (WFS) [59] was presented as a method to schedule batches based on their priority. Tasks with higher priority are introduced first. Another proposed crowd scheduling method is quality of submissions (QOS) [17], a skill-based scheduling method with the purpose of minimizing scheduling while maximizing quality by assigning the task to the most available qualified worker. This method was created by extending standards of Web Service Level Agreement (WSLA) [60]. The third available method is game with a purpose [61], in which a task will not be started unless a certain number of workers registered for it. Similarly, CrowdControl [62], provides a scheduling approach that assigns tasks to workers not only based on their historical performance but also their learning curve of performing tasks. SmartCrowd [63] schedules tasks based on worker skills and their reward requirements to perform the task. Using above scheduling methods, HIT-Bundle based on worker skills and their reward requirements to perform the task. Using above scheduling methods, HIT-Bundle (Human Intelligent Task) [4] provides a batch container which schedules heterogeneous tasks into the platform from different batches. This method makes a more successful outcome by applying different scheduling strategies at the same time. The most recent method is helping crowdsourcing-based service providers to meet completion time SLAs [64]. The system works based on the oldest task waiting time and runs a stimulative evaluation to recommend best scheduling strategy in order to reduce the task failure ratio.

VII. Conclusion

To ensure an effective and successful crowdsourcing platform in such dynamic market, there is a need for improved understanding and visibility into the characteristics and risks associated with CSD processes. While most of the CSD approaches are based on the static aspects of CSD marketplaces, its dynamic aspects have often been ignored. In this paper, we presented a hybrid simulation model to forecast crowdsourcing task failure risk in crowdsourcing platforms. The simulation model is composed of three components: the macro-level systems dynamic simulation reflecting the crowdsourcing platform dynamics, the meso-level of discrete event simulation representing the task life cycle, and the micro-level agent-based simulation illustrating the crowd workers’ decision-making processes. This hybrid simulation model is evaluated through three CSD decision scenarios to demonstrate its effectiveness, using a real-world historical dataset. The
results show that, task openness in the platform in terms of task similarity and agents’ experience are two of the most effective factors that should be considered while planning for a more effective task execution. In details: 1) A pool of open tasks with a higher degree of task similarity in the platform leads to higher level of failure prediction, 2) Attracting agents with mid-level experience to compete on the task helps to achieve a lower level of failure prediction, 3) Agents with mid-level experience provide a higher level of task submissions ratio. The proposed simulation model empowers crowdsourcing managers to explore potential crowdsourcing outcomes with respect to different task scheduling options. In our ongoing and future work, we are considering providing a model to predict agents’ trust-ability besides agents’ reliability.

REFERENCES


