Collaborative Software Development Platforms for Crowdsourcing

Xin Peng, Muhammad Ali Babar, and Christof Ebert

Outsourcing to the crowd, or crowdsourcing, has launched extremely successful businesses, such as Linux. But platforms for efficient collaboration and crowdsourcing support are still emerging. Authors Xin Peng, Ali Babar, and I provide an overview of current technologies for crowdsourcing. I look forward to hearing from both readers and prospective column authors about this column and the technologies you want to know more about. —Christof Ebert

IN 1991, A 21-year-old student at the University of Helsinki, Finland, posted a short message on Usenet: “I’m doing a (free) operating system (just a hobby, won’t be big and professional like gnu) for 386(486) AT clones. This has been brewing since April, and is starting to get ready. I’d like any feedback on things people like/dislike in Minix....” His name was Linus Torvalds, and with this short message, he attracted such a big crowd of software developers that the first version of this new OS was completed in just three years. Linux 1.0 was made publicly available in March 1994, and it started one of the biggest crowdsourcing initiatives ever launched. By 2008, the revenue from servers, desktops, and software running on Linux was nearly 30 billion euros. Crowdsourcing not only seems to be fun for software engineers, but it also delivers a solid business model.

Crowdsourcing in software development means that you solicit services from a voluntary online community, rather than from traditional employees or suppliers. It rapidly developed in the past decade as a part of Web 2.0 as a process that can be closed or open source. Figure 1 shows the differences among crowdsourcing, outsourcing, open source, and proprietary development. Essentially, crowdsourcing lets members of the crowd participate as providers of software development tasks requested by enterprises—simultaneously, it supports business value transfer between providers and requesters. In contrast, open source development doesn’t support business value transfer between providers and requesters, and traditional outsourcing doesn’t allow open participation.

Today, crowdsourcing is used for large-scale and commons-based peer production of information, knowledge, and culture. Enterprises use it for various purposes such as content creation, innovative design, data analysis, development, and testing. The general motivation behind crowdsourcing is to harness the creative energies of multiple voluntary participants with little or no
financial compensation or formal managerial structure.\textsuperscript{2}

Software development is an innovative and knowledge-intensive process that takes advantage of the collective wisdom, creativity, and productivity of myriad people in an increasingly global context.\textsuperscript{2,3}

In crowdsourced software development, enterprises (as requesters) delegate requirement analysis, design, coding, and testing tasks to external individuals or groups (as providers) with the support of crowdsourcing platforms. Large IT companies use internal crowdsourcing with software development tasks and their own employees to leverage untapped human resources.

**Platform Requirements**

Crowdsourced software development, by its very nature, is collaborative. The stakeholders in a crowdsourced software project form a virtual team with the support of collaboration tools and social media technologies. Various kinds of communication, collaboration, and coordination (3C) happen among the requesters, providers, and platform vendors—for example, requesters and providers communicate about a task’s requirements and evaluation criteria, requesters coordinate the progress and technical decisions of different tasks, and providers collaborate with each other via shared artifacts and workspace.

In addition, developers working on a collaborative project need to be aware of various aspects of the team and the project, which is called group awareness.\textsuperscript{4} Successful teams will combine communication, collaboration, and coordination with awareness to form a 3C+A model of collaborative software development.\textsuperscript{5}

Crowdsourcing platforms can execute a request in different modes, for example, by advertising it in a marketplace and allowing providers to bid for it or running a competition and selecting a winner based on requester-specified criteria. In both of these cases, the platform must support some kind of business model that allows different parties—requesters, providers, and platform vendors—to participate in value creation and sharing.

**Communication**

The crowdsourcing platform supports the exchange of messages and information among requesters and providers to reduce gaps and ambiguity. Providers need to negotiate with requesters about requirements and terms by exchanging information and opinions, and requesters need to learn about provider capability, experience, and reputation. Crowd members, usually geographically distributed, need to communicate with each other about technical or organizational issues via the platform. In addition, different task providers might need to communicate for collaboration and coordination of tasks for the same project.

**Collaboration**

The crowdsourcing platform also supports various collaborations by providing the facilities for sharing workspaces and encouraging user interactions with artifacts synchronously.
or asynchronously. Developers collaborate at different levels—some of them might work on the same piece of the project (source code or UML models) synchronously in collaborative development activities or collaborate on a set of shared artifacts with the support of version control systems. At the project level, different task developers might collaborate on the integration of artifacts from their specific tasks.

**Coordination**

Finally, the crowdsourcing platform supports the management and coordination of people and processes at both the technical and business levels. Essentially, the platform provides the facilities for creating, assigning, executing, evaluating, and rewarding crowdsourced tasks and supervises the commitments of both requesters and providers. For example, the platform might need to resolve possible disputes between requesters and providers; if a task is executed as a competition, it might also need to coordinate the competition among different providers of the same task.

Simultaneously, the platform needs to support requesters and coordinate the development processes of different tasks in the same project. For example, for a component development task, the requester will need to aggregate and provide the required component specification, development tools, libraries, testing data, and environments for providers, all with platform support.

**Awareness**

Group awareness lets members of virtual teams obtain the required knowledge of the working context by understanding the processes, tasks, physical presence, and project status. There are four types of group awareness: informal (or presence), group structural, workspace, and social.

Group awareness is especially important in crowdsourced software development because of the openness and high fluidity among crowd members. The crowd gets involved in a project loosely and temporarily, gathered in virtual communities. Group awareness can help crowd members better understand updated statuses for their tasks, development environments, collaborators, and competitors. Moreover, being aware of others’ work can prompt crowd members to learn from each other and enhance their creativity.

**Business Models**

Through crowdsourcing, an enterprise working as a requester can access a scalable workforce online in a cost-effective way and harness its creative energies. In return, an individual or a group of developers working as a provider can gain monetary rewards from the company and reputation for their work. The platform vendor benefits by receiving agency fees for successful completion of tasks and usage fees for platform resources, such as storage and tools.

A key issue to be addressed in the crowdsourcing business model is the handling of intellectual property (IP) issues. For a crowdsourced task, the platform must provide the mechanisms to coordinate various IP issues between requesters and providers. Moreover, requesters and providers need to agree on how the IP rights of deliverables are transferred to the business and shared by both parties. Finally, the enterprise must ensure that the deliverables don’t infringe on copyrights owned by third parties.

**Collaborative Platforms: Current Practice**

Several platform options let enterprises leverage the intelligence of the crowd. Table 1 lists some of these platforms and compares their capabilities in terms of support for crowdsourced software development.
Crowdsourcing Platform

Current crowdsourcing platforms, such as TopCoder (www.topcoder.com), CoFundos (cofundos.org), Genius Rocket (geniusrocket.com), and Innocentive (www.innocentive.com), offer a Web-based platform on which enterprises and individual developers can register and form an online community. (For a more detailed look at TopCoder as an example, see the sidebar.)

Crowdsourcing platforms have well-defined business models to encourage crowd members to participate in development tasks and submit their solutions. A platform usually charges enterprises for their delivered tasks, but charges no fee from crowd members. A small number of winners selected for a task can get monetary or other kinds of rewards such as employment according to specified terms. Usually, the IP rights to winning solutions are transferred to the crowdsourcing enterprise in exchange for rewards.

Crowdsourcing platforms coordinate the delegation relationships between crowdsourcing enterprises and crowd members. For each project, a platform usually assigns a coordinator (called a copilot in TopCoder), who might also be selected from the crowd. The coordinator helps the enterprise decompose a project into a series of tasks and delivers these tasks to the crowd. The coordinator handles the whole process, including task specification, execution, evaluation, and rewards.

Crowdsourcing platforms support communication by providing task-specific forums for crowd members to ask questions and communicate with each other. The task coordinator can manage and answer questions raised by crowd members; some platforms support communication by letting providers and requesters send messages. However, crowdsourcing platforms provide little support for collaboration among crowd members. Some of them allow the crowd to share artifacts but provide no support for version control.

TOPCODER

A good example of a typical crowdsourcing platform on which enterprises can deliver their software developing tasks and crowd members can compete for them is TopCoder (www.topcoder.com).

If a customer wants to build a website, he or she first finds a copilot on TopCoder to work with on managing the whole process, including setting tasks, pricing each task, developing and submitting artifacts, and evaluating deliverables. Each task is delivered in the form of competition (see Figure A). After the competitors participating in a task submit their work, the best is chosen and compensated; sometimes, the second and third best candidates are paid as well.

Several companies have used TopCoder to develop new Web interfaces that work with legacy systems. At ABB, this project was complicated because it requires not only keeping the original, complicated user operations but also developing the functionalities for meeting a rich set of new user requirements (see http://community.topcoder.com/pdfs/tcs/casestudies/abb_casesudy.pdf for the case study). TopCoder’s copilots assessed the project and proposed a solution to reduce its complexity. Utilizing the TopCoder methodology helped ABB group the project into several components, and the competition mechanism ensured high-quality solutions for each phase. In addition, the TopCoder copilots created developer forum threads to keep developers aware of new problems, thus making the whole process much more effective. Last but not least, TopCoder provided reusable components for future integration, achieving cost and time savings of roughly 30 percent. With the utilities provided by TopCoder, ABB significantly saved in the total cost and gained a higher-quality software product.

**FIGURE A.** Software assembly competitions in TopCoder.
Platform as a Service
PaaS systems such as Google App Engine and Force.com are examples of Web-based application development platforms enabled by cloud computing technologies. PaaS systems provide end-to-end or partial environments for developing full programs online, supporting tasks from editing code to debugging, deployment, runtime, and management.7

PaaS systems usually provide a set of tools and environments for application development that can be used to support various tasks such as modeling, interface design, coding, and testing in an on-demand way. PaaS supports collaboration among developers because the code is managed online, making it easy to access, modify, and return.7 However, PaaS systems provide little support for collaboration among developers and there are no specific communication mechanisms provided to them. Moreover, version control can only be implemented in local environments in an offline mode—although this might change in the future, as Google App Engine recently began to support version control by integrating with Google Code (code.google.com).

Enterprises developing and hosting applications on a PaaS system conduct the development process in a closed way. There’s no business value transfer between providers and requesters: enterprises fully own the IP rights to their data and applications, and the PaaS vendor charges them for resource consumption such as for storage and network bandwidth. In return, enterprises often charge their application users for services in a software-as-a-service model.

Open Source Platform
Open source software (OSS) platforms such as Sourceforge.net and Google Code provide an open platform for users to find, download, create, and publish OSS for free. Users are encouraged to contribute to open source projects as co-developers by submitting additions such as code fixes, bug reports, and feedback.

OSS platforms provide comprehensive support for communication and collaboration by providing various communication mechanisms such as mailing lists, forums, blogs, and wikis. They also integrate version control systems and issue trackers to support collaboration. In contrast to traditional centralized software development, organization structure and roles in an OSS project aren’t clearly defined. Coordination such as conflict mediation is conducted democratically, for example, by voting or using moderator mechanisms.

Open source platforms don’t support transfer of business value between requesters and providers. Developers involved in OSS projects don’t seek monetary rewards but do pursue technical challenges. The source code of an OSS product is available, but the rights to study, change, and distribute it are usually constrained by a license.

Collaborative Testing Platform
Collaborative testing platforms such as UT es (www.utest.com) provide services for enterprises that support various testing types such as functional, usability, localization, load, and security testing.

With a collaborative testing platform, any crowd member can register as a tester. The platform provides online learning materials for registered testers and rates their capabilities. Enterprises deliver various testing tasks on the platform, and the platform then assigns a set of testers for each task based on testing requirements and tester capabilities. Enterprises are charged for delivered testing tasks, and testers are rewarded according to their effort evaluation (for example, the number of bugs found).

For each task, the platform assigns a project manager to help the enterprise coordinate the whole testing progress. To support communi-
cation, it provides forums for general discussion, instant messengers for real-time chatting, discussion threads for conversations on various reports, and direct emails for contact and invitation. Testers involved in a task work independently, so there’s little collaboration among them.

**Enterprise Collaboration Platform**

Enterprise collaboration platforms such as IBM Jazz (https://jazz.net) support collaborative software development within an enterprise, offering full life-cycle tools and process support by integrating information and tasks across different phases.

This kind of platform provides comprehensive and extensible support for each aspect of collaborative software development. Project members can communicate with each other through integrated instant messengers, and group awareness is supported in various ways, including workspace awareness through email notification and RSS, group structural awareness through process and team management features, informal awareness by integrating with messengers, and social awareness through integration with enterprise social software. These platforms integrate version control systems, issue trackers, and build tools to enable project members to work collaboratively. As for coordination, they provide Web-based dashboards and process planning and management facilities.

**Trends in Collaborative Platforms**

Crowdsourcing software development platforms must be able to support both value transfer between crowd members and enterprises and the large-scale collaboration of distributed individuals and groups. We can learn a lot about future collaborative software development tools when we look at the trends in crowdsourcing platforms.

Crowdsourcing platforms will most likely continue to integrate more facilities for communication, collaboration, coordination, and awareness—similar to what’s supported in open source and enterprise collaboration platforms. By integrating version control systems and issue trackers, crowdsourcing platforms can better help crowd members collaboratively work on individual tasks and even whole projects. In return, through the use of email notification, RSS feeds, and dashboards, crowd members become better aware of the related processes, tasks, organizations, and project statuses. Crowdsourcing platforms still need better support for cross-task coordination for complex enterprise projects. Crowdsourcing projects also need a focus on team building, just like any team-building efforts in traditional enterprise projects, but the models will be different—for example, they’ll need to consider the characteristics of crowd-based virtual teams, such as competition and loosely coupled team members.

To provide more efficient development environments for crowd members, it’s worth folding in the advantages of PaaS systems and their on-demand provision of development tools and resources. For example, it’s quite easy for a group to conduct its development tasks if the platform can automatically allocate the required resources, such as virtual machines, tools, libraries, and testing environments. Activity-based computing might provide viable theoretical foundations for developing collaborative software development environments to better support the division of labor, task-centric resources and tool aggregation, and community-based knowledge management and sharing.

The next generation of crowdsourcing platforms will also need to combine internal and crowd-oriented development. Some critical or confidential components will naturally be assigned to internal groups and others will be crowdsourced. With support, however, an enterprise can integrate and manage all the related tasks in a unified way.

By merging best practices from open source development and outsourcing, crowdsourcing leverages and stimulates energy toward distributed value creation. Its popularity only continues to grow: over 600,000 people have registered on the TopCoder website so far, and 15 percent of them have participated in at least one algorithm competition.
Although current crowdsourcing platforms have well-defined business models, they lack comprehensive support for building virtual teams and collaborative development among crowd members. In the near future, we foresee collaborative development tools and environments combining with crowdsourcing business models to form the next generation of platforms to foster more crowdsourced software development.

Acknowledgements
Xin Peng’s work is supported by the National High Technology Development 863 Program of China under grant number 2013AA01A605. Ali Babar’s work is partially funded by the Danish Council for Strategic Research under project 10-092313, “Next Generation Technology for Global Software Development—NeXGSD.”

References

XIN PENG is an associate professor at Fudan University. His research interests include requirements engineering, software maintenance, and self-adaptive systems. Peng received a PhD in computer science from Fudan University. Contact him at pengxin@fudan.edu.cn.

MUHAMMAD ALI BABAR is a professor and chair of software engineering in the School of Computer Science at the University of Adelaide. His research interests include software engineering, software architectures, cloud computing, and global software engineering. Ali Babar received a PhD in computer science and engineering from the University of New South Wales. Contact him at ali.babar@adelaide.edu.au.

CHRISTOF EBERT is managing director at Vector Consulting Services. He’s a senior member of IEEE and is the editor of the Software Technology department of IEEE Software. Contact him at christof.ebert@vector.com.