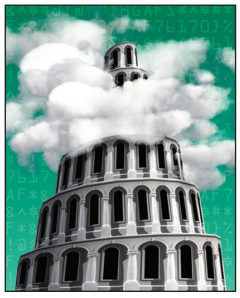
Editor: George Pallis • gpallis@cs.ucy.ac.cy



Cloud-Based Software Crowdsourcing

Wei-Tek Tsai • *Arizona State University*

Wenjun Wu • *Beihang University*

Michael N. Huhns • *University of South Carolina*

In addition to providing large-scale, highly available computational resources, clouds also enable a new methodology for software development via crowdsourcing, in which crowd participants either collaborate or compete to contribute software. Using a crowd to develop software is predicted to take its place alongside established methodologies, such as agile, scrum, pair programming, service-oriented computing, and the traditional waterfall.

Crowdsourcing software development, or *software crowdsourcing*,[1] is an emerging software engineering approach. Software development has been outsourced for a long time, but using a cloud to outsource it to a crowd of developers is new. We've found that all software development tasks can be crowdsourced, including requirements, design, coding, testing, evolution, and documentation. Software crowdsourcing practices blur the distinction between users and developers, and follow the cocreation principle — that is, a regular user becomes a codesigner, codeveloper, and comaintainer. This is a paradigm shift from conventional industrial software development, with developers distinct from users, to a crowdsourcing-based, peer-production software development, in which many users can participate.

A cloud provides a scalable platform with sufficient resources, including computing power and software databases, for a large crowd of developers. With emerging cloud software tools such as DevOps (a portmanteau of development and operations) and large-scale software mining, a cloud significantly reduces the amount of manual labor needed to set up software production environments and empowers peer developers to perform software crowdsourcing tasks efficiently in design, coding, and testing.

Based on its organizational style, software crowdsourcing can be either competitive or collaborative. In competitive crowdsourcing, only winning participants are rewarded. TopCoder (www.topcoder.com), an online programming contest site, is an example of this approach. In collaborative crowdsourcing, people cooperate with each other on various aspects, including funding, concept development, user interface design, code, test, and evaluation. AppStori (www.appstori.com) represents this approach. The process design (such as activities, duration, and number of participants), support infrastructure, and software projects are different for these two approaches.

Crowdsourcing Development Processes

Crowdsourcing can be incorporated into conventional software development processes such as waterfall or agile and can contribute to any software development phase.[2] It can be used with most design techniques, such as object-oriented, service-oriented, and user-centered design (UCD); software-as-a-service (SaaS); and formal methods. For example, TopCoder directly supports the following process: *conceptualization*, *specification*, *architecture*, *component production*, *application assembly*, *certification*, and *deployment*. Each can be crowdsourced competitively.

Published by the IEEE Computer Society

1089-7801/14/$31.00 © 2014 IEEE

Crowdsourcing Goals

Crowdsourcing's benefits accrue from organizations attempting to achieve the following goals:

- *Quality software*. Such software comes from competent participants who try to outdo their peers in submitting innovative concepts, design, code, or tests.
- *Rapid acquisition*. Crowdsourcing organizers can post a competition hoping to find that something similar has already been developed.
- *Talent identification*. An organizer might be interested in identifying talented developers, as demonstrated by their performance in competitive efforts.
- *Cost reduction*. A crowdsourcing organizer can acquire software at a low cost owing to the need to pay only winners, and could even pay below-market costs, given that participants might seek reputation rewards rather than monetary ones.

Other goals include solution diversity, idea creation, broadening participation, marketing, and participant education, such as encouraging people to use or learn specific tools. To ensure a good outcome from software crowdsourcing, organizations can leverage a cloud infrastructure to accelerate the process of setting up the development environment and enabling distributed and large-scale development by a highly dynamic community.

The turnout for a software crowdsourcing event varies due to crowdsourcing's open and online nature. A project with a high reward will attract a relatively large number of crowd workers. Moreover, computational overhead and data traffic for different software development tasks lead to spikes in the resources that crowdsourcing activities require. To address this issue, cloud-based software crowdsourcing combines advantages of both paradigms: pay-for-use and pay-for-performance. An elastic cloud computing resource lets crowdsourcing organizers cope with fluctuations in the numbers of participants as well as computational workloads that occur with crowdsourcing competition activity. For example, the largest crowdsourcing site for data analytics, Kaggle (www.kaggle.com), adopted Windows Azure to run software services and host crowdsourcing contests.

Industrial Software Crowdsourcing

Several websites have been established to support crowdsourcing, such as TopCoder, uTest (www.utest.com), AppStori, oDesk (www.odesk.com), and mob4hire (www.mob4hire.com). Furthermore, most of the industrial software giants are actively engaged in crowdsourcing, including Microsoft and Oracle. Microsoft used crowdsourcing for Windows 8 development by starting blogs,[3] crowdsourcing mobile devices for Windows 8, and offering US$100,000 for security testing.[4,5] Oracle also adopted crowdsourcing for its customer relationship management projects.[6]

Architecture and Models

The distributed development of a software system by a crowd requires the guidance of a reference architecture and models for the development process.

Software Crowdsourcing Architecture

Different software crowdsourcing processes can have different needs, but also share some commonalities. Such common themes for software crowdsourcing processes include

- a cloud service management dashboard for system administrators and software crowdsourcing organizers;
- collaboration and communication tools, such as a distributed blackboard system where each party can participate in discussions;
- participant ranking and recommendation tools;
- software development tools for modeling, simulation, code editing and compilation, design notation and documentation, and testing;
- cloud payment and credit management tools; and
- a repository of software development assets, such as modules, specifications, architectures, and design patterns.

All these elements are encapsulated in the reference architecture for a cloud-based software crowdsourcing system (Figure 1). Metaphorically, we can regard this architecture as synergy between two clouds — machine and human — toward the ultimate goal of developing high-quality and low-cost software products. With the properly specified platform-as-a-service (PaaS) recipes, software project managers can establish a customized cloud software environment to facilitate the following software crowdsourcing process:

1. A manager uses software networking and collaboration tools to design a reward mechanism to motivate crowd workers. Depending on the software product's value and the development scale, a project manager can specify the appropriate budget to attract as many talented developers as possible and provision computing resources to sustain their activities.
2. The manager uses project management tools to coordinate development tasks among the crowd workforce by ranking individuals' expertise and matching their skills with the different task levels and types.
3. Toward a specific software project, a manager sets up a virtual system platform with all the necessary software development gears to assist crowd workers with their tasks.

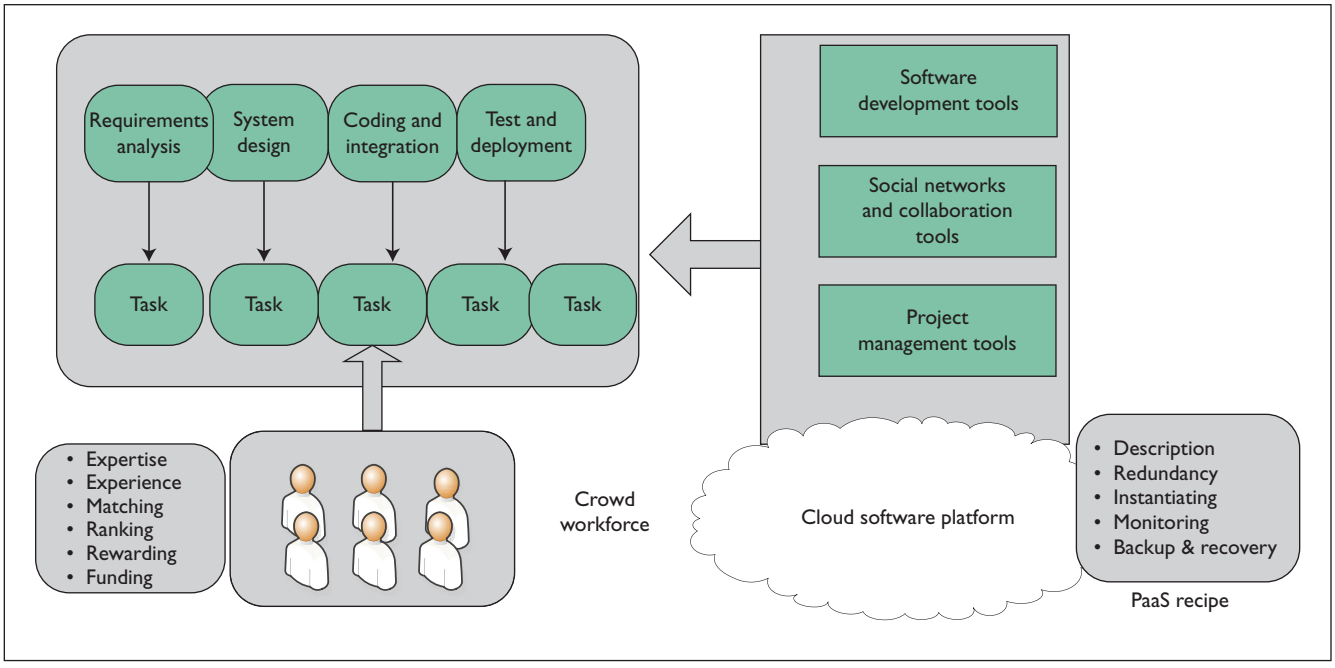


Figure 1. Reference architecture for cloud-based software crowdsourcing. We regard it as a synergy between two clouds — machine and human — toward the ultimate goal of developing high-quality and low-cost software products.

4. Both the platform and its workflow can operate elastically to yield cost-efficient resource utilization.

We discuss the cloud tools in detail in a subsequent section.

Software Crowdsourcing Models

We can characterize software crowdsourcing in terms of the crowd size, software scale and complexity, development processes, and competition or collaboration rules. Formal models for designing and modeling software crowdsourcing can have the following foundations:

- *Game theory*. The nature of contests in competition-based crowdsourcing can be analyzed via game theory. For example, we can determine the reputation reward value based on the number of participants and the reward price; participants are often willing to compete to gain reputation rather than receiving a reward price determined according to a Nash equilibrium.[2] Even the Prisoner's Dilemma game can be viewed in terms of crowdsourcing (www.dellingadvisory.com/blog/2013/4/10/crowdsourcing-the-prisoners-dilemma).
- *Economic models*. Economic competition models provide strategies and incentives to generate crowd participating and reward-structuring rules for organizers to maximize crowdsourcing returns. The recently developed *contest theory*[7] introduces new mathematical tools, such as all-pay auction, to describe the synergy among individual efforts, competition prize structure, and product quality.
- *Optimization theory*. Due to the competitive and dynamic nature of software crowdsourcing processes, coordinating unstable virtual teams, optimizing the partition and allocation of development tasks, and balancing costs, time, and quality can be challenging. Thus, the organizer of a crowdsourced software development effort could apply a search-based software engineering approach[8] to address optimization problems.

By using platforms that incorporate open source software repositories, such as Github and commercial crowdsourcing sites, we can conduct experiments to understand software crowdsourcing's nature and validate theories. Such experiments must model tasks, workers, and costs for effort and resources. We can verify various worker ranking, incentive, and matching algorithms to simulate possible virtual team formation, and to isolate design factors. Both professional engineers and students can be involved in experiments, and investigators can document and make available ontologies of these models, so that others can design new experiments, collect data, and exchange ideas. Several platforms are available, including OW2 Open Source Community and Trustie (http://forge.trustie.net). Simulation techniques, such as NetLogo, that are based on multiagent systems can also play a role in modeling software crowdsourcing behavior.

Cloud Tool Support

Given software crowdsourcing's distributed nature, it needs a powerful

Table 1. Cloud tools for supporting software crowdsourcing.

Cloud tool type	Name	Characteristics
Software development tools	Chef/Puppet	Streamlines the environment configuration for cloud-oriented software development and testing
	PaperTrail	Supports large-scale system log administration and analysis
	CloudIDE	Web-based IDE that simplifies and integrates cloud software development
Social networking and collaboration tools	Confluence	A team-based collaboration tool for content creation and sharing
Project management tools	CloudSpoke on Topcoder	A crowdsourcing website for cloud development that ranks developers' skills and organizes contests for development tasks
	Github	A project-hosting website that supports collaborative software development, and can serve as code repository for cloud-based software crowdsourcing
	Trustie	An online software development site that integrates project hosting, social networking, and programming education

development environment to facilitate software design, coding, test, and deployment across distributed and heterogeneous infrastructures. All community members should be able to access the same software development environment customized for a specific project. Moreover, numerous submissions from community members must be quickly screened, evaluated, reviewed, and integrated. Thus, software crowdsourcing efforts require enhancements to common software development tools for coding, testing, and deployment to support automatic project building, integration, performance analysis, and security checking. By leveraging cloud computing's elastic resource provisioning and virtual appliances, crowdsourcing practitioners have started to develop the evaluation pipeline to automate testing and review for quality, security, and coverage, as well as coding standards. A cloud-based integrated development environment (IDE) must have the following features (Table 1 summarizes the various tools).

Software Development Tools

An IDE for crowdsourcing integrates tools for requirements, design, coding, compilers, debuggers, performance analysis, testing, and maintenance. For example, cloud software configuration tools such as Chef (www.opscode.com/chef/) and Puppet (http://puppetlabs.com) let community members establish their own virtualized development environments. MapReduce-based log-management tools, such as PaperTrail (http://papertrailapp.com), support large-scale system log administration and analysis, and help community members resolve software problems and enhance system reliability using log messages.

Social Networks and Collaboration Tools

Facebook, Twitter, wikis, blogs, and similar Web-based tools let participants communicate for sharing and collaboration. For example, organizers can use Facebook profiles to form a virtual crowdsourcing team, even if the participants don't know each other. A collaborative blackboard-based platform can let participants see a common area and suggest ideas to improve the solutions posted there.

Project Management Tools

Crowdsourcing project management should support project cost estimation, development planning, decision making, bug tracking, and software repository maintenance, all specialized for the context of the dynamic developer community. In addition to these regular functions, it must incorporate ranking, reputation, and award systems for both products and participants. For example, TopCoder introduces a sophisticated ranking scheme, similar to sport tournaments, to rank community members' skills in software development. Community members often decide to participate in a specific contest if they know the ranking of the participants already enrolled.

Ecosystem for Software Crowdsourcing

Both collaboration- and competition-based software crowdsourcing have ecosystems with significant economic implications. These ecosystems create jobs and establish career paths for developers. They also let organizations start new projects, secure funding, identify talent, and create new products. For example, Apple's App Store, a website for iOS applications, has an ecosystem with 150,000 contributors and has accumulated more than 700,000 iOS applications in its four years of operation. People contribute their innovations for reputation or money via the store's micropayment mechanism. With such a large community, several community-based, collaborative platforms have

been created as incubators, and one such incubator is AppStori, an online crowd-funding community in which people can collaborate to develop iOS apps.

TopCoder, founded in 2001, similarly has an ecosystem with a global workforce of more than 500,000 members. Recently, TopCoder and CloudSpokes (www.cloudspokes.com), which has 72,000 registered members with expertise in cloud software, merged to form the largest crowdsourcing community for cloud-based software development. About 1,100 CloudSpokes projects have been posted online to motivate developers to work on the major cloud platforms, such as Amazon Web Services and Force.com.

Crowdsourcing also needs to focus on workers to keep them happy as the world enters a worker-centered crowdsourcing society. Essential features of a crowdsourcing platform should show past performances and assign proper ratings based on metrics developed by the relevant communities.

Team organization is also important in crowdsourcing, and teams can be self-organized by community or recommendation. Crowdsourcing communities must have objective governing rules to arrange bidding, matching, job security, career paths, project management, and often physical environments for participants. Trust among team members is important because teams might be formed by people who don't know each other, and it's necessary to manage trust via multiple strategies such as ranking, recommendation, displayed previous products, and recorded dialogue. Furthermore, a crowdsourcing platform must inform workers whom they work for and what their role in the development effort will be.

Crowdsourcing Roadmap

A recent workshop presented the following four-level roadmap for software crowdsourcing.[9]

Level 1

The first level is characterized by individual developers, well-defined modules, small size, limited development time span (less than a few months), quality products, and current development processes such as the ones by AppStori, TopCoder, and uTest. Coders are ranked; websites contain online repository crowdsourcing materials; participants can rank software; and crowdsourcing platforms have communication tools such as wikis, blogs, and comments, as well as software development tools such as an IDE, testing, compilers, simulation, modeling, and program analysis.

Level 2

At the second level are teams of people (< 10) and well-defined systems; a medium-sized system to be developed, with medium development time (several months to less than one year); and adaptive development processes with intelligent feedback in a common cloud platform where people can freely share thoughts. At this level, a crowdsourcing platform supports an adaptive development process that allows concurrent development processes with feedback from fellow participants; intelligent analysis of coders, software products, and comments; multiphase software testing and evaluation; big data analytics, automated wrapping of software services into SaaS, annotations with terms from an ontology, and cross references to DBpedia and Wikipedia; automated analysis and classification of software services; and ontology annotation and reasoning, such as linking those services with compatible I/O.

Level 3

The third level has teams of people (< 100 and > 10), a large well-defined system, long development time (< 2 years), and automated cross-verification and cross-comparison among contributions. A crowdsourcing platform at this level contains automated matching of requirements to existing components, including matching of specifications, services, and tests, and automated regression testing.

Level 4

The fourth level consists of a multinational collaboration of large and adaptive systems. A crowdsourcing platform at this level might contain domain-oriented crowdsourcing with ontology, reasoning, and annotation; automated cross-verification and test-generation processes; and automated configuration of the crowdsourcing platform. It might also restructure the platform as SaaS with tenant customization.

Pilot Software Crowdsourcing Projects

The University of South Carolina recently started a cloud-based crowdsourcing project to produce the control software for an office robot. The robot behavior will follow a three-step process: perceiving, reasoning, and acting, commonly known as a PRA architecture. Specifically, perceiving occurs using ultrasonic sensors to detect walls and obstacles. Moving to an office is an action that requires reasoning using data collected in the perceiving subsystem. A crowd consisting of students and faculty will develop software for the robot at the granularity of a behavior. The goal is to show that the crowd size need not be large. Developers can act autonomously and collaboratively. To ensure interoperability, software produced will be stateless services, so that they can be independent of each other, and each module will expose a service specification so that specifications can be formally reasoned. Furthermore, software development will be done using a common IDE, a cloud-based repository (such as Bitbucket or GitHub), and a Ubuntu Linux platform; testing can occur either in the cloud or on the actual robot.

Cloud-based software crowdsourcing is a new approach for low-cost, rapid software development, and the existence of large ecosystems has shown that this approach is viable. Moreover, it is a ripe area for research on how a crowd of anonymous developers can produce coherent models, analyses, simulation components, experimental results, and a support environment. Many research directions are possible, including theoretical models, optimization methods, infrastructure support features, and social issues, with a clear roadmap.

Acknowledgments

The ideas in this article were inspired by participants at the recent Dagstuhl Workshop on Cloud-Based Software Crowdsourcing.[9]

References

1. W. Wu, W.T. Tsai, and W. Li, "Creative Software Crowdsourcing: From Components and Algorithm Development to Project Concept Formations," *Int'l J. Creative Computing*, vol. 1, no. 1, 2013, pp. 57–91.
2. W. Wu, W.T. Tsai, and W. Li, "An Evaluation Framework for Software Crowdsourcing," *Frontiers of Computer Science*, vol. 7, no. 5, 2013, pp. 694-709.
3. S. Thomas, "Microsoft Launches Crowdsourcing Blog for Windows 8," *Memeburn*, 16 Aug. 2011; http://memeburn.com/2011/08/microsoft-launches-crowdsourcing-blog-for-windows-8/.
4. S. Simpson, "Crowdsource Your Next Windows 8 Device?" Windows RT Source blog, 10 June 2013; www.winrtsource.com/2013/06/10/crowdsource-your-next-windows-8-device/.
5. L. Bell, "Microsoft Offers a $100,000 Bug Bounty for Cracking Windows 8.1," *The Inquirer*, 20 June 2013; www.theinquirer.net/inquirer/news/2276303/microsoft-offers-a-usd100-000-bug-bounty-for-cracking-windows-81.
6. A. Diana, "Oracle Integrates Crowdsourcing into CRM," *InformationWeek*, 16 Mar. 2011.
7. L.C. Corchón, "The Theory of Contests: A Survey," *Rev. of Economic Design*, vol. 11, no. 2, 2007, pp. 69–100.
8. M. Harman, "The Current State and Future of Search Based Software Engineering," *Proc. IEEE Future of Software Eng.* (FOSE 07), 2007, pp. 342–357.
9. M.N. Huhns, W. Li, and W.-T. Tsai, "Cloud-Based Software Crowdsourcing (Dagstuhl Seminar 13362)," *Dagstuhl Reports*, vol. 3, no. 9, 2013, pp. 34–58; doi: 10.4230/DagRep.3.9.34.

Wei-Tek Tsai is a professor at Arizona State University. His research interests include software as a service, service-oriented computing, and crowdsourcing. Tsai has a PhD in computer science from the University of California, Berkeley. He's a member of IEEE. Contact him at wtsai@asu.edu.

Wenjun Wu is a professor at Beihang University. His research interests include crowdsourcing, cloud computing, and eScience. Wu has a PhD in computer science from Beihang University. He's a member of the China Computer Federation (CCF). Contact him at wwj@nlsde.buaa.edu.cn.

Michael N. Huhns holds the NCR Professorship and is chair of the Department of Computer Science and Engineering at the University of South Carolina. He also directs the Center for Information Technology. Huhns has a PhD in electrical engineering from the University of Southern California. He is a senior member of ACM and a fellow of IEEE. Contact him at huhns@sc.edu.