VOL. 10, NO. 6, APRIL 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

AN AGENT BASED SIMULATION STUDY OF ASSOCIATION AMONGST CONTESTANTS IN CROWDSOURCING SOFTWARE DEVELOPMENT THROUGH PREFERENTIAL ATTACHMENT

Nitasha Hasteer[1], Abhay Bansal[1] and B. K. Murthy[2]
[1]Amity School of Engineering and Technology, Amity University Uttar Pradesh, Noida, India
[2]Centre for Development of Advanced Computing, Noida, India
E-Mail: nitasha78@gmail.com

ABSTRACT

Software development is creative, challenging and ever evolving. With the increasing deployment of cloud technologies and benefits of crowdsourcing, an emerging form of software development is Software Crowdsourcing. The members of the crowd use various platforms to participate in competitions of software design and development to earn reputation and reward. In this paper we analyze and model the association amongst contestants in a software crowdsourcing platform to earn reputation. Agent based modelling is being used to simulate actions of agents (contestants) and measure the resulting system behaviour and outcomes over time. We model the preferential attachment behavior amongst the contestants and analyze the data retrieved from a crowdsourced software platform. This research proposes that agents that compete together for a certain task are more likely to be associated with each other for future competitions.

Keywords: crowdsourcing, software development process, agent based modelling, preferential attachment.

INTRODUCTION

Software development has always been creative, challenging and ever evolving. Organizations use various software development process models and methodologies for developing software. Crowdsourcing is an emerging form of outsourcing software development. It is a name given to a revolution that marks the rise of online community composed of like minded enthusiasts who work together, creating innovative solutions and lowering the production cost [1]. According to a report on 'Top Ten Technology Predictions' by Gartner, more than half of consumer goods manufacturers will receive 75 percent of their consumer innovation and research and development capabilities from crowdsourced solutions by 2017 [2]. Crowdsourcing in software development implies that services of voluntary online community are taken to build software in place of taking the services of traditionally employed workers. The objective of software crowdsourcing is to produce high quality and low cost software products by harnessing the power of the crowd. Almost all software development tasks can be crowd sourced. Software crowdsourcing practices blur the distinction between end users and developers, and allow the co-creation principle. A regular end user co designs and co creates the software [3].

Enterprises can outsource the task of developing software to the general crowd in either collaborative or competitive manner [3, 4]. In a collaborative crowdsourcing environment, people collaborate to produce software products [5]. Appstori.com is a collaborative crowdsourcing environment where people cooperate with each other on various aspects of mobile application development. Competitive crowdsourcing on the other hand is reward based. Topcoder.com is a crowd sourcing platform on which enterprises deliver their software development tasks and crowd members compete with each other to obtain solutions to the given problem and the winning crowd participant is rewarded. This environment promotes innovative ideas and obtains diverse solutions to the problems. The reward and the reputation earned by the winning participant is a driving factor for continuous contributions and addiction towards completion of the tasks [6]. To analyze the participation pattern of the contestants in various competitions, we model the preferential attachment behavior amongst them in a crowdsourced software development platform.

Software Process Simulation is a well established technique used to study behavior patterns, predicting future events, performing what-if and trend analysis and thereby improving software development. Agent-based modeling and simulation (ABMS) is an approach to model systems comprised of autonomous, interacting agents. Agent-based simulation of processes provides a natural way to describe communication between individuals, model their characteristics and can be implemented using various tools [7, 8, 9]. NetLogo, a programmable modeling environment is well suited for modeling complex systems developing over time. Modelers can give instructions to hundreds or thousands of 'agents' all operating independently. This makes it possible to explore the connection between the micro-level behavior of individuals and the macro-level patterns that emerge from their interaction [10].

In this paper we model the 'preferential attachment' interconnection amongst the contestants in a crowd sourced software development environment with the help of agent based modelling. A preferential attachment process is any process in which some quantity (some form of wealth or credit), is distributed among a number of individuals or objects according to how much they already have, so that those who are already wealthy receive more than those who are not. The principal reason

for scientific interest in preferential attachment is that it can, under suitable circumstances, generate power law distributions [11]. The rest of the paper is organised as follows: Literature Review section reviews the general crowdsourcing work; Crowdsourced software development section describes the software development process of TopCoder as a case study; Contestant Collaboration Network Model section lays out the Model; Model implementation sections sets the rules and simulates the model in NetLogo; Results and Discussions section shows the results and Conclusion and future work section concludes the paper and highlights the future research directions.

LITERATURE REVIEW

There are studies on various aspects of crowdsourcing that have been undertaken in the past. Many researchers have analyzed the economics of crowdsourcing contests. Huberman *et al.* demonstrated through an analysis of a massive data set from YouTube that the productivity exhibited in crowdsourcing exhibits a strong positive dependence on attention, measured by the number of downloads [6]. In his work Vukoic. M presented a sample crowdsourcing scenario in software development domain to derive the requirements for delivering a general-purpose crowdsourcing service in the Cloud. He proposed taxonomy for categorization of crowdsourcing platforms, and evaluates a number of existing systems against the set of identified features [12]. DiPalantino and Vojnović modeled crowdsourcing as business auction and leverage the research of auction theory to build models for reward system and effective strategies for crowdsourcing participants [13]. Archak presented an empirical analysis of determinants of individual performance in multiple simultaneous crowdsourcing contests for the portal TopCoder.com [14]. Zhenghui H. and Wu W. applied the famous game theory to model the 2-player algorithm challenges on TopCoder. They demonstrated that if a competitor's probability to make a successful challenge exceeds some certain value, then he will always choose to challenge the opponent [15]. LaToza *et al*. developed an approach to decompose programming work into micro tasks for crowdsourced software development and implemented it in CrowdCode [16]. In their work Stol and Fitzgerald presented an in-depth industry case study of crowdsourcing software development at a multinational Corporation and highlighted the challenges of the same [17].

Most of the research in this area has been on the studying the mechanism of crowdsourcing systems, like pricing, bidding strategies, rewarding rules etc. Since the creative work like software development requires a large degree of knowledge integration, coordinated effort and interaction among workers, this work focuses on modelling the association amongst the contestants with the help of agents to simulate real world competitions on a crowd sourced platform.

CROWDSOURCED SOFTWARE DEVELOPMENT

Software crowdsourcing is becoming increasingly popular with many portals like TopCoder.com; AppStori.com; uTest.com; mob4hire.com getting thousands of enthusiasts who are collaborating or competing to develop software. As a case study, we have analyzed the world's largest competitive software development portal, TopCoder. The portal reports 700, 000 registered workforces at the site. The site hosts competitions and challenges under graphics design, software development and data science tracks. The graphics design and software development challenges are of different types varying from idea generation and conceptualization to component development and generating test scenarios. The software development process at this portal is a simplified process wherein the company interacts directly with the client company to formulate application requirements, timelines and budgets. Once the application requirements are defined, the application enters the architecture phase and is split into a set of components. Any registered contestant who satisfies the minimum legal requirements can submit a design to any posted design competition. Winning design submission goes as input into the development competition, which has a similar structure. Output from development competitions is assembled together into a single application, which is later delivered to the customer. Each hosted competition belongs to some catalog and has two associated deadlines: the registration deadline and the submission deadline [14]. Figure-1 shows the TopCoder development process.

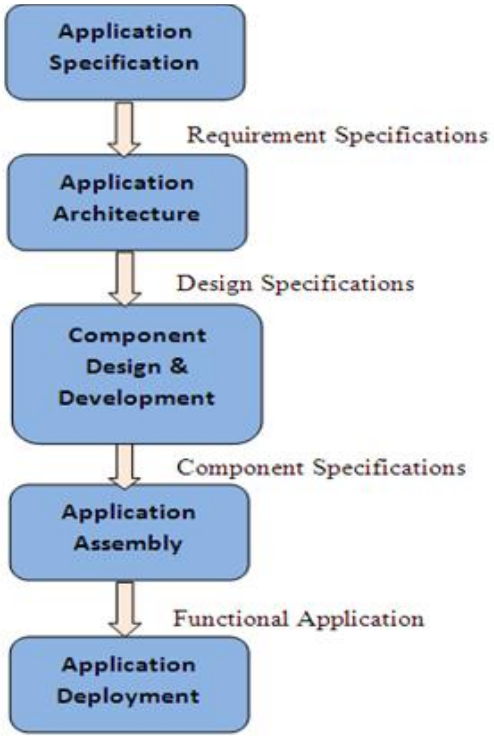

Figure-1. TopCoder development process.

The requirement analysis phase precedes specification phase and broadly consists of conceptualization, studio ideation and developing a GUI interface of the application. The platform allows the contestants to engage in wireframes or storyboards competitions to quickly create new ideas and express them in the form of mockups, thereby stimulating crowd creativity. The detailed requirement specifications are produced during the application specification phase based on the inputs from the earlier phases. The design documents are completed and the application's architecture is developed in the architecture phase. During the component design and development phase, contestants compete to convert the set of architecture documents into component specification documents and develop the application. All developed components are then linked together with the application flow and the application is delivered to the customer [18]. Figure-2 shows the tasks undertaken for developing software at the TopCoder portal.

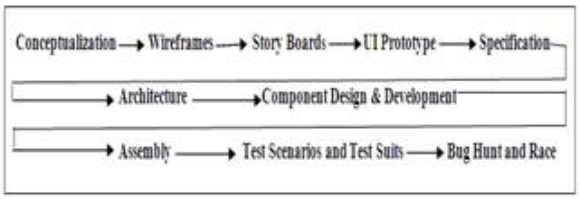

Figure-2. Tasks undertaken for developing software at TopCoder.

The process requires the contestants to compete during all the phases and after each phase the set of deliverables are generated. The winning entry serves as an input to the next phase. The software development tasks are accomplished through a series of competitions and matches after breaking down projects into units of work that consists of the entire build. The community has Program Managers who oversee customer projects and choose co-pilots within the community to act as an interface between customers and developers, and to help choose winners for the various contests. Co-pilots are experienced TopCoder community members who have proven themselves in the past on this platform. They manage the technical aspects of crafting and running competitions through to successful delivery [19].

CONTESTANT COLLABORATION NETWORK MODEL

Real world networks are like open systems that grow by the continuous addition of new nodes. Starting from a small nucleus of nodes, the number of nodes increases throughout the lifetime of a network by the subsequent addition of new nodes. We consider the environment of a crowdsourced software development as a network which grows with the addition of new nodes. This is a continuously expanding network, with 163, 351 nodes in 2008 [14], which grew to 450,000 nodes by 2013[18]. As the network grows the new contestants gets associated with the earlier existing contestants. We construct an association network model amongst the contestants, the Contestant Collaboration Network Model (CCNM), where the nodes are the contestants and two nodes are connected if the two contestants have competed for a same competition. This could be represented as a bipartite graph. Figure-3 shows a schematic representation of a bipartite graph, the graph of competitions and the contestants who have competed in them. In this small graph we have four competitions, labeled 1 to 4, and eleven contestants, labeled A to K, with edges joining each competition to the contestant who has competed for it. The bottom Figure shows the one-mode projection of the graph for the eleven contestants, [20].

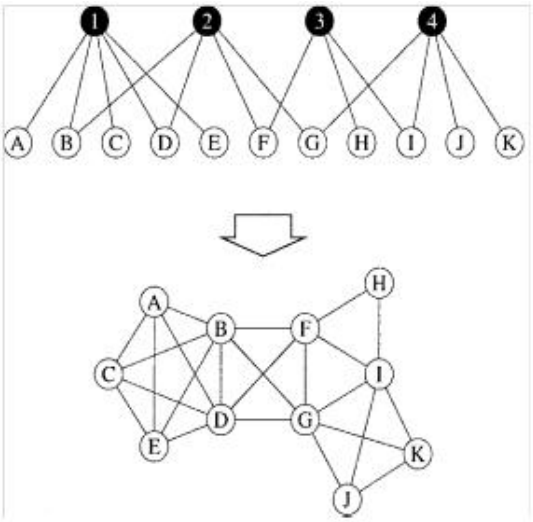

Figure-3. Representation of the competition-contestant graph and contestant collaboration network.

In a contestant collaboration network, two nodes are connected if the contestants have competed for the same competition. Nodes in a network have varying node degree depending on the number of edges it has. The spread in the node degrees is characterized by a distribution function P (k), which gives the probability that a randomly selected node has exactly k edges. Earlier studies have shown that for most large networks, degree distribution has a power law tail [21] [22]. Such networks are called scale free networks [23]. A central ingredient of all models aiming to generate scale-free networks is preferential attachment, i.e., the assumption that the likelihood of receiving new edges increases with the node's degree. The two ingredients of the Barabasi Albert (BA) model are growth and preferential attachment. The power-law scaling in the BA model indicates that growth and preferential attachment play important roles in network development [23].

a) **Growth:** Starting with a small number of nodes, at every time step, we add a new node and link it to different nodes already present in the system
b) **Preferential attachment:** When choosing the nodes to which the new node connects, we assume that the probability that a new node will be connected to node i depends on the degree k_i of node i, such that:

$$\pi(k_i) = \frac{k_i}{\sum_j k_j}$$

MODEL IMPLEMENTATION

NetLogo is an agent based simulation tool written in Java language at Northwestern University's Center for Connected Learning in Unite States [10]. It can be run on all major platforms. The environment uses three types of agents: turtles, patches and observer. Turtles are agents that move inside the world of bi-dimensional lattice composed by patches. Observer can be regarded as an entity that observes the world composed by turtles and patches. We use NetLogo for implementing our model and in this section we present how our problem was modeled and which abstractions were used to achieve it.

As agents, we defined the contestants registered at the crowd sourcing platform. Patches are stationary and arranged in a grid and agents move over the patches. They get connected through links. We set up our model based on our hypothesis that a contestant is more likely to get a new link, if it already has more links. For the purpose of simulation, we start with a fixed number N of contestants and implement the scenario based on the BA model of preferential attachment as described in the previous section. The number N however can be increased to 100 in our model. When the simulation setup starts, contestants are distributed over the network. The contestants are then arranged and laid out in order. Pressing the "go" button once (one tick), connects two contestants randomly. The choice of a contestant getting connected to the other is done stochastically as the simulation progresses. Evolved on the dimension of time, we simulate the forming and evolving process of CCNM. Figure-4 shows the interface at the beginning and at the end of the simulation.

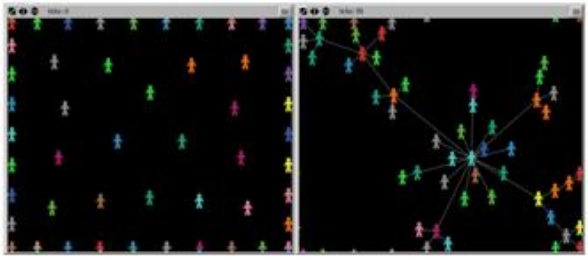

Figure-4. Network simulation.

The degrees of the nodes represented by the histogram, depicting the result of running the simulation program in NetLogo are shown in the following Figure-5.

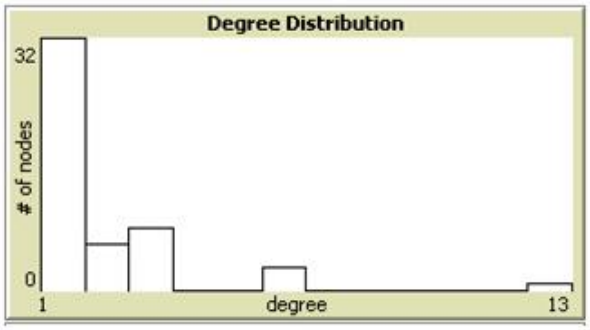

Figure-5. Degree distribution.

RESULTS AND DISCUSSIONS

In order to examine the association amongst the contestants in a crowdsourced software platform, we extracted and analyzed the data from the TopCoder web portal. Apart from design, development and data science challenges, the TopCoder portal also hosts Single Round Matches (SRM) every alternate weekend. These matches are of three hours duration and witness participation of around 2000 contestants for each match [24]. The contestants participate in these matches to earn social incentives like reputation and for fun. One of the key features of crowdsourcing is that a large number of people gets attracted to work on problems posted on the web, and this is consistent with TopCoder data where the platform shows that many people from different countries participate in various competitions. The originality of contestants at TopCoder platform is shown in Figure-6.

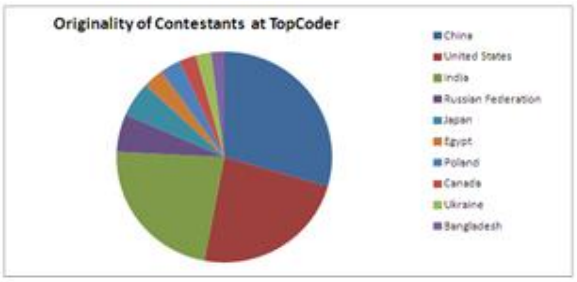

Figure-6. Originality of contestants

The dataset that we used for our study included various Single Round Matches ran by TopCoder from 01/01/2012 to 22/07/2014. There were in total 100 SRMs that we analyzed. For the purpose of thorough examination, we grouped the SRMs in four different groups having uniform number. The descriptive statistics of the SRMs is as shown in Tables 1 and 2 below:

Table-1. Statistics of data set.

S. No.	Group name	Single round matches
1	SG1	SRM529-SRM553
2	SG2	SRM554-SRM578
3	SG3	SRM579-SRM603
4	SG4	SRM604-SRM628

Table-2. Statistics of data set.

Group name	Count of SRMs	Total contestants (Multiple entries)
SG1	25	42678
SG2	25	38696
SG3	25	42553
SG4	25	41632

We then randomly choose ten contestants, one each from the top ten countries and analyzed their participation in various SRM competitions within each group. We examined the association with regard to their participation in various SRMs over a period of time. For the Group SG1, 92 percent of the times, more than 02 contestants out of 10 randomly selected have competed for the same SRM. For the groups SG2, SG3 and SG4 we found that 92, 92 and 80 percent of the times respectively more than two contestants have competed in the same SRM. Figure-7,8,9 and10 shows the result of the analysis.

In order to investigate the collaboration between pair of contestants, we did pair wise collaboration analysis of the contestants competing in various competitions. We have found that the association of the contestants grows with time which is an essential feature of scale free networks. For a random pair chosen the collaboration has grown from 8 percent of the times in SG1 and SG2 to 20 percent of the times in SG3 and SG4. Figure-11 to Figure-14 shows the result of analysis for one of random pair chosen.

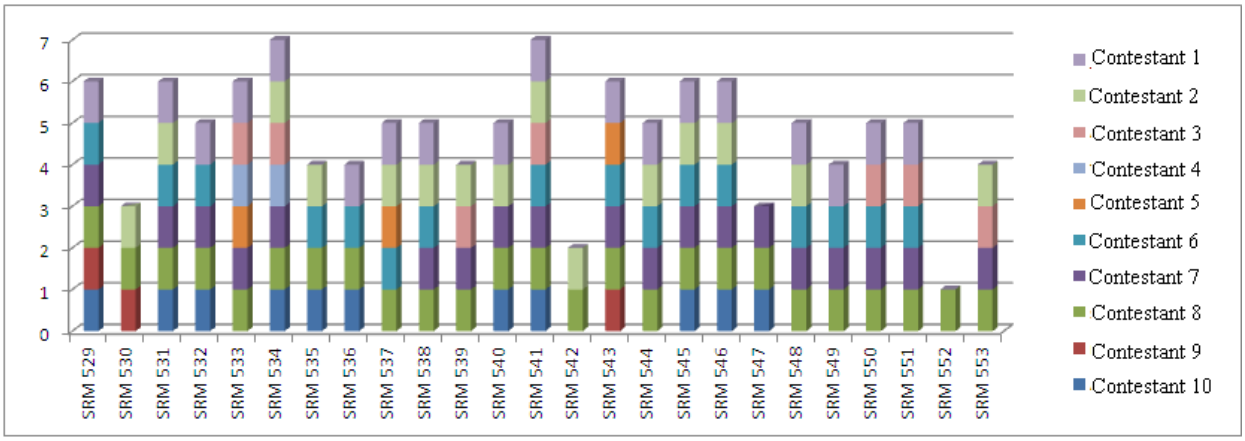

Figure-7. Contestant collaboration for SG1.

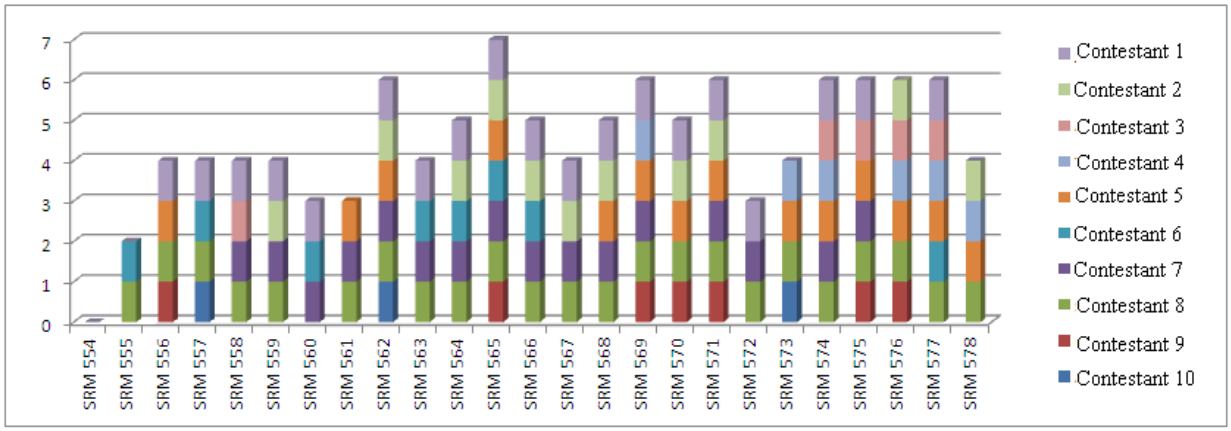

Figure-8. Contestant collaboration for SG2.

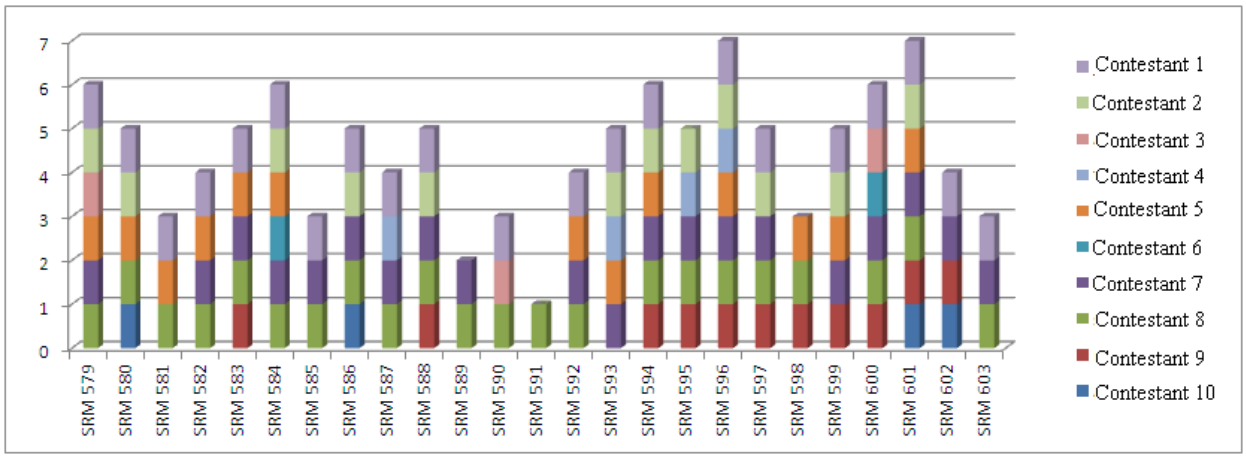

Figure-9. Contestant collaboration for SG3.

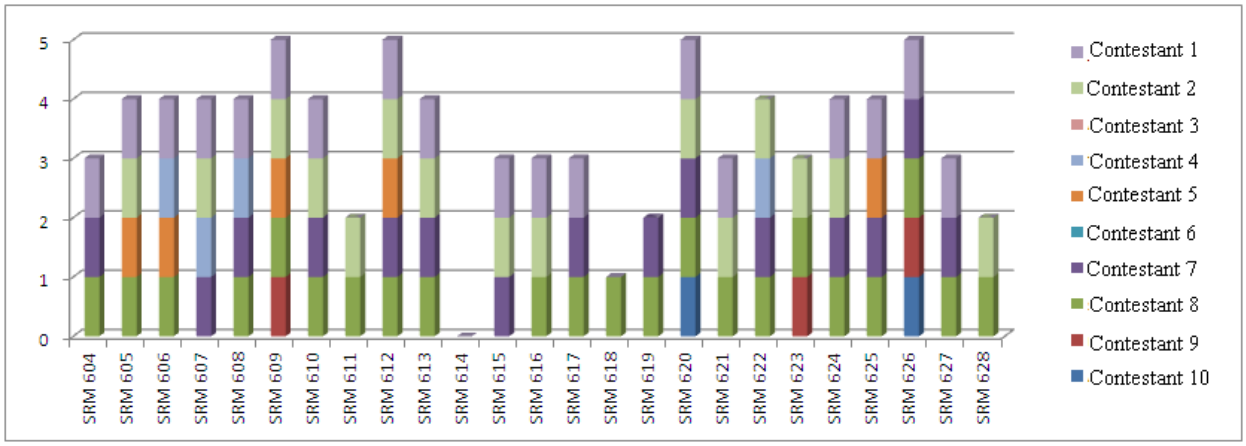

Figure-10. Contestant collaboration for SG4.

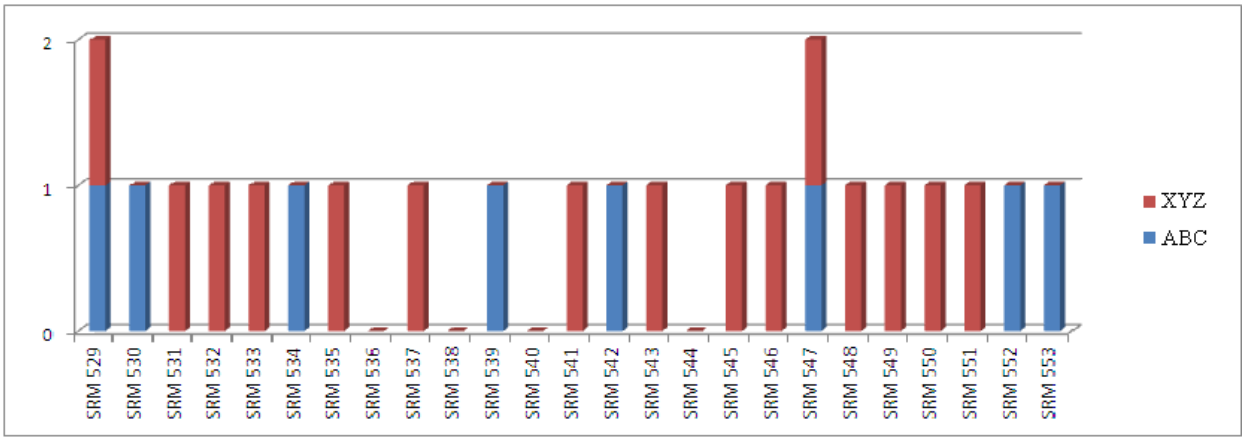

Figure-11. Pair collaboration for two random contestants in SG1.

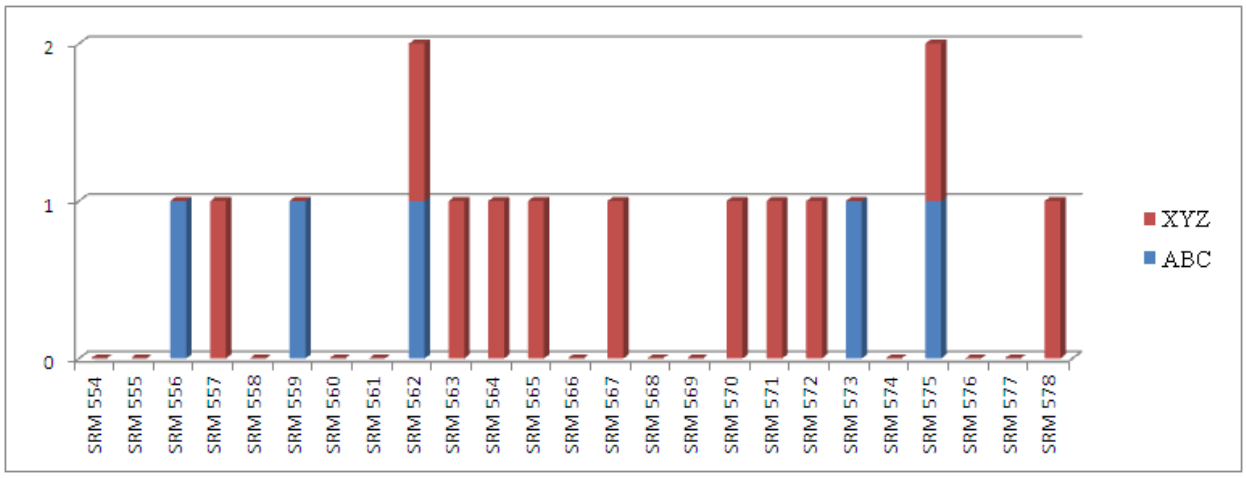

Figure-12. Pair collaboration for two random contestants in SG2.

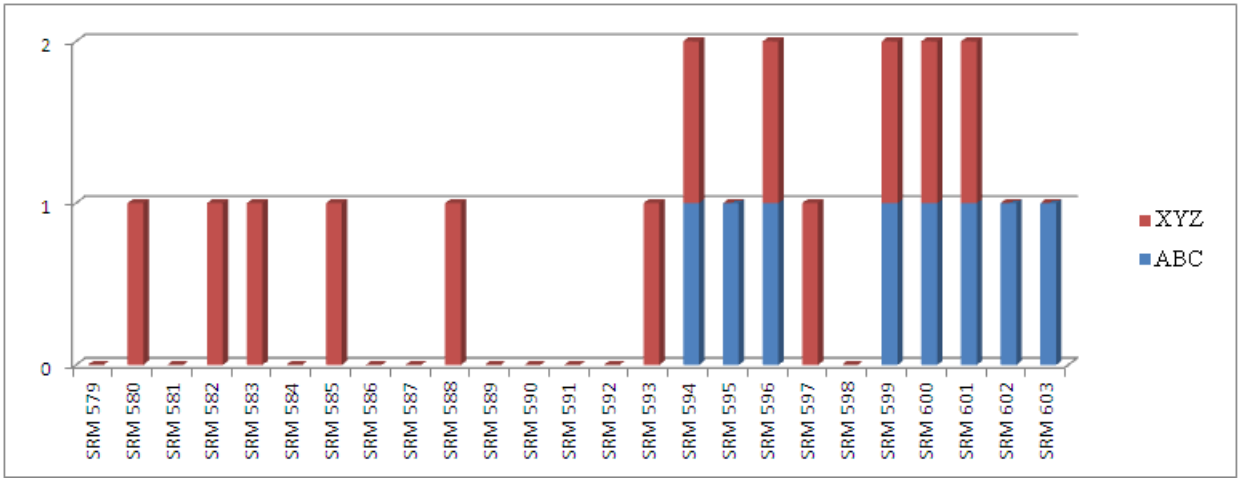

Figure-13. Pair collaboration for two random contestants in SG3.

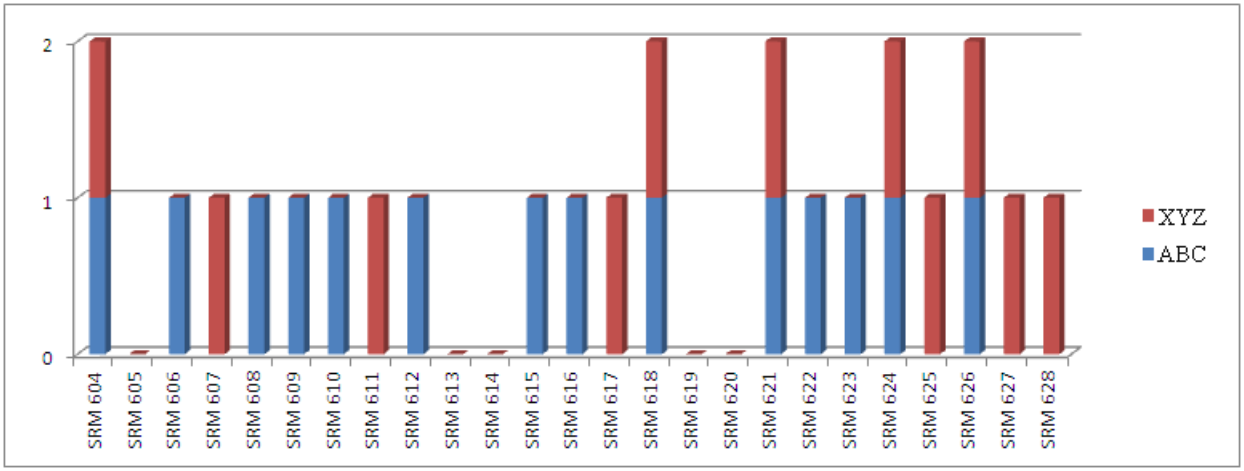

Figure-14. Pair collaboration for two random contestants in SG4.

CONCLUSION AND FUTURE WORK

The increasing number of crowd workers registering on the crowdsourcing platform signifies growth. We find that highly connected contestants increase their connectivity faster than their less connected peers subject to the growth of motivation to earn reputation as seen during various single round matches held by TopCoder. Crowdsourcing software development is a distinct and emerging approach to software development. It is different from the traditional software development scenario and uses the power of crowd to obtain solutions to problems. The magnitude and diversity of the crowd promotes creativity and innovation. This topic has recently started gaining attention by the software engineering research community. While the focus in the literature has been more towards the study of crowdsourcing mechanism and formation of pricing and reward rules, this paper contributes towards analyzing the association of the contestants competing on a software crowdsourcing platform. The preferential attachment model used for the study reveals that the contestants who get associated with each other during a competition, are more likely to compete together in future competitions to earn reputation.

Crowdsourcing delivers high quality solutions and makes the software development process more effective in terms of time and cost savings as compared to traditional development. Due to the social or financial incentive attached to the tasks hosted on to these platforms, the community of members of crowdsourced software platforms is bound to increase. Many research directions are thus possible including the research on coordination amongst contestants for collaborative development and the factors on which the productivity of a crowdsourced software development platform depends. The approaches and framework of software crowdsourcing needs to be investigated through multiagent system models for complex projects involving cross task coordination.

REFERENCES

[1] J. Howe. 2006. The rise of crowdsourcing," Wired Magazine, 2006.

[2] Gartner. 2014. Top 10 Technology Predictions for 2014.

[3] W.-T. Tsai, W. Wu and M. N. Huhns. 2014. Cloud-Based Software Crowdsourcing. Internet Comput. 18(3): 78-83.

[4] L. Hoffmann. 2009. Crowd Control. Communications of the ACM. 52(3): 16-17.

[5] X. Peng, M. A. Babar and C. Ebert. 2014. Collaborative Software Development Platforms for Crowdsourcing. Software. 31(2): 30-36.

[6] B. A. Huberman, D. M. Romero and W. F. 2009. Crowdsourcing, attention and productivity. J. Inf. Sci. 35(6): 758-765.

[7] A. T. Crooks and A. J. Heppenstall. 2011. Introduction to Agent Based Modelling. in Agent-Based Models of Geographical Systems. 164(2011): 85-105.

[8] T. Wickenberg and P. Davidsson. 2003. On Multi Agent Based Simulation of Software Development Processes. in Multi Agent Based Simulation. II, pp. 171-180.

[9] W. N. Robinson and Y. Ding. 2010. A survey of customization support in agent-based business process simulation tools. ACM Trans. Model. Comput. Simul. 209(3): 1-29.

[10] U. Wilensky. 1999. NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer Based Modeling Northwestern University Evanston IL. [Online]. Available: http://ccl.northwestern.edu/netlogo/.

[11] A. Capocci, V. D. P. Servedio, F. Colaiori, L. S. Buriol, D. Donato, S. Leonardi and G. Caldarelli. 2008. Preferential attachment in the growth of social networks: the case of Wikipedia. Phys. Rev. E. 74: 1-5.

[12] M. Vukovi. 2009. Crowdsourcing for Enterprises Maja Vukovi. in Congress on Services-I. pp. 686-692.

[13] D. Dipalantino and M. Vojnovic. 2009. Crowdsourcing and All-Pay Auctions. in EC'09. pp. 119-128.

[14] N. Archak. 2010. Money, Glory and Cheap Talk: Analyzing Strategic Behavior of Contestants in Simultaneous Crowdsourcing Contests on TopCoder. Com. in WWW 2010. pp. 21-30.

[15] Z. Hu and W. Wu. 2014. A Game Theoretic Model of Software Crowdsourcing. in Service Oriented System Engineering (SOSE), 2014 IEEE. pp. 446-453.

[16] T. D. Latoza, W. Ben Towne, C. M. Adriano and A. Van Der Hoek. 2014. Microtask Programming: Building Software with a Crowd. in User Interface Software and Technology Symposium. pp. 43-54.

[17] K. Stol and B. Fitzgerald. 2014. Researching Crowdsourcing Software Development: Perspectives and Concerns. in CSI-SE. pp. 7-10.

[18] W. Wu, W. Tsai and W. Li. 2013. Creative software crowdsourcing: from components and algorithm development to project concept formations. Int. J. Creat. Comput. 1(1): 57-91.

[19] K. Stol and B. Fitzgerald. 2014. Two's Company, Three's a Crowd: A Case Study of Crowdsourcing Software Development. in ICSE 2014. pp. 187-198.

[20] M. E. J. Newman, S. H. Strogatz and D. J. Watts. 2001. Random graphs with arbitrary degree distributions and their applications. Phys. Rev. E. 64: 1-17.

[21] R. Albert and H. Jeong. 1999. Diameter of the World-Wide Web. Nature. Vol. 401, no. September, pp. 398-399.

[22] H. Jeong, Z. Neda and A. L. Barabasi. 2003. Measuring Preferential Attachment for Evolving Networks. Europhys. Lett. 61(4): 567-572.

[23] R. Albert and B. Albert-Laszlo. 2002. Statistical mechanics of complex networks. Rev. Mod. Phys. vol. 74, January.

[24] A. Begel, J. Bosch and M.-A. Storey. 2013. Social Networking Meets Software Development: Perspectives from GitHub, MSDN, Stack Exchange and TopCoder. Software. 30(1): 52-66.