Trustworthiness in Crowdsourcing

Kumar Abhinav

Computer Science

Indraprastha Institute of Information Technology, Delhi (IIIT-D), India

A Thesis Report submitted in partial fulfilment for the degree of

MTech Computer Science

29 July 2015

1. Dr. Anurag Dwarakanath (Thesis Advisor)

2. Dr. Pushpendra Singh (Thesis Advisor)

3. Dr. Kuldeep Yadav (External Examiner)

4. Dr. Vikram Goyal (Internal Examiner)

Day of the defense: 29 July 2015

Signature from Post-Graduate Committee (PGC) Chair:

Abstract

Crowdsourcing is an exciting new trend where a group of geographically distributed individuals contribute willingly, sometimes for free, towards a common goal. Previous research and studies have shown how Crowdsourcing can be effectively used for Software Development. In this work, we investigate one of the key concerns in Crowdsourcing of Software Development Trustworthiness of the Crowd. Through a comprehensive review of related work, we present a taxonomy of the various concerns around the Trustworthiness of the Crowd. We study the current state of the art techniques to address these concerns both from academic literature and the Crowdsourcing vendors (platforms). From the study, we find that there are algorithmic techniques to tackle a few of the issues. However, other issues would still need to have a process. Thus to tackle Trustworthiness of the Crowd in a comprehensive way, a combination of algorithmic techniques and a process is required. We also investigate the performance of the algorithmic techniques in the context of Software development through a simulation of crowd behavior.

I dedicate my M.Tech Thesis to my parents and my brother who have always encouraged me in all phases of life and is my greatest source of inspiration.

Acknowledgements

I would take this wonderful oppurtunity to express my deepest gratitude to my advisors Dr. Anurag Dwarakanath and Dr. Pushpendra Singh for their continuous guidance, support, constant motivation and patience throughout my thesis. Without their guidance and support this thesis would not have been possible. I feel really blessed to have them as my thesis advisors.

Besides my advisors, I would like to thank the rest of my thesis committee: Dr.Vikram Goyal and Dr. Kuldeep Yadav for their encouragement, insightful comments, and hard questions.

I would also like to thank Dr. Alpana Dubey, N.C. Shrikanth, Gurdeep Virdi, Sakshi Taneja for their insightful comments, suggestions and constant support during the course of my thesis.

I am also grateful to my friends for their support and motivation. I would like to thank God for all his blessings.

Finally, I would like to thank my parents and brother for their constant support, encouragement, love and trust in me.

Declaration

This is to certify that the MTech Thesis Report titled **Trustworthiness in Crowdsourcing** submitted by **Kumar Abhinav** for the partial fulfillment of the requirements for the degree of *MTech* in *Computer Science* is a record of the bonafide work carried out by him under my guidance and supervision at Accenture Technology Labs, Bangalaore and Indraprastha Institute of Information Technology, Delhi. This work has not been submitted anywhere else for the reward of any other degree.

Dr. Anurag Dwarakanath
Accenture Technology Labs, Bangalore

Dr. Pushpendra Singh
Indraprastha Institute of Information Technology, New Delhi

Contents

List of Figures

List of Tables

1

Introduction

Crowdsourcing is an emerging area and has constantly evolved as a powerful practice to leverage the collective intelligence of an anonymous crowd. It has been applied in various domains ranging from creative resolution of a problem to improving the business process using several platforms. Crowdsourcing Software Development is one such emerging practice where the entire process of Software Development Life Process (SDLC). Examples of Crowdsourcing platform specific to Software development include TopCoder [1], Upwork [2], Freelancer [3] etc. Crowdsourcing gives various advantages including - faster time to market (due to parallel execution), higher quality (due to multiple opinions from the crowd) and lower cost (due to quicker access to the right talent) [4].

However, there is a general perception that the risk associated with Crowdsourcing is extremely high. Some of the factors associated with the risk perception is the anonymity of the crowd - it is not always apparent who is contributing to a task and whether this contribution is of adequate quality. The lack of direct control over the crowd also increases the perception of risk due to the inability to enforce quality through a strong penalty for deviances. The question of Intellectual Property (IP) further increases the question of risk i.e. by publicizing the tasks to be done, has an enterprise given away it's differentiation from its competitors.

In this paper we present a taxonomy of trustworthiness issues pertaining to Crowdsourcing Software Development. We studied related work to comprehend the existing tools and methodologies tackling these concerns. We studied the methods adopted by successful crowdsourcing vendors dealing with these issues. Further, we discussed an algorithmic method to address the key challenges and used simulation technique to study the behaviour of the algorithm proposed.

1.1 Research Motivation

The main motivation behind our research is the risks involved are often what prevent the company from engaging in an open innovation setting. The anonymity of the crowd raises doubt on trustworthiness and puts crowdsourcer in dilemma whether to crowdsource the task or not especially tasks containing sensitive information. Crowdsourcing tasks with sensitive information to an anonymous crowd poses risk and might lead to security and intellectual property issues. Clients discern that crowdsourcing might give competitors an insight to their business processes and future business plans. It is also apprehended by clients that crowd who developed the solution might forbid the client to use that solution or demand hefty payments of royalties after the solution gets incorporated in the business process. The solutions from an anonymous crowd might contain infringing and non-authorized material which can lead to intellectual property conflicts. These challenges have motivated to do this research and this is one of the first work done so far in this area. While many successful cases of crowdsourcing have been witnessed, we will in the next section call-out specific instances of where the crowd has led to a failure of the task.

1.1.1 *DARPA Shredder Challenge*

'Shredder Challenge' was a public competition organized by the US Pentagon's research and development branch, the Defense Advanced Research Projects Agency (DARPA). The challenge was to put together nearly 10,000 pieces from different documents that had been cross-shredded. University of California, San Diego (UCSD, one of the participating teams) launched an online program to work out the puzzles which enlisted a crowd comprising of 3,600 people from all over the world. This approach of soliciting the wisdom of the crowd made UCSD the top three out of more than 9,000 teams in just five days in spite of entering two weeks after the competition began. The day team's ranking was updated on DARPA website, there was an attack that night itself. An individual, acting as a crowdsourcing volunteer, sabotaged the UCSD team's puzzles by removing pieces that had been correctly matched. The following night another attack happened in which the saboteur moved the correctly assembled pieces far away from the UCSD team's puzzle board in a way that it could not be seen. As a result of these attacks, UCSD could not win the competition and placed sixth in the contest [5]. The saboteur in a mail to UCSD has highlighted consideration of security to be the most

significant measure while implementing crowdsourcing. The competition illustrates the advantages of using the collective intelligence of the crowd and at the same time points out the potential risk owing to the anonymous nature of the crowd.

1.1.2 *Villa Fresh Italian Kitchen's Dub The Dew campaign*

In Dub The Dew Campaign, crowd was asked to submit innovative names for a green apple-flavored Mountain Dew drink. Few malicious members of the crowd suggested names such as "moist nugget", "diabeetus" and "fapple". Another set of crowd started upvoting these silly names instead of the legitimate ones for fun. Promptly responding to these name suggestions, organizers pulled this campaign and shut down the website. Also the campaign received bad PR [6].

1.1.3 *Leakage of AOL search data*

On August 3, 2006, three-month search data containing twenty million search keywords of over 650,000 users was released on one of AOL's websites which was accessible to entire internet users in order to benefit academic researchers. In order to protect users' identity and maintain anonymity, AOL removed the username and IP address from the released data and replaced it with unique identification number. However, the search queries contained personally identifiable information which was used by two NewYork Times reporter, Michael Barbaro and Tom Zeller, to recognize the user (User No. 4417749) [7]. AOL's only intention was to strengthen research but the detailed records of search done by the 650,000 users were already circulated widely on internet and was available on mirror sites [8]. AOL was accused of violating the Electronic Communications Privacy Act and of fraudulent and deceptive business practices, among other claims, and the lawsuit was filed against AOL by Electronic Frontier Foundation (EFF) [9]. AOL apologized for the data release and fired the researcher and supervisor working on this release. This was followed by the resignation of company's CTO. At last the entire research division of AOL was shut down [7].

1.1.4 *Netflix Prize Contest*

To attract and retain customers to visit Netflix site for accurate movie recommendations and to improve Netflix movie recommendation algorithm Netflix launched "Netflix Prize Contest". On October 2, 2006, Netflix released 100 million records of movies rated from December 1999 to December 2005 by half a million Netflix users. The Netflix Prize Contest was beneficial to research in many ways. However, after two weeks of the release

it was claimed that with slight information of users' movie watching preferences from external sites such as Internet Movie Database (IMDb), it was possible to identify the users and find out all the movies they have rated on Netflix [7]. It was also asserted that the released data can reveal users personal information such as political and religious views. After the announcement of the first Netflix prize winner, Netflix was planning to launch the second Netflix Prize contest. However as per the lawsuit against the company filed by the customers in 2009, Netflix was accused of various US state and federal privacy laws. After the involvement of US Federal Trade Commission (FTC), Netflix declared the settlement with customers and scrapped the sequel [7].

1.2 Research Aim

1. To identify a taxonomy of Trustworthiness in Crowdsourcing Software development.
2. To study the current best practices and state of the art approaches to mitigate the concerns.
3. To propose and develop a statistical framework to simulate crowd behavior.
4. To investigate the performance of state of the art algorithms using the framework.

2

Background of Crowdsourcing

Crowdsourcing, coined by Jeff Howe, is a compound word formed from crowd and sourcing which means obtaining the work by giving it to unknown people or the crowd. Jeff Howe [10] has defined Crowdsourcing as

"the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call."

Crowdsourcing is still in its nascent stage and constantly evolving with time and development in technology. It is a problem solving as well as a production model [11] used to leverage the collective intelligence of the crowd to render brilliant solutions and ideas. It is a multi-disciplinary research area and has been incorporated in various domains and platforms. Crowdsourcing has been successfully applied by various institutions by utilizing the wisdom of the crowd. With crowdsourcing it is possible to accomplish massive projects by using the workforce available across the world. Today enterprises are looking for innovation in their business practices. The traditional approach of enterprises towards innovation has its own shortcomings such as limited resources with limited ideas and time constraint. Hence, fresh and innovative ideas can be generated by going beyond the conventional process and tapping the wisdom of the crowd. Crowdsourcing leverages the collective intelligence of an unknown crowd to bring in fresh perspectives. In Crowdsourcing people across the globe come forward to bring innovation in tasks ranging from micro-task to complex interdependent tasks. One of the successful instances of innovation through crowdsourcing was the Toyota Contest in 1936 [12] [13], to redesign the logo. From an overwhelming response of 27,000 entries, the winning logo for Toyota was selected.

The main entities involved in crowdsourcing are crowdsourcer, crowd and platform. It

is necessary to understand the perspective and role of each entity before implementing crowdsourcing.

1. **Crowdsourcer (or Requestor)**: This entity of crowdsourcing initiates the process by raising the request for a task to be done by the crowd. The requestor can be an individual, an institution, a non-profit organization, or a company [14]. One of the primary roles of requestor is to provide incentives to the crowd which can be in the form of monetary, social recognition or reputation. Requestor makes an open call to an unknown crowd to take part in the task which is provided in form of different modules. Maintaining ethical standards and confidentiality of the private information of the crowd is the responsibility of the requestor or crowdsourcer.

2. **Crowd**: The crowd is an anonymous group of people connected by internet who volunteer to participate and work on the task to provide innovative solution. Hobbyists and part-timers [10] constitute this group of volunteers with required knowledge and skills. The crowd can be public crowd which consists of unidentified group of people, a private crowd composed of specific experts who are invited to participate in the crowdsourcing activity or an internal crowd consisting of people known to the organization. The primary motives of crowd to participate are difficult to gauge owing to the heterogeneity and anonymity of the crowd and is a challenging area in crowdsourcing.

3. **Platform**: The platform acts as an interface between the crowdsourcer and the crowd and identifies the talent pool for a crowdsourcer. Today numerous platforms are used by the requestor and provider to complete a task on various domains. Several platforms are available for tasks ranging from micro tasks to complex integrated tasks. Amazon Mechanical Turk (AMT) is one of the most popular platforms for micro tasks which may include content-creation, testing, image tagging and document labelling. The most common platforms used in software development comprising of interrelated complex tasks include Topcoder which functions by making use of competitions [15].

The graph in Figure 2.2 shows the massive increase in revenue of 15 leading crowd service providers from 140.80 million US$ in 2009 to 375.70 million US$ in 2011 which is a phenomenal growth merely in two years with an expectation of market growth doubling in coming years [16]. Enterprises are adopting different approaches and are looking for new options to bring innovation in their current practices. To harnesses

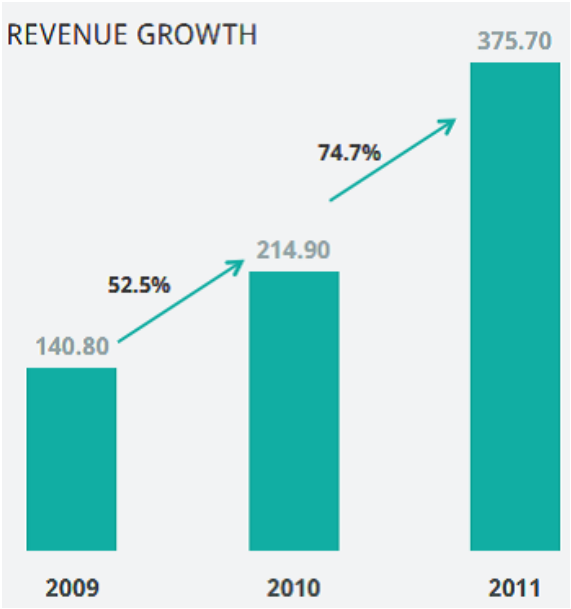

Figure 2.1: Crowdsourcing Revenue growth in two years

the intelligence and wisdom of the crowd and to obtain fast and scalable throughput, Enterprise Crowdsourcing is being used which in a sense is an innovative delivery model. Crowdsourcing has emerged as a viable option for these enterprises in all the domains like Banking, HealthCare, and Services etc. and enterprises are extending the boundary of tasks that can be done by the crowd. The graph in Figure 2.1 depicts implementation of enterprise crowdsourcing in different sectors [16]. The inevitable internet age with an exponentially increasing online crowd provides a pool of talent to the enterprises adopting crowdsourcing. The art of leveraging this opportunity is to implement enterprise crowdsourcing in an organized and secure manner to develop a new value system in the enterprise and to have a sustainable competitive advantage.

2.1 Crowdsourcing Software Development

The innovation through Crowdsourcing has been achieved in software development as well which has emerged as a model for problem solving and delivery in an innovative way. Crowdsourcing software development includes the process of documentation, design, coding and testing for software to be performed by a crowd which was earlier done by the employees or outsourced to a third party. With involvement of a large mass of people, there is a high probability of generation of new innovative ideas for the problem at hand and development of ground breaking solutions which were previously

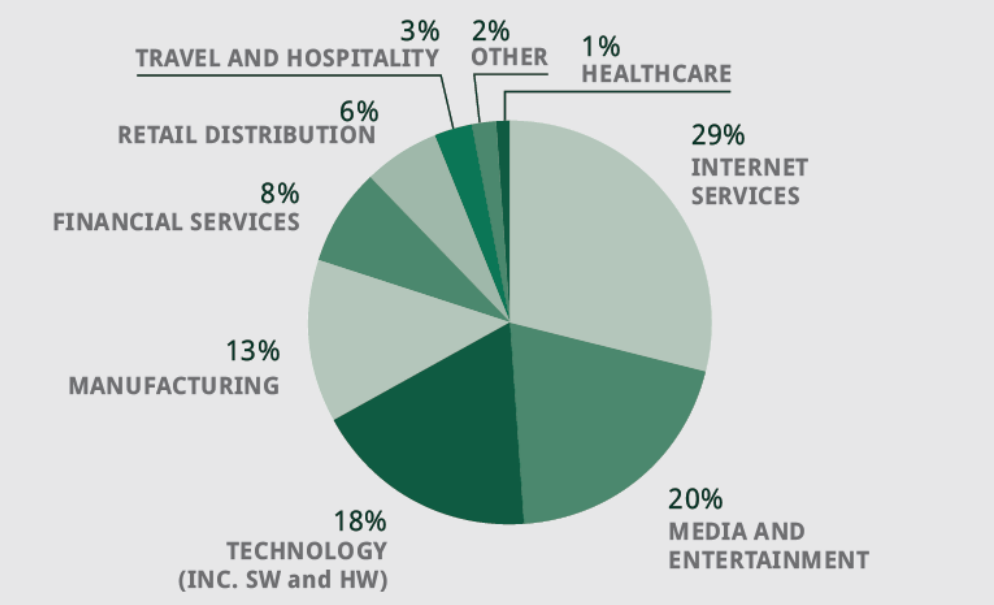

Figure 2.2: Crowdsourcing Activity used in Different Business Sectors

not imagined. The complex and interdependent tasks of a project are decomposed into modules and anyone from the crowd can participate by contributing solutions to these modules. The collective wisdom tapped through crowdsourcing improves the quality of software, The software is developed at a lesser cost and takes less time. As the crowd works at a problem, a variety of solutions is obtained. This provides the opportunities to crowd to earn social recognition. At the same time, the enterprises get a pool of talent which can be their potential employees. This is a win-win situation for both the participants and the organization. Two of the popular models used for crowdsourcing are:

1. **Competitive Model**: Tasks are published in a contest where teams participate and prizes are offered to the best solutions. This model encourages creativity and improves quality; however, rigid and tough competitions might limit active participation from the crowd.

2. **Hiring Model**: In this model, crowdsourcer hires crowd through interviews for secific projects.

2.1.1 Benefits of Crowdsourcing Software Development

Crowdsourcing has been applied in several domains to leverage the collective intelligence of the crowd. Crowdsourcing as an efficient problem solving and production model promises numerous benefits [17] [12] [4] in terms of quality, cost, time and creativity of the crowd specific to software development context. We briefly provide the different benefits below.

1. **Cost Efficient**: Crowdsourcing uses productivity based pricing model in which the payment is made to the work done and not to the employees and makes it relatively less expensive on account of no infrastructure, no hiring and training cost.

2. **Time Efficient**: The parallel processing of the entire process by highly skilled people across the time zone reduces the time invested to accomplish the task.

3. **On-Demand Availability**: The skilled crowd is available across the globe whenever there is any requirement. This on-demand availability of crowd reduces dependency on the existing workforce in case any emergency arises.

4. **Better quality**: The pool of highly skilled and specialised talent selects the task to be done as per their interest and expertise. The peer-review, competitions etc. are various approaches adopted to select the best solution and to ensure quality.

5. **Creativity and innovation**- Crowdsourcing provides a platform to explore the most innovative ideas by soliciting the wisdom of the crowd. The crowd consists of different people with different background who submit their solutions. This diversity in the crowd brings creativity and innovation.

2.1.2 Challenges with Crowdsourcing Software Development

Although Crowdsourcing offers several benefits to an organization but there are various challenges that are faced while implementing it [17] [4]. Few of the challenges are listed below.

1. **Task Modularisation**: To effectively decompose a complex and interdependent software product into tasks is what is termed as workflow design problem [18]. Efficient decomposition tend to increase parallel processing. The key challenge here is to maintain the balance between specifying the details for the tasks with clear requirements and limiting the interdependencies [17].

2. **Information Alignment and Conveyance**: This is one of the major challenges involved in crowdsourcing software development as it involves complex and interdependent tasks. The specification and the requirements to be communicated to the wide audience for each module must be clear. Another major issue here is to coordinate with a network of anonymous and unidentified people and the problem increases with increase in the size of the crowd.

3. **Organizing and Scheduling**: The control on the completion of the tasks on stipulated time, no longer remains in organization's hand [17]. The parallel processing of the interdependent tasks requires a meticulous planning and scheduling of the tasks and sufficient allocation of time to the crowd based on the size and scope of each task. The adherence of the crowd to the desired schedule for the successful accomplishment of the task is a big challenge in crowdsourcing.

4. **Quality**: The major quality attributes performance and maintainability can be controlled by peer review approach in crowdsourcing. However, the assumption of the presence of required expertise in an unknown crowd is a concern. Also, there is a tendency of decrease in quality of work with increase in participants.

5. **Incentives and Motivation**: The incentive rewarded pertaining to the required duration and the complexity of the task in hand may not be attracting enough to acquire and retain the talent pool owing to the diversified crowd with different expectations and needs.

6. **Talent Identification**: The identification of right talent in a massive crowd is a challenge. Another associated challenge is to retain the talented crowd with the right expertise for some specific task.

3

Related Work and Research Contributions

3.1 Related Work

There has not been much research done in the area of crowdsourcing software development and trustworthiness of the crowd. The work presented in this report belongs to the area of crowdsourcing software development. In this section, we discuss some closely related work (to the experiment presented in this report) and present the novel research contributions in context of existing work. We categorize the related work in 3 areas - Crowdsourcing Software Development, Crowdsourcing risks and Aggregation Approaches to evaluate trustworthiness.

3.1.1 Crowdsourcing Software Development

In 2006 Jeff Howe coined the term Crowdsourcing [10] . In 2010, Estellés-Arolas et al. [19] analyzed the existing definitions of crowdsourcing to draw out eight common characteristics and proposed a definition which includes majority of the existing processes involved in crowdsourcing. They defined Crowdsourcing as follows:

"Crowdsourcing is a type of participative online activity in which an individual, an institution, a non-profit organization, or company proposes to a group of individuals of varying knowledge, heterogeneity, and number, via a flexible open call,

the voluntary undertaking of a task. The undertaking of the task, of variable complexity and modularity, and in which the crowd should participate bringing their work, money, knowledge and/or experience, always entails mutual benefit. The user will receive the satisfaction of a given type of need, be it economic, social recognition, self-esteem, or the development of individual skills, while the crowdsourcer will obtain and utilize to their advantage that what the user has brought to the venture, whose form will depend on the type of activity undertaken."

Hosseini et al. [14] derived a taxonomy of crowdsoucing which discusses about the four main pillars of crowdsourcing- the crowdsourcer, the crowd, the crowdsourced task and the crowdsourcing platform to be used to represent the different configurations of crowdsourcing. Vuković, et al. [20] presented sample crowdsourcing software development scenarios for deriving general purpose crowdsourcing services and proposed a taxonomy which represents categorization of crowdsourcing platforms and evaluates a number of existing systems. Stol et al. [17] presented the first industry case study of Crowdsourcing Software development. They also discussed the key issues in Crowdsourcing Software Development through interviews with the client company who shared the experience of Crowdsourcing Software Development using TopCoder platform. They [21] also proposed a research protocol regarding the background, design and execution of this case study. Further they [22] mentioned the lessons learnt from the case study [17] and expressed their own advice for crowdsourcing software development. Prikladnicki et al. [23] discussed emergence of the three entities of Crowdsourcing (Crowdsourcer, Crowdsourcing platform and the Crowd) and identified the associated challenges in context of Crowdsourcing Software development in Brazil. Zogaj et al. [24] discussed the three main challenges of managing the crowd, managing the process and managing techniques faced by the Crowdsourcing platforms or intermediaries and presented a case study of how a German start up, testCloud, which is a crowd testing platform, solved some of these issues.

3.1.2 Crowdsourcing risks

Liberstein et al. [25] discussed the intellectual property and confidentiality risks of Crowdsourcing and presented the risk factors associated with three types of design- internally sourced, crowdsourced and agency sourced. Further they also proposed various approaches to minimize these risks. Wolfson et al. [9] pre-

sented the legal issues related to employment, data security, patent, copyright and crowdfunding that may arise due to Crowdsourcing and proposed suggestions to consider whenever a task is crowdsourced. Stefanovitch et al. [5] discussed the attacks on Crowdsourcing systems and provided the way by which crowd can help to recover from these attacks. Henkel et al. [26] presented challenges of protecting intellectual property and analyzed the reason for selectively revealing of any code until and unless it does not give advantage to competitors. Varshney et al. [27] used redundancy approach in which several workers perform same task and then use voting rule to determine the final output, to solve reliability issues and used random noise to hide sensitive data for privacy issues. Aner et al. [28] discussed the threats, vulnerabilities and risks involved in open-source software and proposed a framework for security evaluation and assessment of open-source software based around threat modeling to reduce the risks of confidentiality, integrity and availability.

3.1.3 Aggregation Approaches

Gadiraju et al. [29] identified different types of workers in the crowd and studied the behaviour exhibited by trustworthy and untrustworthy workers. They also identified the malicious activity on crowdsourcing platforms and proposed a method to measure the maliciousness of worker based on the acceptance of response. Balamurali et al. [30] presented a case study on crowdsourcing and automatic validation of completed tasks based on user modelling. Dawid et al. [31] proposed a model that estimates the error rate of workers performing the task (in form of confusion matrix) and predicted the true response for each task when the true label is not known. Smyth et al. [32] addressed the same problem in the context of labeling volcanoes in satellite images of Venus and used similar approach to combine labels from multiple experts to form a ground truth for items with homogeneous levels of difficulty. Raykar et al. [33] proposed an algorithm to iteratively compute ground truth when true label for the task is not known and measured performance of the annotators. Wang et al. [34] [35] proposed a hybrid scheme that can use any combination of redundancy and gold standard to jointly estimate the correct answer for each task and a “quality score” for each worker which can be used to separate systematic worker biases from unrecoverable errors. They also [35] proposed a pricing scheme based on workers quality. Venanzi et al. [36] used fusion algorithm to replicate a single task and

then to aggregate the answer of several users and constructed a likelihood model of the user's trustworthiness to overcome the challenges of fusing untrustworthy reports of observing crowd and at the same time identifying the trustworthiness of individuals. Whitehill et al. [37] proposed a system GLAD which is capable of handling millions of parameters used in processing large datasets at a minimal computational cost. The system combines human labellers and automatic classifiers and recovers the most accurate true labels in comparison to Majority Vote heuristic. Sheshadri et al. [38] discussed the evaluation of various consensus methods using different data sets. Hung et al. [39] presented a guideline to select the most appropriate aggregation technique as per the requirement, by evaluating and comparing these techniques used in crowdsourcing on the basis of several performance metrics. Kazai et al. [40] studied crowd worker's characteristics and presented the observation of the behavioural pattern of the worker and their personality traits. Based on their study and observations, they defined five types of workers as Spammer, Sloppy, Incompetent, Competent, Diligent. Raykar et al. [41] proposed a model to score and rank the annotators for crowdsourced binary, categorical and ordinal labelling tasks and formalized the notion of a spammer. KhudaBukhsh et al. [42] presented an algorithm to detect colluding groups in crowdsourcing. Kamar et al. [43] proposed an effective decision making system that combines computer vision, machine learning and decision-theoretic planning to decide when to hire new workers, and to predict next vote that a system would receive based on worker characteristics and task features so that it maximizes the efficiency of large-scale crowdsourcing processes.
We summarize this work in the below Table 3.1.

Table 3.1: Related Work Study.

Begin of Table		
Study/Projects	Type	Goal/Objective
Estellés-Arolas et al. [19]	Crowdsourcing Software Development	Analysed the existing definitions of crowdsourcing to draw out eight common characteristics and proposed a definition which includes majority of the existing processes involved in crowdsourcing
Hosseini et al. [14]	Crowdsourcing Software Development	Proposed a taxonomy of crowdsourcing which discusses about the four main pillars of crowdsourcing

Continuation of Table		
Vuković, et al. [20]	Crowdsourcing Software Development	Presented sample crowdsourcing software development scenarios for deriving general purpose crowdsourcing services and proposed a taxonomy which represents categorization of Crowdsourcing platforms
Stol et al. [17]	Crowdsourcing Software Development	First industry case study of crowdsourcing software development and discussed the key issues relevant to Crowdsourcing Software development
Prikladnicki et al. [23]	Crowdsourcing Software Development	Discussed emergence of the three entities of crowdsourcing and identified the associated challenges in context of crowdsourcing software development in Brazil
Zogaj et al. [24]	Crowdsourcing Software Development	Case study of how a German start up, testCloud, which is a crowd testing platform solved some of the challenges of managing the crowd, managing the process and managing techniques faced by the crowdsourcing platforms
Liberstein et al. [25]	Crowdsourcing risks	Discussed the intellectual property and confidentiality risks of crowdsourcing
Wolfson et al. [9]	Crowdsourcing risks	Discussed the legal issues that may arise due to crowdsourcing and proposed suggestions to consider whenever a task is crowdsourced
Stefanovitch et al. [5]	Crowdsourcing risks	Discussed the attacks on crowdsourcing systems and provided the way by which crowd can help to recover from these attacks
Henkel et al. [26]	Crowdsourcing risks	Addressed the challenges of protecting intellectual property by selective revealing of code

Continuation of Table		
Varshney et al. [27]	Crowdsourcing risks	Used redundancy approach to solve reliability issues, and random noise to hide sensitive data for privacy issues
Aner et al. [28]	Crowdsourcing risks	Discussed the threats, vulnerabilities and risks involved in open-source software and proposed a framework for security evaluation and assessment of open-source
Gadiraju et al. [29]	Trustworthiness evaluation	Identified different types of workers in the crowd and studied the behaviour exhibited by trustworthy and untrustworthy workers
Balamurali et al. [30]	Trustworthiness evaluation	Case study on crowdsourcing and automatic validation of completed tasks based on user modelling
Dawid et al. [31]	Trustworthiness evaluation	Estimates the error rate of workers and predicts the true response for each task when the true label is not known
Smyth et al. [32]	Trustworthiness evaluation	Approach to combine labels from multiple experts to form a ground truth for items in the context of labeling volcanoes in satellite images of Venus
Raykar et al. [33]	Trustworthiness evaluation	Algorithm to iteratively compute ground truth when true label for the task is not known and measure performance of the annotators
Wang et al. [34] [35]	Trustworthiness evaluation	Proposed algorithm to jointly estimate the correct answer for each task and a "quality score" for each worker which can be used to separate systematic worker biases from unrecoverable errors
Venanzi et al. [36]	Trustworthiness evaluation	Used fusion algorithm to replicate a single task and then to aggregate the answer of several users and constructed a likelihood model of the users trustworthiness

Continuation of Table		
Whitehill et al. [37]	Trustworthiness evaluation	Proposed a system GLAD which is capable of handling millions of parameters used in processing large datasets at a minimal computational cost
Sheshadri et al. [38]	Trustworthiness evaluation	Evaluation of various consensus methods using different data sets
Hung et al. [39]	Trustworthiness evaluation	Evaluation of different aggregation techniques on the basis of several performance metrics
Kazai et al. [40]	Trustworthiness evaluation	Studied crowd workers' characteristics and presented the observation of the behavioural pattern of the worker and their personality traits
Raykar et al. [41]	Trustworthiness evaluation	Algorithm to detect colluding groups in crowdsourcing
Kamar et al. [43]	Trustworthiness evaluation	Effective decision making system,to decide when to hire new workers, and predict the next vote based on worker characteristics and task features

3.2 Research Contribution

In context to existing work, the study presented in this paper makes the following research contributions:

(a) The work presented in this report is the first step in the direction of evaluating trustworthiness in Crowdsourcing Software development. We proposed Taxonomy of the various concerns around the Trustworthiness of the Crowd.

(b) We presented current state of the art techniques to address these concerns both from academic literature and the Crowdsourcing vendors

(c) We evaluated performance of the algorithmic techniques in the context of software development through a simulation of crowd behavior.

4

Taxonomy of Trustworthiness Issues

Crowdsourcing Software development is a promising and emerging field. It acts as a platform where the crowd performs the entire software development tasks given by crowdsourcer or requester which are generally enterprises and establishes a business value transfer between the crowd and requester. The literature study brings out the benefits of crowdsourcing software development and uncovers the associated risks. However, Trustworthiness in Crowdsourcing Software Development is an area which has not been analysed by researchers until now. Trustworthiness in crowdsourcing software development can be analyzed from the three perspectives: perspective of the crowdsourcer, perspective of the crowd and perspective of the crowdsourcing platform. We will limit our study in this paper to the Trustworthiness of the crowd.

Trustworthiness of the crowd is a potential concern especially in software development context. Massive participation of the dispersed crowd and the associated uncertainty makes trustworthiness of the work performed by an anonymous crowd and the entire software, one of the major challenges in crowdsourcing software development. There can be different scenarios where assessing trustworthiness becomes a challenge. For instance it is difficult to ensure that workers have not left any back door in the code which can later be exploited by them. There is also a possibility of illegally reusing the technology for which crowdsourcer has paid to develop the software. Another major challenge is ensuring that the intellectual property of the task is not stolen and monetized by any member of the crowd

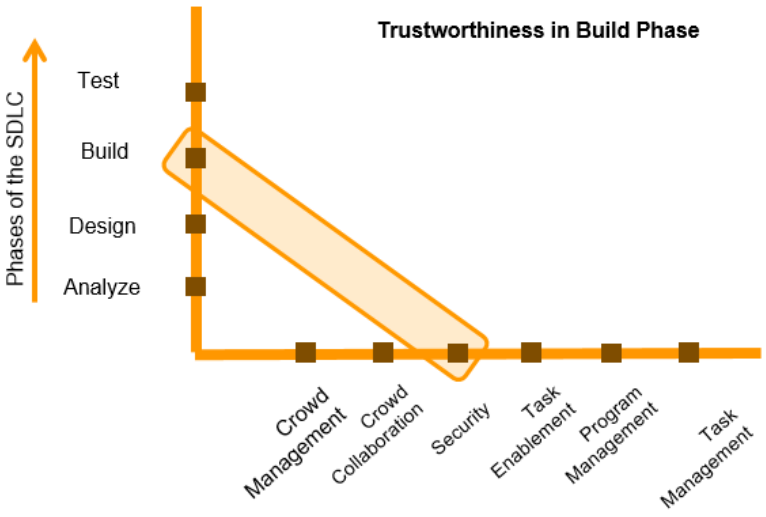

Figure 4.1: Trustworthiness in Build Phase

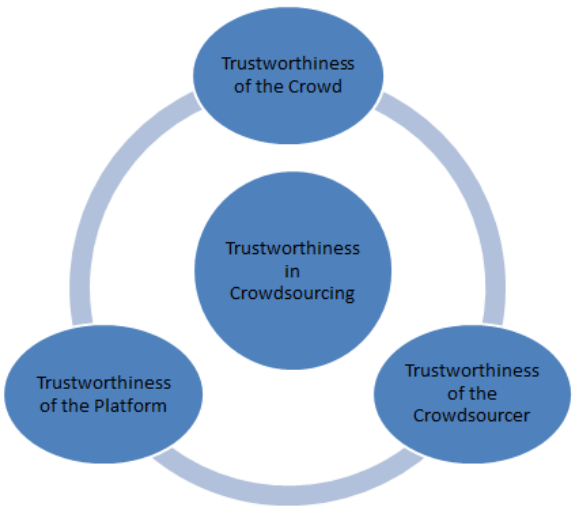

Figure 4.2: Trustworthiness of Crowdsourcing

which can give advantage to competitors. There can be many other possibilities from business point of view pertaining to trustworthiness of the crowd. Hence identification of these issues becomes necessary before any problem arises. In this work we have developed a taxonomy which deals with several issues in relation with trustworthiness of the crowd.

4.1 Taxonomy for Trustworthiness of the Crowd

Based on a literature review on crowdsourcing, this paper presents a taxonomy of concerns of trustworthiness of the crowd (shown in Fig. 4.3) that are of particular importance in context of Software development.

4.1.1 *Trojan Code Submission*

Within the realm of trustworthiness of crowd, submission of Trojan code is a concern in crowdsourcing as there is always a possibility of submissions consisting more than the required business functionality. The crowd can insert malicious or harmful code into a program, which he/she could later exploit, leading to various privacy and security breaches. Trojan code submission can either be a sheer act of carelessness or can be a malicious intention of some individual in the crowd. However, both the cases manifest dire consequences and most importantly are not easy to discover. Consider an instance of creating a login module which contains sensitive information in form of passwords. An individual submits the task which the system testing finds functioning well without any error. But what if, at the same time the data is flowing to some other system and leaks sensitive information. This data flow of sensitive information is not easily detectable and if it's a malicious intent it's likely that some attempts would have been made to conceal this data leakage.

4.1.2 *Non-adherence to best practices*

In context of software development, security of the system is subject to the technology and method incorporated along with the people involved in the process. When software development is crowdsourced, the number of people involved is massive and this in turn adds to the potential risk of security breaches in various forms. There is a set standard to be followed with its appropriate and

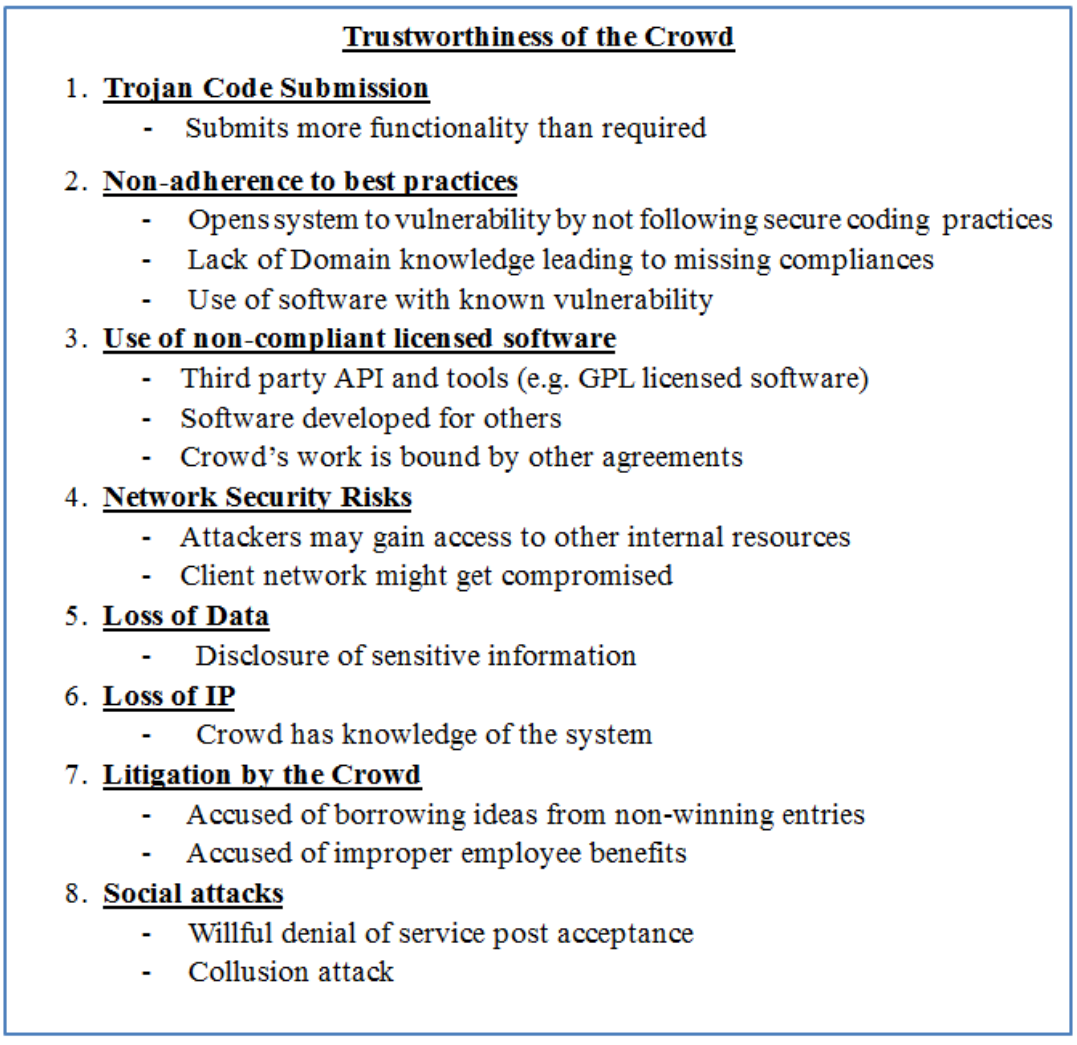

Figure 4.3: Taxonomy of Trustworthiness of Crowd

accurate implementation to ensure security in the product or service. While developing software, it is necessary to be aware of the policies of the business and apprehend the post implementation risks and compliance to various privacy regulations. Sometimes these basic security requirements such as confidentiality, authentication, authorization etc. are not followed by the crowd to save time. This intensifies the possibility of exposure and vulnerability and its impact can be disastrous. Moreover the required domain knowledge of the individuals in the crowd is in question and is open to doubt or suspicion. As an example, the software pertaining to sensitive information like password or credit card number requires encryption process to protect the confidentiality. It is likely that to submit the task requirement before the deadline and to get monetary benefits as soon as possible, encryption is not incorporated because the individual does not possess the detailed information of the client's requirements specific to banking and financial services and is not aware of the environment in which the software has to function or in simple words the individual does not understand the business purpose of the software to be developed. If the required domain knowledge does not exist in the crowd the client's requirement will not be understood clearly and consequently the potential challenges and its impact will not be assessed by the crowd working on the software. Another issue is the use of open source software with known vulnerability by individuals in the crowd to save time and effort of writing long codes or whatsoever be the reason. As per the study conducted by Aspect Security[44] to evaluate software for vulnerabilities, over a period of 12 months 31 popular open source libraries downloaded out of which more than a third of the 1261 versions of these libraries had a known vulnerability and about a quarter of the downloads were tainted . Contrast Security highlights that 29.8 million out of 113 million downloads from the central repository i.e. 26% of the library downloads consist of known vulnerabilities wherein majority of library glitches remain undetected. Crowdsourcing software development increases the risk as the crowd is unknown, handling this security issue of vulnerabilities in open source code libraries, with carelessness can have extremely disastrous impact such as data leakage or corrupted data, sensitive and important data flow to attackers or sometimes complete buy out of the party using it.

4.1.3 *Use of non-compliant licensed software*

In relation to trustworthiness of the crowd, understanding of compliance requirements is a requisite to understand the business. Validating whether the submissions from a massive unknown crowd adhere to compliance regulations is a tedious task. An individual might replicate the solution from third party API and tools which is not an open source code library. This usage of licensed software is a violation of compliance to regulations and privacy, consequently it increases the risk and can lead to serious impact. In another case of use of non-compliant licensed software, any individual can resubmit his/her developed software to some other clients (or competitors) in need of the same software violating the compliance regulations and keeping a check on this is challenging. In addition, sometimes crowd's work is bound by some agreements, violation of which affects the final product and can lead to legal disasters. As an example [45] consider intellectual property agreement of a university. At the time of assigning the rights to the solution, it is found out that the best solution selected by the company through crowdsourcing has been submitted by a student of such a university which, as per the intellectual property agreement, claims all the inventions. As a consequence, the company will not be able to incorporate the solution and the submitter will not get paid for the work done by him/her. Another misfortunate possibility is when the submitter would have signed the intellectual property assignment and the company would have incorporated the solution without the disclosure of the intellectual property agreement causing disastrous legal consequence. Moreover, cost and difficulty to verify all the submissions is very high.

4.1.4 *Network Security Risks*

To facilitate the application development and deployment, the Company has to provide necessary network access to the crowd. This access permission may expose sensitive information, if not monitored properly, will become serious threat to confidentiality. For instance, service accounts will be created for the applications to access the Database (DB) server, a care should be taken to minimize the access permission to the service accounts like disabling delete permission and admin privilege. If service accounts have admin privilege one can log in to DB server with admin privilege and can gain the complete control over server.

4.1.5 *Loss of Data*

One of the areas where crowdsourcing risks often run ahead of law and evokes trustworthiness of the crowd is Data Security. Crowdsourcing is opted in many cases to facilitate and stimulate research by leveraging the crowd's ingenuity but at the cost of sharing and disclosing data which might contain sensitive information. While, the purpose of the business entities behind releasing supposedly anonymous customer records might be to study user data, nonetheless this genuine endeavour can breach privacy regulations by not protecting personal information. There is always a loss of sensitive information when crowdsourcer share data with the crowd even after anonymizing the sensitive information, which leads to various privacy and security breaches. As it can be seen in incidents of AOL and Netflix which cite the potential reward and the associated risk of crowdsourcing. The insecure storage of data, inadequate evaluation of the vulnerability of the process, and not incorporating strong measures against hackers are few areas where the crowdsourcer can consider while implementing crowdsourcing to protect the confidential user information. However, the anonymity of crowd impedes the process of trust building. Exhibiting too much trust on an unknown crowd might turn malicious whereas limiting the trust might be dysfunctional. The extent to which trust needs to be displayed is a challenging task in crowdsourcing keeping in mind the anonymous, unknown and unidentified nature of the crowd as even a fair and honest attempt to leverage crowdsourcing to analyse user's data can lead to data security problems and violation of privacy.

4.1.6 *Loss of Intellectual Property*

Several intellectual property and confidentiality risks loom ahead of the company when crowdsourcing approach is adopted pertaining to software development. The task is given in the form of modules and there is no single contributor that can formulate a comprehensive understanding of the problem of the crowdsourced project. Therefore, these modules must be designed in a way that it should not portray a big picture to the crowd, otherwise, it might give advantage to competitors. Moreover, the reluctance of enterprises to share too many details of certain task with an anonymous crowd is fully apprehended in crowdsourcing context. Still they have to provide sufficient details to make sure the requirements for the task is clearly understood by the crowd members. Striking this balance is one of the major challenges in crowdsourcing and presents the potential risks in relation

to trustworthiness of the crowd. Loss of intellectual property might lead to loss of competitive advantage and hence, trustworthiness of the crowd has crucial importance in crowdsourcing. Furthermore, ownership of innovations exposes risk to an infringement lawsuit. After submission, there is no clarity on who owns the solution- the company which initiated crowdsourcing or the individual who submitted the solution. The question that arises is how to ensure lucidity and avoid any confusion related to ownership especially in software development context which requires more creativity for complex tasks. The solutions are drawn from an unknown crowd and there is a high possibility of plagiarism i.e. the submissions can be a work of others or can contain open source code with restrictive GPL. The level of risk associated with intellectual property depends on the nature of the crowd[25]. As far as internal crowd is concerned, it extends the lowest risk on account of the control the company has over the ownership and the work done by internal crowd or employee qualifies as works made for hire. However in many cases, companies seek external sources to leverage their latent talent and collective intelligence. There is medium level of risk involved if the crowd is private as it is comparatively easier to verify the work and to terminate the agreement or contract with work for hire provision, if any. If the crowd is public, it involves the highest risk of infringement due to the contribution from a massive group of unknown people as it is not known with whom the company is dealing.

4.1.7 *Litigation by the crowd*

Every solution possesses some element of innovation and contains confidential innovation related information. In certain cases, an individual from non-winning entries might accuse the client for borrowing his/her ideas and incorporating it in the business process. To ensure that none of the future work of the company has any similarity with any of the solutions submitted or trenches on any of the rejected solutions, is a challenge. One area in crowdsourcing where there is still ambiguity is who qualifies as an “employee”. Several federal and state laws specify the relationship to be maintained between employer and employee particularly, how to treat an employee. However, crowdsourcing adopted in several domains in various ways and with no clear definition of an employee face federal and/or state regulation over employment practices. Subject to crowdsourcing and considering the amount of control the employer has over the worker, crowdsourced work might

not qualify as works made for hire [25][9]. Under this ambiguity, the crowd might accuse the employer of improper employee benefits.

4.1.8 *Social Attacks*

Consider the case where an individual responsible for submitting the solution backs off after agreeing on completing the task within stipulated time. In another instance a group of multiple participants in the crowd, either to have a competitive advantage or as a malicious intention, conspire and decline to submit after consenting to finish the task on the stated deadline. In both the scenarios, the wilful denial of service by crowd post acceptance demands re-initiating the entire process creates additional burden on the company in terms of extra cost and time incurred to get the solution. Moreover, manual review of the code submitted by the crowd is a daunting task and to execute this, the company picks out a disparate crowd. There is always a possibility of multiple people in crowd to collude either to have a competitive advantage or monetary benefits and as a result of the incorrect or unfair review the final solution is contaminated.

5

Existing Approaches to Mitigate issues

In this section, we present the current state of art techniques, both from academic literature and the crowdsourcing vendors, to address the problems discussed in previous chapter. Few of the challenges discussed such as Trojan Code, Non-Adherence to best practices (Secure coding, Domain knowledge), Third party APIs can be tackled by verifying the quality of the submission using code quality tools(VeraCode, SonarQube, Blackduck) and approaches such as peer review. The limitation is that these tools cannot address issues such as reuse of others software by crowd, crowd bound by another contract, Litigation by the crowd. Methodologies such as reducing anonymity, background check, Contracts and long term incentives are used to prevent any deviation from conformity of the norms and address most of the issues of trustworthiness of the crowd. However genuine misses in the submission such as unknowingly introducing vulnerability cannot be tackled by these approaches. The concern of client on confidentiality of sensitive information can be addressed by using IP modularity and Selective revealing. IP Modularity facilitates in protecting intellectual property by breaking down the task into modules and in this way spreading out and shrouding sensitive information. Organizations decompose complex tasks into modules or components on which crowd perform independently, these modules are then integrated to function together as a complete solution for the task. This partitioning of tasks into components has many other advantages such as; it reduces complexity of the task, assists in proper segmentation of labour, and provides flexibility. In selective approach certain confidential and sensitive information is anonymized

and only revealing certain crucial information protects intellectual property and ensures data security We also investigated the approach employed by popular crowdsourcing software development platforms such as Upwork, Freelancer and TopCoder to tackle these issues. These platforms use Legal contracts and Non-Disclosure Agreement(NDA) to tackle most of the issues. Moreover, Upwork provides custom contracts for clients. TopCoder employs various approaches to mitigate these risks. Modular Software development is one such approach where complex tasks are broken down into modules. This compartmentalized Software development approach [15] prevents malicious crowd to put in any unwanted or harmful code (as they have to work in modules and are unaware of the overall task) and also helps in protecting IP. Another approach used by TopCoder is the community based peer review where expert community members review all code submissions and select the contest winner, which also helps in ensuring code quality and security [15] [46]. The Peer review is used to assess trustworthiness of the crowd. In order to protect client confidentiality, TopCoder anonymizes client name and entire project information (actors and use cases of the project). Taking into consideration the concern of few clients to protect confidential information, TopCoder give the option to go for CCA (Competition Confidentiality Agreement) Contests. These are private contests where members have to agree to an additional confidential agreement, thus adding an extra layer of protection. The study of these tools and methodologies employed by popular crowdsourcing platforms have been presented in Tabular form

Issues	Tools and Techniques available	Tools and Techniques used by Vendors		
Trojan Code	Static and Dynamic Code Analysis, VeraCode[47] , Manual Review	Custom Contracts	Peer Community Review, CheckMarx[48]	Not Applicable
Non-Adherence to best practices	SonarQube[49], BlackDuck[50], VeraCode, Manual Review	Custom Contracts, Legal Contracts	Peer Community Review, CheckMarx	Legal Contracts
Non-Compliant Licensed Software	Ninka[51], Binary Analysis Tool[52] , Background checks, Legal Contracts, Manual Review	Custom Contracts, Legal Contracts	Peer Community Review, Legal Contracts	Legal Contracts
Loss of Data	Privacy Preserving Data Publishing[53] , k-anonymity[54]	Not Applicable	Simulated Data, DMZ[46]	Not Applicable
Loss of IP	IP Modularity[55][56], Selective Revealing[57], Legal Contracts	Non-Disclosure Agreement, Custom Contracts	Task Breakdown, Anonymize Client/Project, Contracts, Private Crowd, CCA, Non-Disclosure Agreement [46][15]	Non-Disclosure Agreement
Litigation by the crowd	Legal Contacts	Legal Contacts, Employment Relationship	Legal Contacts	Legal Contacts

Table 5.1: Existing Approaches to mitigate risks

6

Evaluation of Trustworthiness of the Crowd

In a software engineering context, client may not have knowledge of the crowd who develops the software and are unaware of the processes followed. It is likely that the developer decides on such a course of action that satisfies the minimum requirements to submit the task but such actions could bring liabilities to the enterprise. Hence, owing to the anonymity of the crowd, evaluating trustworthiness of the crowd becomes a major challenge in crowdsourcing software development. The current state of art mitigates these concerns using well known standard mechanisms.

The evaluation of trustworthiness of the crowd can be done in broadly two ways:

(a) Establishing the trustworthiness of the crowd through this actions (i.e. the submissions he makes)

(b) Establishing the trustworthiness of the crowd through his associations (i.e. associated with a contract, associated with a university, etc).

In this section, we investigate the methods to establish trustworthiness through his actions (i.e. point (a)). Evaluation of trustworthiness of the crowd on three parameters- whether the code is Trojan or not, does the code adhere to the best practices, and does the code comply with licensed software, can be done by code review based on the submission of the crowd. There can be various approaches for reviewing the code either by internal crowd or by giving it back to crowd. However our past experience [4] suggests that the time and effort incurred in internally

reviewing the code was almost equal to the time and effort of development of the application itself. The possible approaches to evaluate trustworthiness in crowdsourcing software development are shown in the workflow (Figure 6.2). In our approach we divided the software development task into modules. Each module is then give to crowd for development. Based on the assumption that there will be multiple submissions for a module, a disparate crowd is hired which is different form the set of developers. This disparate crowd is asked to verify the submissions based on three parameters- whether the code is a Trojan code or not, whether the code adheres with best practices or not and whether there is any use of non-compliant licensed software in the code. The review response of the disparate crowd is then aggregated using known techniques to identify the true response of the review and thereby identify the best submission.

There are different computational approaches [58] [35] to evaluate the trustworthiness of the crowd. In this paper we will discuss the following approaches which are commonly used in crowdsourcing

(a) **Reputation based Approach**: In this approach, historical data of quality of work submitted by the crowd and interaction done with crowdsourcer, is used to generate a reputation score for each worker [58]. The past performance of the workers assesses the quality of the workers and hence is used to measure trustworthiness of the crowd. The crowdsourcer prefers workers with high reputation scores and this reputation score of worker serves as a parameter to get better monetary benefits and attainment of social recognition.

(b) **Gold Standard Approach**: In this approach a set of questions is put in the task for which answers are already known to the crowdsourcer. Based on the discrepancy between response submitted by the crowd and correct answer for predefined set of questions, trustworthiness can be assessed [35]. Using this approach it is possible to measure trustworthiness of the crowd and compute error rate of each worker (performance of worker). A trust belief is formed on the user's capability using the performance of the worker on same task to differentiate trustworthy and untrustworthy crowd and to filter out low quality workers. However, the crowd might get digressed from providing solution for the actual task. Hence, integration of the set of questions, with known answers, in the actual task is a challenge. Also, this incurs an extra cost on the crowdsourcer.

(c) **Consensus based Approach**: This is the most common approach to determine the true response and in turn to asses the trustworthiness of the crowd [35] [58] [43]. In this approach consensus is built by the crowd. Each response is considered as a vote and is based on the belief that eventually the most accurate solution will get maximum votes. This approach relies on redundancy i.e. ask multiple workers to complete the same task and as the number of participation increases, the possibility of building consensus on the most accurate solution also increases.

In this paper, we adopted consensus based approach, where we gave the submitted code for review to crowd members and asked to review each line of code pertaining to the three issues- Trojan code, Non-adherence to best practices and Non-compliant licensed software (Fig. 6.1). However, the challenge with this approach is to aggregate the response from the crowd and find out the best solution. There are various ways to aggregate the crowd's response and predict the true value. In this paper, we will discuss two of these approaches.

(a) **Majority Voting (MV)**: This is the most common and simple consensus based method [33] [38] [39]. In majority voting, the label agreed with majority is treated as correct or true label. It assumes majority of workers in the crowd are quality workers who work independently and ultimately the majority of crowd workers' vote will agree on ground truth. This approach is widely applied from binary classification to multi-class classification and multiple-choice. However, MV does not consider model worker behavior and also doesn't take into account trustworthiness of the crowd and assumes each participant in the crowd to be trustworthy. Another issue with MV is that for tie, random draws lead to uncertainty (or instability) in results.

(b) **Expectation Maximization (EM) Algorithm**: This is an algorithm for finding the probabilities of latent variables, which can be used to estimate the true labels and the workers' accuracy. This algorithm was applied by Dawid and Skene[31] in medical diagnosis to estimate the error rates of physicians when they examine patients for which there is no correct answer.

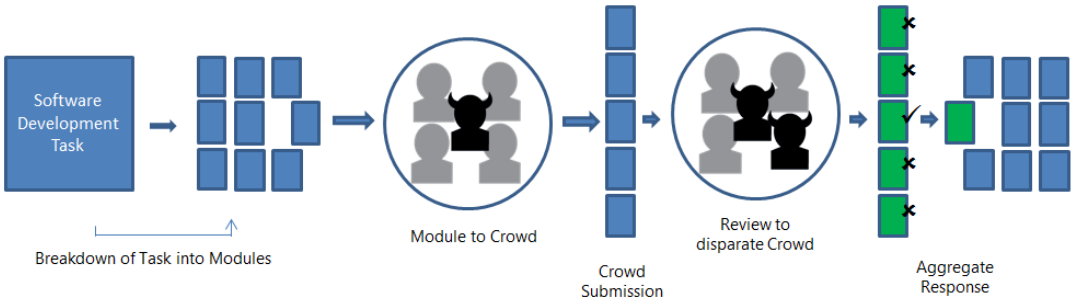

Figure 6.1: Our Approach[2]

Algorithm 1: Expectation Maximization Algorithm

Input : Responses from each reviewer for every task

Output: Confusion matrix for each reviewer, Class Priors for each class, True labels for each task

1 Initialize error rates for each reviewer, error rates will be predicted on the basis of gold standard tasks;

2 Initialize true label for each task, using majority vote;

3 **while** *not converged* **do**

4 Estimate the true label for each task based on the error rates for each reviewer

5 Estimate the error rate of each reviewer based on the true label predicted in step 4 and the label assigned by reviewer

6 **end**

[2]Inspired from [59]

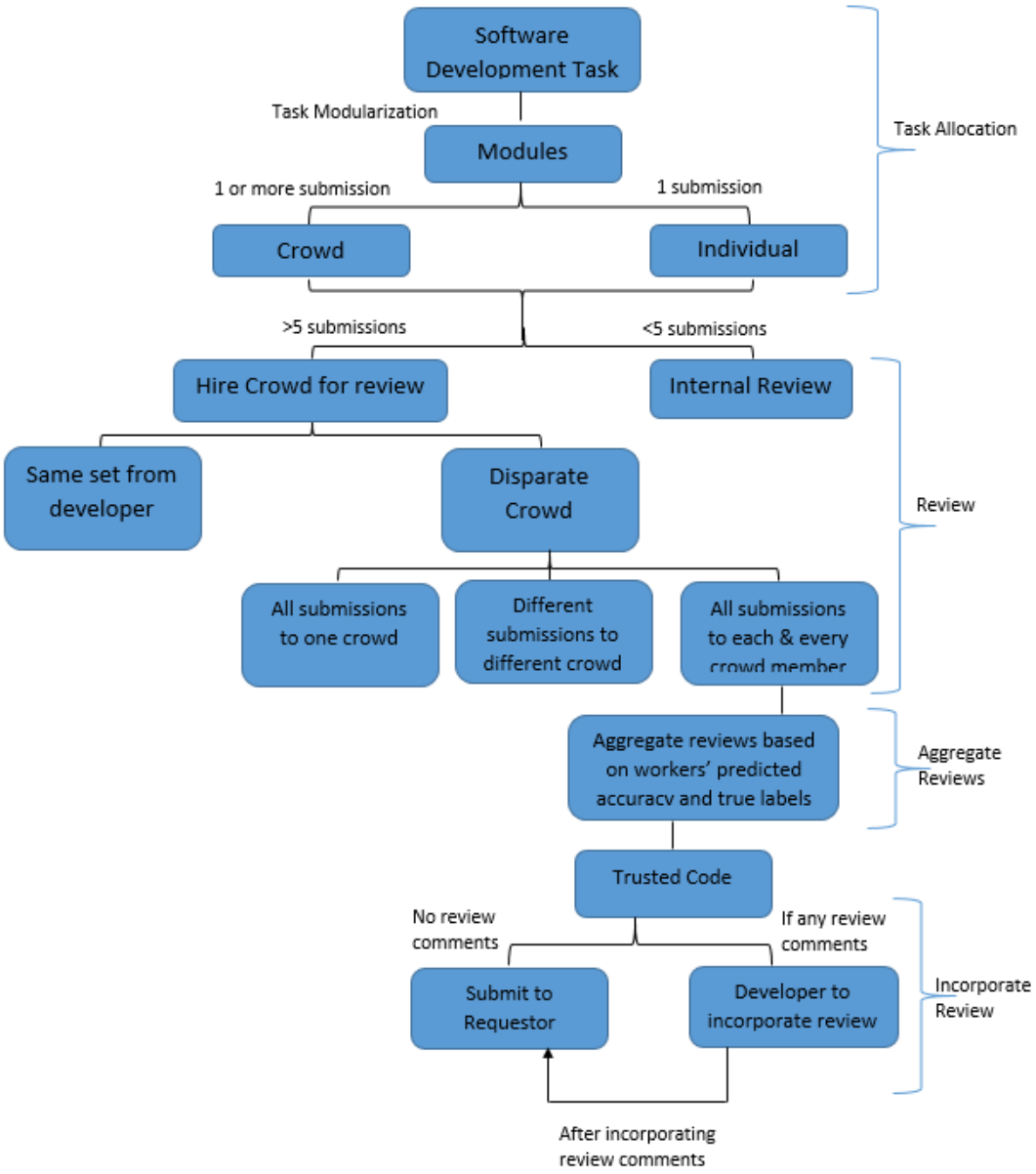

Figure 6.2: Proposed Workflow

7

Experimental Dataset and Evaluation

Due to lack of availability of real world datasets on which we can test our Majority voting and Expectation Maximization algorithm, we generated synthetic datasets based on the simulation of workers' behavior and prior probabilities for each category. We simulated the behaviour of the crowd as a probabilistic system. Each crowd worker is assumed to follow a Bernoulli distribution [60] to give a binary answer to a question. Every question has an answer following the Bernoulli Distribution, but with a skewed prior probabilities e.g. the chance of having a Trojan code is very low with 0.2 probability and the chance of having Non-adherence to best practices is very high with 0.7 probability (shown in Table 7.1). The implementation of the project is available on github[1].

There can be different types of crowd workers in crowdsourcing.

(a) **Expert worker**: expertise in the area with profound domain knowledge and the questions answered correct with a high probability.

(b) **Biased worker**: intentionally gives incorrect answers.

(c) **Random Spammer**: gives random answers for any question.

(d) **Uniform Spammer**: with a specific motive give same answers for all the questions.

(e) **Adversarial Colluded worker**: give wrong answers by colluding with other workers having malicious intention. Adversarial Colluded Leader (ACL)

[1]https://github.com/kumarabhinav04/Trustworthiness-in-CS

with malicious intention marks all answers as wrong. Adversarial Colluded Followers (ACF) follow their leader and mark all answers same (with high probability) as leader.

(f) **Non-Adversarial Colluded worker**: give correct answers by colluding with other workers for the sake of monetary benefits. Non-Adversarial Colluded (NACL) Leader marks all right answers and Non-Adversarial Colluded Followers (NACF) follow their leader and copy the answer marked by leader.

Mathematically, we have defined different types of workers' responses in a probabilistic manner as shown in Table 7.2

Begin of Table	
Types	Prior Probabilities(Pr)
Trojan Code	$\Pr(TC = Yes) = 0.2$ $\Pr(TC = No) = 0.8$
Non-Adherence to best practices	$\Pr(NA = Yes) = 0.7$ $\Pr(NA = No) = 0.3$
Non-Compliant licensed Software	$\Pr(NC = Yes) = 0.5$ $\Pr(NC = Yes) = 0.5$

Table 7.1: Prior Probabilities for each category

Begin of Table	
Worker Types	Probability
Expert Worker	$p\,(EW = TRUE \mid Actual = TRUE)$=0.65 AND $p\,(EW = FALSE \mid Actual = TRUE)$=0.35
Biased Worker	$p\,(BW = FALSE \mid Actual = TRUE)$=0.8 AND $p\,(BW = TRUE \mid Actual = TRUE)$=0.2
Random Spammer	$p\,(RS = TRUE \mid Actual = TRUE\,or\,FALSE)$=0.5 AND $p\,(RS = FALSE \mid Actual = TRUE\,or\,FALSE)$=0.5

Uniform Spammer	$p\,(US = TRUE \mid Actual = TRUE\,or\,FALSE)$=0.9 AND $p\,(US = FALSE \mid Actual = TRUE\,or\,FALSE)$=0.1
Adversarial Colluded Worker	$p\,(ACL = TRUE \mid Actual = TRUE)$=0.2 $p\,(ACF = TRUE \mid ACL = TRUE)$=0.9 AND $p\,(ACL = FALSE \mid Actual = TRUE)$=0.8 $p\,(ACF = FALSE \mid ACL = FALSE)$=0.9
Non-Adversarial Colluded Worker	$p\,(NACL = TRUE \mid Actual = TRUE)$=0.8 $p\,(NACF = TRUE \mid NACL = TRUE)$=0.9 AND $p\,(NACL = FALSE \mid Actual = TRUE)$=0.2 $p\,(NACF = FALSE \mid NACL = FALSE)$=0.9

Table 7.2: Probability for different worker types

We conducted various experiments to observe how the accuracy of Majority voting and Expectation Maximization algorithm varies with different types of workers. Based on our experiment, we concluded

(a) **Accuracy vs No. of experts**: The aim of the experiment was to find how accuracy varies with no of experts. In this experiment no of tasks was fixed to 100 and the number of experts varied. First 1-20 experts varied in interval of 1. Later for 20-40 experts we increased the interval to 5. Based on the observations we came to the conclusion that for expert workers, both algorithms have similar (shown in Fig. 7.1).

(b) **Accuracy vs No. of Tasks**: In this experiment we observed how accuracy varies with number of tasks. The number of experts were fixed to 5. The number of tasks were varied with range from 100 to 400 in interval of 50. Based on the observations we came to the conclusion that EM performs best for Trojan, worst for Non-compliant (shown in Fig. 7.2).

(c) **Accuracy vs No. of Biased Worker**:The aim of this experiment was to find how accuracy varies with the number of biased workers. For a fixed number of 100 tasks we varied the number of biased workers. Out of total

10 workers, first we kept one worker as biased and remaining 9 to be expert workers. Followed by 2 workers as biased and remaining eight as expert and so on. At last nine workers were biased workers and one as expert worker. We observed that for Biased workers, EM performs better than MV (shown in Fig. 7.3).

(d) **Accuracy vs No. of Random Spammer**: We conducted this experiment in order to find how accuracy varies with number of random spammers. For a fixed number of 100 tasks we varied the number of random spammers out of total 10 workers. First we observed the accuracy when there was one random spammer and remaining nine were expert workers. Later two workers were random spammers and remaining eight were expert workers and so on. Finally nine workers were random spammers and remaining one was an expert worker. Based on this experiment we came to the conclusion that for Random spammers, both EM and MV are equally affected (shown in Fig. 7.4)

(e) **Accuracy vs No. of Uniform Spammer**: Similarly the aim of this experiment was to find how accuracy varies with number of uniform spammers. For a fixed number of 100 tasks we varied the number of uniform spammers out of total 10 workers. First we observed the accuracy when there was one uniform spammer and remaining nine were expert workers. Later two workers were uniform spammers and remaining eight were expert workers and so on. Finally nine workers were uniform spammers and remaining one was an expert worker. Based on this experiment we came to the conclusion that for Uniform spammers, both EM and MV are equally affected (shown in Fig. 7.5).

(f) **Accuracy vs No. of Adversarial Colluded**: In this experiment we observed how accuracy varies with number of adversarial colluded workers. For a fixed number of 100 tasks we varied the number of adversarial colluded workers. Out of total 10 workers, first we kept two workers as adversarial colluded and remaining eight to be expert workers. Followed by 3 workers as adversarial colluded and remaining seven as expert and so on. At last nine workers were adversarial colluded workers and one as expert worker. Based on our observation we came to the conclusion that for Adversarial Colluded workers, EM is more affected than MV (shown in Fig. 7.6).

(g) **Accuracy vs No. of Non-Adversarial Colluded**: Similarly the aim of this experiment was to find how accuracy varies with number of non-

adversarial colluded workers. For a fixed number of 100 tasks we varied the number of non-adversarial colluded workers out of total 10 workers. First we observed the accuracy when there were two non-adversarial colluded workers and remaining eight were expert workers. Later three workers were non-adversarial colluded workers and remaining seven were expert workers and so on. Finally nine workers were non-adversarial colluded workers and remaining one was an expert worker. Based on our observation we came to the conclusion that for Non-adversarial Colluded workers, both algorithms have similar performance (shown in Fig. 7.7).

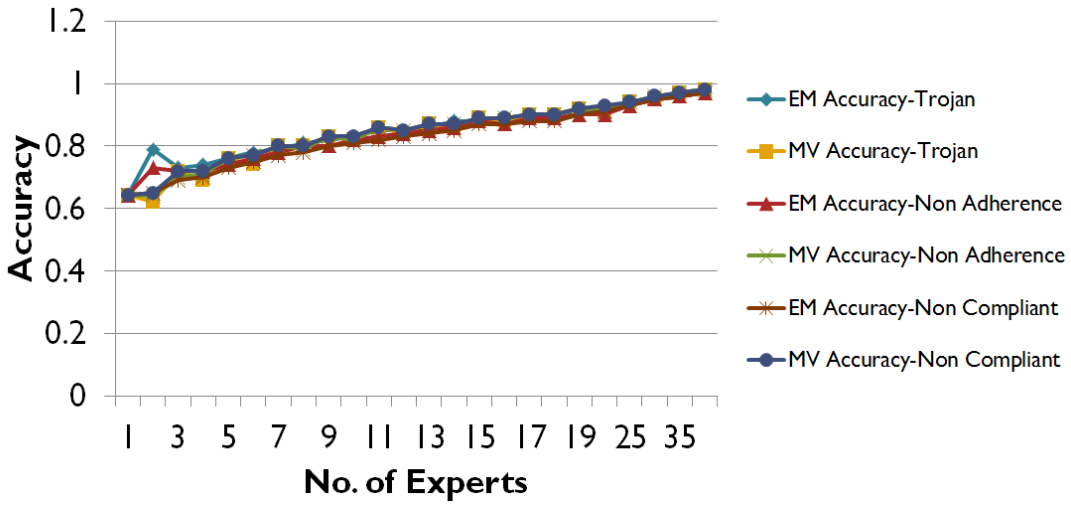

Figure 7.1: Accuracy vs Number of Experts

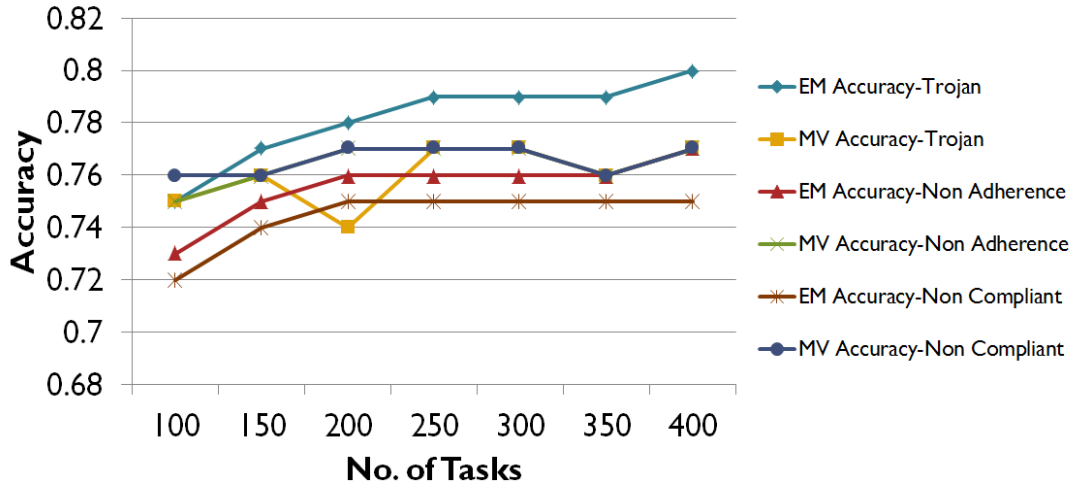

Figure 7.2: Accuracy vs Number of Tasks

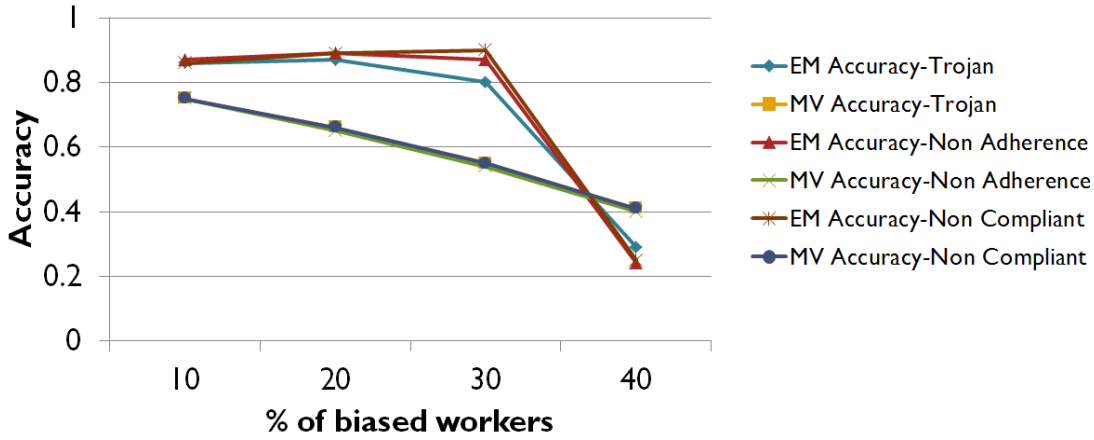

Figure 7.3: Accuracy vs % of Biased workers

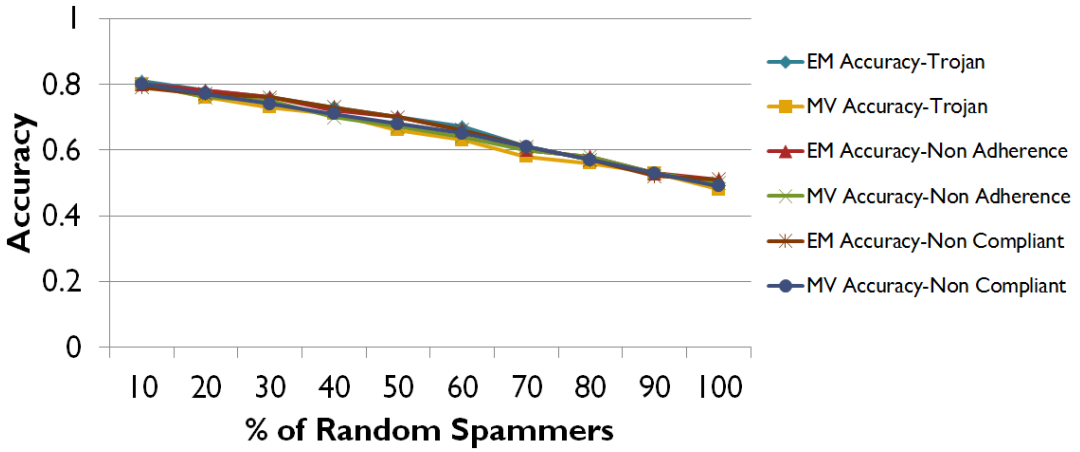

Figure 7.4: Accuracy vs % of Random Spammers

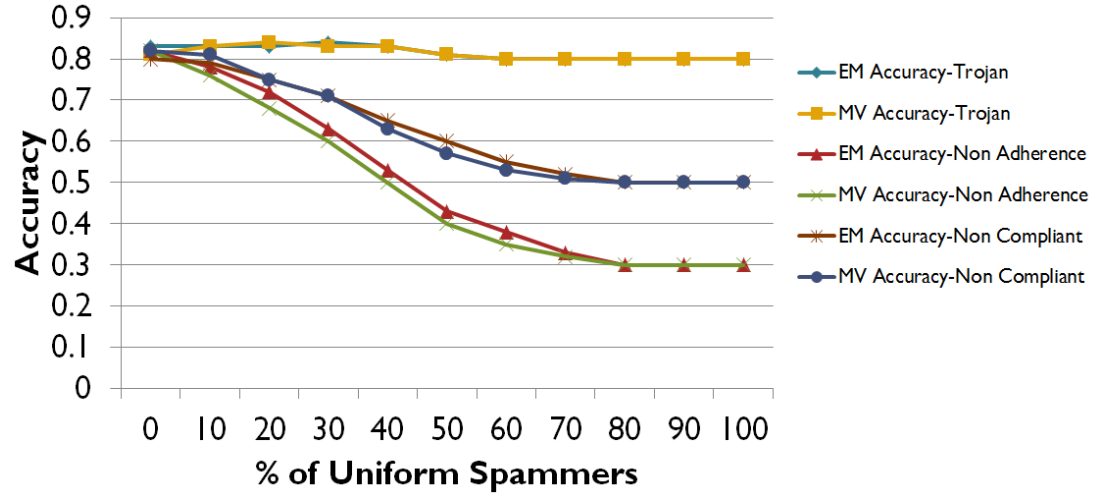

Figure 7.5: Accuracy vs % of Uniform Spammers

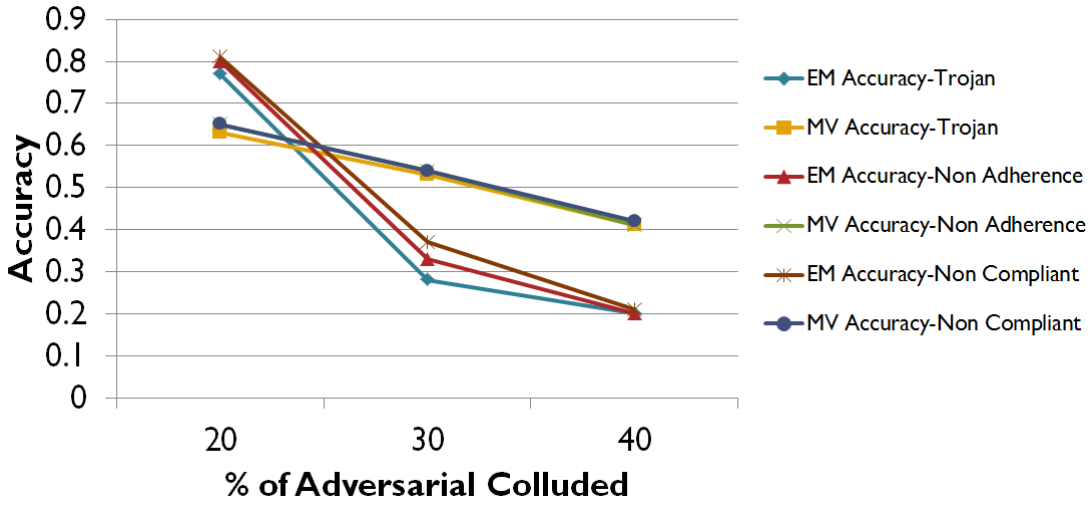

Figure 7.6: Accuracy vs % of Adversarial Colluded workers

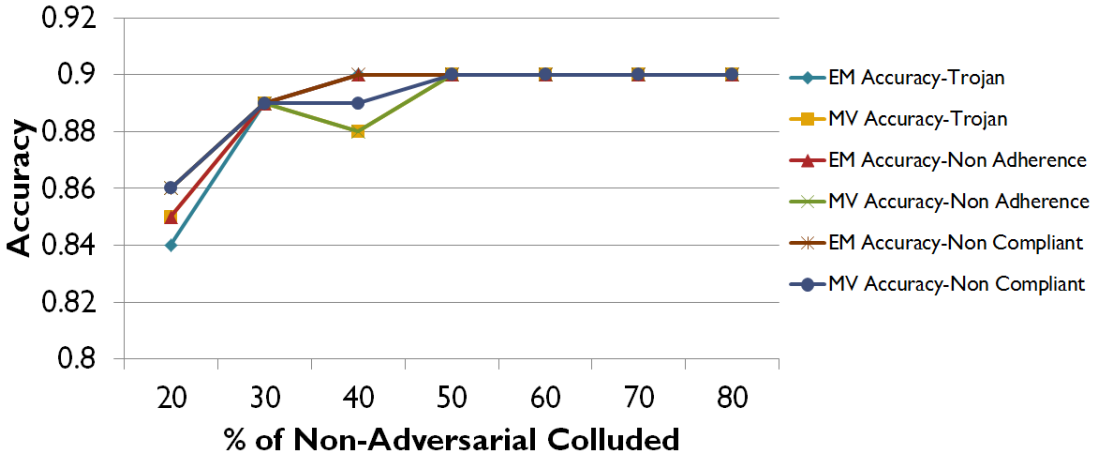

Figure 7.7: Accuracy vs % of Non-Adversarial Colluded workers

8

Conclusion

Through a review of related work, we have developed a taxonomy of the issues in crowdsourcing with respect to trustworthiness of the crowd in Software development context. We have presented the existing techniques, both from an algorithmic options and process options. Furthermore, we have evaluated the Peer review mechanism through a simulation of the crowd behaviour. Our results show that accuracy of both Majority voting and Expectation Maximization Algorithm are affected by different types of workers in crowdsourcing and skewed prior probabilities.

9

Future Work

The work presented in this thesis report is limited to the trustworthiness of crowd. In future, we will extend our work to trustworthiness of other two crowdsourcing entities i.e. platform and crowdsourcer. In addition to this, we have proposed a peer review mechanism to mitigate the three issues of Crowdsourcing Software development. Our future work will address the other four issues. As we have seen in our results that accuracy of both Majority voting and Expectation Maximization algorithm get affected by workers' type, the future work will be prior detection of these workers.

References

[1] https://www.topcoder.com/community/how-it-works/. 1

[2] https://www.upwork.com/i/howitworks/freelancer/. 1

[3] https://www.freelancer.in/info/how-it-works.php. 1

[4] Anurag Dwarakanath, Upendra Chintala, N C Shrikanth, Gurdeep Virdi, Alex Kass, Anitha Chandran, and Sanjoy Paul. **CrowdBuild: A Methodology for Enterprise Software Development using Crowdsourcing**. 1, 9, 30

[5] Nicolas Stefanovitch, Aamena Alshamsi, Manuel Cebrian, and Iyad Rahwan. **Error and attack tolerance of collective problem solving: The DARPA Shredder Challenge**. *EPJ Data Science*, **3**(1):1–27, 2014. 2, 13, 15

[6] http://www.crowdsourcing.org/editorial/when-crowdsourcing-goes-wrong/17996. 3

[7] Paul Ohm. **Broken promises of privacy: Responding to the surprising failure of anonymization**. *UCLA Law Review*, **57**:1701, 2010. 3, 4

[8] http://www.nytimes.com/2006/08/09/technology/09aol.html?pagewanted=all&_r=1&. 3

[9] Stephen M Wolfson and Matthew Lease. **Look before you leap: Legal pitfalls of crowdsourcing**. *Proceedings of the American Society for Information Science and Technology*, **48**(1):1–10, 2011. 3, 12, 15, 26

[10] Jeff Howe. **The rise of crowdsourcing**. *Wired magazine*, **14**(6):1–4, 2006. 5, 6, 11

[11] Daren C Brabham. **Moving the crowd at iStockphoto: The composition of the crowd and motivations for participation in a crowdsourcing application**. *First monday*, **13**(6), 2008. 5

[12] http://www.tcs.com/SiteCollectionDocuments/White%20Papers/Reimagining-enterprise-innovation-crowdsourcing-1114-1.pdf. 5, 9

[13] http://blog.designcrowd.co.in/article/202/crowdsourcing-is-not-new--the-history-of-crowdsourcing-1714-to-2010. 5

[14] Mahmood Hosseini, Keith Phalp, James Taylor, and Raian Ali. **The four pillars of crowdsourcing: A reference model**. In *Research Challenges in Information Science (RCIS), 2014 IEEE Eighth International Conference on*, pages 1–12. IEEE, 2014. 6, 12, 14

[15] Karim R Lakhani, David A Garvin, and Eric Lonstein. **Topcoder (a): Developing software through crowdsourcing**. *Harvard Business School Case*, **610**:032, 2010. 6, 28, 29

[16] http://www.lionbridge.com/files/2012/11/Lionbridge-White-Paper_The-Crowd-in-the-Cloud-final.pdf. 6, 7

[17] Klaas-Jan Stol and Brian Fitzgerald. **Two's company, three's a crowd: a case study of crowdsourcing software development**. In *Proceedings of the 36th International Conference on Software Engineering*, pages 187–198. ACM, 2014. 9, 10, 12, 15

[18] Anand Kulkarni, Matthew Can, and Björn Hartmann. **Collaboratively crowdsourcing workflows with turkomatic**. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1003–1012. ACM, 2012. 9

[19] Enrique Estellés-Arolas and Fernando González-Ladrón-de Guevara. **Towards an integrated crowdsourcing definition**. *Journal of Information science*, **38**(2):189–200, 2012. 11, 14

[20] Maja Vuković. **Crowdsourcing for enterprises**. In *Services-I, 2009 World Conference on*, pages 686–692. IEEE, 2009. 12, 15

[21] K-j Stol and Brian Fitzgerald. **Research Protocol for a Case Study of Crowdsourcing Software Development**, 2014. 12

[22] Brian Fitzgerald and Klaas-Jan Stol. **The Dos and Donts of Crowdsourcing Software Development**. In *SOFSEM 2015: Theory and Practice of Computer Science*, pages 58–64. Springer, 2015. 12

[23] Rafael Prikladnicki, Leticia Machado, Erran Carmel, and Cleidson RB de Souza. **Brazil software crowdsourcing: a first step in a multi-year study**. In *Proceedings of the 1st International Workshop on CrowdSourcing in Software Engineering*, pages 1–4. ACM, 2014. 12, 15

[24] Shkodran Zogaj, Ulrich Bretschneider, and Jan Marco Leimeister. **Managing crowdsourced software testing: a case study based insight on the challenges of a crowdsourcing intermediary**. *Journal of Business Economics*, **84**(3):375–405, 2014. 12, 15

[25] Mark A Liberstein. **Crowdsourcing: Understanding the Risks**. *NYSBA Inside*, **30**(2):34–38, 2012. 12, 15, 25, 26

[26] Joachim Henkel. **Selective revealing in open innovation processes: The case of embedded Linux**. *Research policy*, **35**(7):953–969, 2006. 13, 15

[27] Lav R Varshney, Aditya Vempaty, and Pramod K Varshney. **Assuring privacy and reliability in crowdsourcing with coding**. In *Information Theory and Applications Workshop (ITA), 2014*, pages 1–6. IEEE, 2014. 13, 16

[28] FRIEND OR FOE. **Open-Source Security Assessment**. 13, 16

[29] Ujwal Gadiraju, Ricardo Kawase, Stefan Dietze, and Gianluca Demartini. **Understanding malicious behavior in crowdsourcing platforms: The case of online surveys**. In *Proceedings of CHI*, **15**, 2015. 13, 16

[30] AR Balamurali. **Can the Crowd be Controlled?: A Case Study on Crowd Sourcing and Automatic Validation of Completed Tasks based on User Modeling**. 13, 16

[31] Alexander Philip Dawid and Allan M Skene. **Maximum likelihood estimation of observer error-rates using the EM algorithm**. *Applied statistics*, pages 20–28, 1979. 13, 16, 32

[32] Padhraic Smyth, Usama Fayyad, Michael Burl, Pietro Perona, and Pierre Baldi. **Inferring ground truth from subjective labelling of venus images**. 1995. 13, 16

[33] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. **Learning from crowds**. *The Journal of Machine Learning Research*, **11**:1297–1322, 2010. 13, 16, 32

[34] Jing Wang, Panagiotis G Ipeirotis, and Foster Provost. **Managing crowdsourcing workers**. In *The 2011 winter conference on business intelligence*, pages 10–12, 2011. 13, 16

[35] Jing Wang, Panagiotis G Ipeirotis, and Foster Provost. **Quality-based pricing for crowdsourced workers**. 2013. 13, 16, 31, 32

[36] Matteo Venanzi, Alex Rogers, and Nicholas R Jennings. **Trust-based fusion of untrustworthy information in crowdsourcing applications**. In *Proceedings of the 2013 international conference on autonomous agents and multi-agent systems*, pages 829–836. International Foundation for Autonomous Agents and Multiagent Systems, 2013. 13, 16

[37] Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier R Movellan, and Paul L Ruvolo. **Whose vote should count more: Optimal integration of labels from labelers of unknown expertise**. In *Advances in neural information processing systems*, pages 2035–2043, 2009. 14, 17

[38] Aashish Sheshadri and Matthew Lease. **Square: A benchmark for research on computing crowd consensus**. In *First AAAI Conference on Human Computation and Crowdsourcing*, 2013. 14, 17, 32

[39] Nguyen Quoc Viet Hung, Nguyen Thanh Tam, Lam Ngoc Tran, and Karl Aberer. **An evaluation of aggregation techniques in crowdsourcing**. In *Web Information Systems Engineering–WISE 2013*, pages 1–15. Springer, 2013. 14, 17, 32

[40] Gabriella Kazai, Jaap Kamps, and Natasa Milic-Frayling. **Worker types and personality traits in crowdsourcing relevance labels**. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1941–1944. ACM, 2011. 14, 17

[41] Vikas C Raykar and Shipeng Yu. **Ranking annotators for crowdsourced labeling tasks**. In *Advances in neural information processing systems*, pages 1809–1817, 2011. 14, 17

[42] Ashiqur R KhudaBukhsh, Jaime G Carbonell, and Peter J Jansen. **Detecting Non-Adversarial Collusion in Crowdsourcing**. In *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014. 14

[43] Ece Kamar, Severin Hacker, and Eric Horvitz. **Combining human and machine intelligence in large-scale crowdsourcing**. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 467–474. International Foundation for Autonomous Agents and Multiagent Systems, 2012. 14, 17, 32

[44] `http://www.networkworld.com/article/2187134/software/open-source-code-libraries-seen-as-rife-with-vulnerabilities.html`. 22

[45] `http://www.ideaconnection.com/blog/2014/04/how-to-fix-crowdsourcing/`. 23

[46] `https://www.youtube.com/watch?v=BB9iaGbBpBI`. 28, 29

[47] `http://www.veracode.com/security/malicious-code`. 29

[48] `http://appirio.com/category/resource/crowdsourcing-for-enterprise-it/`. 29

[49] `www.sonarqube.org/features/`. 29

[50] `https://www.blackducksoftware.com/solutions/code-scanning-and-code-matching`. 29

[51] `http://ninka.turingmachine.org/`. 29

[52] Armijn Hemel, Karl Trygve Kalleberg, Rob Vermaas, and Eelco Dolstra. **Finding software license violations through binary code**

clone detection. In *Proceedings of the 8th Working Conference on Mining Software Repositories*, pages 63–72. ACM, 2011. 29

[53] Benjamin Fung, Ke Wang, Rui Chen, and Philip S Yu. **Privacy-preserving data publishing: A survey of recent developments**. *ACM Computing Surveys (CSUR)*, **42**(4):14, 2010. 29

[54] Sai Wu, Xiaoli Wang, Sheng Wang, Zhenjie Zhang, and Anthony KH Tung. **K-anonymity for crowdsourcing database**. *Knowledge and Data Engineering, IEEE Transactions on*, **26**(9):2207–2221, 2014. 29

[55] Carliss Y Baldwin and Joachim Henkel. **Modularity and intellectual property protection**. *Strategic Management Journal*, 2014. 29

[56] Joachim Henkel, Carliss Baldwin, and Willy C Shih. **IP modularity:: profiting from innovation by aligning product architecture with intellectual property.** *California management review*, **55**(4):65–82, 2013. 29

[57] Joachim Henkel, Simone Schöberl, and Oliver Alexy. **The emergence of openness: How and why firms adopt selective revealing in open innovation**. *Research Policy*, **43**(5):879–890, 2014. 29

[58] `http://eprints.soton.ac.uk/368451/13/__userfiles.soton.ac.uk_Users_slb1_mydocuments_Venanzi%20-%20Thesis.pdf`. 31, 32

[59] `http://hcjournal.org/ojs/index.php?journal=jhc&page=article&op=download&path%5B%5D=22&path%5B%5D=44`. 33

[60] `https://en.wikipedia.org/wiki/Bernoulli_distribution`. 35

[61] Jeff Howe. *Crowdsourcing: How the power of the crowd is driving the future of business*. Random House, 2008.

[62] Eric Schenk and Claude Guittard. **Crowdsourcing: What can be Outsourced to the Crowd, and Why**. In *Workshop on Open Source Innovation, Strasbourg, France*, 2009.

[63] Nguyen Hoang Thuan, Pedro Antunes, and David Johnstone. **Factors influencing the decision to crowdsource**. In *Collaboration and Technology*, pages 110–125. Springer, 2013.

[64] Aniket Kittur, Jeffrey V Nickerson, Michael Bernstein, Elizabeth Gerber, Aaron Shaw, John Zimmerman, Matt Lease, and John Horton. **The future of crowd work**. In *Proceedings of the 2013 conference on Computer supported cooperative work*, pages 1301–1318. ACM, 2013.

[65] Panagiotis G Ipeirotis, Foster Provost, and Jing Wang. **Quality management on amazon mechanical turk**. In *Proceedings of the ACM SIGKDD workshop on human computation*, pages 64–67. ACM, 2010.

[66] Daniel M German, Yuki Manabe, and Katsuro Inoue. **A sentence-matching method for automatic license identification of source code files**. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*, pages 437–446. ACM, 2010.

[67] Chris Wysopal, Chris Eng, and Tyler Shields. **Static detection of application backdoors**. *Datenschutz und Datensicherheit-DuD*, **34**(3):149–155, 2010.