Workshop Report from Web2SE 2011: 2nd International Workshop on Web 2.0 for Software Engineering

Christoph Treude
University of Victoria
Victoria, Canada
e-mail: ctreude@uvic.ca

Margaret-Anne Storey
University of Victoria
Victoria, Canada
e-mail: mstorey@uvic.ca

Arie van Deursen
Delft University of Technology
Delft, the Netherlands
e-mail: arie.vandeursen@tudelft.nl

Andrew Begel
Microsoft Research
Redmond, USA
e-mail: andrew.begel@microsoft.com

Sue Black
University College London
London, UK
e-mail: s.black@cs.ucl.ac.uk

DOI: 10.1145/2020976.2020977
http://doi.acm.org/10.1145/2020976.2020977

Abstract

Web 2.0 technologies, such as wikis, blogs, tags and feeds, have been adopted and adapted by software engineers. With the annual Web2SE workshop, we provide a venue for research on Web 2.0 for software engineering by highlighting state-of-the-art work, identifying current research areas, discussing implications of Web 2.0 on software engineering, and outlining the risks and challenges for researchers. This report highlights the paper and tool presentations, and the discussions among participants at Web2SE 2011 in Honolulu, as well as future directions of the Web2SE workshop community.

1 Introduction and Motivation

Social software is built around an "architecture of participation" where user data is aggregated as a side effect of using Web 2.0 applications [18]. Web 2.0 implies that processes and tools are socially open and that content can be used in several different contexts. Web 2.0 tools and technologies support interactive information sharing, data interoperability and user-centered design. For instance, wikis, blogs, tags and feeds help us organize, manage and categorize content in an informal and collaborative way [24]. Some of these technologies have made their way into software development processes and platforms [1]. These processes and environments are just scratching the surface of what can be done by incorporating Web 2.0 approaches and technologies into collaborative software development. Web 2.0 opens up new opportunities for software developers to form teams and collaborate, but it also comes with challenges for developers and researchers.

One goal of the annual Web2SE workshop is to investigate how Web 2.0 technologies can improve software development tools and practices. However, as Ian Davis phrased it in a blog post[1], Web 2.0 is more than just mere technologies, it is an attitude. Web2SE aims at improving our understanding of how this attitude, manifested in technologies such as mashups or dashboards, can change the culture of collaborative software development.

Web2SE 2011 at the International Conference on Software Engineering (ICSE) in Honolulu, Hawaii, USA, was attended by 23 registered participants. In four sessions, a total of six research papers were presented and discussed by all participants. For each paper, there were ten minutes of presentation by the authors, followed by discussion initiated by a number of questions prepared in advance by one or two workshop participants, as well as free-format discussion with all participants. In addition, we had an invited demonstration of a Web2SE-related tool – Pex4Fun from Microsoft Research – and a session in which the participants were encouraged to demonstrate their own tools in an informal setting. The workshop concluded with a wrap-up discussion in which participants got together in small groups and brainstormed ideas on the future of social media in software development.

In the spirit of the workshop topic, throughout the day we took notes and shared our findings by means of several Web 2.0 technologies, using a shared Google Document[2] as well as our @web2se Twitter account[3] along with the #web2se hashtag[4]. The remainder of this paper is structured as follows. After a short review on the objectives of Web2SE, we outline the papers and demonstrations that were presented at Web2SE 2011 in Sections 3 and 4, and we review the wrap-up discussion in Section 5. Section 6 concludes the report by outlining future work of the Web2SE community. A workshop report from one of the workshop participants can be found on the ICSE 2011 blog[5].

[1] http://blog.iandavis.com/2005/07/04/talis-web-2-0-and-all-that/

[2] A read-only version of the Google Document is available online at http://tinyurl.com/web2se11notes.

[3] http://twitter.com/web2se

[4] An archive of the #web2se Twitter stream is available online at http://tinyurl.com/web2se11tweets.

[5] http://icse2011w.posterous.com/web2se-2011

2 Objectives

The Web2SE workshop has several objectives:

- To collect an **overview** of the latest developments with regard to the use of Web 2.0 technologies in software development. Some Web 2.0 technologies, such as wikis [15], facebook [4], blogs [19], social bookmarking [9], micro-blogging [5] and tagging [23], have already been adopted by software developers and development environments. In addition, projects such as Cloud9 IDE[6] and websites such as Stack Overflow [26] bring Web 2.0 into software development. At Web2SE 2010, the potential move of the IDE into the browser was one of the main discussion points [27].
- To explore **new opportunities** that Web 2.0 creates in software development. Web 2.0 enables new ways for developers to form teams to create software, to bypass traditional hierarchies of power in organizations, to get ideas heard and acted upon from any organization member, to socialize and promote ideas, and to use new forms of computation, such as Mechanical Turk[7], to solve previously difficult, yet impactful problems [3].
- To investigate to which extent the "socially open" **attitude** of Web 2.0 applies to software development, and how to balance the architecture of participation and individual productivity. Web 2.0 applications are increasingly driven by data and the key advantage of Internet applications is the extent to which users add their own data [18]. A promising route uses data collection and analysis techniques from the field of *software repository mining* for recommending relevant bugs, developers, or artifacts that can be enriched with data provided by software engineers or collected from the way they interact with their development tools [6].
- To explore how Web 2.0 technologies can be incorporated into and adapted to software engineering **processes and methods**.
- To discuss potential **risks** of using Web 2.0 in software development. Protecting the privacy and reputation of individuals participating in Web 2.0 systems, and making sure that data stored in those systems has proper access restrictions is essential.
- To highlight **challenges for researchers** who are studying the use of Web 2.0 in software development. Researchers have a responsibility to ensure that tools are consistent with the values of the organizations in which they are deployed. Another challenge lies in the difficulty of creating new social networks to test new tool ideas. We discuss ways in which researchers can create and evaluate new tools despite the difficulties of achieving wide-spread adoption, something that is essential to the success of Web 2.0 tools.

[6]http://cloud9ide.com/
[7]https://www.mturk.com/

3 Accepted Papers

In this section, we briefly review the six research papers that were presented at Web2SE 2011 and we highlight important points made during the discussion of those papers.

Measuring API documentation on the web. In the first paper presented at Web2SE 2011, Parnin and Treude [20] discussed an empirical study on the extent to which the methods of one particular API – jQuery – are documented on the Web. Websites, such as blogs, forums and Q&A portals, have changed the way software is documented by allowing developers to create and communicate knowledge and experiences without having to rely on a central authority for official documentation. To understand how documentation via social media augments more traditional forms of documentation, the authors analyzed the first ten search results for each of the methods in the jQuery API. They found that 87.9% of the API methods were covered by software development blogs, and that these blog posts were written by a large group of authors. They conclude that API documentation through social media can provide high levels of coverage and that it gives readers a chance to engage with authors.

The discussion for the paper focused on the nature of blog posts (advanced vs. introductory) and on the motivations of bloggers to contribute content. One of the findings – the importance of blogging for evangelists in a particular community – was confirmed by a recent paper by Pagano and Maalej [19]. Another interesting topic that came up during discussion is the nature of API methods that are documented using social media vs. the nature of those that are not. Are the API methods that have not been blogged about simply the ones that developers do not have problems with? Or are those parts of the API orphaned?

Towards understanding Twitter use in software engineering: preliminary findings, ongoing challenges and future questions. The second presentation at the workshop explored the use of the micro-blogging service Twitter by software engineers and its effect on communication within software engineering projects. Bougie, Starke, Storey and German [5] used archival analysis to quantify some basic parameters of Twitter use by software engineers and triangulated their findings with qualitative data obtained through the manual coding of 600 tweets. The authors found that the software engineering community leverages Twitter's capabilities for conversation (through @-replies) and information sharing (through hyperlinks) more than the general Twitter audience, as found by Java *et al.* [11]. Topics discussed by software engineers on Twitter include: current projects, seeking or providing technical help, gadgets and technologies, current events, and daily chatter. Twitter brings the "water cooler atmosphere" to distributed developers.

A challenge that was discussed at the workshop is how to identify software engineers on Twitter. LDA topic analysis, social network analysis, follower analysis, and trying

to mine source code repositories and compare author names were among the suggested techniques. In addition, different projects attract different communities on Twitter. For example, most Linux-related Twitter accounts are from Linux *users*, whereas most Eclipse-related Twitter accounts belong to Eclipse *developers*. The role of Twitter compared to other communication and coordination mechanisms for software developers was also discussed. Does Twitter replace instant messaging? Is it possible to move a conversation from one medium to another?

Leveraging social media to gather user feedback for software development. In their paper on the role of social media to gather user feedback, Bajic and Lyons [2] used data from the collaborative feedback tool UserVoice[8] and interview data to analyze how software companies are finding ways to use social media techniques to gather feedback from users. Their work is motivated by the fact that social media service offerings, such as Facebook and LinkedIn, have emerged over the past few years, and that companies, such as Dell, are finding creative ways of using social media[9]. The authors analyze four factors and the way they influence the use of social media: company size, transparency, software deployment, and number of social media tools in use. Several challenges come with managing user feedback: identifying the best ideas, sustaining a community, and avoiding disclosure of ideas to competitors.

The discussion for this paper commenced by pondering whether companies actually feel like they are missing out on feedback from customers or whether they shun collaborative feedback because of trust and noise issues. We also discussed whether users on websites such as UserVoice are reflective of the general user population of a product. If companies have users vote on feature requests, are they more likely to share hard-to-implement features vs. features that are easy to realize? What is the tipping point for the number of people asking for a feature before the company implements it?

Automatic status updates in distributed software development. King and Lyons [12] presented a paper on automatic status updates in distributed software development. They developed an Eclipse plugin that automatically determines a user's activity in their Eclipse IDE and publishes that activity information as the status in an instant messenger client. The status is updated as soon as the user changes their activities in the IDE. The tool was evaluated by demonstrating it to 81 academics and industry workers, and by interviewing them about the perceived benefits and usefulness of the tool. The authors found various factors that impact the need for such automatic status updates. These factors include the amount of work experience of a given user, their experience with distributed software development, and their seniority level on the team. Automatic status updates might also play a role in interruption management.

The potential use for interruption management sparked discussion at the workshop. As Christoph Walesch writes in his blog post on Web2SE 2011[10], *"Interruptions through IM messages, Skype calls, email popups, Twitter, etc. have an impact on focus and productivity. Each interruption moves our focus to an incoming email for example, even if we decide to ignore it. It takes some time to re-focus on the original task. And even worse, you have no chance to get into deep focus, a focus you need to carefully analyze difficult problems. So I think we should extend the scope of such a tool to all interrupters and then conduct research about the productivity and quality impact of an interruption-aware working style."* Other discussion items included whether team size determines the usefulness of such a tool, and how researchers could measure "prevented interruptions" in a meaningful way.

Supporting the cooperation of end-user programmers through social development environments. Singer and Schneider [22] presented a paper on how insights from Communities of Practice research (e.g., Wenger *et al.* [28]) can be implemented using social media in software development. Their work is motivated by the fact that many programs are being created by end users without formal training in software development, and that these end users are often unaware of software engineering principles. The authors connect insights from Communities of Practice research to social software in order to form communities of end-user programmers. Some of the lessons from the Communities of Practice research include: building a community by enabling public and private exchanges, encouraging contributions by embracing and engaging lurkers as well as supporting all levels of involvement, and stabilizing the community by making value visible and by supporting reputation building.

As the most important lesson from this work, the workshop participants discussed the theme "Know your community". Knowing one's community means learning what they do, what they want, what their goals are, and to support their interests rather than subjecting them to your own interests. Several pieces of related work came up during the discussion: the book "Designing for the Social Web" by Porter [21], a recent paper by Monroy-Hernández *et al.* on attribution [17], and work by Luther *et al.* on success factors in online creative collaboration [16].

Wikigramming: a wiki-based training environment for programming. In the last paper presentation at Web2SE 2011, Hattori [10] introduced Wikigramming[11], a web-based collaborative programming environment for Scheme that aims at being as simple to use as a wiki. Each page of a Wikigramming project contains source code for one Scheme function, and users can edit any function at any time and see the results of their edits instantly. The work is motivated by the fact that, particularly in open source, de-

[8] http://uservoice.com/
[9] http://en.community.dell.com/
[10] http://icse2011w.posterous.com/web2se-2011
[11] http://wikigramming.com/

velopers are often unable to see their changes right away due to complex patch and verification processes. To avoid spam and mischief, modified versions of a working program have to pass unit tests that were contributed by other users. Wikigramming is particularly aimed at training new developers and prototyping for end users.

Apart from its inherent simplicity, one of the advantages of Wikigramming discussed at the workshop is the lack of version control along with its obstacles and coordination problems for small projects. Since Wikigramming had not been evaluated, a classroom experiment was suggested to verify the ideas and concepts. Related tools that were mentioned during the discussion at the workshop include Pastebin[12], codepad[13], and the recently published Collabode [7].

4 Demonstrations

At Web2SE 2011, we had one invited demonstration – Pex4Fun from Microsoft Research – and a one-hour session where we encouraged workshop participants to demonstrate their own Web2SE-related tools in an informal setting. All demonstrations are described in the following paragraphs.

Programming on the Phone and in the Cloud with Pex4Fun. Pex4Fun[14] from Microsoft Research is a web-based gaming environment for teaching programming. Users edit C#, Visual Basic or F# code in a browser, and Pex4Fun executes and analyzes it in the cloud. In particular, Pex4Fun determines interesting input values that help users understand what their code is actually doing. Pex4Fun also features coding duels where users write code trying to match the output of a secret implementation. Pex4Fun finds the discrepancies between the users' code and the specification. Pex4Fun adds functionality to design courses that teach programming at various levels.

Social features in Pex4Fun include the display of usage statistics, a live feed of users' interactions with certain puzzles, and an application for the Windows Phone 7[15]. The phone applications allows users to write scripts for the phone on the phone. It uses the touch screen as its programming paradigm, and utilizes the various input and output options of a mobile phone.

Pex4Fun was demonstrated by Nikolai Tillmann from Microsoft Research [25].

Pollicino. Pollicino[16] is an Eclipse plugin developed by Guzzi *et al.* [9] that introduces collective code bookmarks into the IDE by enabling users to create links to locations in files. The concept of text bookmarks is preserved while functionality for creating, deleting and managing bookmarks are added. Pollicino aims at supporting software comprehension activities. Arie van Deursen demonstrated Pollicino at Web2SE 2011.

Work Item Explorer. Work Item Explorer is a research prototype that facilitates the flexible, iterative exploration of IBM Jazz[17] data, focusing primarily on work items. It provides several visualizations, such as bar charts, tag clouds and graphs, and automatically coordinates them. Work Item Explorer is built on the Choosel framework[18] for web-based visual data exploration [8]. Christoph Treude demonstrated the tool at the workshop.

StakeSource 2.0. StakeSource 2.0[19] is a web-based tool that uses social networks and collaborative filtering to identify and prioritize stakeholders and their requirements [14]. The approach utilizes crowdsourcing and stands in contrast to traditional approaches to requirement elicitation that employ system analysts. StakeSource was demonstrated by Soo Ling Lim.

Automatic Status Update. The Automatic Status Update tool that was presented by Abayomi King [12] earlier in the workshop was also part of the informal demonstration session (see the previous section for a description).

5 The Future of Social Media in Software Development

We concluded Web2SE 2011 with a discussion session in which the workshop participants got together in four small groups to brainstorm ideas on the future of social media in software development. In particular, participants were asked to envision what the future of social media in software development would look like. The main ideas from each of the groups are outlined below.

Group 1. The first group envisioned hobbyists that are able to develop software that suits their needs by being connected to experts. Large software companies will no longer exist and marketing will be done using social and virtual channels. There was discussion on whether social media will lead to fewer or more developers. In any case, the influence of social media in software development will lead to democratization. On a more technical level, the first group envisioned improved discovery of libraries and reuse of components at a higher level than today.

Group 2. Group 2 envisioned a state where everybody is a developer, programming will be closer to literate programming [13] and not as "nerdy" anymore, and the distinction

[12] http://pastebin.com/
[13] http://codepad.org/
[14] http://pex4fun.com/
[15] http://research.microsoft.com/en-us/projects/touchstudio/
[16] http://www.st.ewi.tudelft.nl/~guzzi/pollicino/
[17] https://jazz.net/
[18] http://thechiselgroup.org/choosel
[19] http://www.stakesource.co.uk/

between users and developers will no longer be clear. Development will be collaborative, making use of content created by other people, and focusing on mashup products. Coupling between components will be done in visual editors, with a secondary textual view. IDEs will be better at understanding natural language to capture context, and developers will use a Twitter-like language to define requirements. These requirements will be used to perform a social search that results in a web of developers that have previously dealt with similar requirements. IDEs will also be better at data mining and adjust based on the abilities and mindset of the developer. Helper tools, such as Pex[20], will be built into IDEs.

Group 3. The third group also envisioned a future in which everybody is a developer. Software development will be moving to the masses. In the spirit of Abraham Lincoln's Gettysburg Address, the group proclaimed, *"Software of the people, by the people, for the people, shall not perish from the earth."* Programming will be done using natural language or by acting programs out. In fact, programming will be effortless and the only remaining problem will be defining the requirements. Requirements will be determined by capturing and observing everything we are doing in order to find out what we need. Based on that, many alternatives will be generated that will be presented to the users to choose from. Requirements will be communicated by showing examples or similar products. Software will also be able to adjust to the moods of the developers.

Group 4. The last group wondered whether the distinction between professional and end-user developers will start to blur. The constraints around contributing to software may start to disappear and the programmer base will change. Social media will enable more people to participate, however, the number of professional developers will stay roughly the same. Domain-specific languages and programming using gestures and natural language will be more prevalent. Devices will get smaller, and I/O will become ubiquitous. Peripheral devices, such as mobile phones, will be added into what will be a ubiquitous programming environment.

6 Conclusions and Future

The Web2SE community has moved forward since the first workshop held in 2010. At Web2SE 2010, defining what we meant by Web 2.0 and social media, as well as explaining particular mechanisms and tools such as Twitter, took up a considerable amount of time. At Web2SE 2011, all participants had a clearer understanding of what tools and mechanisms social media has brought to software development, and the workshop was an exciting opportunity to explore further applications and implications of social media in software engineering. Social media has also become more of a mainstream topic in the software engineering research community and Web2SE-related papers are now part of many major conferences and journals in our area.

It was interesting to see common themes emerge from the session in which the participants of Web2SE 2011 envisioned the future of social media in software development. All groups thought that the line between developers and users of software will start to blur, and that collaboration between developers and users will be enabled through the use of social media in one way or another. The groups also envisioned that communication with development machines will become easier. Computers might be able to determine requirements by themselves or at least with less user involvement, and programming languages might become more similar to natural language.

There are still research areas and challenges to be addressed by the Web2SE community. The use of social media by software developers is a moving target, and new mechanisms and tools are adopted and adapted faster than we can study them. To understand and influence these fast moving socio-technical environments, the Web2SE community will continue to work toward its objectives: study the current use of Web 2.0 tools, explore further potential uses, investigate the implications of socially-open software development, explore implications for processes and methods, discuss the risks of social media, and highlight challenges for researchers in Web2SE.

Acknowledgments

We thank all workshop participants and authors for their valuable contributions and presentations at the workshop. We also appreciate the organizational support from the ICSE 2011 workshop co-chairs and the logistical help from Cassandra Petrachenko.

We also thank the members of our program committee for providing timely and relevant reviews: Jorge Aranda (University of Victoria, Canada), Marcelo Cataldo (Carnegie Mellon University, USA), Li-Te Cheng (IBM Watson Research Center, USA), Kate Ehrlich (IBM Watson Research Center, USA), Harald Gall (University of Zurich, Switzerland), Adrian Kuhn (University of Bern, Switzerland), Michele Lanza (University of Lugano, Switzerland), Kelly Lyons (University of Toronto, Canada), Mira Mezini (TU Darmstadt, Germany), Peri Tarr (IBM Watson Research Center, USA), Gina Venolia (Microsoft Research, USA), Thomas Zimmermann (Microsoft Research, USA).

References

[1] N. Ahmadi, M. Jazayeri, F. Lelli, and S. Nesic. A survey of social software engineering. In *ASE Workshops '08: 23rd international conference on Automated Software Engineering*, pages 1–12, Washington, DC, USA, 2008. IEEE.

[2] D. Bajic and K. Lyons. Leveraging social media to gather user feedback for software development. In *Proceedings of the*

[20] http://research.microsoft.com/en-us/projects/pex/

2nd international workshop on Web 2.0 for software engineering, Web2SE '11, pages 1–6, New York, NY, USA, 2011. ACM.

[3] A. Begel, R. DeLine, and T. Zimmermann. Social media for software engineering. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, FoSER '10, pages 33–38, New York, NY, USA, 2010. ACM.

[4] A. Begel, Y. P. Khoo, and T. Zimmermann. Codebook: Discovering and exploiting relationships in software repositories. In *Proceedings of the 32nd international conference on Software Engineering - Volume 1*, ICSE '10, pages 125–134, New York, NY, USA, 2010. ACM.

[5] G. Bougie, J. Starke, M.-A. Storey, and D. M. German. Towards understanding twitter use in software engineering: preliminary findings, ongoing challenges and future questions. In *Proceedings of the 2nd international workshop on Web 2.0 for software engineering*, Web2SE '11, pages 31–36, New York, NY, USA, 2011. ACM.

[6] M. Bruch, E. Bodden, M. Monperrus, and M. Mezini. Ide 2.0: collective intelligence in software development. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, FoSER '10, pages 53–58, New York, NY, USA, 2010. ACM.

[7] M. Goldman, G. Little, and R. C. Miller. Collabode: collaborative coding in the browser. In *Proceedings of the 4th international workshop on Cooperative and human aspects of software engineering*, CHASE '11, pages 65–68, New York, NY, USA, 2011. ACM.

[8] L. Grammel and M.-A. Storey. Poster: Choosel - web-based visualization construction and coordination for information visualization novices. IEEE Information Visualization Conference, InfoVis '10, 2010.

[9] A. Guzzi, L. Hattori, M. Lanza, M. Pinzger, and A. van Deursen. Collective code bookmarks for program comprehension. In *Proceedings of the 19th international conference on Program Comprehension*, ICPC '11, Washington, DC, USA, 2011. IEEE.

[10] T. Hattori. Wikigramming: a wiki-based training environment for programming. In *Proceedings of the 2nd international workshop on Web 2.0 for software engineering*, Web2SE '11, pages 7–12, New York, NY, USA, 2011. ACM.

[11] A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, WebKDD/SNA-KDD '07, pages 56–65, New York, NY, USA, 2007. ACM.

[12] A. King and K. Lyons. Automatic status updates in distributed software development. In *Proceedings of the 2nd international workshop on Web 2.0 for software engineering*, Web2SE '11, pages 19–24, New York, NY, USA, 2011. ACM.

[13] D. E. Knuth. Literate programming. *The Computer Journal*, 27:97–111, 1984.

[14] S. L. Lim, D. Damian, and A. Finkelstein. Stakesource2.0: using social networks of stakeholders to identify and prioritise requirements. In *Proceedings of the 33rd international conference on Software engineering*, ICSE '11, pages 1022–1024, New York, NY, USA, 2011. ACM.

[15] P. Louridas. Using wikis in software development. *IEEE Software*, 23(2):88–91, 2006.

[16] K. Luther, K. Caine, K. Ziegler, and A. Bruckman. Why it works (when it works): success factors in online creative collaboration. In *Proceedings of the 16th international conference on Supporting group work*, GROUP '10, pages 1–10, New York, NY, USA, 2010. ACM.

[17] A. Monroy-Hernández, B. M. Hill, J. Gonzalez-Rivero, and d. boyd. Computers can't give credit: how automatic attribution falls short in an online remixing community. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 3421–3430, New York, NY, USA, 2011. ACM.

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

[19] D. Pagano and W. Maalej. How do developers blog?: an exploratory study. In *Proceedings of the 8th working conference on Mining software repositories*, MSR '11, pages 123–132, New York, NY, USA, 2011. ACM.

[20] C. Parnin and C. Treude. Measuring api documentation on the web. In *Proceedings of the 2nd international workshop on Web 2.0 for software engineering*, Web2SE '11, pages 25–30, New York, NY, USA, 2011. ACM.

[21] J. Porter. *Designing for the social web*. New Riders Press, 2008.

[22] L. Singer and K. Schneider. Supporting the cooperation of end-user programmers through social development environments. In *Proceedings of the 2nd international workshop on Web 2.0 for software engineering*, Web2SE '11, pages 13–18, New York, NY, USA, 2011. ACM.

[23] M.-A. Storey, J. Ryall, J. Singer, D. Myers, L.-T. Cheng, and M. Muller. How software developers use tagging to support reminding and refinding. *IEEE Transactions on Software Engineering*, 35(4):470–483, 2009.

[24] M.-A. Storey, C. Treude, A. van Deursen, and L.-T. Cheng. The impact of social media on software engineering practices and tools. In *Proceedings of the FSE/SDP workshop on Future of software engineering research*, FoSER '10, pages 359–364, New York, NY, USA, 2010. ACM.

[25] N. Tillmann, J. de Halleux, and T. Xie. Pex4Fun: Teaching and learning computer science via social gaming. In *Proceedings of the 24th conference on Software Engineering Education and Training, Practice and Methods Presentations & Tutorials (PMP&T)*, CSEET '11, 2011.

[26] C. Treude, O. Barzilay, and M.-A. Storey. How do programmers ask and answer questions on the web? nier track. In *Proceedings of the 33rd international conference on Software engineering*, ICSE '11, pages 804–807, New York, NY, USA, 2011. ACM.

[27] C. Treude, M.-A. Storey, K. Ehrlich, and A. van Deursen. Workshop report from Web2SE: First Workshop on Web 2.0 for Software Engineering. *SIGSOFT Software Engineering Notes*, 35:45–50, 2010.

[28] E. Wenger, R. McDermott, and W. Snyder. *Cultivating Communities of Practice: A Guide to Managing Knowledge*. Harvard Business School Press, Boston, MA, USA, 2002.