Proceedings of the IASTED International Symposium
Software Engineering and Applications (SEA 2015)
October 26 - 27, 2015 Marina del Rey, USA

LEVERAGING CROWDSOURCING IN CLOUD APPLICATION DEVELOPMENT

Eman Aldhahri, Abdullah Abuhussein, and Sajjan Shiva
Computer Science Department
The University of Memphis
Memphis, USA
{aldhahri, bhussein, sshiva}@memphis.edu

ABSTRACT
The emergence of crowdsourcing has enabled workforce seekers to delegate various tasks to the unknown public to accomplish. Crowdsourcing serves in Software development where projects often fail due to the inability to find and allocate expertise. In cloud application (i.e. Software as Service – [SaaS]) development, projects severely suffer from shortage of expert developers due to its recent and rapid evolution. Therefore, many software manufacturers resort to crowdsourcing to find and recruit experts in SaaS development. To address this need, we conducted a survey of SaaS crowdsourcing to identify its challenges and to explore the crowdsourcing facilities that support addressing these challenges. Furthermore, we review two widespread existing approaches for software development crowdsourcing and propose a novel approach. Additionally, we discuss the challenges in SaaS development crowdsourcing and evaluate our proposed approach for its ability to address these challenges. Finally, we provide future adopters with a list of attributes to assist them in choosing the proper crowdsourcing service. This paper aims to provide a state-of-the-art assessment of the work carried out so far in applications development crowdsourcing and proposes recommendations to enhance it.

KEY WORDS
Software engineering; crowdsourcing; cloud application development; software-as-a-service; cloud computing;

1. Introduction

Recruiting workforce with sufficient experience to develop software has consistently been a great challenge. In particular, there is an unexpectedly high demand for workers to move organizations' data and computation to the cloud. However, experts in SaaS development are hard to find [1]. This problem has led to the need to outsource tasks in software development to experts off-site. Software crowdsourcing is one of the emerging approaches to hire others from outside the organization to complete some software development tasks.

In 2006, Wired Magazine first introduced the term crowdsourcing as "the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call" [4]. Crowdsourcing is the process where a crowdsourcer (i.e. Requesters) outsource tasks to crowdsourcees (i.e. a large network of crowd workers) for monetary rewards. The advantage of crowdsourcing resides in the ability of the crowdsourcers to access a large pool of highly skilled crowdsourcees who are able to process the outsourced tasks in a reduced amount of time and cost compared to in-house workers [4, 5, and 6].

Crowdsourcing platforms consist of a set of internal and external developers, a collective set of software development tools, and collaboration tools that enable resource sharing and automated collaboration to implement and maintain software. This is known as the crowdsourcing ecosystem. Through such a system, developers can collaboratively build software on top of a single platform [7, 8]. Ecosystems support software development crowdsourcing by providing a unified platform that enables experts to contribute to various project tasks (e.g. requirements, design, implementation, testing, etc.). It forms an interface for interaction between crowdsourcers and crowdsourcees [8]. There are a number of commercial ecosystem platforms, such as clinkerhq or junos platform [9, 51], that have a wide range of tools to simplify the development process. The clinkerhq ecosystem for instance, provides tools for configuration management, building, continuous integration, code analysis, documentation, planning, tasks, backups, and artifact management.

The usefulness of crowdsourcing is not limited to solving the problem of finding the right experts. Companies are increasingly calling on outsiders to hear different voices, eliminate expert bias, and maximize access to a variety of solutions [2]. In 2011, Nokia started the "Ideasproject" which is an online community that was created by Nokia to allow users and developers from all around the world to brainstorm [3]. Nokia employed the idea of crowdsourcing because they believed that user-centered innovation offers great advantages over manufacturer-centric innovation. The Ideasproject was founded on the philosophy of democratized innovation to enable users to innovate products and services for themselves.

Despite the crowdsourcing features (e.g. infinite pool of skilled workers, on-demand hiring, obtaining alternative solutions), adopting crowdsourcing for SaaS development remains nontrivial. SaaS development still faces challenges such as requirements engineering and testing [10, 11, 12, and 13] that are yet to be completely addressed. Our objective is to highlight the capability of crowdsourcing in developing effective SaaS. Hence, we pose the following questions:

- Can crowdsourcing be utilized in SaaS development to cover the shortage of experts and other obstacles?
- What are the challenges involved in SaaS crowdsourcing?

In the remainder of this paper, we demonstrate the difference between SaaS and the traditional software development. In section 3, we go over our motive for this research and present related works and their motivations. In section 4, we propose existing approaches for software crowdsourcing and demonstrate their limitations to support SaaS development. Section 5 represents the facilities of crowdsourcing and their influences on software development. In section 6, we investigate the critical challenges that hinder SaaS crowdsourcing success. In section 7, we suggest attributes to benchmark utilizing software crowdsourcing platforms for SaaS development projects and follow that in section 8 with a discussion of research limitations and future work. Section 9 offers a summary and conclusions.

2. SaaS vs Traditional Software

SaaS aims to increase software adoption, accelerate upgrades/updates, and provide less strenuous scalability and supportability. Although all SaaS development methodologies in literature are considered adaptations of the traditional software development lifecycles with additional phases (e.g. evaluation, Subscribing, etc.), these phases are critical for SaaS success [52]. SaaS applications are developed, deployed and delivered to customers using various cloud computing related architectures (e.g. multi-tenancy, multi-instance, etc.) and design principles (reusability, composability, etc.). In addition to the application's functional and non-functional requirements, SaaS development possesses cloud-related requirements that can be compositional (coordination, conformance, monitoring, and QoS) and managerial (certification, rating, SLA, and support) [53]. These architectures, requirements and principles are relatively new to traditional software development; as a result, developing software with these advantageous features adds new dimensions of considerations to the traditional software development process

It is apparent that a lack of experts is the key challenge that makes SaaS development more problematic than traditional software development. Therefore, we are seeking solutions that provide a sufficient quantity and quality of experts who can deal with SaaS complexities. Additionally, when developing SaaS for customers, developers are playing in someone else's garden and they need to play by their rules. For example, when developing SaaS for a customer in different country, challenges like compliance and understanding currency and taxation are preferably addressed by experts from the same country.

3. Motivation

This research is motivated by well-known approaches in software engineering (SE) such as (1) "Global distributed software development" where software is developed in multiple locations [14], (2) "Outsourced software development" where work is performed by known individual/organization and maybe "shipped" off-shore [15], and (3) "open-sourced software development" where software source code is publicly available to be used, changed, and/or shared [16]. Conducting software projects in multiple global locations or hiring others is likely to result in benefits such as cost reduction and reduced time-to-market [1, 17]. Therefore, these development approaches have been increasingly utilized by the industry [18].

Table I illustrates a comparison between software development crowdsourcing and the three aforementioned software development approaches in terms of publicity of participants, the anonymity of participants, sacrificing intellectual property, location diversity, and multiplicity. These aspects and their impacts were extensively researched and effective solutions were proposed in the previous literature. However, Table 1 demonstrates that crowdsourced software development shares all four aspects with the other three SE approaches. Thus, crowdsourcing software development tends to be subject to all the challenges--and perhaps some additional challenges--that may emerge due to the incorporation of two or more aspects.

TABLE 1. COMPARISON OF IN-HOUSE, GLOBAL DISTRBUTED, OUTSOURCING, CROWDSOURCING SOFTWARE DEVELOPMENT

Software Development					
Aspect/ Approach	**In-house**	**Global Distributed**	**Open-sourcing**	**Out-sourcing**	**Crowd-sourcing**
Public Participants	NO	NO	YES	NO	YES
Unknown Participants	NO	NO	YES	NO	YES
Transfer Intellectual Property	NO	NO	NO	YES	YES
Multiple or diverse Locations	NO	YES	YES	YES	YES

The literature suggests there has been a substantial work where software manufacturers delegated to a third party to develop software fully or partially. However, providing SaaS is becoming a trend in the software industry nowadays. Software engineers use service-oriented architecture (SoA) to patternize applications, guide the development process, and thus develop effective SaaS. SoA is defined as a set of architectural tenets for building autonomous yet interoperable systems [19]. SoA defines eight principles that guide its development, maintenance, and service usage: abstraction, autonomy, composability, discoverability, formal contract, loose coupling, reusability, and statelessness [20]. A glance at the SoA principles reveals that SoA principles and software crowdsourcing share some commonalities. Software crowdsourcing can contribute to any software development phase(s) or can be incorporated into various conventional or agile development processes. As a result, it is favorable for the crowdsourced tasks to be abstract, loosely coupled, and independent from the underlying infrastructure and application logic in order to create SaaS with maximal features (e.g. reusability and composability).

To crowdsource, a phase of the software development process is not something new. Several research papers addressed crowdsourcing requirement engineering [21], stakeholders' analysis [22], testing, support, and maintenance [23]. Cloud-based software crowdsourcing is fairly researched too [24]. In this paper, we want to provide

future adopters (e.g. crowdsourcers and crowdsourcees) and potential researchers with a thorough, systematic review of the existing status of software crowdsourcing and specifically SaaS development [25].

4. Crowdsourcing for SaaS Development

SaaS development can be crowdsourced using two different approaches. The first approach (Figure 1.a) demonstrates a software project that is defined, analyzed, and then decomposed into smaller tasks to be outsourced to the public. These tasks can be requirement elicitation, design, requirement implementation, or testing. Accomplished tasks are then gathered and integrated into a working SaaS. This approach is being adopted by many crowdsourcing services that are available online like Topcoder [26]. The second approach (Figure 1.b) involves crowdsourcing the whole SaaS development project as a single unit where public workers perform the requirement elicitation, design, development, and post-deployment phases as one task. This particular approach fits very small or tiny projects like the ones crowdsourced in Upwork [27].

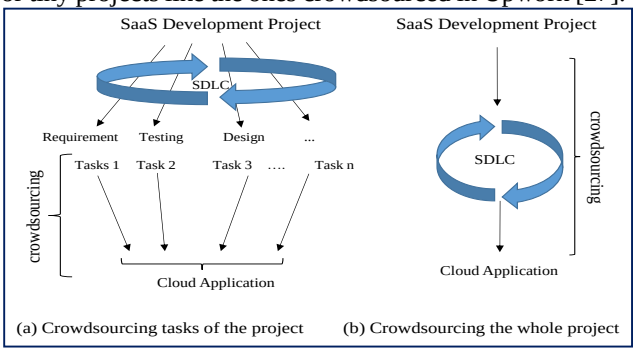

Figure 1. The existing approaches followed in Crowdsurcing Software

5. Crowdsourcing Facilities

Developing an effective SaaS can be a challenging process through centralized organizations [28, 29, and 30]. In centralized organizations, all the project members are located in one location [32]. Challenges like finding proper experts, reducing project cost, and reducing project duration, etc. are common in software development projects. Researchers are still looking for contemporary solutions that could facilitate the development process and address the aforementioned challenges [32].

Harnessing crowdsourcing in SaaS development can contribute effectively toward this purpose due to the multiple efficacious facilities that crowdsourcing possesses. These facilities are illustrated as follows:

A. Access to Pools of Global Distributed Skilled Workers

Crowdsourcing platforms are open for public participation and thus encourage creativity. Contrary to centralized organizations, crowdsourcing customers are not limited to specific workers. The Netflix Prize for example is a remarkable crowdsourcing experiment that demonstrates the usefulness of having access to minds from all over the globe [33].

B. Access to Lower-Cost Workers

Crowdsourcers benefit from lower wages and the purchasing power of currencies in developing countries. For instance, some Asian countries offer skilled workers (software engineers in this context) with a significantly lower wage compared to North American workers who have similar capabilities [29].

C. Temporary Hiring and On-demand Hiring

Unlike centralized organizations that are obligated to pay monthly salaries to workers who may be underutilized over a period of time, crowdsourcers benefit from the pay per task facility and they can request the service on-demand. This dynamic availability results in reducing either cost or duration of a task; thus, project managers have the advantage of sacrificing time for budget or vice versa in struggling projects.

D. The Ability to Obtain Alternative Solutions

The crowdsourcing platform has two behavioral natures that will be explained later in this paper (i.e. competitive and hiring). In competitive crowdsourcing platforms, crowdsourcers only pay for the winner. In other words, there are no extra charges when a crowdsourcer receives alternative solutions. Therefore, crowdsourcing encourages creativity through engendering the spirit of competition.

E. Direct Access to the Voice of the Customer

Crowdsourcing acts like an avenue where customers' opinions and feedback about a service can be heard. It motivates customers to share their thoughts that can contribute effectively to a software project's success. IdeaStorm, for instance, is a project launched by Dell to give their customers an avenue to share their thoughts and cooperate with each other and Dell. Customers' thoughts have generated 432 inventive implementations [54]. This facility can be utilized throughout the life cycle of the software by crowdsourcing software design, testing, and evaluation.

F. Eliminate Experts' Bias

To eliminate expert bias, crowdsourcees are assessed based on their successful contribution to the crowdsourcing platform instead of their positions, titles, or background. Experts from the public are motivated to participate regardless of their backgrounds. Students can compete with professors to win a crowdsourcing competition. The Longitude Prize in 1714 is a good example that confirms the fact that better solutions could be produced from less experienced people [34]. The competition was to determine the longitude of ships at sea. Experts in the field worked on the problem without success; however, a clock-maker solved the problem in the end. The following list demonstrates how the aforementioned crowdsourcing facilities can be used to cater SaaS application development.

- Adding more creativity to the development processes: due to crowdsourcing, a pool of skilled workers as well as innovative and superior solutions is becoming reachable. Software manufacturers are able to form an innovative team of workers who share a variety of best

practices perspectives because of their individual backgrounds [29].

- Recruiting more skilled software engineers with a lower cost: through crowdsourcing, customers could access engineers from developing countries who cost much less than engineers with similar capabilities from developed countries [29].
- Avoiding redundant skills: because hiring workers is on-demand, customers can scale the number of workers up and down depending on their needs.
- Achieving the best possible efficiency: better efficiency can be achieved due to the ability to find lower cost, skilled software engineers.
- Having the right skills on demand: there is no obligation to provide a monthly salary for workers who are underutilized for a certain amount of time. Alternatively, workers are paid based on what they have achieved.
- Utilizing cross-sourcing: software manufacturers can crowdsource some or all independent tasks concurrently to different workers. This in turn reduces the software development time. It is true that cross-sourcing is not exclusive to crowdsourcing and it can be emphasized wherever task execution can occur independently and concurrently. However, task parallelism is limited to the number of the workers who can contribute to a task. Because crowdsourcing gives access to skilled workers globally, cross-sourcing tasks can be simpler.
- Utilizing time-zone advantages: the fact that workers in crowdsourcing can be located in different time zones could be very useful in speeding up the software development process. When a sequence of (start-to-finish) dependent tasks is distributed effectively, dependent tasks can start immediately after successive tasks finish if located properly in consecutive time-zones, resulting in reduced wait times that may add up to a significant duration [29].
- Meeting stakeholders' needs: client-site application development provides developers with more insight into the system requirement [35, 29]. Likewise, in SaaS development, stakeholders are located all over the world, so it would be more effective to elicit SaaS requirements from stakeholders who understand other cultures' needs.
- Improving task modularization: decomposing tasks to modules represents a substantial role in managing coordination between workers [36]. The fact that the SaaS development can be decomposed into several tasks in crowdsourcing enables crowdsourcers to effectively create independent tasks. This is achieved by dividing tasks into services that can be integrated seamlessly into different applications [37]. Afterwards, each service is assigned to a worker. A detailed explanation of this model is illustrated in the discussion.
- Reducing coordination cost: assigning tasks to workers who are globally distributed contributes positively in reducing the coordination cost [38].
- Tracking communication logs: the nature of crowdsourcing implies the absence of simultaneous communication. Because workers are distributed globally in different time zones, they can rely on electronic communication through emails or instant messengers [38]. Thus, communication history will be preserved which yields traceability and accountability [39].
- Improving documentation: distributing tasks among workers requires documentation. Documenting tasks and their statuses in every phase aims to make tasks well supported and communication between task performers unambiguous and more transparent [40, 41]. The absence of face-to-face meetings necessitates that workers document solutions in a more detailed and descriptive way.
- Defining processes clearly: in centralized application development, processes are not predominantly formalized [29]. Distributing tasks among different workers from different backgrounds raises the necessity of formalized and standardized processes to ensure consistency.
- Enhancing testability: it is more feasible to test SaaS by the crowd who are the potential users. It provides more insight into the end-user needs. Moreover, it is more cost effective than hiring a permanent local worker to test the application

6. Challenges in Software crowdsourcing

Delegating software development tasks to workers in different geographical areas has become widespread and accepted as a business necessity to overcome some of the traditional software development drawbacks such as high cost [42]. Although modern communication tools are available to support task distribution, software crowdsourcing is still considered a serious challenge [41] and most crowdsourcing commercial platforms tend to highlight the success stories in software crowdsourcing with little or no attention to the challenges. In the following list, we investigate the potential challenges that could encounter in general crowdsourcing application development but with more emphasis on SaaS development.

A. Communication issues:

Application development requires substantial communication among workers and customers, especially at early stages. This communication usually occurs in two different ways: (1) formal communication, which includes decomposing tasks, responsibility assignment, and updating the application status [43]; and (2) informal communication, which occurs frequently between workers when tasks are interdependent. Because crowdsourcing workers are distributed globally, this could hinder simultaneous communication among workers due to time zone differences. Consequently, productivity also decreases [44].

B. Language barriers

Language differences can cause project failure when application development processes are crowdsourced. Workers might misunderstand task requirements or other

co-workers' communications, which may lead to failure to meet task requirements.

C. Cultural issues

Human resources are an essential element in the software development industry. Cultural differences among human resources are also a concern. The cultural gaps can be represented in time commitment, communication manner, and behavior towards team members. [34]. These differences can be critical, especially for interdependent tasks.

D. Tasks and Workers Coordination Issues

Planning for project decomposition and distribution among workers and then assigning task dependencies can also become a dilemma. For example, task parallelism is useful when tasks are independent. However, interdependent tasks that different crowdsources have developed can lead to another type of integration challenge known as tasks incompatibility. For this, inadequate task coordination can be fatal in the application development process.

E. Worker Collaboration Issues

Because crowdsourcing workers are distributed globally, sharing experiences, practices, tools, and decisions may not be an easy task. Managing such a collaboration technique among crowdsourcing workers is another considerable challenge.

F. Planning and Scheduling

In crowdsourcing, tasks are usually assigned to unknown workers. This may cause a loss of control over many aspects (e.g. intellectual property). In addition, schedule uncertainty due to a lack of knowledge of the level of workers' experience is a challenge that is yet to be completely resolved.

G. Quality Assurance

Crowdsourcing providers claim that crowdsourcing's solutions provide high quality work [45, 43]. However, in software development, it is hard to guarantee the quality because it is a relative measure [46]. Although crowdsourcing is meant to be a solution to the shortage of experts, there is no guarantee that the assigned worker can finish a task on time. In addition, crowdsourcing, like other forms of temporary employment, can be superficial work. Not only are crowdsourcers subject to an ongoing hassle from task doers (e.g. late or low quality task), but task doers (crowdsourcees) also face difficulties to guarantee being paid on time and keeping a steady stream of work. All above are enough reasons to produce low quality and quick work from crowdsourcee side.

H. Hidden Costs

Software development projects are usually decomposed into several tasks that may be handled in a parallel or in a sequential manner. However, assigning a task to a worker does not guarantee the task's completeness and correctness. Consequently, this may result in reassigning incomplete or incorrect (unaccepted) tasks to other workers. It is worth mentioning that for tasks that are not accepted, there is no payment obligation, but development slippage does affect the application's development cost.

I. Copyrighting and Intellectual Property

Protecting copyrights and intellectual property for a project's ideas and solutions can be a challenge. Crowdsourcing platforms are open to the public from anywhere on the globe to sign up, explore tasks, possibly participate in a task, etc. The task payment guarantee is an issue because it depends on the crowdsourcer's decision after that entity is granted the solution [47]. For that, the need for reputation systems has emerged.

After introducing the crowdsourcing facilities' influence on SaaS development and challenges, Table II explains how potential software crowdsourcing services can be compared in terms of their likelihood to address these challenges. We consider three prominent commercial software crowdsourcing services, namely, TopCoder [26], UpWork [27] and Freedomsponsors [47]. We compare these crowdsourcing platforms in a tabular form, as shown in Table II. The first column enumerates the challenges of adopting crowdsourcing for software development; the remaining columns explain how the three crowdsourcing platforms address these challenges. Some commercial crowdsourcing services may not publish enough details about their platform and the way they address challenges

7. Benchmarking SaaS-crowdsourcing

Like any commercial service, there has been a significant amount of work on trying to rank crowdsourcing platforms (e.g. ranker.com) [48]. Contrary to any other crowdsourcing ranking intermediaries that rank crowdsourcing platforms based on popularity, this work is to enable crowdsourcers to discover the crowdsourcing platform that best fits their SaaS development projects (if any) by identifying the platforms' capabilities and readiness to address the challenges of software crowdsourcing. The following are attributes that enable both crowdsourcers and crowdsourcees to make well-informed decisions.

A. *Behavioral Nature:* A crowdsourcee needs to know the behavioral nature (e.g. competitive or hiring) of the crowdsourcing platform. In the competitive behavior, any crowdsourcee may contribute and process the task without permission from the crowdsourcer to start. By contrast, in the hiring behavior, crowdsourcers need to grant their permission to the crowdsourcee before he/she can start processing the task. In competitive crowdsourcing, a prize goes to one or more crowdsourcees who provide the best solution. However, in hiring crowdsourcing, a crowdsourcee receives a prize for solution correctness and conformity to crowdsourcer requirements. Knowing the behavioral nature of the crowdsourcing platform would help the crowdsourcee to predict the probability of winning the prize.

B. *Reputation systems:* Crowdsourcing platforms need to be equipped with a means to assess workers (crowdsourcees) based on their expertise and behavior. Such information can be collected from the worker's history. In order to achieve this, a crowdsourcing environment requires a reputation system that assesses crowdsources and crowdsourcers and predicts the trustworthiness of user contributions [49].

we saw two approaches that are widely adopted for software

TABLE I. CROWDSOURCING CHALLENGES AS ADDRESSED BY THREE SOFTWARE CROWDSOURCING PLATFORMS

Challenges	TopCoder	Upwork	Freedomsponsors
Communication	-Avilable.	- Not avilable.	-Aviable. .
Language	-Remains a challenge	-Remains a challenge	-Remains a challenge.
Culture	-Remains a challenge.	-Remains a challenge.	-Remains a challenge.
Coordination	- Remains a challenge	- not a challenge bacuase the project is developed by a single crowdsourcee and not decompsed into tasks.	- Remains a challenge
Collaboration	- Provides a communication tool to support sharing experiences and practices among crowdsoursees.	- Remains a challenge	-Provides a simple forum as a communication tool. It enables developers to collaborate.
Planning and Scheduling	Tasks are posted as a competition for all crowdsorsees. There is no risk in assigning the task for non suitable crowdsourcee.	-Remains a challenge.	-Remains a challenge.
Quality Assurance	-Remains a challenge	- Remains a challenge	-Remains a challenge.
Hidden Costs	-Remains a challenge.	-Remains a challenge.	-Remains a challenge.

C. *Supporting Tools:* Crowdsourcing platforms vary in providing software development facilities such as ecosystems and communication/ collaboration tools. These facilities aim to support, facilitate the development process and eventually improve the software quality.

D. *Copyright Protection:* A major challenge in software crowdsourcing is to protect copyrights from infringement. Ideas rights for the posted tasks should be reserved to the crowdsourcer. Likewise, unless a crowdsourcee agrees to abandon his/her rights, solution rights should be also reserved to the crowdsourcee. The main concern is to ensure copyrights are created correctly and that they preserve the right of both sides. Thus, any crowdsourcing platform needs to develop a set of solutions that guarantees copyright protection and complies with the laws and regulations governing both parties.

E. *Recommendation system:* Quite a large number of tasks are posted every day in crowdsourcing platforms. For workers, finding the appropriate task that fits their skills, time, and expertise is time consuming. Similarly, crowdsourcers need to be able to learn about the task doer's ability to solve similar problems. Therefore, a recommendation system aims to shorten the distance between both parties by providing suggestions that can effectively support their decision making. This need is addressed by a matching process that is crucial to the crowdsourcing platform's success.

F. *Popularity:* All the aforementioned attributes can attract more contributors (i.e. crowdsourcees or crowdsourcers). Once a crowdsourcing platform becomes known and popular, other contributors may be more inclined to utilize it because people tend to obtain/use popular services that are recommended or used by others.

8. Discussion and Future Work

Crowdsourcing SaaS development processes can be quite empowering. However, we discuss some of its challenges that can be critical if not addressed. In Figure 1 (a and b), crowdsourcing. Table II shows the two approaches in column two and three and their preparedness to address these challenges. As noted, when the whole project is crowdsourced, it is subjected to all challenges in column 1 except the communication and collaboration challenges. In the second approach (i.e. project decomposition into tasks), the project is subject to all challenges. As an alternative, we propose a third approach that suits the nature of SaaS development. This approach is a service-based decomposed SaaS. In a service-based decomposition approach, the SaaS development project is decomposed into services. Each service is then outsourced to the crowd. Implemented services are collected afterward and composed into a larger server of services. This approach is not utilized in the existing crowdsourcing platforms and to the best of our knowledge; it is first introduced in this paper. A service-based decomposition of a crowdsourced SaaS project is motivated by two key points: (1) SoA that SaaS applications promote and (2) crowdsourcing works better for specific software development tasks that are less complex and stand-alone without interdependencies [50]. The rationale behind this proposed approach is that every service will function separately without any interdependencies with the other services' functionality and is independent from the underlying infrastructure and/or programming language. In other words, the output from one service can be used as input for the other services.

TABLE III. CHALLENGES AS APPLIED TO THE THREE SOFTWARE CROWDSOURCING APPROACHES

Challenges	**Whole Project**	**Project decomposed into tasks**	**Project is decomposed into services**
Communication	YES	NO	YES
Language	NO	NO	YES
Culture	NO	NO	NO
Coordination	NO	NO	YES
Collaboration	YES	NO	YES
Planning and Scheduling	NO	NO	NO
Quality Assurance	NO	NO	NO
Hidden Costs	NO	NO	NO

Figure 2 shows a crowdsourcing model that decomposes the SaaS development project into sub-services. Each service is then crowdsourced to the public throughout a crowdsourcing platform "ecosystem". The ecosystem platform has facilities that guide and control the development processes such as software development tools, project management tools, communication tools, and collaboration tools. All software development phases (i.e. requirement, design, implementation, testing, and post deployment) are performed on every sub-service. The software development phases take place after the project is decomposed into services and after each service is clearly defined in terms of its input, output, and process. This model works better for both small and large-scale SaaS projects. The proposed approach should include a quality control process to ensure that tasks (i.e. services in this context) are correct and complete. The proposed model benefits from qualities like replacing project managers with a system that is able to assign workers to tasks and predict tasks and project duration based on pre-collected data. In the future, we are planning to build a web-based crowdsourcing platform equipped with an ecosystem and all aforementioned supporting tools. In addition, we plan to probe and propose more SaaS development crowdsourcing models to enable crowdsourcers to easily conceptualize and plan for a software development project, elicit its requirements, and decompose it into tasks to be developed, tested, and submitted.

Although crowdsourcing may present solutions to overcome some SaaS development challenges, at this time, crowdsourcing software development is still superficial and immature. We are aware of some limitations for the previously mentioned approach. For example, a crowdsourcer may need to contact the developer to perform some changes to the service after the project is delivered. This limitation, among others, forms potential issues to be researched and further investigated in the future.

9. Conclusion

Crowdsourcing can be very effective for SaaS development. The facilities of crowdsourcing support building cohesive and loosely coupled SaaS if used soundly. In this paper, we aimed to provide a useful reference point for researchers and professionals who wish to work on SaaS crowdsourcing and, more generally, software crowdsourcing. We described the facilities of crowdsourcing and their influence on SaaS development. We also identified and described the approaches currently used in SaaS crowdsourcing and the challenges in utilizing crowdsourcing to develop SaaS. We used this list of challenges to identify the likelihood and the ability of existing software crowdsourcing platforms to address these challenges. We proposed a service-based crowdsourcing approach for SaaS development and demonstrated its ability to deal with some of these challenges for large-scale projects. We attempted to start the process of detangling the intricacies of the crowdsourcing SaaS development process. In the future, we aim to expand this work and propose the architecture and implementation for a crowdsourcing platform that can effectively address the challenges and that can expand software crowdsourcing adoptability.

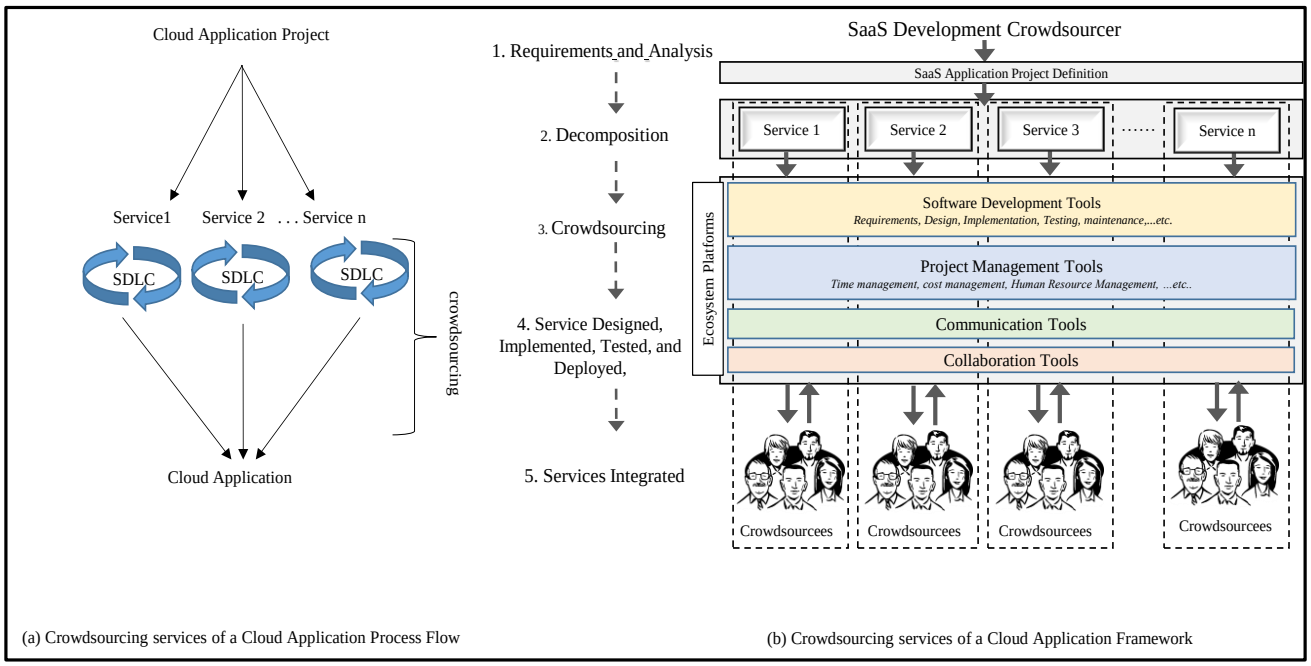

Figure 2: A SaaS development crowdsourcing model that decomposes a project into services.

References

[1] Lee, S. G., Chae, S. H., & Cho, K. M. (2013). Drivers and inhibitors of SaaS adoption in Korea. International Journal of Information Management, 33(3), 429-440.

[2] Hypios, "Why use crowdsourcing? Three fundamental arguments" acccessed from http://www.hypios.com/news/blog/210711/why-use-crowdsourcing-three-fundamental-arguments

[3] David Doherty " IDEAS PROJECT: Your chance to help Nokia develop new mHealth services" 2011 accessed from: http://mhealthinsight.com/2011/05/30/idea-project-your-chance-to-help-nokia-develop-new-mhealth-services/

[4] Howe, J. (2006). The rise of crowdsourcing. Wired magazine, 14(6), 1-4.

[5] Brabham, D. C. (2008). Crowdsourcing as a model for problem solving an introduction and cases. Convergence: the international journal of research into new media technologies, 14(1), 75-90.

[6] Whitla, P. (2009). Crowdsourcing and its application in marketing activities.Contemporary Management Research, 5(1).

[7] Bruegge, B., Creighton, O., Helming, J., & Kögel, M. (2007). Unicase–an Ecosystem for Unified Software Engineering Research Tools. In Third IEEE International Conference on Global Software Engineering, ICGSE (Vol. 2008).

[8] Bosch, J., & Bosch-Sijtsema, P. (2010). From integration to composition: On the impact of software product lines, global development and ecosystems.Journal of Systems and Software, 83(1), 67-76.

[9] Clinkerhq, Software Development Ecosystem accessed from: http://clinkerhq.com

[10] Gao, J., Bai, X., & Tsai, W. T. (2011). Cloud testing-issues, challenges, needs and practice. Software Engineering: An International Journal, 1(1), 9-23.

[11] B.J.D Kalyani, Swarna Bharathi, " Challenges in the Cloud Application Development" , International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Volume 2, Issue 1, January 2013 (pp. 310-313)

[12] Homer, A., Sharp, J., Brader, L., Narumoto, M., & Swanson, T. (2014). Cloud Design Patterns: Prescriptive Architecture Guidance for Cloud Applications.

[13] Jana, D., & Bandyopadhyay, D. (2014, May). Management of security and privacy issues of application development in mobile cloud environment: A survey. In Recent Advances and Innovations in Engineering (ICRAIE), 2014(pp. 1-6). IEEE

[14] Herbsleb, J. D., & Moitra, D. (2001). Global software development. Software, IEEE, 18(2), 16-20.

[15] Krishna, S. (2003). Global IT outsourcing: software development across borders. Cambridge University Press.

[16] Feller, J., & Fitzgerald, B. (2002). Understanding open source software development. London: Addison-Wesley.

[17] F. Lanubile, D. Damian, and H. L. Oppenheimer, "Global software development: technical, organizational, and social challenges," SIGSOFT Software Engineering Notes 28(6): 1 - 4.

[18] Hawthorne, M. J., & Perry, D. E. (2006). Software engineering education in the era of outsourcing, distributed development, and open source software: challenges and opportunities. In Software Engineering Education in the Modern Age (pp. 166-185). Springer Berlin Heidelberg.

[19] Jammes, François, Antoine Mensch, and Harm Smit. "Service-oriented device communications using the devices profile for web services." In Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing, pp. 1-8. ACM, 2005.

[20] Thomas Erl: SOA Design Patterns, Prentice Hall PTR; 1 edition (2009)

[21] Hosseini, M., Phalp, K., Taylor, J., & Ali, R. (2013). Towards crowdsourcing for requirements engineering. In 20th International working conference on Requirements engineering: foundation for software quality-Empirical Track.

[22] Lim, S. L., Quercia, D., & Finkelstein, A. (2010, May). StakeSource: harnessing the power of crowdsourcing and social networks in stakeholder analysis. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2 (pp. 239-242). ACM.

[23] Utest.com. The Professional Network for Testers. accessed from: http://www.utest.com/

[24] Tsai, W. T., Wu, W., & Huhns, M. N. (2014). Cloud-Based Software Crowdsourcing. IEEE Internet Computing, (3), 78-83.

[25] Keele, S. (2007). Guidelines for performing systematic literature reviews in software engineering. In *Technical report, Ver. 2.3 EBSE Technical Report. EBSE*.

[26] TopCoder. 2010. 10 Burning Questions on Crowdsourcing: Your starting guide to open innovation and crowdsourcing success. http://www.topcoder.com/blog/10-burningquestions-on-crowdsourcing-and-open-innovation .

[27] Upwork accessed from: https://www. upwork.com/

[28] ul Haq, S., Raza, M., Zia, A., & Khan, M. N. A. (2011). Issues in Global Software Development: A Critical Review. Journal of Software Engineering and Applications, 4(10), 590.

[29] Ågerfalk, P. J., Fitzgerald, B., Olsson, H. H., & Conchúir, E. Ó. (2008). Benefits of global software development: the known and unknown. In Making Globally Distributed Software Development a Success Story (pp. 1-9). Springer Berlin Heidelberg.

[30] Conchúir, E. Ó., Ågerfalk, P. J., Olsson, H. H., & Fitzgerald, B. (2009). Global software development: where are the benefits?. Communications of the ACM,52(8), 127-131.

[31] Daly, E. B. (1979). Organizing for successful software development.Datamation, 25(14), 107-120.

[32] Wu, W., Tsai, W. T., & Li, W. (2013). Creative software crowdsourcing: from components and algorithm development to project concept formations.International Journal of Creative Computing, 1(1), 57-91.

[33] Koren, Y. (2009). The bellkor solution to the netflix grand prize. Netflix prize documentation, 81.

[34] Sobel, D. (2007). Longitude: The true story of a lone genius who solved the greatest scientific problem of his time. Bloomsbury Publishing USA.

[35] Herbsleb, J.D., Moitra, D.: Guest Editors' Introduction: Global Software Development. IEEE Software 18(2), 16–20 (2001).

[36] Parnas, D.L.: On the criteria to be used in decomposing systems into modules. Communications of the ACM 15(12), 1053–1058 (1972).

[37] Hashmi, S. I., Clerc, V., Razavian, M., Manteli, C., Tamburri, D. A., Lago, P., ... & Richardson, I. (2011, August). Using the cloud to facilitate global software development challenges. In Global Software Engineering Workshop (ICGSEW), 2011 Sixth IEEE International Conference on (pp. 70-77). IEEE.

[38] Espinosa, J.A., Carmel, E.: The Effect of Time Separation on Coordination Costs in Global Software Teams: A Dyad Model. In: 37th Hawaiian International Conference on System Sciences. Big Island, Hawaii, IEEE, Los Alamitos (2004)

[39] Boland, D., Fitzgerald, B.: Transitioning from a Co-Located to a Globally-Distributed Software Development Team: A Case Study and Analog Devices Inc. In: 3rd International Workshop on Global Software Development, May 24, 2004, Edinburgh, Scotland (2004) 30. Damian, D., Zowghi, D.: The impact of stakeholders geographical distribution on managing requirements in a multi-site organization. In: IEEE Joint International Conference on Requirements

Engineering, IEEE Computer Society Press, Los Alamitos (2002).

[40] Delone, W., Espinosa, J.A., Lee, G., Carmel, E.: Bridging Global Boundaries for IS Project Success. In: 38th Annual Hawaii International Conference on System Sciences (HICSS 2005) - Track 1, vol. 01, IEEE Computer Society, Los Alamitos (2005).

[41] Gumm, D.: Distribution Dimensions in Software Development Projects: A Taxonomy. IEEE Software 23(5), 45–51 (2006).

[42] Herbsleb, J. D., & Grinter, R. E. (1999, May). Splitting the organization and integrating the code: Conway's law revisited. In Proceedings of the 21st international conference on Software engineering (pp. 85-95). ACM.

[43] Herbsleb, J. D., & Moitra, D. (2001). Global software development. Software, IEEE, 18(2), 16-20.

[44] Shrivastava, S. V. (2010). Distributed agile software development: A review.arXiv preprint arXiv:1006.1955.

[45] Schenk, E., & Guittard, C. (2009, December). Crowdsourcing: What can be Outsourced to the Crowd, and Why. In Workshop on Open Source Innovation, Strasbourg, France.

[46] Stol, K. J., & Fitzgerald, B. (2014, May). Two's company, three's a crowd: a case study of crowdsourcing software development. In ICSE (pp. 187-198).

[47] Freedomsponsors accessed from https://freedomsponsors.org/

[48] Ranker.com, "Best Crowdsourcing Websites & Pay Per Task Sites" accessed from : http://www.ranker.com/crowdranked-list/list-of-the-best-crowdsourcing-websites-and-pay-per-task-sites

[49] Wu, C., Luo, T., Wu, F., & Chen, G. (2015, April). An endorsement-based reputation system for trustworthy crowdsourcing. In *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on* (pp. 89-90). IEEE.

[50] Stol, K. J., & Fitzgerald, B. (2014, May). Two's company, three's a crowd: a case study of crowdsourcing software development. In ICSE (pp. 187-198).

[51] Kelly, J., Araujo, W., & Banerjee, K. (2009, August). Rapid service creation using the JUNOS SDK. In Proceedings of the 2nd ACM SIGCOMM workshop on Programmable routers for extensible services of tomorrow (pp. 7-12). ACM.

[52] Kommalapati, H., & Zack, W. H. (2011). The SaaS Development Lifecycle.InfoQ: http://www. infoq. com/articles/SaaS-Lifecycle.

[53] Papazoglou, M. P., Traverso, P., Dustdar, S., & Leymann, F. (2007). Service-oriented computing: State of the art and research challenges. Computer, (11), 38-45.

[54] "DELL: IdeaStorm" (2015) accessed from: http://www.ideastorm.com/