CONTINUOUS INTEGRATION FOR CROWDSOURCING SOFTWARE DEVELOPMENT

Author : Vishnu Tej MP (vm9069@rit.edu)

Advisor : Dr. Mei Nagappan (mei@se.rit.edu)

Rochester Institute of Technology

2015

Software Engineering

Introduction

Software development has evolved over the years and nowadays, largely takes place in small co-located and remote teams [2]. Crowdsourcing is an open call for participation in any task that can be assigned to anyone in the general public.

There are benefits associated with crowdsourcing related to cost, time to market, however, it is the concerns including planning, intellectual property and quality assurance(QA) that make it a relatively unpopular development process.

In this research, the plan is to mitigate concerns related to QA with Continuous Integration, which is a development practice where developers integrate code frequently and then verified by an automated build.

Research Questions

RQ1

Does continuous integration as a development practice for crowdsource software development improve the quality of the product?

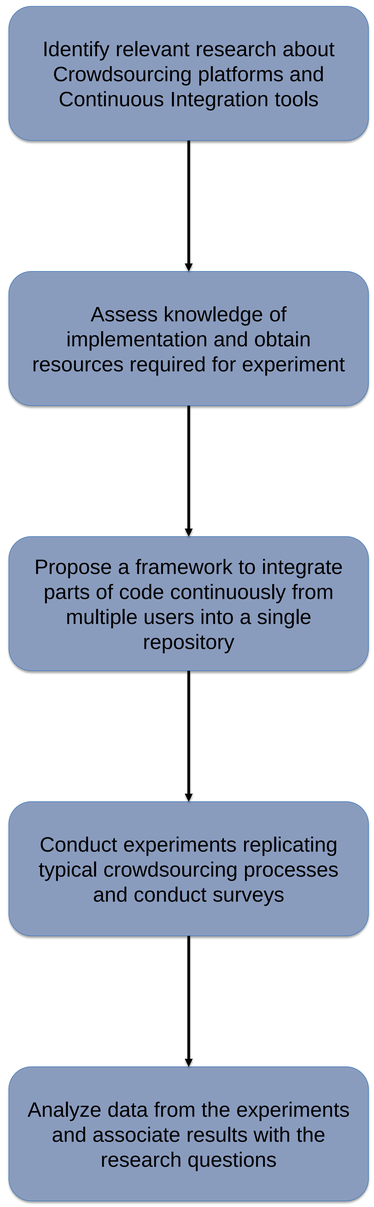
RQ2

Does continuous integration make the crowdsourcing software development quicker by improvement of frequency of submissions?

RQ3

Would developers be encouraged to adopt crowdsourcing software development with continuous integration over traditional software development processes?

Methodology



Study

The experiment specified in the methodology is a core part of the study as there isn't much research in this area due to its rather recent discovery and it can provide critical data required to further assess the crowdsourcing development process [3].

The experiment would include replication of a crowdsourcing process where a competition would be conducted for developers to submit code for a task as part of an application. However, there would be two similar projects, one using continuous integration and the other without.

We then analyze the data and conduct surveys with the developers and record their experiences to further investigate the viability of crowdsourcing in software [1].

Experiment

The idea behind conducting the experiment was to replicate the general crowdsourcing process and request the developer to fill in the survey. The experiment is similar to a competition where developers would be provided with a part of the code of a polling web application and they were required to develop a chat module that would help users chat with each other.

Only one set of developers would be given the capability to use Jenkins, which is a popular continuous integration tool, to help with the development. Once the developers have submitted their code, they would then complete their respective survey and after evaluating all of the submissions, a winner is picked and a prize is given based on design, functionality and code complexity.

Results

As the experiment was of a comparatively smaller scope as opposed to general industry grade crowdsourcing practices, it is difficult to determine and calculate the exact amount of success such a tool can provide, but there are positive indications of it from the gathered survey data and the submissions.

To answer RQ1, the submissions do indeed suggest that quality of product does increase, with the combination of crowdsourcing and continuous integration as the experiment had better overall submissions with the pool of developers using the continuous integration tool.

To answer RQ2 and RQ3, the data suggests that the developers did think it would improve the speed, but did not think it would replace standard software development.

References

[1] Saengkhattiya, Makon, Mikael Sevandersson, and Unai Vallejo. "Quality in Crowdsourcing-How software quality is ensured in software crowdsourcing." (2012).

[2] Stol, Klaas-Jan, and Brian Fitzgerald. "Researching crowdsourcing software development: perspectives and concerns." Proceedings of the 1st International Workshop on Crowdsourcing in Software Engineering. ACM, 2014.

[3] Stol, Klaas-Jan, and Brian Fitzgerald. "Two's company, three's a crowd: a case study of crowdsourcing software development." Proceedings of the 36th International Conference on Software Engineering. ACM, 2014.

Abbreviations

QM – Quality Management

QA – Quality Assurance