Social Media for Software Engineering

Andrew Begel, Robert DeLine, Thomas Zimmermann
Microsoft Research
One Microsoft Way
Redmond, WA 98052 USA
{andrew.begel,rdeline,tzimmer}@microsoft.com

ABSTRACT

Social media has changed the way that people collaborate and share information. In this paper, we highlight its impact for enabling new ways for software teams to form and work together. Individuals will self-organize within and across organizational boundaries. Grassroots software development communities will emerge centered around new technologies, common processes and attractive target markets. Companies consisting of lone individuals will able to leverage social media to conceive of, design, develop, and deploy successful and profitable product lines. A challenge for researchers who are interested in studying, influencing, and supporting this shift in software teaming is to make sure that their research methods protect the privacy and reputation of their stakeholders.

Categories and Subject Descriptors

K.4.3 [**Organizational Impacts**]: [Computer-supported collaborative work]; H.5.3 [**Group and Organization Interfaces**]: [Organizational design, Computer-supported cooperative work, Web-based interaction]

General Terms

Human Factors, Economics, Management

Keywords

Privacy, Social Networking, Software Engineering, Web 2.0

1. INTRODUCTION

Over the past decade, researchers have given increased attention to the social aspects of software engineering, both to test hypotheses about software development (e.g. socio-technical congruence [7]) and to create tools to improve practice (e.g. team awareness tools [26]). We can find inspiration for further understanding in organizational science, which has long studied the social aspects of how teams work. Applying one model of teaming from Tuckman and Jensen [30] to the software lifecycle, we see that software engineers first organize into teams (*forming*), come to consensus about their goals (*storming*), choose and implement their software methodology and engineering processes (*norming*), collaborate and coordinate with one another to create a new product (*performing*), and finally reflect on their accomplishments and failures in order to improve themselves in their next endeavor (*adjourning*). The social processes around software development are thus highly dependent on engineers' abilities to find and connect with individuals who share similar goals and complementary skills, to harmonize each team member's communication and teaming preferences, to collaborate and coordinate during the entire software lifecycle, and advocate for their product's success in the marketplace.

Unfortunately, these aspects of teaming are difficult to enact successfully and can lead to poor project outcomes. Software engineers typically spend a large fraction of their work day communicating with their coworkers in order to exchange tacit knowledge and coordinate shared work [21, 25, 20]. While electronic media such as email, instant messaging and audio conferencing are used for these purposes, there is a strong bias for face-to-face communication within and between teams [2]. Yet, necessary communications do not occur often enough (or at all) [1], include too many or too few stakeholders [8], and can all too easily misconvey the intentions and priorities of the members of teams who do not share a common identity or knowledge base [19]. Some of the factors that cause these communication problems include distance [18, 22], large team size [3], lack of awareness [16], poor knowledge flow and communication breakdowns [9], and even architectural modularity designed into the software [10].

As early as the 1960s, researchers felt that global networking technologies (some of which became the Internet) had the potential to improve communication and coordination between coworkers [12]. Various forms of media developed since then, such as email, file sharing, instant messaging, the Web, search engines, and audio and video conferencing have all helped communities of coworkers to create and maintain relationships with their colleagues. Each generation of communication and collaboration technology is more lightweight, easy to use, and scalable than the last. The current generation of technology, collectively known as *social media* or *Web 2.0*, adds to the ubiquitous and searchable medium of the Web by making it easy for users to share information with the people in their social network. The utilization of social relationships enhances both the production and consumption of information. For content authors, social relationships make the audience visible and personally familiar, providing a motivation for creating content and enabling the content to be tailored to the audience. For information consumers, social relationships make the content more relevant, rapidly available, and make all consumers of the same content visible to one another so that communities may form around them.

FoSER 2010, November 7–8, 2010, Santa Fe, New Mexico, USA.

In this paper, we describe the potential for social media to both improve communication and coordination in software development teams and support the creation of new kinds of software development communities. We begin with a short overview of how software engineers use existing social media to improve their work practice and relationships with customers. We then reference Tuckman and Jensen's teaming model to describe some future scenarios that the widespread use of social media would afford. Finally, because social media are inherently very personal, we describe some challenges that software engineering researchers face when designing social media tools for communities of developers, namely respecting individual reputations and privacy.

2. CURRENT USE OF SOCIAL MEDIA

Considering that software engineers are creators and early adopters of new communication technologies, it comes as no surprise that they are already using modern social media tools to organize software development-related work more efficiently. Here we provide short descriptions of these media and their current use.

Blogs are an ongoing series of short articles from the same author (or group of authors) on narrow topics, with feedback in the form of reader comments. Microsoft and other software companies frequently use blogs to share technical information and opinions with their employees, and very profitably, with their customers, both internal and external. Blogs and other social media, in particular, support scalable, two-way communication, enabling a company to quickly react to customer feedback (positive or negative), and use this feedback to help design more targeted software features and marketing messages.

Microblogs like Twitter are "lightweight" blogs that enable users to share extremely short (160 characters) messages with a set of "followers" who have subscribed to receive them. Messages can be read and written on many devices, even phones that only support SMS text messages, enabling anytime, anywhere access to timely information blasts. Among engineers, messages are written to publicize links to technical information, to announce spontaneous meetings, and to keep coworkers apprised of status and progress on work-related tasks [29]. To keep this status information confidential within corporations, new companies like Yammer [32] are providing microblogging services for the enterprise.

Social networking sites such as Facebook and LinkedIn build a searchable and browsable graph of social (including work-related) relationships between people. Information about a person (e.g. their status, photos, recommendations, questions, or expertise) can be obtained from neighboring edges in the graph and made available to that person's "friends." In open source software development, social networking sites are fairly common. Ohloh.Net [14] and Github [15] are web portals that connect people through the software projects they create and use. Our own project, Codebook, is a platform inspired by social networking, and connects Microsoft employees to one another not only through social connections, but also through automatically discovered relationships between their shared work artifacts (e.g. code, bugs, documentation, etc.) [1].

Community Q&A sites such as Yahoo! Answers or Answers.Com, are places for asking and answering questions about any topic posed by someone visiting the site. Often, they are used for seeking opinions, such as products to buy (e.g. the best lawn mower) or worthwhile places to visit on vacation (e.g. what one should see when touring Lisbon). Users can apply tags to questions and answers to identify topic areas and apply badges to reward users who offer good questions and useful answers. Searching by tags and filtering by badges helps novices sift through search results to find the highest quality information. Stack Overflow [28], a popular Q&A site for software engineers, focuses on technical questions about computer programming. As of September 2010, the site has 750,000 answered questions, 155,000 unanswered questions, and employs just under 30,000 technology-related tags, all applied to the questions and answers by site visitors.

3. THE POTENTIAL OF SOCIAL MEDIA FOR SOFTWARE ENGINEERING

Open source and distributed software development communities already take advantage of Internet-based communication and coordination technologies to function effectively. As social media has gained adoption, opportunities for creating software in new ways have risen to enhance and augment the old.

3.1 Forming

Social networking sites often provide a complete environment to enable "communities" of people to self-organize online, report their current status, and stay aware of the status of the people in their communities. While many social networking communities originated offline, as the services have grown, online-only communities have formed, including grassroots political movements (e.g. the Obama 2008 presidential campaign), celebrity, TV and brand fan groups, computer science conference outreach groups, and more.

We see grassroots software development communities forming around technologies (e.g. users of a language or library), processes (e.g. Agile teams), markets (e.g. teams shipping products to developing countries), or user communities (e.g. teams designing for end-user programmers) [4]. Company-internal social networking web sites enable workers to find and communicate with colleagues who have the expertise to handle a particular business situation, or to find and recruit qualified workers to work on a team that will create a new product (i.e. "skunkworks" projects) [11]. Even in a "Company of One," a lone software developer can conceive of an idea for a new application and use social media to find, connect with, and work with people (and the knowledge they have spread and left behind on social media web sites) who have the talent and skills necessary to help that developer build an application.

3.2 Storming

As new communities of software engineers organize into teams, they must communicate their aspirations and goals and come to consensus about their shared purpose. Online forums enable coworkers to discuss application ideas, future trends, potential markets, and their own personal requirements for working in a team. Some engineers use blogs like position papers; they stand on their electronic soapboxes and call for particular technological or user problems to be solved. Corporations like Microsoft have also used blogs, microblogs, and social networking personas as a form of marketing (e.g. to announce products) to their customers. The two-way communication channel that these social media support offer the added benefit to receive customer feedback (positive and negative) in a scalable manner that was infeasible using older media forms. This feedback can be used to quickly identify and respond to the changing user needs of the target customer group.

Any size company, even one with a single developer, can leverage pre-existing low-cost, low-overhead marketplaces (e.g. the Apple iPhone, Android, or Microsoft App Stores) to make their software available and visible to millions of potential customers. They can then employ social networking sites to virally market the software, first to friends and work colleagues of the company founders and employees, and from there to millions of potential customers who hear about the product from friends of friends of friends. Social tagging built into the App Stores enable customers to rate and

provide feedback on application design and performance, which helps the company realign and focus on winning more business.

3.3 Norming

Teams using social media must learn to communicate and coordinate effectively in order to successfully build a product. This requires processes and tools to support knowledge management, knowledge transfer, team awareness, and mutual cooperation.

Microblogs are currently being used by engineers to spread knowledge, ideas, and suggestions to others in their work communities. As the communities grow more connected through their persistent use of social media, the distribution of knowledge within the community can become more complete more quickly, minimizing misunderstandings between colleagues who do not meet face-to-face very often, or at all.

Grassroots organizational processes have been used by open source software development (FLOSS) for decades, but by co-opting FLOSS' successful methods for supporting distributed team collaboration and software development, small teams of industrial software developers can now coordinate to create products with more agility and responsiveness to customer needs. At the same time, these teams can microblog their accomplishments and attract the attention of corporate leadership, even if the traditional organizational hierarchy puts up barriers to disseminating those innovations across, and especially, up, the hierarchy.

3.4 Performing

Once a team gets down to the business of creating software, social media tools can be leveraged in all the ways mentioned in Section 2 to help the individuals and subteams find relevant expertise, coordinate work and schedules, find and spread information about their code, bugs, documentation, shared resources, and monitor their progress towards success using the electronic trail left behind as they use the tools.

Social bookmarking sites such as Delicious [31] or CiteULike [23] support two synergistic goals. For an individual software engineer, they offer a highly available, web-based storage site for links and documents. In making these links visible either to the public, or anyone in the engineer's social network, they accomplish a second, altruistic goal of knowledge sharing without extra effort. Similarly, programming tools in the cloud could inspect the work of participating teams to infer best practices (e.g. API usage rules, user interface designs, or licensing schemes), which could then be promoted back to the contributing software engineers as recommendations.

Crowd sourcing tools such as Amazon's Mechanical Turk (MTurk) can enable smaller teams to avoid the challenges of engineering complex software. Employing "artificial" artificial intelligence, engineers divide up their desired computation into many small tasks that humans are good at solving (e.g. CAPTCHAs), and that computers are not. Then, MTurk's social work distribution system attracts site visitors (Turkers) to compute each task for a micropayment of one to ten cents. Callison-Burch evaluated MTurk as a platform for performing speech recognition [5], and found that the quality of the resulting transcriptions exceeded online machine translation systems. In fact, two very small companies, CastingWords [6] and SpeakerText [27] now exploit MTurk's crowd sourcing capabilities to sell speech transcription services at rates much lower than average.

3.5 Adjourning

Once a product has shipped and the team is ready to disband, each team carries out a post-mortem of the product plan, design, development and marketing process, with the desire to improve their processes and tools for their next project, done together or with new collaborators. The complete record of the team's interactions using social media presents an attractive artifact to preserve institutional memory not just inside the head of a team "historian" [21], but accessible by all team members.

4. CHALLENGES

An intrinsic attribute of social media requires revealing data about individuals who employ the services. As engineers say, this is both a feature and a bug. Researchers interested in studying how social media is used by software engineers and how this use may evolve to support new kinds of teaming structures must conduct their work with care. First, social media have a strong influence over the reputations of the participants, which in turn has career consequences within an organization or community. Researchers, even when trying out prototypes, have a responsibility to ensure that the reputations that the tools help people to form are consistent with the values of the individuals and organizations in which they are involved. Second, social media publish new information and make existing information about individuals available in new ways. While all researchers want to protect the privacy of their tool users, in the light of current European privacy laws, this goal is also a legal requirement.

4.1 Protecting Reputations

A social medium that reveals a user's knowledge, expertise, activities, or availability has a different "feel" when consumed by the user's peers than it does when consumed by his manager. The former could seem like a useful tool for finding and connecting with a knowledgeable and helpful colleague, while the latter could feel like corporate spyware. A poor design for a social media tool for software engineers has the potential to create individual reputations that are inaccurate or inconsistent with the team's goals and values. More subtly, such a tool can also create an incentive for users to change their actions in order to improve how they are reflected in the tool, that is, to "game" the system.

Traditional approaches to protecting individuals, such as anonymity and aggregation, are possible in realm of social media, but can come at the cost of usability. For example, a recommendation system whose purpose is to suggest experts in a particular topic area can preserve the experts' anonymity by using software as a go-between, but at the cost of eliminating highly desirable face-to-face conversations. Similarly, to ensure individual privacy to non-teammates, a team awareness tool could show only an aggregate of all team members' activity, but prevents coworkers within the team from using the system to coordinate their shared work with one another.

An alternate approach is to use research methods like participatory design [13] or action research [17], in which the researchers and user community form a long-term partnership, to ensure the authenticity, utility, relevance, and ethics of user needs and tools created to support these needs. Indeed, the informal motto of these methods is "nothing about us without us." The benefit of these methods is that the values of the individuals and enterprises being studied are considered at every stage. The downside is that the research may lack generality and may need to be replicated in different settings.

4.2 Protecting Privacy

The point of social media tools is to reveal information about individuals, at least to those who are socially connected to them. However, these tools raise privacy concerns, even if they merely recombine information already considered "public." For researchers

working with European professionals, protecting privacy is a legal requirement, based on European Union Directive 95/46/EC [24]. At a high level, this Data Protection Directive allows tools to process data that identifies individuals only with the explicit consent of those individuals, and only to the extent necessary to meet a legitimate business purpose. Many large software companies are multinational, and include at least some European employees. These companies often choose to apply the European standard to both European and non-European employees to avoid creating different classes of employees.

While supporting communication and collaboration may be a legitimate purpose for software companies (the authors know of no court cases that have tested this issue), obtaining explicit consent from all users of a social media tool is not straightforward. First, many useful tools are based on data mining, for example, extracting information from code revision histories or bug databases. The utility of these tools is that they provide an instant "critical mass" of data. If a tool is only allowed to mine data from those users who have given explicit consent, then the team as a whole may never see the value of the tool because too few users have joined. Second, requiring consent again creates two classes of employees: those who have given consent and those who have not. The latter class will be invisible in the tool, which may adversely affect their reputations. For example, a team awareness tool that displays the activities of all team members except one (who withheld consent) will likely place that hidden member at a disadvantage with respect to coordination, communication, and reputation.

5. CONCLUSION

Software development is inherently a social activity involving interconnected communities of engineers and users. In this paper, we propose how social media and Web 2.0 technologies can improve software engineering practices, and, in particular, how they can enhance the ways that people come together to form teams, agree on their goals and practices, communicate and collaborate with one another to build applications, and reflect on their successes and failures in their work. Researchers interested in pursuing studies of these new teaming practices and helping to support them with new tool designs must take care to respect the reputation and privacy of each stakeholder, in order to ensure regular collaboration and cooperation with research goals.

6. REFERENCES

[1] A. Begel, Y. P. Khoo, and T. Zimmermann. Codebook: Discovering and exploiting relationships in software repositories. In *Proceedings of ICSE*, 2010.

[2] A. Begel, N. Nagappan, C. Poile, and L. Layman. Coordination in large-scale software teams. In *Proceedings of CHASE*, pages 1–7, 2009.

[3] F. P. Brooks, Jr. *The Mythical Man-Month*. Addison-Wesley Professional, 2nd edition, 1995.

[4] J. S. Brown and P. Duguid. Organizational learning and communities-of-practice: Toward a unified view of working, learning, and innovation. *Organizational Science*, 2(1), 1991.

[5] C. Callison-Burch. Fast, cheap, and creative: Evaluating translation quality using amazon's mechanical turk. In *Proceedings of EMNLP*, pages 286–295, 2009.

[6] CastingWords, LLC. Castingwords transcription services. http://www.castingwords.com.

[7] M. Cataldo, D. Damian, P. Devanbu, S. Easterbrook, J. Herbsleb, and A. Mockus. 2nd international workshop on socio-technical congruence, May 2009.

[8] M. Cataldo, P. A. Wagstrom, J. D. Herbsleb, and K. M. Carley. Identification of coordination requirements: implications for the design of collaboration and awareness tools. In *Proceedings of CSCW*, pages 353–362, 2006.

[9] B. Curtis, H. Krasner, and N. Iscoe. A field study of the software design process for large systems. *Communications of the ACM*, 31(11):1268–1287, 1988.

[10] C. R. B. de Souza, D. Redmiles, and P. Dourish. "Breaking the code", Moving between private and public work in collaborative software development. In *Proceedings of GROUP*, pages 105–114, 2003.

[11] K. Ehrlich, C.-Y. Lin, and V. Griffiths-Fisher. Searching for experts in the enterprise: combining text and social network analysis. In *Proceedings of GROUP*, pages 117–126, 2007.

[12] D. Engelbart. A conceptual framework for the augmentation of man's intellect. In P. Howerton and D. Weeks, editors, *Vistas in Information Handling*, volume 1, pages 1–29. Spartan Books, Washington, D.C., 1963.

[13] C. Floyd, W.-M. Mehl, F.-M. Resin, G. Schmidt, and G. Wolf. Out of scandinavia: Alternative approaches to software design and system development. *Human-Computer Interaction*, 4(4):253–350, December 1989.

[14] Geeknet, Inc. Ohloh, the open source network. http://www.ohloh.net.

[15] GitHub, Inc. Github social coding. http://github.com.

[16] C. Gutwin, R. Penner, and K. Schneider. Group awareness in distributed software development. In *Proceedings of CSCW*, pages 72–81, 2004.

[17] G. Hearn and M. Foth. *Topical Issues in Communications and Media Research*, chapter Action Research in the Design of New Media and ICT Systems, pages 79–94. Nova Science, New York, NY, 2005.

[18] J. D. Herbsleb and R. E. Grinter. Splitting the organization and integrating the code: Conway's law revisited. In *Proceedings of ICSE*, pages 85–95, 1999.

[19] P. Hinds and C. McGrath. Structures that work: social structure, work structure and coordination ease in geographically distributed teams. In *Proceedings of CSCW*, pages 343–352, 2006.

[20] R. E. Kraut and L. A. Streeter. Coordination in software development. *Communications of the ACM*, 38(3):69–81, 1995.

[21] T. D. LaToza, G. Venolia, and R. DeLine. Maintaining mental models: a study of developer work habits. In *Proceedings of ICSE*, pages 492–501, 2006.

[22] A. Mockus, R. T. Fielding, and J. D. Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM TOSEM*, 11(3):309–346, 2002.

[23] Oversity, Ltd. Citeulike. http://www.citeulike.org.

[24] E. Parliament and C. of the European Union. EU Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the Protection of Individuals with Regard to the Processing of Personal Data and on the Free Movement of such Data. *Official Journal of the European Communities*, L(281), November 1995.

[25] D. E. Perry, N. Staudenmayer, and L. G. Votta. People, organizations, and process improvement. *IEEE Software*, 11(4):36–45, 1994.

[26] A. Sarma, Z. Noroozi, and A. van der Hoek. Palantír: raising awareness among configuration management workspaces. In *Proceedings of ICSE*, pages 444–454, 2003.

[27] SpeakerText. Speakertext: Read my clips. http://www.speakertext.com.

[28] Stack Overflow Internet Services, Inc. Stack overflow. http://stackoverflow.com.

[29] C. Treude and M.-A. Storey. Awareness 2.0: Staying aware of projects, developers and tasks using dashboards and feeds. In *Proceedings of ICSE*, Cape Town, South Africa, May 2010.

[30] B. W. Tuckman and M. A. C. Jenson. Stages of small group development revisited. *Group and Organizational Studies*, 2(4):419–427, 1977.

[31] Yahoo! Delicious. http://delicious.com.

[32] Yammer. Yammer: Enterprise microblogging. http://www.yammer.com.