Programming in a Socially Networked World: the Evolution of the Social Programmer

Christoph Treude, Fernando Figueira Filho, Brendan Cleary, Margaret-Anne Storey
Dept. of Computer Science
University of Victoria
ctreude@uvic.ca, ffilho@uvic.ca, bcleary@uvic.ca, mstorey@uvic.ca

ABSTRACT
Social media has changed how software developers collaborate, how they coordinate their work, and where they find information. Social media sites, such as the Question and Answer (Q&A) portal Stack Overflow, fill archives with millions of entries that contribute to what we know about software development, covering a wide range of topics. For today's software developers, reusable code snippets, introductory usage examples, and pertinent libraries are often just a web search away. In this position paper, we discuss the opportunities and challenges for software developers that rely on web content curated by the crowd, and we envision the future of an industry where individual developers benefit from and contribute to a body of knowledge maintained by the crowd using social media.

Author Keywords
Social Media; Software Development; Social Coding

ACM Classification Keywords
D.2 Software Engineering

General Terms
Human Factors.

INTRODUCTION

While the Internet has made a vast body of knowledge accessible to software developers, it is the advance of social media that has introduced effective mechanisms for a large crowd of developers to curate content on the web. Users can endorse articles through mechanisms such as Facebook's *"Like"*, they can give positive or negative ratings to questions and answers on Q&A websites, and they can annotate and comment on a wide spectrum of blog posts. Content on social media sites ranges from tutorials and experience reports to code snippets and samples.

One of the most successful social media resources for software developers is Stack Overflow[1], a site that facilitates the exchange of knowledge between software developers through a Q&A portal. Since its foundation in 2008, more than 2.3 million questions have been asked on Stack Overflow, and nearly 5 million answers have been provided, contributing to a large repository of software development knowledge. Many of the questions and answers contain code snippets that can easily be copied and pasted into one's source code.

A question asked on Stack Overflow has a median answer time of 11 minutes [4]. However, users can sometimes start receiving answers only minutes or even seconds after their initial post. This virtually real-time access to a community of millions of other programmers willing and eager to help is an almost irresistible resource, evidenced by the more than 12 million visitors and 135 million page views which Stack Overflow receives each month[2]. Add to the live community answering questions on demand, an archive of over 7 million posts representing the rated and classified knowledge of millions of programmers, readily accessible through search engines, and Stack Overflow has the potential to have a significant impact on the practices of millions of programmers worldwide.

The easy access to such a vast repository of knowledge raises several research questions: Will developers who focus on reusing content from the web have sufficient understanding of the inner workings of the software they produce? Are web resources going to cover all important aspects of a topic? What meta-data is needed to facilitate technical information-seeking? How can we address security and copyright concerns that come with using other developers' code?

In this position paper, we discuss the past, present, and future of software developers that have access to an unprecedented amount and diversity of resources on the web. We focus our discussion on the Q&A portal Stack Overflow due to space constraints, but we believe that our observations are applicable to other social software and social computing environments as well.

BACKGROUND AND RELATED WORK

Social media is changing how software developers communicate and coordinate, and how they produce and consume documentation. The current adoption of social media in processes and integrated development environments is just scratching the surface of what can be done by incorporating social media approaches and technologies into software development.

Storey et al. [7] discuss the impact of social media on software engineering practices and tools. Historically, **wikis** and

[1] http://stackoverflow.com/

[2] http://www.quantcast.com/stackoverflow.com

blogs were the first social media mechanisms used by software developers, utilized mostly in the areas of requirements engineering and documentation, and to communicate high-level concepts. **Microblogs**, such as Twitter, play a role in conversation and information sharing between software developers [1], whereas **tags** can help software developers communicate their concerns in task management [9] and add semantic information to source code [6].

Among those technologies, the Stack Overflow Q&A portal not only provides a unique medium for the interaction between several communities of practice of developers [2], but also stands out due to the daily involvement of its design team within those communities [4]. In a preliminarily categorization of the questions found on Stack Overflow, we found that the website is particularly effective at providing code reviews and at answering conceptual questions, and that roughly 85% of the questions on Stack Overflow are answered [8]. Stack Overflow also attracts a lot of web traffic and can reach a high level of coverage for a given topic. In a recent study, we analyzed the Google search results for one particular API – jQuery – and found at least one Stack Overflow question on the first page of the search results for 84% of the API's methods [5].

BEFORE STACK OVERFLOW

Before the advent of Stack Overflow, the main mechanisms for Q&A consisted of technical forums, where content is made available in the form of threaded discussions. The main problem with this approach is that useful information is typically mixed with irrelevant context. Stack Overflow, on the other hand, made popular the concept of collaborative filtering to rank best answers, which are shown up front thus exempting the user from analyzing several pages of content.

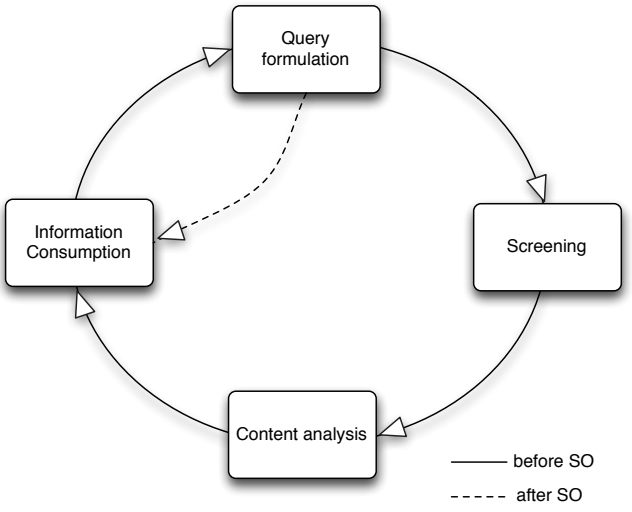

Figure 1. Search process before and after Stack Overflow.

Figure 1 illustrates the search process before and after Stack Overflow. A typical search scenario requires effort from the user to select the most promising results (i.e., screening) and to analyse each result before eventually consuming information. The collaborative filtering approach enables a shorter search process, since relevant information is pre-ranked based on community expertise [3].

Another advantage of Stack Overflow over previous knowledge exchange portals is that knowledge reuse is encouraged within the community and supported by design. Before posting a new question, the system requires the user to check if the question has already been answered, thus avoiding the same question being asked repeatedly. This approach facilitates technical information scrutiny, as it produces fewer candidates to be analyzed after a search query.

AFTER STACK OVERFLOW

The increasing socialization of software development has only begun to be studied and leads us to consider the questions: Do we need to rethink our concept of what a programmer is and what they do? Do we need to refactor the programmer?

The Evolution of the Social Programmer

Contrary to popular stereotypes, programmers have always been social creatures (at least within their own communities). Indeed, the new electronic communication technologies of the last several decades (email, bulletin boards, Usenet, IRC, the web) were most often first colonized by programmers. The emergence of Stack Overflow is the latest evolution of this historical trend with programmers inventing or adopting a technology to meet their need to discuss what it is they do with other programmers. However, with sites such as Stack Overflow indexed and made available through search engines, are we approaching the point where the archive of stored programming knowledge reaches a critical mass and where new programming practices and behaviors will emerge? Have we already reached that point and what kinds of impact might we see on programmer practices and the software development industry as a result?

What Makes a Good Programmer

In a world where a large percentage of programming knowledge is archived and curated by millions of "experts", do we have to redefine the attributes of a good programmer? Is the metric of a good programmer someone with a deep understanding of programming and software engineering principles, or someone who can leverage and synthesize the programming community to achieve the same results? When you are very unlikely to be the first person in the world to encounter a particular problem, does a smart programmer attempt to diagnose the problem independently or just ask the community? Has Google become the world's fastest debugger? Will an entirely new category of programmers emerge, a *Just in Time Programmer*, without formal training but with the ability to combine snippets to craft solutions that will just meet their needs? What tools will these new programmers require?

Software Development as a Massively Distributed Activity

If future programmers will require the ability to synthesize a single solution from the contributions of the many, it raises interesting questions (both positive and negative) about the nature of the software that those programmers produce. Given

an environment where practically all software is developed with reliance on a shared knowledge base and community, who actually owns the intellectual property? Is there a risk that programmers do not really understand how their software works? Or will it in fact lead to better efficiency by reducing time spent fixing bugs or re-inventing the wheel? Could the distributed development approach to programming actually increase quality by promoting best practice solutions? Is this a realization of software componentization, different from the vision of component based development perhaps but effectively a similar result? What organizational changes will this shift entail in terms of social offshoring (e.g., TopCoder[3]) and the ad hoc creation of teams?

Impact on Education and Career Planning

Social media has rapidly changed how programmers advertise their skills and how they manage their time coding, learning, and strategically planning their careers. For example, one can observe a growing interest from developers in including statistics from their Stack Overflow profiles in job applications. Providing good answers to questions on Stack Overflow can be a valuable resource when competing for a position[4]. As a result, Stack Overflow not only provides means for exchanging information, but also enables a level of transparency never seen before in community portals. For users that answer questions, Stack Overflow is an opportunity to help people, to build credibility, and to learn about the problems encountered by their peers[5].

Although social media has gained momentum in the software industry, we are only beginning to explore what can be realized with those technologies in education. We argue for a greater use of social media for teaching as a form of preparing the student to become a socially networked programmer. The use of collaborative technologies is a fundamental part in this process and is giving rise to various forms of virtual learning environments (e.g., Code School[6]). This emergent scenario will demand a redefinition of what we understand as good education for a software developer. How do we reformulate the curricula of academic courses in a world where programming skills have become less individualized and more socialized?

CONCLUSION

The stereotype of a typical computer programmer has evolved from an isolated individual who learns by reading computer science books, to a person who manages his persona across many social coding sites on the web and collaborates with other software developers all over the world. This persona will redefine the social attributes of a good programmer, and also what we think is essential for planning the curricula of future academic courses. While we have focused on the Q&A portal Stack Overflow in this position paper, future research opportunities lie in the investigation of other social media and social computing environments and their implications on the future of collaborative software development.

[3]http://www.topcoder.com/

[4]http://blog.stackoverflow.com/2011/08/reputation-not-rep/

[5]http://meta.stackoverflow.com/questions/22400

[6]http://www.codeschool.com

REFERENCES

1. Bougie, G., Starke, J., Storey, M.-A., and German, D. M. Towards understanding twitter use in software engineering: preliminary findings, ongoing challenges and future questions. In *Proc. of the 2nd Intl. workshop on Web 2.0 for software engineering*, Web2SE '11, ACM (New York, NY, USA, 2011), 31–36.
2. Figueira Filho, F. M., de Albuquerque, J. P., and de Geus, P. L. Broadening the perspective on classification systems in the web: Analysing web classification as a situated activity whithin communities of practice. In *Proc. of the Intl. Conf. on Collaborative Technologies*, CT '10, IADIS Press (July 2010), 117–124.
3. Figueira Filho, F. M., Olson, G. M., and de Geus, P. L. Kolline: a task-oriented system for collaborative information seeking. In *Proc. of the 28th Intl. Conf. on Design of Communication*, SIGDOC '10, ACM (New York, NY, USA, September 2010), 89–94.
4. Mamykina, L., Manoim, B., Mittal, M., Hripcsak, G., and Hartmann, B. Design lessons from the fastest q&a site in the west. In *Proc. of the Conf. on human factors in computing systems*, CHI '11, ACM (New York, NY, USA, 2011), 2857–2866.
5. Parnin, C., and Treude, C. Measuring api documentation on the web. In *Proc. of the 2nd Intl. workshop on Web 2.0 for software engineering*, Web2SE '11, ACM (New York, NY, USA, 2011), 25–30.
6. Storey, M.-A., Ryall, J., Singer, J., Myers, D., Cheng, L.-T., and Muller, M. How software developers use tagging to support reminding and refinding. *IEEE Trans. on Software Engineering 35* (July 2009), 470–483.
7. Storey, M.-A., Treude, C., van Deursen, A., and Cheng, L.-T. The impact of social media on software engineering practices and tools. In *Proc. of the FSE/SDP workshop on Future of software engineering research*, FoSER '10, ACM (New York, NY, USA, 2010), 359–364.
8. Treude, C., Barzilay, O., and Storey, M.-A. How do programmers ask and answer questions on the web? (NIER track). In *Proc. of the 33rd Intl. Conf. on Software Engineering*, ICSE '11, ACM (New York, NY, USA, 2011), 804–807.
9. Treude, C., and Storey, M.-A. Work item tagging: Communicating concerns in collaborative software development. *IEEE Trans. on Software Engineering 99*, PrePrints (2010).

BIOGRAPHIES OF THE AUTHORS

Christoph Treude is a PhD candidate in computer science at the University of Victoria and an organizer of the workshop on Web 2.0 for Software Engineering (Web2SE). In his PhD research, he is exploring the role of social media artifacts in collaborative software development. He has already studied the use of tags, dashboards, feeds and a community portal by professional software developers using IBM's Jazz.

Fernando Figueira Filho received his PhD from the University of Campinas, Brazil, in 2011 and is currently a post-doctoral fellow at the University of Victoria. His thesis focused on social search and how we can evolve from traditional information retrieval approaches in order to incorporate social media aspects into search. His main research interests are in the areas of software engineering, human-computer interaction and computer supported cooperative work.

Brendan Cleary received his PhD from the University of Limerick in 2007 and is currently a post-doctoral fellow at the University of Victoria. Brendan has managed research projects totalling over 2 million euro and is co-founder of a university start-up in the social learning space. His main research interests are in the areas of software engineering, recommendation systems and software comprehension.

Margaret-Anne Storey is a professor of computer science at the University of Victoria, a Canada Research Chair in Human Computer Interaction for Software Engineering and a principal investigator for the National Center for Biomedical Ontology, US. Her research goal is to understand how technology can help people explore, understand and share complex information and knowledge. She applies and evaluates techniques from knowledge engineering, social software and visual interface design to applications such as collaborative software development, program comprehension, biomedical ontology development, and learning in web-based environments.