The Impact of Social Media on Software Engineering Practices and Tools

Margaret-Anne Storey, Christoph Treude
University of Victoria, BC, Canada
{mstorey,ctreude}@uvic.ca

Arie van Deursen
Delft University of Technology
Delft, the Netherlands
Arie.vanDeursen@tudelft.nl

Li-Te Cheng
IBM TJ Watson Research Lab
Cambridge, USA
li-te_cheng@us.ibm.com

ABSTRACT
Today's generation of software developers frequently make use of social media, either as an adjunct or integrated into a wide range of tools ranging from code editors and issue trackers, to IDEs and web-based portals. The role of social media usage in software engineering is not well understood, and yet the use of these mechanisms influences software development practices. In this position paper, we advocate for research that strives to understand the benefits, risks and limitations of using social media in software development at the team, project and community levels. Guided by the implications of current tools and social media features, we propose a set of pertinent research questions around community involvement, project coordination and management, as well as individual software development activities. Answers to these questions will guide future software engineering tool innovations and software development team practices.

Categories and Subject Descriptors
D.2 [**Software Engineering**].

General Terms
Human Factors.

Keywords
Social media, web 2.0, software engineering tools.

1. INTRODUCTION
Software engineering is a highly collaborative activity, at the team, project and community levels. To facilitate coordination, developers need to maintain awareness of team members' activities, awareness of the workspace and project, and also social awareness of the interests and opinions of all stakeholders [4]. Communication of information is also a critical component in collaborative development, especially when designing large scale modern software systems that have to meet the needs of a diverse group of users and stakeholders. At the community level, developers can furthermore learn and collaborate with other developers on solving problems.

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

Understanding collaboration processes in software development and designing effective collaborative tools are active topics in software engineering research. Modern integrated development environments (IDEs) and software project portals integrate many features to support collaboration, coordination and communication. Example features include version control, mailing lists, and issue tracking. These environments may also provide additional lightweight collaboration features, such as chat, to increase awareness and support informal communication. The need to have tool support for both formal and informal activities is well recognized in software engineering [10].

Today's generation of developers frequently makes use of social media, also referred to as Web 2.0, to augment tools in their development environments [1]. That today's developers use social media to support collaborative development is not surprising as the computer industry is currently witnessing a paradigm shift in how everyday users work, communicate, and play over the Internet. This paradigm shift is due to the decentralization of computer systems and also to the wide array of innovative social media tools that are being adopted and adapted by this new generation of users. For example, social media applications such as Facebook and Twitter are household names.

Social media tools can be characterized by an underlying "architecture of participation" that supports crowdsourcing as well as a many-to-many broadcast mechanism [14]. Their design supports and promotes collaboration, often as a side effect of individual activities, and furthermore democratizes who participates in activities that were previously in the control of just a few stakeholders.

Software engineers make use of a variety of social media tools to coordinate with one another, to communicate with and learn from users, to become informed about new technologies and to create informal documentation. Despite the apparent widespread use of these technologies within software engineering, there is little known about the benefits or risks of using such tools, and the impact they may have on the quality of software. There is also little advice on how, or indeed if, social media features should be integrated within modern software development environments.

In this position paper, we advocate for future research on understanding how social media plays a role in collaborative activities at the team, project and community levels. This research is needed so that we can guide future software engineering tool innovations as well as practices. In the following, we first review the wide variety of tools that support individual, team, project and community collaborations in software engineering (Section 2). In Section 3, we discuss specific social media features, and how they currently play a role across the spectrum of development tools. In

Section 4, we provide a research outlook guided by the implications of current tools and social media features, focusing on research questions on community involvement, project coordination and management as well as individual software development activities. Finally we discuss some of the challenges with conducting this research and conclude the paper in Section 5.

2. COLLABORATION AND SE TOOLS

To support individual and team-based development activities, software engineers can choose from a wide variety of tools ranging from code editors and issue trackers to integrated development environments and web based portals for hosting projects. In this section we review this spectrum of tools from a collaboration perspective.

2.1 Integrated Development Environments

IDEs were initially designed as a soloist tool, with features for the lone engineer to author, debug, refactor, and reuse code. Larger scale software projects are team based and require management of the artifacts under development. Thus most IDEs also have data and control integration mechanisms to support team-based activities through version control, release management and issue tracking tools. In addition to these tools, mailing lists, forums and wikis are commonly used to manage collaboration. These integrations predominantly rely on the Internet or the intranet as a platform. They are critical in managing task articulation work during distributed development. Task articulation refers to how engineering tasks are identified, assigned and coordinated [6]. Without such tool support, we would not have seen the current growth of global software development.

2.2 Collaborative Development Environments

Recently, the design of IDEs is leaning towards a federation of tools, as well as the addition of collaboration features [4]. IDEs that are designed with collaboration as a major focus are referred to as Collaborative Development Environments (CDEs) [11]. The primary goal of CDEs is to reduce friction in collaborative processes. Jazz, one example of a CDE, supports the integration of tools along the project lifecycle[1]. The Jazz environment furthermore provides a web interface for accessing collaboration information such as issue tracking and a web interface for customizing dashboards to provide information on project status [21]. This web interface extension of the CDE helps facilitate community involvement as they do not need to use the full blown IDE to browse and augment project information. The Microsoft Team Foundation Server has similar features to Jazz such as work item tracking and a team project portal to support collaboration[2].

2.3 Software Project Portals and Forges

There are a wide variety of web based portals to support software development activities. Some of these portals are specifically designed for hosting projects, whilst others are for exchanging information or community building. Source code project forges host independent projects on a website. Software Forge applications provide integrated web interfaces and tools, such as forums, mailing lists, wikis, bug trackers, and social networking, for accessing and managing the stored projects in a collaborative manner. Examples include SourceForge and Google Code[3].

Forges may also have specialized software for setting up the project, with services for downloading the archives, and services for setting up and maintaining mailing lists, forums, wikis and bug trackers. Forges are very popular for open source projects, and lately integrate more and more social media features. For example, Github[4] markets itself as a "social coding environment", because of its underlying philosophy in supporting social interactions around the project. Github combines social networking with the Git distributed source-control; developers can follow other developers and they can watch specific projects, for example via "activity streams".

There are also other innovative websites that support developers in exchanging information and managing collaborative work. Stackoverflow[5] is a community site where developers can ask and answer each others' questions. TopCoder[6] hosts programming competitions with the underlying business goal to connect companies with talented programmers. More recently, it is acting as an outsourcing service, where companies can assign challenging tasks to TopCoder programmers. Freshmeat[7] helps users keep track of software releases and updates, and hosts reviews and articles about software projects.

Some websites further integrate source code development features (such as authoring and compiling) within project websites. An interesting example is Skywriter[8], Mozilla's experiment with a HTML5-based code editor running entirely in a browser. Skywriter offers support for versioning and following of co-developers. The move of the IDE towards the browser further opens up opportunities for integration of social features [22].

These social development websites share the common theme of providing support for development activities using web-based social mechanisms. We use the term Social Development Environment (SDE) to refer to this broad spectrum of websites.

3. SOCIAL MEDIA AND SE

The use of social media crosscuts many of the traditional categories in software development: it goes across teams, projects and communities, and integrates a wide range of development processes and tools from IDEs to CDEs and SDEs. Social media usage can support software development activities ranging from requirements engineering and development to testing and documentation. The lightweight nature of social media channels allows developers to adapt them to their current context and has the potential to revolutionize the way collaborative software development is done. We are starting to see social media features becoming adopted and either used as an adjunct, or being directly integrated in the development environments. This is particularly the case for SDEs which make use of social media features extensively. To understand how social media is used to support software engineering, we take a crosscutting view. We first

[1] http://jazz.net
[2] http://msdn.microsoft.com/en-us/teamsystem/
[3] http://www.slideshare.net/oostendo/the-oss-forge-ecosystem-today-and-tomorrow
[4] http://www.github.com/
[5] http://stackoverflow.com/
[6] http://www.topcoder.com/
[7] http://freshmeat.net/
[8] http://mozillalabs.com/skywriter (formerly named Bespin)

review how various social media channels are being used to support collaborative software development practices, and we then review existing research that has specifically investigated the use of social media within software engineering.

3.1 Social Media Features in SE Practice

Wikis were designed with development collaboration directly in mind [12]. Since Wikis have been around for some time, their adoption in software engineering is widespread. Wikis are used to support defect tracking, documentation, requirements tracking, test case management and for the creation of project portals [13].

Blogs are frequently used by developers to document "how-to" information, to discuss the release of new features[9] and to support requirements engineering [15]. Moreover, some practitioners advocate that every developer should have a blog[10], arguing that blog posts help exchange technical knowledge among a larger audience than email messages.

Microblogs such as Twitter are also actively used to exchange information between developers[11]. They provide support for lightweight coordination and communication. Recent research proposes the inclusion of microblogging in the IDE [8] [17].

Tagging (social bookmarking) is used in many project portals (such as Github) for tagging issues, in Freshmeat for tagging project releases and by the Jazz CDE for tagging work items and builds [20]. A research project called TagSEA also explored the role of tagging of source code and artifacts within the IDE [19]. The main advantages of using tags in software development are their flexibility and their lightweight, bottom-up nature.

Feeds are used on the Internet to provide subscribers with updates from websites with frequently updated content. In software development, tools such as the Jazz CDE and forges use feeds to provide awareness about workspaces, developers and processes.

Social networking allows for the creation of "virtual communities" through sites such as Facebook. Services such as the web-based hosting service Github provide social networking functionality to display how developers and their work on versions are connected. In Codebook [3], developers can "become friends with work artifacts", keep track of dependencies and discover connections using a web interface.

Mashups combine data or functionality from several external sources. In software development, users can further participate as co-developers by leveraging data and services that are exposed to them through Mashup technologies [7]. As Raman notes, we are moving from a "web of documents, to a web of applications" [16].

Crowdsourcing: Through social media, users act as co-developers by providing requirements for new features and feedback on bugs. Google uses crowdsourcing to help in testing alternative designs, using an A/B testing approach[12], where two versions of a webpage are developed and each one is presented to two different groups of users. Statistics, such as click-throughs, are collected for both to see which page version is more effective.

3.2 Research Studies on Social Media in SE

Despite the apparent widespread use of Web 2.0 in collaborative software development and the wide array of social media features used, there have only been a sprinkling of empirical studies that investigate the adoption rate and implications from social media use in software engineering. These studies include how tagging is used to support collaboration around source code [19], how tagging is used for documenting tasks in the Jazz Environment [20], a study of how web based dashboards and feeds are used to support awareness in collaborative development [21] and how developers use Wikis for documentation [5]. The use of Codebook by software developers has also been studied [3].

For the most part, the studies done so far report on how one or two forms of social media are used in isolated projects, but this research does not consider the bigger picture of how a wide variety of social media is used in conjunction with traditional tools to support collaborative software engineering. One exception is a paper that presents a case study of how Web 2.0 features are used by practitioners on a single open source project [18]. Their paper describes how to use the features from the perspective of other practitioners wishing to adopt them, but the authors do not provide many insights on the implications of using these features.

4. RESEARCH OUTLOOK

So far we have presented an overview of the broad spectrum of tools that exist for collaborative activities in software engineering and described how social media features crosscut these tools to support collaborative activities. These summaries provide a technical perspective of the tools that are available and provide some preliminary insights on how they are used. However, as discussed in Section 3, there is considerable evidence that these tools are being used in innovative ways in practice, yet there have been very few research studies that articulate the benefits and limitations of these tools on software engineering. From our review of how social media is used in practice, the following questions about the possible implications from the use of these tools emerged:

- How can social media play a role in increasing **community** and **end user** involvement in software engineering?
- How can social media facilitate **project coordination** and **management**?
- How can social media improve individual **software development activities**?

We discuss each of these potential implications below in turn, and suggest ten research questions that could form the basis for future research projects.

4.1 Community and End User Involvement

Web 2.0 and social media are designed with an "architecture of participation" in mind [14]. The increasing use of the Web as a platform for information based systems has led to services-based software development and "ultra-large-scale" software systems. Within these large-scale systems, users are not perceived as "passive recipients of goods but as co-creators of value" [9]. Through social media, users can be involved in the software development process in many different ways. For example, the

[9] See http://planeteclipse.org/planet/ for examples

[10] http://channel9.msdn.com/posts/Glucose/Hanselminutes-on-9-Social-Networking-for-Developers-Part-1-Every-Developer-Needs-a-Blog/

[11] See http://wiki.eclipse.org/Twitter for examples

[12] http://www.google.com/websiteoptimizer

Jazz project uses the model of "Open Commercial Software Development"[1], exposing the current development status and allowing the community to contribute to their issue tracking system. The benefits and risks of this approach are not well understood as of yet. In many projects, users also blog about their experiences and opinions, but it is not clear how this kind of information could be used to benefit development activities.

For web application development, end users can be involved using A/B testing. It is an open question to what extent a similar approach could be adapted to applications beyond the web. Going a step further, mashups allow users to build their own applications through the free combination of data and functionality from different sources [2]. For example, mashups may offer great potential to involve users as co-developers that guide future requirements.

The benefits and risks of the various ways that end users and community members can be involved in a project are not well understood. This leads to the following specific research questions:

1. What are the benefits, limitations and risks of using social media for gathering **requirements**?
2. How effective is social media for supporting **testing** of new features and alternate user interface designs?
3. How can social media support **end user development**?

4.2 Project Coordination and Management

Managing and coordinating projects brings many challenges in distributed software development. Challenges arise when teams are distributed across time zones and geographic locations, and lack informal mechanisms for communication. Social media is regarded as something that can replace the virtual water cooler by supporting informal and serendipitous interactions across the team and project, thus supporting task articulation and coordination.

Another important aspect of project coordination is resource sharing. Social media can help people coordinate access to the documents, as well as keep up to date on changes to resources (e.g. through feeds). Social media can also be used by team members and management to access and aggregate information about the status of the project in a lightweight manner, using tools such as web-based dashboards [21]. However, the flip side of this usage may be a discomfort on the part of developers that information about their activities can be easily aggregated and then compared with other developers. These hypothesized benefits and issues lead to the following research questions:

4. Can social media play an effective role in supporting **coordination** and **task articulation**?
5. What kinds of social media would increase **informal communication,** the **flow of knowledge** and **awareness** across team and project members?
6. Can social media help facilitate the coordination of **resource sharing** in project forges and other repositories?
7. What are the drawbacks from increased **transparency** in team projects? Does this lead to **privacy** concerns?

4.3 Software Development Activities

Software development is a highly creative human endeavour, where developers create and maintain software systems by standing on the shoulders of many others. They reuse components and libraries, and will often refer to online resources and tools to find relevant information that will help them in their tasks. Developers will communicate with teammates and developers in their social networks, to tell them about their programming tasks and to ask them for advice. Social media, even at the level of developing code, is playing an important role. In particular, we see the creation of many informal and emergent forms of documentation that can help developers inform and be informed about important aspects of the code that is relevant to them.

Once again, we need to consider not just the benefits of how social media can provide assistance, but we also consider the limitations and risks, not just on the developers' workload and performance, but also on the quality of the resulting software. The following provides additional research questions:

8. What influence will social media have on the **quality** of software that is collaboratively authored?
9. How effective is social media for the shared authoring of both informal and formal **documentation**?
10. Does social media lead to **interruptions** or **information overload** that could impair a developer's performance?

5. DISCUSSION AND CONCLUSION

To answer the research questions posed above, we need a systematic research approach. Understanding the implications of social media use in software development requires studying a wide variety of projects and contexts. The dimensions that should be considered, include, but are not limited to:

- open, closed and crowdsourced projects;
- globally distributed and co-located projects;
- projects across diverse domains (e.g. safety critical embedded software and web-based entertainment software); and
- projects using agile and traditional development processes.

Finding the right set of research methodologies to study the impact of social media is challenging. Social media use is a moving target, and inventions are adopted and adapted faster than we can study them. To understand these fast moving socio-technical environments, we need to focus on development teams as the unit of analysis and study social media in the entire development context, as social media use crosscuts traditional software development categories. This research also has to be aware of the potential drawbacks of social media use to understand the implications that social media has on privacy and security, as well as its influence on productivity and product quality.

In this paper, we have argued how and why social media will play an increasingly important role in software engineering research and practice. We provided an analysis of the current use of social media in software engineering and used this context to identify a series of research questions in the area of social media in software engineering. We hope that answering these questions will help inform both future software development processes and tools.

6. ACKNOWLEDGMENTS

Some of the ideas in this paper were formulated during a workshop held at Dagstuhl in March of 2010 on New Frontiers for Empirical Software Engineering. In particular, we would like to thank the following colleagues for fruitful discussions on this topic: Martin Robillard, Michael Maximilien, Pankaj Jalote, Gregg Rothermel, Forrest Shull and Jim Whitehead.

7. REFERENCES

[1] Ahmadi, N. Jazayeri, M., Lelli, F. and Nesic, S. 2008. A survey of social software engineering. In *ASE Workshops '08: 23rd IEEE/ACM Intl. Conf. on Automated Software Engineering*, Washington, DC, 1-12.

[2] Ahmadi, N. Jazayeri, M. Lelli, F. and Repenning, A. 2009. Towards the web of applications: incorporating end user programming into the web 2.0 communities. In *Proc. of the 2nd Intl. Workshop on Social Software Engineering and Applications, SoSEA '09*. ACM, New York, NY, 9-14.

[3] Begel, A. and DeLine, R. 2009. Codebook: Social networking over code. In *ICSE Companion '09: 31st Intl. Conf. on Software Engineering - Companion Volume*, Washington, DC, 263-266.

[4] Calefato, F., Gendarmi, D., and Lanubile, F. 2009. Embedding Social Networking Information into Jazz to Foster Group Awareness within Distributed Teams. In *Proc. of the Second Intl. Workshop on Social Software Engineering and Applications*, ACM, 23-28.

[5] Dagenais, B. and Robillard, M., 2010. Creating and Evolving Developer Documentation: Understanding the Decisions of Open Source Contributors. In *Proc. of the 18th ACM SIGSOFT Intl. Symposium on the Foundations of Software Engineering*, November 2010. To appear.

[6] Gerson, E. M. and Star, S. L. 1986. Analyzing due process in the workplace. *ACM Trans. Inf. Systems,* 4, 3 (Jul. 1986), 257-270.

[7] Grammel, L. , Treude, C. and Storey, M.-A. 2010. Mashup environments in software engineering. In *Proc. of the 1st Workshop on Web 2.0 for Software Engineering*, *Web2SE '10*. ACM, New York, NY, 24-25.

[8] Guzzi, A, Pinzger, M., and van Deursen, A. Combining Micro-Blogging and IDE Interactions to Support Developers in their Quests. In *Proc. of the 26th Intl. Conf. on Software Maintenance (ICSM)*. IEEE, 2010. Early Research Achievements (ERA) Track. To appear.

[9] Kazman, R. and Chen, H. 2009. The metropolis model a new logic for development of crowdsourced systems. *Communications of the ACM*, 52, 7 (Jul. 2009), 76-84.

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

[11] Lanubile, F., Ebert, C., Prikladnicki, R. and Vizcaino, A. 2010. Collaboration Tools for Global Software Engineering, *IEEE Software*, March/April 2010, 52-55.

[12] Leuf, B. and Cunningham, W. 2001. The Wiki Way: Quick Collaboration on the Web, Addison-Wesley.

[13] Louridas, P.. 2006. Using Wikis in Software Development. *IEEE Software*, Mar. 2006, 88-91.

[14] O'Reilly, T. 2005. What is Web 2.0: Design patterns and business models for the next generation of software, http://oreilly.com/web2/archive/what-is-web-20.html.

[15] Park, S. and Maurer, F. 2009. The role of blogging in generating a software product vision. *In Proc. of the ICSE Workshop on Cooperative and Human Aspects on Software Engineering, IEEE CHASE*. Washington, DC, 74-77.

[16] Raman, T. V. 2009. Towards 2^W, beyond Web 2.0. *Communications of the ACM*, 52, 2 (Feb. 2009), 52-59.

[17] Reinhardt, W. 2009. Communication is the key -Support Durable Knowledge Sharing in Software Engineering by Microblogging. In *Proc. of the SENSE Workshop, Software Engineering within Social Software Environments* (collocated with the Conf. on Software Engineering (SE2009), Germany, http://www.se2009.de/.

[18] Serrano, N. and Torres, J. M. 2010. Web 2.0 for Practitioners. *IEEE Software,* 27, 3 (May. 2010), 11-15.

[19] Storey, M.-A., Ryall, J. Singer, J., Myers, D., Cheng, L.-T. and Muller, M. 2009. How Software Developers Use Tagging to Support Reminding and Refinding. *IEEE Transactions on Software Engineering,* 35, 4 (Jul. 2009), 470-483.

[20] Treude, C. and Storey, M.-A. 2009. How tagging helps bridge the gap between social and technical aspects in software development, In *Proc. of the 31st ACM/ IEEE Intl. Conf. on Software Engineering (ICSE 2009)*, Washington, DC, 12-22.

[21] Treude, C. and Storey, M.-A. 2010. Awareness 2.0: staying aware of projects, developers and tasks using dashboards and feeds. In *Proc. of the 32nd ACM/IEEE Intl. Conf. on Software Engineering (ICSE 2010),* 365-374.

[22] van Deursen, A., Mesbah, A., Cornelissen, B., Zaidman, A., Pinzger, M., and Guzzi, A. 2010. Adinda: a knowledgeable, browser-based IDE. In *Proc. of the 32nd ACM/IEEE Intl. Conf. on Software Engineering - Volume 2. (ICSE 2010)*. ACM, New York, NY, 203-206.