Workshop Preprints

WAWSE 2015

The Workshop on Alternative Workforces for Software Engineering

co-located with APSEC 2015
New Delhi, December 1st, 2015

Editors:

Alpana Dubey, Accenture Technology Labs Bangalore, India
Elisa Celis, EPFL Lausanne, Switzerland
Table of Contents

Crowdsourcing for API documentation: A Preliminary Investigation
Allahbaksh M. Asadullah and Shilpi Jain

3

Simulation of consensus based approaches to mitigate the challenges in
Crowdsourcing
Kumar Abhinav, Shrikanth N. C. and Anurag Dwarakanath

9

An Exploratory Study of Ethical problems in Digital Service and Engineering
Project Management
Santosh K. More, Julian M. Bass and Farid Meziane

12

People Management Issues in Scrum from COBIT Perspective
Necmettin Ozkan

14

Case Study of Project Outcome Prediction for an IT Vendor
Tomoyuki Kawamura and Kenichi Takano

15

Crowdsourced Testing for Enterprises: Experiences
Prahlad Rao, Alpana Dubey and Gurdeep Virdi

16

Quality Based Software Project Staffing and Scheduling with Budget and
Deadline
Dongwon Seo, Donghwan Shin and Doo-Hwan Bae

18

Organization

Alpana Dubey, Accenture Technology Labs Bangalore, India
Elisa Celis, EPFL Lausanne, Switzerland
Crowdsourcing for API documentation: A Preliminary Investigation

Allahbaksh M. Asadullah
Infosys Labs, Infosys Ltd.
Bangalore, India
Email: allahbaksh.asadullah@infosys.com

Shilpi Jain
FORE School of Management
New Delhi, India
Email: shilpijain@fsm.ac.in

Abstract—Developers and researchers have been using crowdsourcing in a variety of fields related to software development and software engineering. Crowd based documentation is another yield of crowdsourcing where the coder community or workers document the software. In the present work, we have analyzed how crowdsourcing can be used for an API documentation. The study is based on the fact that good programmers write descriptive variables and method names and continue to do so for future references. A variety of tools such as Amazon Mechanical Turk, ETurk and DocIt were evaluated for the purpose. Among these, DocIt and ETurk were built in-house. The evaluation of the documentation was performed by experienced coders. This is a preliminary experiment which was performed in a controlled environment. Results were encouraging and help us to determine that in future crowd based documentation might help to reduce time to market and improve software quality.

Keywords: Crowdsourcing, API Documentation, Amazon Mechanical, Turk, DocIt, E-Turk

I. INTRODUCTION

Development and maintenance of large software systems\(^1\) remains a difficult and daunting problem for any project team. Based on the studies, the percentage of effort goes in requirements phase is 15-20\%, analysis and design is 15-20\%, development effort is 25-30\%, system testing is 15-20\%, and maintenance effort across the software development life cycle is typically 5-10\%. Until early 90s, in a conventional software development, the modification (adding or deleting a module or functionality) of the code had been a great challenge. This paradigm shifted with the emergence of object oriented programming and open source software development that supports modular programming and library reuse. By breaking down the problem into multiple tasks, different developers\(^2\) can work in parallel. Modular programming allows distributed development that shorten the development and documentation time. Moreover, individual modules are easier to design, implement, and test.

Besides source code, several documents accompany a software system, because there is a possibility that the source code perhaps is not sufficient to convey the objective of the project. Hence, moderate to large sized development projects, irrespective of application, generate a large amount of associated documentation such as documentation of code, algorithms, application programming interface (API), UML diagrams, sequence diagrams, class diagrams, design documents etc. The set of these components are popularly known as software artifacts or technical documents.

Usually, the process of documentation is elaborate and requires a significant amount of effort. A substantiate documentation reduces the maintenance work and further improves the productivity and reusability of the code.

Many developers believe that the documentation doesn’t require a high intelligent quotient and it is a waste of time and effort. They consider that the code written by them is sufficient and self-explanatory. On the other hand, another set of programmers appreciate its importance but tend to avoid due to paucity of time or limited resources. Segal [1] observed that professional developers do not volunteer to produce code documents. If necessary, they will write a page or two as a formality. Brief and inappropriate documentation is a matter of concern [2]. To overcome this limitation, new programmers can leverage the abundant repository of unofficial API documents generated by API users on community portals. This unofficial documentation is popularly known as crowd documentation because it is generated from crowdsourcing [3]. These documents have sufficient coverage for practical usages. Blog post and ‘stack overflow’ are two types of crowd documentation that have highest coverage ratios [4]. Latoza et al. [5] describe the advantages of crowd based development in their work. Crowdsourcing introduces agility in the component development of large-scale industrial projects, especially when dealing with changes in API.

Besides having several benefits of API documentation there are not many tools available in the market which can perform

---

\(^1\) At Infosys Ltd., any software project’s code that exceeds 50k line of code (LOC) falls in the category of large software system (project).

\(^2\) Please note that in the current study, words like developer, coder, programmer, and worker are used as synonyms.
this task efficiently. Hence for the purpose, we have developed two prototypes. In the current study we propose to evaluate and compare the performance of Amazon Mechanical Turk (also known as MTurk) with two in-house developed tools (ETurk and DocIt). The other objective is to test whether they can produce consumable APIs.

II. BACKGROUND LITERATURE

API documentation or Programmers documentation, is a deliverable of technical writing in which a technical writer develops instructions about how to effectively use a software API, hardware (SCPIs) or web-API [6]. In addition to established coders, it is even useful for new coders as it helps them to understand and learn best practices and implementation details. Among many, API documentation is a subset of software documentation. It is often embedded within the source code like Javadoc comments in Java. API documentation is written in plain language which requires a thorough understanding of the API, its arguments, its return type and the languages and interface it supports. The text is often supported by images or hyperlink to other elements of the source code. API usability is important because of the spread of APIs in almost every application domain [7]. Among the factors that affect API usability, is the lack of diverse API documents [8].

The extant literature noticed that unlike open source projects, API documents are rarely updated in industrial setting. In open source development, API documentation for deprecated code is taken seriously and this task is mandatory before they release candidate of the library. A set of specialized tools are available for source code documentation, but rarely used [5]. Typically, software engineers decide on their own what kind of documentation is worthwhile to produce and maintain, and adopt selective tools which help them for the purpose [9] [10] [11]. This has been reported that software engineers tend to ignore complex and time-consuming documentation [11]. Literature shows that scientific documentation does not follow recommended standards proposed by SEBoK (www.sebokwiki.org)

Previous studies have shown that a majority of Java developers prefer to use Javadoc [2] [1] instead of APIs. In another research, Latoza et al. [5] explained how a complex task can be decomposed into set of smaller tasks in crowd development. However, they did not discuss or refer any specific case studies for the same. Kittur et al. [12] showcased how an article writing can be achieved by crowdsourcing. Jiau and Yang [13] studied the API documentation of few open source project like GWT, SWT and Swing. They found that the quality of documentation produced in open source forums was of better quality. To the best of our knowledge, we could not find any research study that specifically talks about the development of API documents through crowdsourcing.

A variety of web platforms are available for software crowdsourcing such as Amazon Mechanical Turk (also known as MTurk). MTurk is one of the sites of Amazon Web Services. It is a crowdsourcing Internet marketplace that enables individuals and businesses to collaborate and perform complex tasks that computers are unable to conclude. A user of Mechanical Turk can be either a "Worker" (employee) or a "Requester" (employer). MTurk is one of the sites of Amazon Web Services. Employers post jobs known as HITs (Human Intelligence Tasks), such as choosing the best among several photographs of a storefront, writing product descriptions, or identifying performers on music CDs. Workers can then browse among existing jobs and complete them for a monetary payment set by the employer. On the basis of jobs performed, Turk creates qualification profile for the workers.

MTurk encounter certain limitations. In MTurk, it was difficult for coders to highlight the syntax and due to which code comprehensibility was a challenge. Second limitation, the tool is limited to open source applications and doesn’t support proprietary software. Therefore, for internal enterprise applications, we developed a similar tool named as E-Turk with enhanced features. The tool was tested with a team of Java coders and it was observed that many developers were still finding it difficult to associate the source code with required class file (which might me parent class, implementations etc.). They suggested that it would be helpful to give information about methods and classes that are doing similar nature of work. Hence we added a feature that can develop a connection between similar classes & methods semantically. The new version was renamed as DocIt. To confirm the utility of DocIt and E-Turk over MTurk we performed an experiment based study in industrial setting.

III. RESEARCH DESIGN

The research was conducted in two main phases:

**Phase 1:** Preliminary investigation using surveys with software programmers.

**Phase 2:** Experiment study where software developers wrote API documentation on MTurk and customized prototypes.

Limited time, dynamic requirements, confined research group, and small user base are some of the reasons cited for the absence of documentation [10] [2] [1]. Several researchers raised this concern but how to achieve the goal remain unanswered. To estimate the root cause, in phase I of the research, we conducted a preliminary survey with developer’s community in the Silicon Valley of India, i.e. Bangalore, India. The objective was to understand how frequently and precisely our coders document the code, which is the preferred tool for the purpose. The survey was designed by experts which had 15 questions to capture the responses. The survey was sent to
approximately 127 Java coders with a minimum coding experience of 3 years and maximum of 5 years. Out of 127, only 95 responded, 2 responses were dropped because of incompleteness and finally 93 responses (63 males and 30 females) constituted the final sample size.

A. Phase I Findings

The results indicated that 46.4% of respondents are in habit to include comments before defining any function, method, class or a variable, while 34.7% do it only when they feel it is needed (when writing a complex algorithm or function), and rest i.e. Approximately 19% do it rarely as they consider that their code is self-explanatory and documenting it is a waste of time. In response to another question, 86% developers reported that they follow proper naming conventions and give meaningful names. These findings led the foundation of our research as our research is based on the fact that programmers write descriptive method and variable names. Further, 83% of developers deliberated that appropriately documented source code facilitate in understanding of the code and leads to reduction in code comprehension time, 33% think documentation writing is a dull & boring job, and 16% of developers think that API documentation is not needed. Since projects have stricter deadlines, their focus remains on adding features and deliver. Almost every coder reported that documentation is a time consuming process and in most of the cases hard deadlines doesn’t provide any room for it. Another culprit is ‘frequent changes in requirements’ which completely shifts the focus of a programmer from documentation.

B. Experiment Study

Mechanical Turk: Mechanical Turk (also known as MTurk) is a decent platform but generic in nature. It lack certain special features. In MTurk, the task submitted as a single HIT is difficult to comprehend because of the code dependency. The HIT is an independent work unit. The main advantage of MTurk is that it offers insights into how one should design the platform, what are the key problem that needs to be addressed which a developer may face.

A CSV file from an open source project, Apache Drill3 was selected to upload in MTurk. The worker had to pass the Java programming test before getting assigned to any live project. Once passed, they attempt to HIT. Every HIT included a fully qualified name of the class, a method and to be documented, and class body. The workers were expected to write the API documentation (Java doc) of the method in the text area provided. We have included these three fields so that the workers can understand the package hierarchy from the class name, document the method from the method body provided and use the class body as a reference to identify the relationship of the method from other methods present in the class. Each time the HIT is answered by a specified number of users, the HIT list is updated dynamically with the pool of pending hits. HITs are exhibited to users in random fashion.

Enterprise Turk (E-Turk): E-Turk was designed by mimicking the MTurk and has several modules such as user registration, user modules etc. The user interface of E-Turk is easy to use and almost similar to Mechanical Turk with an exception. Unlike MTurk, in E-Turk, the users were allowed to view all HITs, the source code was rendered with highlighted syntax which makes it easier to comprehend, and submission of the task is easy in comparison to Amazon Mechanical Turk. In MTurk we faced difficulties in posting of certain tasks (HITs) where the code snippet had special characters.

DocIt: This tool was superset of E-Turk. The tool can browse, search and navigate the source code online. Also the tool featured semantic search and gives information on related method and classes. This resembled Eclipse in many aspects.

The UI for DocIt was borrowed from another API explorer tool. When a developer clicks on any previously documented method it gives the latest document which the developer could edit to improve. If there is no documentation for the method then the developer may add the same.

The experiment study with the second platform gave us more insights into the problems faced by the crowd while documenting the source code. For examples, developer could frequently observe the related classes / interfaces. They experienced different patterns in the source code to find out how an object could have created (in case of Factory, Abstract Factory or Prototype Design pattern). These features are fairly common and available in other IDEs. Hence, DocIt was designed to address those key issues.

Prior to the initiation of experiment, certain preparations were made. We followed systems approach and developed a process flow diagram (see Figure 1).

Version Control Plugin (VCP) – We designed a VCP that extracts latest source code from the version control system. Since Team Foundation Server (TFS) is widely used in the organizations, we designed VCP for the TFS. It is important to note here that one can design or extend VCP for any other version control system by using appropriate libraries such as JHG, JGit etc. This plugin pulls the source code from the
Version Control System which is then be acted by different components of the system.

Source Code Parser – The source code which is pulled from Version Control System is input to the Source Code Parser. The source code parser can parse the elements of the source code. For example, in Java it can tell about the various methods in the class, field variables and their initial values etc. The source code parser analyzes the source code which is a .java file and create an Abstract Syntax Tree (AST). The AST provides detailed view of the source code. The necessary information from AST is then extracted and is saved as Comma Separated File (CSV file) and an index file.

   Each .java file created more than 1 row in the CSV file. A row in a CSV file contains details of method. The column attributes are method body, class body, class full name (Package Name +Class Name), which means that for each method (in .java file), we will have a row entry in the CSV file.

   The CSV file and the index is input to the systems like Mechanical Turk, E-Turk and DocIt. Each row in the CSV file correspond to one unit of work which is called as HIT (in Amazon Mechanical Turk).

   Each HIT is independent of the other HIT and the tasks can be completed without any further detailed knowledge. There is no navigation facility available in Amazon Mechanical Turk and E-Turk. Based on our knowledge and other results, it was realized that navigation plays important role in source code comprehension. Consequently, this limitation was addressed in DocIt. Besides navigability, DocIt has additional functionality. It has rich interface that can browse and explore the source code effectively. DocIt uses two inputs: CSV file and Index. The index feature facilitates navigation.

Below is step by step process

1. We first extracted the source code elements and related artifacts from version control system (VCS).
2. The source code is then parsed and a CSV File and Index file is prepared.
3. These artifacts are provided to the crowd developers via Mechanical Turk, ETurk, and DocIt.
4. Based on the available source code and other information, developers wrote API documents.
5. The API documents were evaluated by a handful of experienced system architects.
6. Once the API document for particular method is approved, the source code was updated and committed to the version control.

IV. FINDINGS

In case of E-Turk and DocIt we did not conduct any test to check the knowledge of the developer, instead we handpicked a team of coders (Java) from Infosys. These programmers had scored more than 65% of marks in an internal Java exam. An Enterprise Java application was given for the documentation. They were free to choose their tool from the available 3 options. We advertised about these tools through several internal media channels. Nevertheless, MTurk wasn’t the preferred choice for documentation. Merely 30% of the HITs were resolved from the given list, and only 23 out of 237 (approx.10%) coders participated in the exercise. These findings were quite surprising hence, we asked the rest for the reasons through a semi structured questionnaire. Their responses are summarized in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>Lack of experience with MTurk</th>
<th>Tool Usability (issues with UI and framework)</th>
<th>Complicated HITs</th>
<th>Non-challenging HITs / Ambiguous HITs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>65%</td>
<td>72%</td>
<td>57%</td>
<td>77%</td>
</tr>
</tbody>
</table>

In case of Mechanical Turk, the workers had no other alternative to visualize additional methods which they can use to document the same class. This means that the person documenting the method will be given a random method mij from a class Ci where mij in Ci. Hence the worker has to read the new class and understand it which was time consuming. The HITs were independent of each other, therefore, there are chances that for every HIT worker may probably get a new class, which leads to waste of time of the worker writing the documentation. HITs are released in certain batches, hence there is no guarantee that all the methods of a single class will be released one at a time. About 60% of coders reported that the HITs did not interest them or motivate to respond.

In E-Turk, the results were comparatively better that MTurk. This could be because of two reasons: a. the coder community, tool, and application were internal resources (i.e. available within the organization setup), b. Since these applications were developed for the client, they were well written and followed industry standards of coding and formatting. All developers were experienced Java coders.

In case of DocIt, a cross reference to the classes and methods was available. This helped the developers in accessing more information about the methods and the classes. They could navigate easily to other classes and methods resulting in improved documentation. DocIt was used by 15 developers. We observed that the developers did navigate the source code and read related classes and methods before documenting the method. Many of the developers looked at the other artifacts which were presented in the tool based on the class the developer was browsing in the tool. For example, for a class called HousingLoan, the developers paid attention towards the related documents and look over the threaded discussions on AutomobileLoan. Almost 45% of the developers resorted to reading these descriptions and found them useful in API document preparations. From the logs of the tool, we found that developers spent a considerable amount of time towards...
understanding the classes like ILoan (interface), IInterest etc. which were related to the Loan class before writing the API document.

A. Validation of API Documents

Post experiment, a team of system architects and authors of the class files reviewed API documents. The documents were checked for completeness, context and precision. On an average coders wrote 6 lines per method in the API document and reviewers added 2.5 lines to the submitted API documentation. Three out of four reviewers did spend time to check whether the documentation conveyed intended meaning and relevant to the context. The average time per method spent for review was 7 to 8 minutes. These metrics are decent as successful open source projects follow the same metrics before their source code is committed to any VCS (Version Control System).

Our study was based on the fact that the available tools are insufficient to perform accurate API documentation. For validation, two other source code documentation tools (e.g., TwinText and Doc-O-Matic) were used which comprehend the source code and generates the API documentation. These tools doesn’t require any human intervention.

TwinText uses code comments with source code analysis to generate the API documentation. One can define the style of API documentation in TwinText. This tool has a limitation, API documentation produced here embed originator’s comments. Some of these comments were generic and written merely for understanding purposes (e.g. the code comments written by the developer merely to understand the internal code structure) which might not be worthwhile for code reader or developer who might use these API in future. The tool does not provide any consumable APIs.

Doc-O-Matic uses the source code as primary artifact and add additional information based on the domain through external inputs. This tools uses Java language semantics to identify the package name, member variables, and method names and then generates the API documentation. We used Doc-O-Matic on the Apache Drill project, and the output (API documentation) was inappropriate. The descriptions had random words and control characters. Our guess is that probably their NLP (Natural language Processor) was unable to generate meaningful sentences.

The objective of current study was to evaluate and compare the performance of DocIt with other similar tools available in the market and whether they can produce consumable APIs. We found that none of the available options could solve the purpose. Manually generated API documents were better and meaningful over automatic API documentation tools. Below are some of the metrics which we picked to check how much time and effort it takes to create API documentation.

V. CONCLUSION AND FUTURE WORK

Our preliminary investigation showed that software API documentation can be achieved by crowdsourcing. A variety of developers review API documentation and prepare the final output through code review system. Prior research confirms that the crowd documentation is dependent upon the paradigm of the programming language. The person writing the documentation should have the knowledge of programming paradigm and the domain of the project. Comparatively, developers spent lesser amount of time and effort in documenting a code where modules are interacting using APIs.

We conclude by making an observation that the development of API documentation by crowd sourcing saves time and effort. It further helps the software industry and academia to evolve and generate new software systems and innovate rapidly. We are further exploring additional artifacts that could be helpful in documentation. During the time of writing this paper, we are experimenting with unit test cases and version commit messages. These studies are still in progress.

REFERENCES


http://www.ptlogica.com/TwinText/
http://www.doc-o-matic.com/


Simulation of consensus based approaches to mitigate the challenges in Crowdsourcing

Kumar Abhinav  
IIIT Delhi, New Delhi  
kumar1365@iitd.ac.in

Shrikanth N.C  
Accenture Technology Labs, Bangalore  
shrikanth.n.c@accenture.com

Anurag Dwarakanath  
Accenture Technology Labs, Bangalore  
anurag.dwarakanath@accenture.com

Abstract—Crowdsourcing is an emerging area and has evolved as a powerful practice to leverage the collective intelligence of the crowd. It has been applied in various domains ranging from creative resolution of a problem to improving the business process using several platforms such as CrowdFlower, Freelancer and Amazon Mechanical Turk. Crowd is a creative workforce that has niche abilities to solve complex business challenges across various domains. It can be seen as an alternate workforce by participating in all phases of software development life cycle. However the common problem seen in crowdsourcing is the quality of the work performed by the crowd mostly due to the anonymity of the crowd member. In this work, we evaluated consensus based approach to assess the quality of the work done by the crowd through a simulation of crowd behavior. We also investigated the performance of these techniques for evaluating crowd members.

I. INTRODUCTION

Crowdsourcing Software development is a promising and emerging field. It acts as a platform where the crowd can perform the entire software development tasks given by crowdsourcer or requester. In a software engineering context, client may not have knowledge of the crowd who develops the software and is 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 [1]. Hence, owing to the anonymity of the crowd, evaluating the quality of the work of the crowd becomes a major challenge in crowdsourcing software development. There are different computational approaches in related literature to evaluate the submissions made by the crowd. In this paper we will discuss the following approaches which are commonly used in crowdsourcing:

1) 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. The past performance of the workers assesses the quality of the workers.

2) 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, workers’ quality can be assessed.

3) Consensus based Approach: This is the most common approach to determine the true response and in turn to assess the credibility of the crowd. 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.

In this paper, we adopted consensus based approach. The submissions are given to the crowd members to evaluate quality pertaining to the three issues- Trojan code, Non-adherence to best practices and Non-compliant licensed software. (Figure 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 [2]. In this paper, we will discuss two of these approaches.

1) Majority Voting (MV): This is the most common and simple consensus based method. 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.

2) 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 [3].

Due to lack of availability of real world datasets on which we can test performance of 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 while considering different types of crowd worker [4]. Each crowd worker is assumed to follow a Bernoulli distribution 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 Trojan code is very low with 0.2 probability, Non-adherence to best practices is very high with 0.7 and Non-compliant licensed software with 0.5 probability.

There can be different types of crowd workers in crowdsourcing. Mathematically, we have defined different types of
workers’ responses in a probabilistic manner. For example, probability of giving the correct answer for an expert worker is 0.65, for a biased worker it is 0.2.

1) **Expert worker**: expertise in the area with profound domain knowledge and the questions answered correct with a high probability.
   
   \[ p(\text{EW} = \text{TRUE} | \text{Actual} = \text{TRUE}) = 0.65 \]

2) **Biased worker**: intentionally gives incorrect answers.

   \[ p(\text{BW} = \text{FALSE} | \text{Actual} = \text{TRUE}) = 0.8 \]

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

   \[ p(\text{RS} = \text{TRUE} | \text{Actual} = \text{TRUE} \text{ or } \text{FALSE}) = 0.5 \]

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

   \[ p(\text{US} = \text{FALSE} | \text{Actual} = \text{TRUE} \text{ or } \text{FALSE}) = 0.9 \]

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

   \[ p(\text{ACL} = \text{TRUE} | \text{Actual} = \text{TRUE}) = 0.2 \]

   \[ p(\text{ACF} = \text{TRUE} | \text{ACL} = \text{TRUE}) = 0.9 \]

6) **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.

   \[ p(\text{NACL} = \text{TRUE} | \text{Actual} = \text{TRUE}) = 0.8 \]

   \[ p(\text{NACF} = \text{TRUE} | \text{NACL} = \text{TRUE}) = 0.9 \]

We conducted various experiments to observe how the accuracy of MV and EM algorithm varies with different types of workers. For all the experiments, we kept number of tasks as fixed to be 100, because this gives a high base of accuracy (based on our experiment) and varied the number of workers. We computed the average of each algorithm’s accuracy over 100 runs to obtain the results.

1) **Effect of Expert workers**: The experiment was conducted with number of experts varying from 1 to 40. The accuracy of both the algorithms increases with the increase in number of expert workers. Here both algorithms have similar performance (shown in Figure 2).

![Fig. 2: Accuracy vs Number of Experts](image)

Based on this result, we fixed the number of workers to 10 in all the subsequent experiments as this gives a high starting point of accuracy. As we increased the number of workers, accuracy increased but that came at a cost. Hence, there is a trade-off between cost and accuracy here.

2) **Effect of Biased Workers**: In this experiment, we varied the percent of biased workers from 10 to 40. We observed that for biased workers, EM performs better than MV as EM models the workers’ behavior by confusion matrix. Figure 3 depicts the result.

3) **Effect of Spammers**: In this experiment, we increased the percent of spammers to study how it affects accuracy of both the algorithm. Figures 4 and 5 demonstrate the effects of Random Spammers and Uniform Spammers respectively. In general, both EM and MV are equally affected by presence of spammers. The accuracy of both approaches decreases as the number of spammers increases.
In case of uniform spammers, it is clearly evident from the graph that skewed prior probabilities affect accuracy of the algorithm. The probability of having Trojan code is very low (0.2) and the uniform spammer marks it as "Not Trojan" with 0.9 probability. This response from spammer acts like response from an expert worker, which in turn increases accuracy.

4) **Effect of Adversarial colluded workers**: The effect of adversarial colluded workers is depicted in Figure 6. As we increase the percent of adversarial colluded workers, accuracy of both EM and MV decreases. Based on our observation we concluded that for Adversarial colluded workers, EM is more affected than MV.

5) **Effect of Non-Adversarial colluded workers**: The effect of non-adversarial colluded workers is depicted in Figure 7. As we increase the percent of non-adversarial colluded workers, accuracy of both EM and MV increases up to certain point and then it becomes constant. Here, both algorithms have similar performance.

Based on our experiment, we concluded accuracy of both Majority voting and Expectation Maximization Algorithm are affected by different types of workers in crowdsourcing and skewed prior probabilities.

![Fig. 3: Accuracy vs % of Biased workers](image)

![Fig. 4: Accuracy vs % of Random Spammers](image)

![Fig. 5: Accuracy vs % of Uniform Spammers](image)

![Fig. 6: Accuracy vs % of Adversarial Colluded workers](image)

![Fig. 7: Accuracy vs % of Non-Adversarial Colluded workers](image)

**REFERENCES**


An Exploratory Study of Ethical problems in Digital Service and Engineering Project Management

Santosh K. More
Computer Science and Software Engineering
University of Salford
Manchester, UK
S.More@edu.salford.ac.uk

Julian M. Bass
Computer Science and Software Engineering
University of Salford
Manchester, UK

Farid Meziane
Computer Science and Software Engineering
University of Salford
Manchester, UK

This research focuses on ethical issues in project management within India’s digital service industry, which we define [1] as information technology (IT), telecommunications, software development, consultancy and business process outsourcing (BPO). India is the leading digital service industry nation with 55% of the global market, including the highest market value (above US$150 billion). The digital services industry is comprised of over 75% staff below average age 35, with 5.8 million skilled staff employed [2]. Indian software exports are approximately US$75bn in 2014/15 with an additional US$100bn from the BPO service sector [3].

This research study aims to investigate the ethical problems in project management within the digital service industries in India. The main objective of study is to test the hypothesis that ethical problems are causing high staff attrition rates in the digital service industries in India.

To investigate our hypothesis we used a mixed method research approach using a quantitative practitioner survey questionnaire and to obtain exploratory qualitative information through semi-structured interviews [4]. Initially we sent more than 30 quantitative research questionnaire requests to practitioners and have received 12 responses to date. We have conducted 4 interviews using the quantitative questionnaire, so far. A semi-structured interview guide is under development to collect more information and in which we will triangulate responses from managers, employees and human resource department staff respectively to identify the ethical problems and challenges in digital service industries.

We have investigated the code of ethics developed by international professional bodies like (PMI [5], ACM [6], IEEE [7], and BCS [8] and also the Computer Society of India. The codes of ethics related to IT professionals, code of practice towards organizations, employees, clients and community and action against any members who breach the code as well as the complaints procedure and sequence of action [9]. Gender-based ethical behavior issues in digital service sector, have been investigated by Catalyst India, and concluded that women are getting less salary, as compared with men [10]. This is also one of the drawback and major issue in digital industry, as women have same intelligence as men and use their knowledge to complete the same tasks as men [10].

These preliminary investigations show that there are different ethical issues present in the digital service industries, that ethical issues are a real concern and that these problems are causing people to change their employer. In the past, people wanted to change their job for the better remuneration but nowadays they need sound ethical practices at their current workplace which is not available in the industry. The preliminary results of this study provide evidence of the following eight ethical problems: favoritism, immorality, partiality, enmity and hate, discrimination, gender difference, misuse of management position, internal project disputes and conflicts. Favoritism is the practice of giving unfair preferences to one person or group of people at the expense of another. We use immorality to describe the behavior that is dishonest or does not follow accepted operational standards. Partiality means unfair prejudice or bias in favor of person or group. Enmity and hate is the unethical action and deep-seated hatred among the individuals. Discrimination is the unjust treatment to individuals on the basis of their caste, ethnic group, economic status and religion. A gender differences refers to unfair treatment and inequality trial on the ground of gender. Misuse of management position describes the use available resources and powers improperly or incorrectly. Internal project disputes and conflicts is the psychological, physical and moral struggle within the group of people in the organization. Although it is opposition of needs, values and interests between people working together.

We have compared the code of ethics developed by Computer Society of India and our findings from practitioners in digital sector in India. We also investigated the code of ethics and professional conduct developed by Project Management Institution, USA (PMI) which includes vision and applicability, responsibility, respect, fairness and honest behavior code of practice [5]. We found evidence of several ethical issues which are described in the code of ethics towards individuals’ behavior towards organization, employees, investors and stakeholders. However, we also found evidence of several ethical issues that are not included in the professional code of conduct.

In future, we will conduct more interviews with participants from digital service industry for data collection. The long term goal of this research study is to improve project management decision making process by proposing recommendations that will, if implemented, reduce attrition, reduce ethical problems to make a healthy working environment in digital service industry.
REFERENCES


People Management Issues in Scrum from COBIT Perspective

Necmettin Ozkan
IT Governance Division
Turkiye Finans Participation Bank
Istanbul, Turkey
necmettin.ozkan@turkiyefinans.com.tr

While Scrum aims to maximize the business value of information technology (IT), COBIT (Control Objectives for Information and Related Technology) is to assure the business value of IT throughout the international set of generally accepted IT control objectives [1]. However, from the point of people management, Scrum and COBIT present different approaches in their essence:

- While COBIT is to standardize people to the processes, SCRUM relies on people and their creativity rather than processes [7].
- Instead of a command and control style of management in COBIT, Scrum encourages teams with the resources they need and then trust them to do their jobs well [6].

Albeit Scrum teams have a freedom inside the team, they one way or another still have an interaction with the remaining parts of organizations which have authorities over the same subject which is maximizing the value. Scrum should gain recognition throughout the organization, and be applied appropriately. Otherwise, conserving and protecting the natural structure and mechanism of the teams becomes a challenge.

The focus migrates from people centric management to product centric management by Scrum methods and the structure of Scrum shapes around the product concept. Line managers, who have primary responsibilities over people, disappear. However, still someone should watch over people who are prone to be forgotten somewhere in the product lines, aggressively designed for continuous and unremitting delivery.

Accountabilities and responsibilities of people management functions of teams should be addressed in Scrum. As a part of it, responsibilities of workload and resource capacity management among and inside the Scrum teams should be defined. Performance measurement and reward systems must be suitably designed [4] team based, where collective goals supersede individual accomplishments [8]. Moreover, career path development is a field to study for Scrum which provides a flat structure of organization rather than a hierarchy including steps to managerial positions.

For the teams that are expected to trust each other, the concept of codes of ethics plays a critical role for the success of agile methodologies. And, organizations should be aware of that it may take enormous effort, time, and patience to build a culture of trust and respect among the employees [4].

Documentation as useful artifacts for the backup of information is discouraged in Scrum [5]. Thus, much of the knowledge in agile development resides in the heads of the development team members [4].

Using an agile approach entails formidable responsibility on the client’s part [8]. The success of agile development relies on finding customers who are expected to be collaborative, representative, authorized, committed, and knowledgeable [3]. Great Scrum also needs great product owners [11].

Consequently, Scrum brings the advantages of flexibility and human initiative, yet opens gates to the diversity and unpredictability of people which at the end may inhibit to achieve a level of assurance and control [2], [6]. Thus, organizations within COBIT environments need to strike a balance between the two conflicting interests: agility and control.

**References**

Case Study of Project Outcome Prediction for an IT Vendor

Tomoyuki Kawamura  
Graduate School of System Design and Management  
Keio University  
Yokohama, Japan  
tomoyuki@z5.keio.jp

Kenichi Takano  
Graduate School of System Design and Management  
Keio University  
Yokohama, Japan  
k.takano@sdm.keio.ac.jp

I. INTRODUCTION
Researchers have found that approximately 70% of information systems development projects in Japan have failed, thus increasing the demand for solutions that will raise expected project success rates. It is said that to improve success rates, support should be provided by the organization to which the projects belong. The study aims to identify projects that an organization should support preferentially by predicting a project outcome. Several researches have demonstrated that they could predict project outcomes using risk assessment results in projects with specific characteristics such as in-house embedded software development. In this study, we try to predict project outcomes using the results of risk assessment at the establishment of requirements stage for the projects of a specific IT vendor in Japan.

II. PROJECT OUTCOME PREDICTION SYSTEM
The project outcome prediction was performed for “Company A,” one of Japanese IT vendors. In Company A, risk management specialists of the organization support the identification of project risks at the establishment of requirements stage and then determine the degrees of their participation in the project. We try to develop a project outcome prediction system including “risk assessment sheet” and “project outcome prediction model” to predict project outcome at the stage.

A. Risk Assessment Sheet and Outcome Assessment Sheet
The risk assessment sheet enumerates the risk assessment items of a project at the establishment of requirements stage. The sheet was created by referring to the framework of McLeod and MacDonell [1] and the knowledge of Company A. As a result, a risk assessment sheet consisting of 17 risk assessment items was created.

The outcome assessment sheet enumerates assessment items for project outcomes at the end of the project. The sheet is used for the development of the project outcome prediction model. This research characterizes project outcome as the difference between expected and actual measures of quality, cost, and scheduling.

B. Project Outcome Prediction Model
The data to create the prediction model were collected by assessing the completed projects using the risk assessment sheet and the outcome assessment sheet. As a result, data from 88 projects were collected. Then, each project was classified as a success/failure by considering the assessment results of the outcome assessment sheet. As a result of the classification, 43 projects (49%) were classified as successes, and 45 projects (51%) were classified as failures.

The project outcome prediction model was created by applying logistic regression analysis using the 88 project data. Logistic regression analysis was performed with the stepwise selection method by utilizing the classification results of success/failure as the response variable and utilizing the assessment results of 17 items collected by the risk assessment sheet as the explanatory variables. As a result, a model using five risk assessment items was obtained. Then, the degree of generalization error was checked by applying the 10-fold cross-validation method to the model. As a result, the predictive accuracy of the model is 73.9%.

III. CONCLUSION
The research aimed to develop a project outcome prediction system including “risk assessment sheet” and “project outcome prediction model” in order to identify projects that an IT vendor should support preferentially. As a result, 73.9% of the project outcomes were predicted correctly. Considering that the research object is an IT vendor whose projects have various characteristics and that the prediction is conducted in an early project stage, the system is considered to have achieved the research purpose. Details of the study can be found in our previous paper [2].

REFERENCES
Abstract—Crowdsourced testing is an emerging phenomenon in software testing which utilizes benefits of crowdsourcing for software testing. It brings cost and quality advantage with faster delivery of software products. Among all the software development activities, testing is one of the primary activities considered for crowdsourcing. Certain types of testing demand for testers from diverse ethnicity, culture, or geography; crowdsourced testing has become a viable option to address such needs. This paper presents some of the initial attempts done at Accenture to adopt crowdsourced testing. We present some of the software development projects where testing had been crowdsourced using a crowd based platform and yielded promising results. We also showcase how crowd wisdom, gleaned through a sentiment analysis performed on the users’ feedback, have helped in improving software.

Keywords—crowdsourcing; testing; software development

I. INTRODUCTION

Crowdsourcing is one of the major paradigm shifts that organizations are witnessing these days [4, 5]. Crowdsourcing a task refers to outsourcing the task to an unknown crowd. Crowd workforce has become a viable alternative for traditional workforces for simple tasks, such as image tagging, translation, transcription, etc. However, for complex tasks such as software development tasks, the adoption of crowdsourcing is slowly gaining momentum. Among all the activities in software development, testing is one of the most time consuming activity. Therefore, a lot of attention has been given by the software development community to adopt crowdsourcing during the testing phase [6]. Several platforms, such as Passbrains [1], Applause [2], Testbirds [3] etc., have emerged to fulfill the need for dedicated crowdsourced testing efforts. Crowdsourcing not only provides an alternative to organizational workforce but also compliments the workforce on the specific skills, domains, technologies, and cultures. This aspect of crowdsourcing becomes quite relevant in the testing context where software applications are to be tested by the users coming from different geographies, cultures, frameworks, devices, and backgrounds.

Due to rapid change in technology landscape, a software need to be tested on variety of hardware and software platforms in the shortest time possible. This can only be achieved by parallelizing software testing tasks as much as possible. Crowdsourced testing is one of the promising alternative to test software in parallel on variety of hardware and software platforms. In addition it also provides capability for testing software for localization needs such as for different languages, and geographies. Some of the advantages of crowdsourced testing are listed as follows:

1. Improved quality: with crowdsourced testing, one can afford a diverse set of test scenario exercised by crowd workers and thus the software can be tested thoroughly in a shorter time span.
2. Workforce flexibility: one can attract and retain testers who do not wish to accept full time position.
3. Reduced time to market: the testing tasks can be parallelized, thus crowdsourced testing reduces time to market.
4. Tester diversity: crowdsourcing platforms provide access to a large pool of testers from diverse geography, languages, etc.
5. Agility: The crowdsourced testing helps in more agility in workforces.

Here we present some of the experiences gained while adopting crowdsourced testing within Accenture. We discuss some of the preliminary attempts of utilizing crowd for different types of testing such as functional testing, usability testing, and localization testing. We further demonstrate the value of crowdsourced testing in terms of reduction in overall cost and time to deliver software applications. Firstly, we present an estimate that we performed to assess the value of crowdsourced testing. This estimation was done on a project that has already been tested by internal testers. Next, we discuss a project where we adopted crowdsourced testing on a real project. Towards the end, we discuss some of the ongoing work on utilizing crowdsourced data for software debugging purposes that collects reviews from public forums to identify use case that can replay the software bugs.

II. POINT OF VIEW: VALUE, COST AND TIME

In one of our assessments, we measured cost and time taken in Sandbox testing performed by internal testers and estimated how much cost and time could have been saved when it was performed by crowdsourced testers. In this project the internal testers performed test for six websites which were launched in Japan. The test activity involved Functional testing which was done by the internal testers.

The same test activity when performed by crowdsourced testers was estimated to cost less than half when compared to performed by the internal testers. Also, the duration was estimated to be 10 time lesser when performed by internal tester. In addition, this type of testing has an advantage of being done by the users located in Japan using their own devices and network connectivity; thus the device...
procurement cost was almost negligible. Table 1 highlights the measured and estimated metrics for internal testers and crowd testers.

<table>
<thead>
<tr>
<th>Workforce Type</th>
<th>Day</th>
<th>No of Testers</th>
<th>Total cost</th>
<th>Types of Testing</th>
<th>Mode of Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal (actual)</td>
<td>52</td>
<td>6</td>
<td>45000 USD</td>
<td>Functional</td>
<td>Desktop/Laptops Simulators Intranet</td>
</tr>
<tr>
<td>Crowd (estimated)</td>
<td>4</td>
<td>25</td>
<td>18000 USD</td>
<td>Functional, Usability, Localization</td>
<td>Desktop/Laptops Mobile/ Tablets External network</td>
</tr>
</tbody>
</table>

III. CASE STUDY: LOCALIZATION TESTING

The previous activity helped us in estimating the value proposition for crowdsourced testing. To understand better the true benefits of crowdsourced testing, we adopted crowdsourced testing in a project which was for a world’s leading Hospitality Group to test their Booking engine’s Short Message Service (SMS) communication for any new booking, cancellation, or changes in the service requirement. The above testing required recreation of booking process and confirmation to the user with an SMS message. The SMS needed to have personalized detail, type of room booked and confirmation to the user with an SMS message. The SMS testing required recreation of booking process and testers.

Moreover, it did not bear any procurement cost for necessary hardware. These unpleasant use cases have been used to debug the software.

IV. USING CROWDSOURCING FOR DEBUGGING

Another area where we are exploring the use of crowdsourcing is software debugging to identifying potential use cases that result in defects. We refer such use cases as ‘unpleasant use cases’. Often it is difficult to replay the unpleasant use cases as users do not post the entire scenario that led to an unpleasant experience. In this work we leverage power of crowd to understand how the Mobile Applications (Apps) and software used in (Internet of Things) IoT devices are used and which use cases led the software to an unstable, faulty, or unpleasant situations. This analysis is done based on users’ feedbacks. Users provide feedback on the services provided by App but not about specific aspects of an App that led to an unpleasant experience. Therefore, many a time we do not know which aspect of an App have led to a poor service experience. To collect such use cases, we have adopted a process where we collect user feedback about Apps usage and identify use cases of an App that led to bad user experience. This process is termed as “WhatsAppening”. The user feedback is collected from Apps marketplaces with an analysis tool. We utilized the ‘App review and scoring system’, supported by platform Applause [1], to identify what feature of an App led a user to uninstall the application. The user interaction and the data collected during this process have been used for recreating the potentially unpleasant use cases. These unpleasant use cases have been used to debug the software.

V. CONCLUSIONS

In this paper we presented some initial experiences gained while adopting crowdsourcing for testing activities in an enterprise context. The initial results indicate that crowdsourced testing is one of the plausible alternatives for testing workforces in certain scenarios as it provides a flexible and diverse workforce to meet organizations test requirements. However, there are several challenges that need to be address. There is a need for standard testing practice that includes crowdsourced testing an integral part of standard testing process. Value proposition analysis for crowdsourced testing needs to be validated on wide number of additional projects. Also crowdsourced testing need to be assessed from the perspective of confidentiality and security.

REFERENCES

Quality Based Software Project Staffing and Scheduling with Budget and Deadline

Dongwon Seo∗, Donghwan Shin∗, Doo-Hwan Bae∗

∗School of Computing, Korea Advanced Institute of Science and Technology, Korea
Email: {dwseo, donghwan, bae}@se.kaist.ac.kr

Abstract—Software project planning is becoming more complicated and important as the size of software project grows. Many approaches have been proposed to help project managers by providing optimal staffing and scheduling in terms of minimizing the salary cost or duration. Unfortunately, the software quality, another critical factor in software project planning, is largely overlooked in previous work. In this paper, we propose the quality-based software project staffing and scheduling approach. We provide better software project plans considering quality with either cost bound (Budget) or duration bound (Deadline) for software project managers.

I. INTRODUCTION

Since ineffective software project scheduling can lead the failure of software project [1], it is significant for project managers to schedule the project thoroughly. In software project scheduling, staffing is most important work because software development is people-intensive activity.

Given tasks for a software project and human resources, there are many cases of staffing. Many studies have proposed automatic staffing approaches for scheduling and they aim to minimize salary cost or duration [2]. However, considering salary cost or duration only would lead ineffective software project scheduling because it overlooks relationship between the employee and tasks.

In this paper, we consider the employee, the task, and cost or duration together with the aid of GA (Genetic Algorithm) search. We define the relationship between the employee and the tasks as the quality score of software product in terms of software defects. We search the scheduling by the fitness function which considers quality score with either cost or duration as a bound.

II. QUALITY-BASED SOFTWARE PROJECT PLANNING APPROACH

Software project planning using genetic algorithms is composed of the three steps: (1) assign proper employees to tasks (i.e., staffing), (2) calculate the schedule from the staffing result (i.e., scheduling), and (3) search for better plans by evaluating the current plan (i.e., evaluation). In staffing step, employees are assigned to task and, their working hours on the time unit are decided. To generate various staffing results, we use operation of genetic algorithm (i.e., mutation, crossover, selection). Using the staffing result, scheduling is performed. It decides when each task starts and ends. After scheduling is completed, the scheduling result is evaluated. By defining the cost or duration bound in the fitness function, we get the plan result which satisfying the cost or duration bound, and maximizing quality score.

A. Quality Score

A quality score represents possibility of generating the defect. It is in range from 0 to 1. If the quality score is high, then it means that the software product of the plan has more defect than plan of lower quality score. There are three steps for calculating a quality score as follows.

1) Quality score on the task level: We calculate quality score on the task level (QT) using practical considerations which related to the number of multitasking tasks, the continuity of tasks, and the number of allocated experts [3].

2) Quality score on the phase level: We calculate the quality score on the phase level (QP) by aggregating QT. When aggregating the QT values of the tasks in a phase, we use task Severity. Severity means the vulnerability to defects.

3) Quality score on project level: We calculate the quality score on the phase level (QS) by aggregating QP. When aggregating the QT values of the tasks in a phase, we use the weight of each phase.

B. Fitness function

\[
\text{Fitness score} = \begin{cases} 
\frac{w_q \times QS + w_c \times \text{(cost | duration) bound}}{\text{(cost | duration) bound}} \\
\frac{(w_q \times QS + w_c \times \text{d}) \times 1}{(\text{(cost | duration) ≤ (cost | duration) bound})}
\end{cases}
\]  

III. CONCLUSION

We propose the automatic software project planning approach considering the quality of the software product with either the required cost or duration. Our approach provides software project plans that considers software quality satisfying either the cost bound as called budget or duration bound as called deadline by the project manager.

REFERENCES

