# Dynamic Simulation of Software Workers and Task Completion

Razieh Lotfalian Saremi
School of Systems and Enterprises
Stevens Institute of Technology
Hoboken, NJ 07030, USA
rlotfali@stevens.edu

Ye Yang
School of Systems and Enterprises
Stevens Institute of Technology
Hoboken, NJ 07030, USA
Ye.yang@stevens.edu

***Abstract*—As more and more companies are trying to leverage crowdsourcing to reduce cost and time-to-market of software production, it is very important to better understand the dynamics of crowdsourced software development in order to fully realize its benefits and avoid hidden pitfalls. This paper presents a systems dynamic model for crowdsourced software development platforms based on data gathered from Topcoder. The model is composed of three components: the worker sub-model, the task sub-model, and the interactions between the two sub-models. Results of simulations are compared with empirical data gathered from Topcoder for validation. The major simulation results indicate that task uploading and software workers arrival in the platform follow typical exponential patterns; incentivizing the crowd developers to become more active will lead to more tasks completion and success; and tasks associated with higher awards generally receive few submissions.**

***Index Terms*—Software Worker, Task, Award.**

## I. Introduction

Due to the highly dynamic and complex nature of crowdsourced software development (CSD), it is often very challenging to understand the dynamics of online worker behaviors compared with centralized, in house software projects. As the crowdsourced platforms evolve, the aggregated knowledge needed to support future changes grows exponentially. Therefore, in order to support a successful crowd sourcing platform, it is necessary to design a framework which recognizes and improves the dynamic nature of crowd behaviors [1].

Simulation techniques have been demonstrated to be powerful in modeling and understanding software engineering processes, such as system dynamics and discrete event simulation [2, 3]. While a mixture simulation model of merging systems dynamic and discrete event models has been used to support project planning and improvement of global software development [4], Systems dynamics models have been used in modeling and simulating software processes [5] as well as software investment analysis [6].

Nevertheless, crowdsourced platforms have been simulated via agent based modeling [1, 7, 8, 9], this viewpoint provides the opportunity of analyzing crowdsourced software platforms with a dynamic perspective. However, to the best of our knowledge, there is no existing study using systems dynamic (SD) approach. .

We believe it is beneficial to simulate the platform via SD methodology to predict the patterns of crowd software workers and crowdsourced software tasks. When managers plan for crowdsourcing projects, they need to determine strategies for tasks decomposition and task publishing or uploading to a crowdsourcing platform. It is important to analyze the optimal granularity of task unit and task prices, consequently understanding their impact on the total number of tasks, the award per task, and expected outcomes of task completion. The underlying influential factors may include how many active workers, how many tasks from other requesters are currently open, and the successful rate of overall task completion at the time. For example, it is critical to study through simulation how to price micro tasks in a reasonable way based on the size and complexity of the tasks [7]. Increasing the award would decrease the demand for tasks, because high reward tasks typically mean more complex and difficult tasks which reduce the value of high reward tasks for software workers [8].

Crowdsourcing platforms act as the marketplace between requesters and Software workers. Such platforms need to be designed to ease software workers' understanding of crowdsourcing tasks, as well as form the relationships and practical communication between software workers and requesters [9]. In this paper, we propose a systems dynamic model for crowdsourced software platforms similar to Topcoder [10], which is a largest software development crowdsourcing platform with an online community of over 700000 software workers. The simulation model is created based on task workflows on Topcoder. We also use available empirical data from Topcoder platforms to run the dynamic simulation.

## II. Background

In order to achieve both effective task completion and Software worker pleasure, past research recommends requesters should (A) clearly recognize Software workers motivation, (B) understand Software workers behavior, and (C) design task structures and models appropriately [8]. These

978-1-4673-7040-0/15 $31.00 © 2015 IEEE
DOI 10.1109/CSI-SE.2015.11

factors influence Workers arrival and task completion in the platform.

### A. Software worker Motivations

Based on diversity of software workers in the fields of outsourcing and crowdsourcing, motivational factors for joining the CSD Platforms are typically divided into two categories: intrinsic factors and extrinsic factors. Intrinsic clusters contain enjoyment factors and community values that can get influences from age, location, personal career, society and even task identity. Extrinsic clusters include financial and social point of views which are the direct effect of educational back ground, household income and task award [11, 16]. Another high ranking motivation factor is requesters with brand names, such as Google and NASA, which attracts Software workers to apply for tasks, that potentially can be used in Software worker's resumes and affects software workers ratings indirectly or directly [9,14].

### B. Software worker behaviors

Software workers' arrival in the platform and the pattern of taking tasks to completion are the essential elements to shape the worker supply and demand in crowdsourcing platforms. Many software workers tend to optimize their utility of choosing the task based on different attributes and personal utility [8, 13]. For example, most experienced workers are very interested in competing for famous companies' tasks, or some of them take tasks based on the award. Also for newcomers or beginners, it takes time to improve and turn into an active worker after their first arrival [8, 13]. Therefore, most of them focus on registering and gaining experience by competing with peers, but the chance of them winning the competition is rather low. It is also typical that the workers need to communicate with the requesters in order to better understand the problems to be solved [11, 13]. Another typical issue related to workers is that some workers decide to drop certain tasks after registering for competition or possibly become inactive due to various reasons, such as different time zones and geographical distributions and software workers native language issues [4, 11, and 16].

### C. Simulation models

Agent based modeling techniques have been employed to simulate individuals and social groups' behaviors in crowdsourcing, as well as the interaction between requesters and software workers [1, 2, 11, 16]. While in agent based simulation, the number of stakeholders (agents) are constant, in dynamic simulation the number of stakeholders will vary by passing time. Such difference makes it possible to predict the number of active software workers and their arrival distribution [5, 6]. 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 [4,16]. It is of our interest to study the dynamic approach of CSD, in order to expand better understanding of crowdsourcing decision making. More specifically, growing numbers of active software workers and tasks by passing time and change of market situations will be discussed in this paper for the first time in this domain.

## III. Research Methodology

By simulating crowdsourced software development platforms, it is directed at finding out answers to the following key research questions:

(1) Which distribution pattern is fit for software workers to become active members since their arrival to the platform?
(2) Which distribution patterns are followed by workers behavior on taking tasks and making submissions to tasks?
(3) Is there any specific relationship between Award, Experience point and submission task rates based on the simulation result and empirical data from Topcoder?

### A. Software Crowdsourcing Processes

A typical software crowdsourcing process framework is adapted from [12]. For typical software development challenges, the requesting company would prepare a list of challenges and upload them into the platform. Ideally, this needs to follow a task taken distribution and trend to target maximum possible tasks taken. Crowd software workers would register to work on each task, and submit the task. Registering for competitions that are based on software worker arrival time and also personal interest in the specific task. Submitted tasks will be scored and the two top winners will be awarded the prize. Based on the project phase the associated project manager will communicate with the requester company to identify the project objectives and task plan and also exchange new information. This phase would be managed by platform. Results of different phases are combined in the assembly phase. It is important for the requestor company to have software workers register for competing on tasks by knowing workers arrival and improvement trend and following correct distribution of task uploading. The last phase would be deploying the complete functional solution and send it to the requester company's quality department to check if the task is acceptable or needs rework.

Following this work flow can be effective for requester companies, since it can not only (1) reduce many in-house development costs, it would offer (2) more unique and different ideas with the possibility of optimizing creativity and also prepare (3) higher security for the product. As the newly launched product would be made from combinations of multiple tasks with different developers and coding methods, decoding it would prove difficult. This paper is focusing on the dynamic approach of this chain and tries to analyze the effects of uploading new tasks on crowd source software workers and contrariwise.

### B. Tools and Techniques

To create a systems dynamic model, stakeholders should be defined, and then connection among them will be discussed. Direct stakeholders are: Software worker as workers, Demand Companies as requester and crowd source platform, indirect stakeholders are end users in market and also

research/educational system. This paper is focused on modeling the direct stakeholders. Vensim will be used to create the dynamic model and running the simulation. Vensim is a simulation software toolkit for improving the performance of real systems, which is used for developing and analyzing dynamic feedback models [18]. To run this model, empirical data gathered from Topcoder is going to be used, 26 attributes were needed to come up with the final model, which contains 5 stock and 13 flows.

### *C. Data Set*

The data extracted from the Topcoder platform are the design and development tasks from Sep 2003 to Sep 2012. During this period 2895 software design tasks and 3015 software development tasks were gathered. After cleaning the data by finding outliers data to minimize their influence on model and normalizing the data set through the Z-score standardization method, 1072 components remained which includes 18306 data points, 26 variables and 362 observations. Each component would refer to two tasks of design and development, respectively. Collected data was used to run the Systems Dynamic simulation model in order to see the Crowd software worker behavior and uploaded task trend in the platform.

### *D. SD Models*

To address the platform trend a systems dynamic model was created in Vensim to predict software workers behaviors, task taken and task submitted trend, based on the historical data gathered from the Topcoder platform. The winning score is defined according to Topcoder's policy of at least 75% of peer review of processing each task and score calculation formula defined by Topcoder.

To create an initial model to capture all research goals and appropriate factors of CSD platform, the first step was studying the relations between different attributes and their influence on each other to assign them to different levels of stock, flow or axillary variables. Stock and flow are central concepts of Dynamic simulation. Stock is any attribute that accumulates or depletes over time, and flow is the rate of change in stock, while auxiliary variables are attributes that directly influence stocks and directly or indirectly effect stockholders' decision. Therefore it was very important to do a literature review on case studies and crowdsourced software development platform analysis. All the attributes using different research and also the relation among them were listed. And it was attempted to extend and match attributes to other papers. Research results showed 5 common attributes which are used as Stock and another 21 factors used as flow and auxiliary variables. For each attribute, a brief description is provided, along with the related paper IDs that studied the same subject in the last column. Two lists of attributes used in the simulation model are briefly summarized in tables 1 and 2.

This model is divided into three parts: (A) Dynamics of Software worker behavior in platform, (B) Dynamics of task uploading behavior in platform and (C) the final layout is a mixture of the first two models which shows the interaction and influence among software workers and uploaded tasks and also the effect of experience point and award on the whole system.

#### *1) Dynamics of Software worker behavior*

The variation of software worker makes the shift of tasks and as a result the final product and market competition will occur. In this part it is attempted to simulate crowdsourced software worker entrance to the platform, software worker actively register to complete tasks and connection in the platform across this competitive market.

As illustrated in fig.1 (software worker behavior part), once registered on the crowdsourcing platform, potential software workers may register to compete for new tasks and become active workers, or may change their mind and drop the task and quit the competition at any later time due to various reasons. Gaining experience by active software workers positively

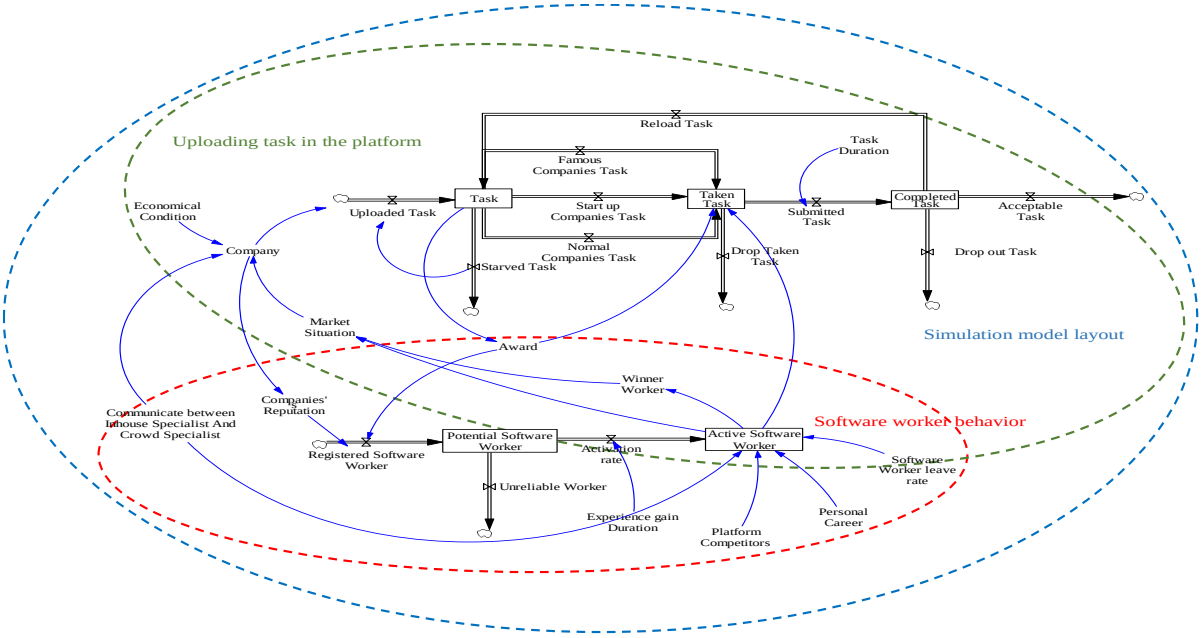


Fig 1: Software worker simulation

influences the number of winner workers as they will be more familiar with the rules. As a result, workers required skills will improve, consequently achieving higher scores in the competition by active workers. In this flow company's reputation directly affects potential Software worker to create an account in the platform, award associated with tasks and also type of tasks make them to decide to continue working with the platform or leave it. Once Software workers decide to continue working with the platform it takes some time to learn all necessary requirements and become an active member.

TABLE 1: ATTRIBUTES USED IN THE SOFTWARE WORKER BEHAVIOR MODEL

| Type | Attribute Name | Description | Reference |
|---|---|---|---|
| Software worker | Registered worker | An individual who registers in the platform | [2,3,6,9] |
| | Active worker | An individual who actively participates in tasks | [2] |
| | Winner Worker | An active worker with winning history | [2,4] |
| | Unreliable worker | An active worker who ever fail to complete tasks | [2,9] |
| Motivation factors | Personal Career | Member's personal career in real life | [9] |
| | Company's Reputation | Market belief of an specific company | [2,9] |
| | Score | Point value of submission depend on the task level | [8,12,16] |
| | Award | Specific monetary prize associated with a task | [2,4,5,6] |
| | Platform Competitors | Other platforms existed in market | [2,6] |
| Constraints | Activation Rate | Fraction of active workers in all registered workers | [2] |
| | Leaving rate | Fraction of registered workers who become inactive | [6,7] |
| | Experience gaining duration | Time taken for a new worker to excel | [2] |

Even at this point many factors such as the platform competitor's situation, member's personal career, and software worker leaving rate from the platform will effect individual decisions to stay or leave the platform.

*2) Dynamics of Task uploading in the platform*

Tasks are the basis of crowdsourced software development which can evolve by passing time. The goal of this model is to analyze trends of taken tasks, submitted tasks and completed task situation on these attributes.

It is extremely important to normalize tasks elaboration as it directly affects the task award. The award is one of the essential motivations for the Software worker to register for the task. However task complexity and also the size of the task are highly influential in choosing the tasks by software worker.

Tasks are never chosen by workers are typically called as starved tasks. As illustrated in fig 1 (uploading task section), based on market situation and economic conditions projects are divided to small micro tasks and uploaded into the platform to be chosen by the workers. As discussed in [9], the reputation of the company has a high effect on the software worker's motivation to register for a task. we model the tasks into three groups: (1) Start up companies' tasks, tasks uploaded by small companies with the idea of decreasing overall cost and getting tasks done through inexpensive software workers; (2) Normal companies' tasks, tasks uploaded by companies that are established and in addition to reduce the cost, try to create a buzz similar to the more well-known companies' reputation , and (3) Famous companies' tasks, task uploaded by companies which are already famous among end users and software worker community hence desire of working on their project is absolutely high, for these kind of companies the main idea of crowdsourcing can be project time line, security and overall cost. Historical data gathered from Topcoder indicated that almost 40% of registrants always registered in famous companies' tasks, 25% for normal companies, around 10% for start-up companies. The remaining tasks will get no registrants and will be starved. Simulation was run with the same category.

TABLE 2: ATTRIBUTES IN THE TASK UPLOADING MODEL

| Type | Attribute Name | Description | Reference |
|---|---|---|---|
| Software mini-tasks | Tasks | All uploaded tasks | [2,4,5,6] |
| | Taken Tasks | Tasks that have registered worker(s) | [2,6,9] |
| | Submitted tasks | Tasks that have associated submissions | [2,6] |
| | Completed tasks | Tasks that have associated submissions and past it due date | [2,6] |
| | Drop Taken tasks | Tasks that have registrants but with no submission | [2,6] |
| | Reload tasks | Tasks that have been uploaded more than once | [1] |
| | Drop out tasks | Tasks with submissions of unacceptable quality | [1] |
| | Acceptable tasks | Tasks with acceptable submissions | [1] |
| | Starved tasks | Tasks that have no registrants | [1,6] |
| Constraints | Task uploading rate | The rate of uploading small Tasks into the platform | [2,4,9] |
| | Task duration | Acceptable time between uploading and submitting a task | [1,6] |
| | Market situation | Total demand of new products and Software workers in the platform | [1,6,7] |

When a task is taken it may be dropped before submission or submitted as a processed task. In completed task phase, it may (1) accepted and good to use in final product, (2) there is need to rework and reload it into system or (3) not deemed acceptable and drop out. Applicable attributes in creating the SD model of task uploading in the platform are briefly explained in Table 2.

*3) The Final Layout*

A crowdsourced platform system continues working as long as there are uploaded tasks to perform and enough interested software workers to consistently process tasks.

The final layout illustrates the dynamic interaction model of such platforms is a combination of the previous two sub-models, as shown in Figure 1. In the new model task complexity has a direct impact on the award associated with the task and consequently on software workers' decisions on whether to take the task or not.

In this model the influence of task complexity on task size normalization is clear, since these subjects are influenced by the award and subsequently by demand for taking the task by software workers and submission tasks.

## IV. Results and Discussions

To run the model via Vensim, historical data from Topcoder was used. Systems dynamic simulates the model for a long run, so the results and available analysis are very helpful in market analysis and company policy decisions. The model provides options to run over 60 periods of time, with each period being considered as a day, a week or a month, respectively. Since the average time line of uploading a new task until submitting the final file in Topcoder is 30 days, this time period is considered as 60 days in our study.

The proposed models are analyzed with respect to 3 different decision scenarios with two groups of shocks (changes in external factors, attributes behavior, or actions by direct or indirect stakeholders in the system) including: (A) Shock occurs in award: how will the platform condition such as task taken and software worker situation and winning change if a shock with tolerance of 10% happens in Award? (B) Shock happens in Task uploading rate: how will a shock in task uploading rate influence the number of software workers and associated awards in platform and consequently the number of completed tasks? (C) Shock occurs in Active worker arrival: How can a changing number of active software workers influence the platform in terms of the number of task taking, granting experience point and submitting completed?

To analyze how the model simulates and predicts the above circumstances, we ran our simulation models three times per scenario: (1) First run (simulation) which is the initial result of the model was based on available gathered data from Topcoder with no shock, obviously this run has the same result for all 3 different scenarios; (2) Second run (simulation 1) was run for shock in specific attribute in the platform with an adjustment for -10%; and (3) The last run (simulation 2) was by assuming if the same attribute improved +10% in the platform.

### A. Result from Scenario A, B and C for first run

Results of the first run shows that by passing time the number of active software workers will increase and chance of taking the task by software workers will grow as well. In addition, by passing time and gaining more experience by software workers, the number of winners and also that for submitting completed task would rise. Moreover this run indicates that experience point gaining by winners and associated awards are following almost the same pattern, however the experience point drops first and then increased smoothly. This difference may be the result of lacking enough workers experience when they are joining the platform.

In the following sections outcomes of different shocks (Fig 2, Fig 3, and Fig 4) for second and third run of simulations in different scenarios will be discussed.

### B. Result from Scenario A (Shock in Award) for second and third run

As Fig 2 illustrates, while dropping the award by 10% in simulation 1, as expected it was seen that after 30 periods of time (days) not only the number of active workers increased by 15% and subsequently the number of taken tasks increased by 19%, the chance of getting more correct submitted task and having more winners increased respectively by 13% and 27% as well. And in simulation 2, when the award was increased by 10%, after 30 days the number of active workers, number of tasks taken and numbers of winners decreased in order by 12.5%, 23% and 29%, and consequently the chance of getting correct submissions would be reduced by 18%. This result demonstrates that raising the award has a negative impact on task, and consequently active software workers become less interested in completing the tasks, which confirms the results report in [8]. Moreover, a higher reward is usually associated

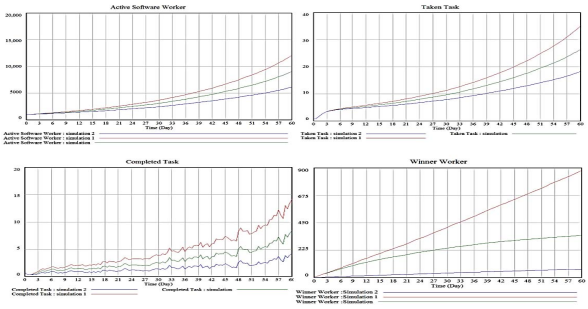


Fig 2: Scenario A, Shock in Award

with more complex tasks which is more time consuming, hence, the utility of such tasks will be decreased for software workers.

### C. Result from Scenario B (Shock in Task uploading rate) for second and third run

When a project gets decomposed to many smaller tasks, it makes for more task uploads into the platform and as a result it will directly effect on active software workers completed task and also associated award with tasks. In simulation 1, task uploaded rate was decreased by 10%, Fig 3 shows that after 30 days the number of tasks taken decreased by 15.4%, the number of active software workers reduced by 18% and chances of getting more completed tasks dropped 14% while as expected the award rose by 13%.

As in simulation 2, the task upload rate was increased by 10%, after 30 days the number of active workers, number of tasks taken and chance of submitting more completed tasks increased respectively by 17%, 13%, and 18%, and therefore associated awards decreased by 11%, and consequently the number of winner workers will increase as well. It was known when a project decomposed into smaller tasks, task complexity would decrease, and therefore the associated award would drop, when the number of composed tasks decreased, complexity would rise and accordingly the award would rise as well.

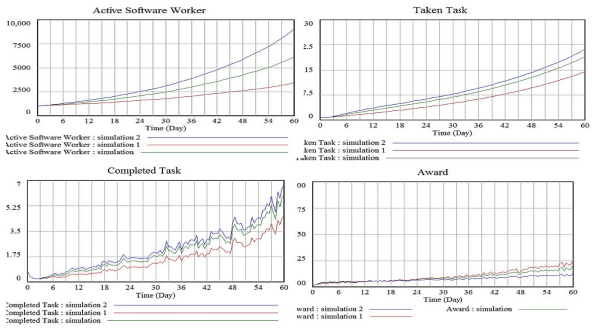


Fig 3: Scenario B, shock in Task uploading rate

### *D. Result from Scenario C (Shock in Active software worker arrival) for second and third*

Changing the number of potential software workers means there are more suppliers to perform the task and higher chance to receive correct completed tasks and obviously a higher number of winner workers. Fig 4 represents results of scenario C, when simulation 1, ran with an assumption of a 10% reduction in the rate of active software worker arrival, it is clear that number of tasks taken drop by almost 12% after 30 days, the chance of getting more completed tasks decreased by 19% and consequently the number of winners decreased, expectedly experience point per task decreased by 6%. In simulation 2, the rate of active software worker arrival improved by 10%, the number of active workers and number of tasks taken after 30 days increased in orderly by 16% and 5% and chance of submitting more completed tasks rose by 14%, therefore the rate of experience point per task increased by 4%.

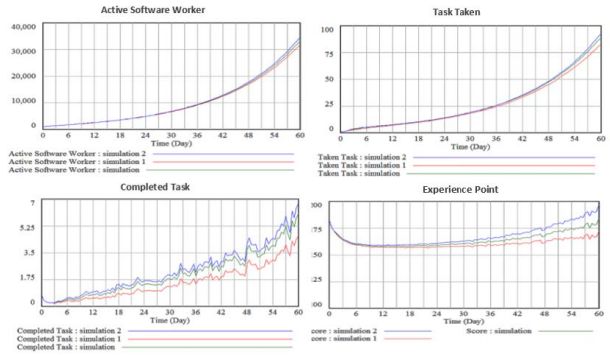


Fig 4: Scenario C, shock in Active software worker arrival

This conclusion is understandable, when the rate of software workers arrival is increasing and rate of uploaded tasks remains constant, the number of competitors per task and clearly number of well experienced workers rises fast, and consequently granted experience point grows.

According to simulation results (Figures 2, 3 ,4) the number of Active Software Workers are exponentially rising by time which makes taken tasks more likely to follow almost the same pattern as Active software workers, this result agrees to the result of [8, 9 ]. This observation support that decomposing projects to smaller size tasks with less complexity and lower associated award would guarantee more number of submitted tasks and higher chance of receiving acceptable completed tasks, and as a result a shorter project time. Also this practice would directly affect the platform market situation and involve more numbers of active software workers.

With respect to the available empirical data and simulation behavior it is indicated that by passing time and gaining more experience, Software workers register to participate in more tasks at the same time and also the win rate would increase per software worker. Based on simulation the winning rate increased from almost 12% of task registrations in day 9 to 45% in in day 60 for active software workers. This outcome is illustrated in Fig 5.

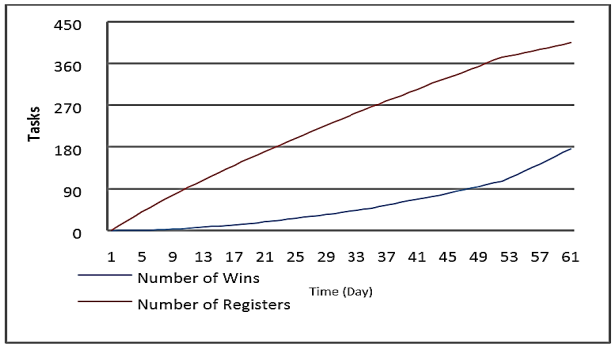


Fig 5: Number of Wins V.S Registers for new tasks by gaining Experience

Fig 6 displays relation between award, Experience point and submission rate. Results of dynamic simulation shows, granted Experience point and award have a direct relation while the submission rate and award have inverse relation. This result is also supported by historical data gathered from Topcoder. As task complexity is one of the basic factors of award allocation and also task completion process, such result was expected. In conclusion, the following practical insights can be derived from the results and discussions above:

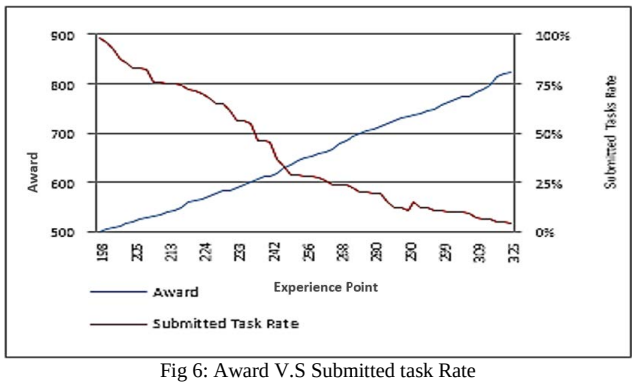


Fig 6: Award V.S Submitted task Rate

(1) By decomposing the project to smaller size tasks and uploading those in the platform by 10% higher rate the chance of getting more registrants and receiving more number of submitted tasks will be raised on average of

15% as the complexity of tasks will decrease which means shorter project time.

(2) Since a higher award is usually associated with more complex tasks, increasing the award by 10% will negatively effect on the total demand for tasks by 23%, consequently more cancelled or starved tasks will occur.

(3) In raising the number of software workers while the number of available tasks is constant, the chance of receiving correct completed tasks will increase by 14%, as chances are more numbers of experienced workers register for the task increased by almost 15%.

## V. CONCLUSION

By simulating a dynamic model of CSD platform, it is indicated that active software workers and task taken are following exponential distribution, while worker experience affects the number of taken tasks. By passing time and gaining more experience the active software worker will register for more tasks, while at the same time the chance of submitting completed tasks will increase.

However it is known that by increasing the Award, demand for task will decrease, simulation showed that award and experience point are directly related and affected by time while rate of submitted tasks and award are following task complexity as one of most important factors in micro task administrating. Simulation outcome and statistical analysis of available Topcoder data are agreeing on the same conclusion.

For future work analyzing market situation based on task demand, complexity and software worker experience is suggested. This research can help companies to decide when and where crowdsourced their project and how to normalized uploaded tasks. Also it can predict what fraction of the project should be crowdsourced and how much should be done in house.

## REFERENCES


[1] Klaas-Jan Stol, Brian Fitzgerald, Two's Company, Three's a Crowd: A Case Study of Crowdsourcing Software Development (2014).

[2] He Zhang, Barbara Kitchenham, D. Ross Jeffery: Toward trustworthy software process models: an exploratory study on transformable process modeling. Journal of Software: Evolution and Process 24(7): 741-763 (2012)

[3] He Zhang, D. Ross Jeffery, Dan Houston, LiGuo Huang, Liming Zhu: Impact of process simulation on software practice: an initial report. ICSE 2011: 1046-1056(2011)

[4] Siri-On Setamanit, Wayne Wakeland, David Raffo. Planning and improving Global Software Development Process Using Simulation. GSD, 06, May 23, Shanghai, China (2006)

[5] R. Madachy, Software Process Dynamics, Wiley-IEEE Press, Washington D.C., (December 2007)

[6] D.Pfahla, K.Lebsanftb. Using simulation to analyse the impact of software requirement volatilityon project performance (2000).

[7] Ross, J., Irani, L., Silberman, M.S., Zaldivar, A., and Tomlinson, B. Who Are the Crowdworkers? Shifting Demographics in Amazon Mechanical Turk. alt.chi '10, ACM Press (2010).

[8] S. Faradani, B. Hartmann, and P.G. Ipeirotis, "What's the Right Price? Pricing Tasks for Finishing on Time", In Proc. Human Computation, 2011.

[9] Dellarocas, C. Analyzing the economic efficiency of eBay-like online reputation reporting mechanisms. Proceedings of the 3rd ACM Conference on Electronic Commerce, (2001), 171–179.

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

[11] M. & Marsella, S. C. (2014). Encode Theory of Mind in Character Design for Pedagogical Interactive Narrative. Advances in Human-Computer Interaction. vol. 2014, Article ID 386928.

[12] Ke Mao, Ye Yang, Mingshu Li, Mark Harman, "Pricing Crowdsourcing-Based Software Development Tasks", ICSE 2013, San Francisco, CA, USA, New Ideas and Emerging Results

[13] AniketKittur, Jeffrey V. Nickerson, Michael S. Bernstein,Elizabeth M. Gerber, Aaron Shaw, John Zimmerman, Matthew Lease, and John J. Horton," The Future of Crowd Work", CSCW '13, February 23–27, 2013, San Antonio, Texas, USA.Copyright 2013 ACM

[14] G Silberman, M.S., Irani, L., and Ross, J. Ethics and tactics of professional crowdwork. XRDS 17, 2, 39–43. erber, E. and Dontcheva, M. Career Aspirations for Crowdworkers. In preparation. (2010)

[15] Hoffmann, L. Crowd Control, Commun. ACM, 52, 3(2009)

[16] Kaufmann, N., Schulze, T. and Veit, D. More than fun and money. Worker Motivation in Crowdsourcing - A Study on Mechanical Turk. Proc. 17th AMCIS.(2011)

[17] Jesus S. Aguilar-Ruzi, Isable Ramos, Jose C. Riquelme, Miguel Toro.An evolutionary approach to estimation software development projects. Elsivier Science B.V(2001)

[18] Venisim website: http://vensim.com