CrowdEV: Crowdsourcing Software Design and Development

Duan Wei(✉)

Beihang University, Beijing, China
duanwei@act.buaa.edu.cn

Abstract. The Internet based software is growing very fast with the soaring of the Internet, traditional approaches, which specify requirement offline and develop in isolated teams, are no longer the best approach. Especially after the web 2.0 and the crowdsourcing came up, it has shown some very promising qualities in speeding up the software development. Taking advantage of this, we present a novel software developing approach, which specify requirement by online crowdsourcing with expert supervision and develop using micro-task based crowdsourcing. To verify the feasibility of our approach, we established an online platform CrowdEV and ran a user study on that, which resulted in a successful development of a SNS software within campus in 5 days, comparing with an open-source project that took 17 days for similar functions.

Keywords: Requirement engineering · Crowdsourcing · Micro-tasks · Software development

1 Introduction

After so many years since it's come to being, the traditional software development has created some very mature methodology and models, these approaches, however, are all within the domain of developing software with very small groups together, thus is very hard to meet the fast-growing need of modern software.

Based on this, we came up with a micro-task based crowdsourcing approach with expert supervision to speed up the development. Our key insight lies mainly in two aspects, the collaborative requirement specification and developing software using micro-task in a massive parallel way.

2 Related Work

Considering the specification and refinement of the software requirement, Patrick Finnegan [4] took the way of having the open-source groups participated the development of the commercial software, yet cannot guarantee the time and quality of software; Wang Hao [1] analyzed the spatial-temporal clustering characteristics of the workers on the Internet that share the same or similar skills and specialities, and optimized the traditional requirement specification process.

S. Wang and A. Zhou (Eds.): CollaborateCom 2016, LNICST 201, pp. 527–532, 2017.
DOI: 10.1007/978-3-319-59288-6_51

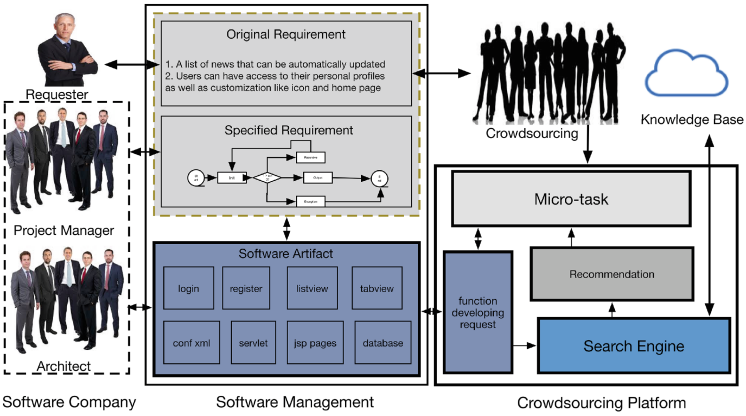

Fig. 1. The overall of the workflow.

In the field of crowdsourcing software development based on micro-tasks, many approaches aim at the decomposition of the project and the generation of the micro-tasks, for example, Little, Greg [7] analyzed the existing crowdsourcing algorithms and developed a script that can automatically decompose the tasks into a bunch of micro-tasks and publish them on the MTurk, CrowdForge [2] took the approach of Map Reduce and managed to have the workers into the process of the task decomposition. CrowdCode [6] however, took a dynamic way of generating micro-tasks to meet the changing need during the development.

3 Design

The key insight of our approach is that requesters can use CrowdEV to collaboratively specify and optimize the requirement of the software product by working with the online crowd as well as the expert project manager, and we have implemented a crowdsourcing software developing approach based on micro-tasks with code recommending to help multiple developers to finish the developing process in a highly parallel way thus to greatly decrease the time from the requesters' original idea to a highly usable software product into the market.

3.1 Workflow Overall

Our approach consists of 4 roles, Requester, Project Manager, Architect and Worker, the Project Manager(PM) and Architect are platform experts, using Product Requirement Document (PRD) and Prototype, these 4 roles will translate the original idea from Requester to final software product. To support communication and parallel development, we also provide 3 kinds of Tasks, which are Requirement Enrich Task (RET), Module Develop Task (MDT) and Function Develop Task (FDT). As is shown in Fig. 1.

From the graph we can see the whole process consists of two main parts, the collaborative requirement specification in which the requirement is defined and edited by requester and project manager and enriched by online worker, and the micro-task based software developing where modules of different granularity are decomposed, developed and assembled by architect and online worker.

3.2 Collaborative Requirement Specification

Traditional approaches use offline methods like focus group, questionnaires or one-to-one investigation, unlike those, we developed an online approach. After the Requester publishes the original ideas about the software product onto the platform in plain natural language, the PM will analyze and abstract software requirements from the description and format them into a formal PRD, which consists of two parts, the Feature and Scenario, the Feature part is organized as a tree structure, each represents an RET task, while every scenario also stand for one. The Crowd can only add new comments or new sub-requirement to the existing ones, the PM will choose from all the replies after the task is closed and return the result to the Requester.

3.3 Micro-task Based Crowdsourcing Software Development

Our approach implemented a task decomposing and designing mechanism that keeps evolving, as we all know, the requirement of the software is very hard to get fully specified before the development of the software, so is the design of the software, new problems and new requests often show up during the software development, taking reference from the Evolution Model for software development, we decomposed the responsibility to design the particular software architecture into the design of some core functional modules and iteratively decomposed it into the design of multiple minor artifacts of the software, during the designing process, new request may come up, developers can edit and expand their design by publishing new MDT or FDT.

When a worker is dealing with a FDT, to accelerate the development and lower the threshold for taking the task, we also get some similar code from github.com with an embed search engine as some reference for the worker.

Moreover, unlike typical micro-tasks which are independent and needs little human labor, the micro-tasks in software development are usually interdependent and requires communication and coordination during the development process. To deal with the problem, we also allow Worker to publish new FDT to help with his own task. The whole process is shown in Fig. 2.

3.4 Implementation

CrowdEV can be divided into two parts based on its function. The whole system is built mainly as a website, with an Android client being auxiliary. Using Android client, the worker can browse all the available micro-tasks and check

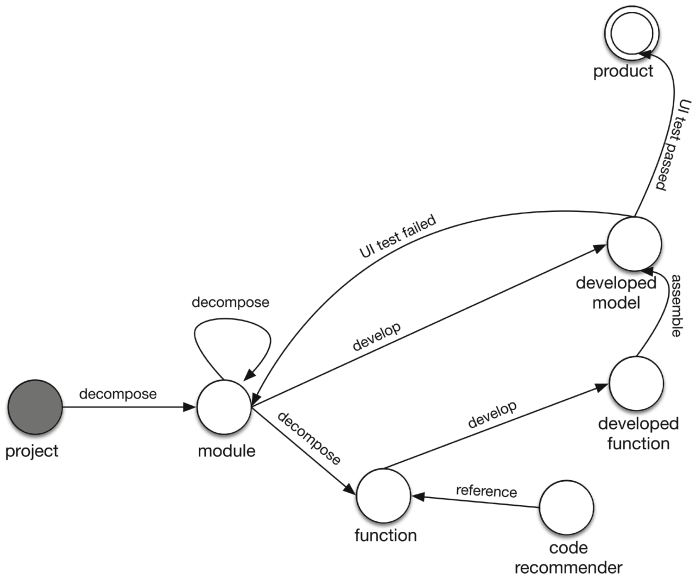

Fig. 2. Workflow of the crowdsourcing software development

out the micro-tasks he already accepted after signing in the system, the worker can also take part in the software requirement refining phase and publish their own enriching opinion, he can also agree or disagree with other workers' opinions using the client.

The platform provides a model for the software product, which capsulate the original request, the complete requirement data, the software prototype and the related micro-tasks with their answers. The platform also provides access to all those data for the requester of the project.

4 User Study

To verify the feasibility of our platform, we recruited 5 undergraduate students from our school as platform worker by email and contacts, these students all major in computer science, all took the Android developing related courses, all have at lease half a year of experience in Android developing, and all with the basic ability to write the simple JUNIT test cases, also, we recruited 4 post-graduate students who also major in computer science, yet with more experience in Android developing with an average of around 2 years, they are later going to be responsible for the decomposing of the MDT. At the same time, we hired one professional project manager as the expert manager of the platform and a experienced developer as the registered Architect on our platform from outside the campus.

Also to verify the advantage of faster development, we found an open-source project on Github.com which took the 9 participants 17 days to finish the core

functions of the Android SNS project. We first analyze the function of the project and played as the Requester and publish the function on the platform, the PM, after communicating with us and the Crowd, finished the prototype in half a day, the Architect took another day and decomposed the project into a batch of MDT.

5 Result

The development of the whole project took 5 days. And in total, 11 participants (9 Workers, 1 Project Manager, 1 Architect) has finished the design and implementation of a software project which consists of 1044 lines of code by publishing 3 micro-tasks and got 7 submissions in software requirement specification and by generating 70 micro-tasks during development process. Though number of participants and user incentive may vary between two projects, this experiment does show some advantage in speeding up the software development.

The key insight of our approach lies in 3 aspects: first, the collaborative requirement specification finished by Requester, PM and Crowd can speed up the specification and lower the barrier of communication; second, the micro-task based software development, by developing software in a massive parallel way, can significantly speed up the development process comparing to some other crowdsourcing ways; finally, the function recommend and pseudo-call replacement can solve the interdependence among different tasks on some level (Table 1).

Table 1. Statistics about the experiment

Project	Requirement	Module	Task
IM	User account management	login and register	task 1–7
		user homepage	task 8–21
	Publish News Feed	news publish activity including UI	task 22–45
	Browse News Feed	news list activity including UI	task 46–58
		news detail activity including UI	task 59–70

6 Discussion

Our experiment created a minor yet fully usable software in only 2 days with 8 students and 2 professionals in total. Which shows the promising ability for crowdsourcing to speed up the software development.

One key insight of our approach is the crowdsourcing workflow with expert supervision, which showed great efficiency in speeding up the software development. In fact, getting expert involved into the crowdsourcing process is generally popular and often interpreted as golden test in current crowdsourcing markets [3,5,8].

Meanwhile, our platform has revealed some problems. The first one is the imbalance of the workload, the design of the core data structure, the architect still has a heavy burden in designing and assembling the product, which could be a bottle-neck for the time control of the whole project process. Also, our platform only support Android development, which is a bit limited scenario.

References

1. Wang, H., Wang, Y., Wang, J.: A participant recruitment framework for crowdsourcing based software requirement acquisition. In: 2014 IEEE 9th International Conference on Global Software Engineering, pp. 65–73 IEEE (2014)
2. Kittur, A., Smus, B., Khamkar, S., et al.: Crowdforge: Crowdsourcing complex work. In: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, pp. 43–52 ACM (2011)
3. Sarasua, C., Simperl, E., Noy, N.F.: CrowdMap: Crowdsourcing ontology alignment with microtasks. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012. LNCS, vol. 7649, pp. 525–541. Springer, Heidelberg (2012). doi:10.1007/978-3-642-35176-1_33
4. Naparat, D., Finnegan, P.: Crowdsourcing Software Requirements and Development: A Mechanism-based Exploration of ?Opensourcing?[J] (2013)
5. Retelny, D., Robaszkiewicz, S., To, A., et al.: Expert crowdsourcing with flash teams. In: Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology, pp. 75–85. ACM (2014)
6. LaToza, T.D., Towne, W.B., Adriano, C.M., et al.: Microtask programming: Building software with a crowd. In: Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology, pp. 43–54. ACM (2014)
7. Little, G., Chilton, L.B., Goldman, M., et al.: TurKit: Human computation algorithms on mechanical turk. In: Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology, pp. 57–66. ACM (2010)
8. Stol, K.J., Fitzgerald, B.: Two's company, three's a crowd: A case study of crowdsourcing software development. In: Proceedings of the 36th International Conference on Software Engineering, pp. 187–198. ACM (2014)