Keynote:
Selecting Research Methods for Studying a Participatory Culture in Software Development

Margaret-Anne Storey
Dept. of Computer Science
University of Victoria
Victoria, BC, Canada
mstorey@uvic.ca

ABSTRACT
Recent innovations in social media have led to a paradigm shift in software development, with highly tuned participatory development cultures contributing to crowdsourced content and being supported by media that have become increasingly more social and transparent. Never before in the history of software development have we seen such rapid adoption of new tools by software developers. But there are many unanswered questions about the impact this tool adoption has on the quality of the software, the productivity and skills of the developers, the growth of projects and technologies developers contribute to, and how users can give feedback on and guide the software they use. Answering these questions is not trivial as this participatory development culture has become a virtual network of tightly coupled ecosystems consisting of developers, shared content and media channels.

In our studies, we have found that combining research methods from the social sciences with data mining and software analytics to be the most promising in terms of revealing benefits and challenges from this adoption of social tools. In this talk, I share some of the findings from our studies, discuss the particular research methods we have used, and share our experiences from using those research methods. I also discuss how we as researchers leverage social tools and interact with the participatory development culture to assist with and help us gain feedback on our research. I close with a discussion about how other software engineering researchers could benefit from using social tools and the challenges they may face while doing so.

Categories and Subject Descriptors
H.5.3 [Group and Organization Interfaces]: Computer-supported collaborative work

General Terms
Human Factors

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

EASE ’15 April 27 - 29, 2015, Nanjing, China
Copyright 2015 ACM 978-1-4503-3550-4/15/04 ...$15.00.
http://dx.doi.org/10.1145/2745802.2747957

Keywords
Research Methods, Social Media, Software Engineering, Collaboration, Empirical Software Engineering

1. INTRODUCTION
Selecting suitable research methods for empirical software engineering research is ambitious due to the various benefits and challenges each method entails. One must regard the different theoretical stances relating to the nature of the research questions being asked, as well as practical considerations in the application of methods and data collection [6]. Furthermore, studying the human and social factors that occur across distributed virtual or online communities is particularly difficult.

Our research has investigated how development tools and communication media that are infused with social features impact software development communities of practice. Our findings to date indicate that the adoption of social media by these communities of developers leads to an emerging participatory development culture [21].

In this extended abstract, we describe some of the characteristics of this emergent development culture and the tools that are used to support it. We then discuss the different tradeoffs and challenges faced in selecting and applying research methods for studying development in online communities of practice. Finally, we briefly explore how social tools could foster a more participatory culture in software engineering research and how that may help to accelerate our impact on software engineering practice.

2. EMERGENCE OF A PARTICIPATORY DEVELOPMENT CULTURE
Over the past few decades, software development has transitioned from a predominantly solo activity where developers create small standalone programs, to a widespread distributed exercise where hundreds and even thousands of developers operate within an online community of practice [25] and create complex software ecosystems (e.g., the Ruby on Rails project has more than 2500 contributors[5]). This large-scale distributed development effort has been made possible by innovations that include the Internet, the World Wide Web, sophisticated development tools (e.g., integrated development environments, bug trackers, source code repositories), and rich communication channels (e.g., email and

https://github.com/rails/rails
online chat tools).

Development and communication tools play a critical role in hosting software, as well as in capturing the history of how the code was developed and documenting how the code can be used or potentially changed. The more recent adoption of social tools, such as social networking and blogging tools, has led to the formation of a participatory development culture [21] with lower barriers to entry, strong support for co-creation of artifacts, mentorship opportunities, and appreciation of social relationships [8]. This participatory culture has lead to the emergence of the social programmer [22], introduced new means for developers to learn and stay up to date [15], and changed how developers assess themselves and others [17, 11].

We now see an increase in both the number of tools developers use and in the social features these tools provide [24]. These include socially-enabled code hosting tools (e.g., GitHub, BitBucket), management and collaboration task boards (e.g., Trello, ZenHub), focused software development communication tools (e.g., Slack, HipChat), collaborative screen-sharing tools (e.g., Screenhero and Nitrous.io), development activity automation tools (e.g., Hubot), personal and team metric tools (e.g., WakaTime, iDoneThis), and tools to keep developers up to date and help them discover important new technologies (e.g., Hacker News and Twitter).

In Figure 1, we loosely categorize a selection of developer and communication tools according to four kinds of knowledge they help capture or communicate [23, 24]: knowledge embedded in people’s heads that may be tacit and is best exchanged one-on-one or in small group interactions; knowledge embedded in development artifacts that can be accessed directly through a tool; knowledge stored in a community resource that is socially generated, maintained and exchanged within emergent communities of practice; and knowledge about people and social networks. We see that some communication channels overlap multiple types of knowledge as developers may use many of these tools in flexible ways.

In our recent research, we have been striving to understand and keep up to date with how developer, communication, and social tools support or potentially hinder development within this participatory culture. We studied how developers use, benefit from, and are challenged by tools that include email [15], social networking tools [1], Stack Overflow [2, 14], software tagging features [20, 23], micro-blogging tools [18, 5], social coding environments [4, 26], and screen-casting channels [10]. We also conducted a large-scale survey with over 2000 developers to investigate which communication channels developers use for certain activities and the challenges they may face from using a large number of communication channels [21]. Though this a related research, we have observed that tools shape and are shaped by the developers who design and use them.

Never before in the history of software development have we seen such rapid adoption of new tools by software developers. There are many unanswered questions about the impact this tool adoption has on the quality of the soft-
ware, the productivity of the developers and the growth of projects they contribute to, on the users and how they can give feedback on and guide the software they use, or on the skill or technological development of the community as a whole. Answering these questions is not trivial as it’s influenced by intertwined social, human and technological issues. In the next section, we discuss the main research methods we have used in our research, and describe the challenges we are facing (and expect to continue facing) in future work.

3. RESEARCH METHODS FOR STUDYING A PARTICIPATORY CULTURE

In our studies of this participatory development culture, our goal has not been to investigate or experiment with new tools or ideas (which would point in the direction of experimental studies), but rather to understand the impact of the social tools developers already adopt on their development practices and on the community. To study existing practices, there are two main approaches to choose from. The first relies on mining and analyzing the trace data that developers leave behind when they use development and communication tools. The second approach involves the collection and analysis of participant data from observing, surveying and interviewing community members. We discuss these two approaches below and then describe how and why we blend both in our research.

3.1 Mining software repositories

The prevalence of development, communication and social tools has led to an increase in the availability of operational and trace data from programs and developer activities. From program data, we have runtime traces, program logs, system events, failure logs, and performance metrics. From users, we have access to usage logs and user reports, and from development tools, we have different versions, bug data, commit data, and testing results.

The research areas of Mining Software Repositories and Software Analytics develop methods and tools for mining and analyzing repository and trace data with the intent to improve developer productivity and software quality. This research has resulted in some important innovations and insights, such as bug prediction models and recommendation tools.

One advantage of research methods that rely on this data is that the participants make these records with no interference from the experimenters, and therefore the participants studied are not subject to reactivity. These tools also make it possible to collect data from a wide array of sources, increasing the external validity of the findings. In the cases where data is publicly available, it can be easier for researchers to replicate findings.

A disadvantage is that as more and more tools capture operational data (not from just from developers, but also from users), researchers have to deal with the increasing variety, velocity and volume of this big data, making it difficult and expensive to study. But another—and perhaps more important—disadvantage concerns the internal validity of these studies, as such methods may be limited at explaining why or revealing factors that may influence why certain behaviors or phenomena occur.

3.2 Borrowing methods from the social sciences

Although data mining and data analytics have been effective at describing or predicting certain behaviors, they are not so effective at explaining why certain behaviors may occur, or how certain development practices or tools could potentially be improved. Trace data may be poorly linked to the concepts we wish to explore or understand, concepts such as stakeholder motivations or barriers to community participation. When development data is mined, there is often the underlying assumption that developers are rational “animals”, when in fact they may not be at all rational and may have private motivations for their actions.

Furthermore, much of the work that is done in developing software is invisible work and cannot be studied by considering the development artifacts alone. Naur points out that visible software is more than just the externalized code and documentation—it is also dependent on extensive tacit knowledge that resides in the heads of developers who authored it, maintain it, or use it. And even when development activities or development knowledge may be visible or explicit, this information may be subtly fragmented across multiple channels and tools. This fragmentation of developer knowledge and communication is exacerbated as developers create and adopt more and more tools to support their development work. Finally, when studying social media tools, “Dark Matter Developers” that lurk or simply use information posted on social channels do not leave a trace behind them; however, our studies need to be aware of such developers’ needs and influences.

To study these more subtle yet important human and social issues, we have borrowed and adapted research methods (e.g., ethnographies, interviews and surveys) from the human and social sciences. These methods have given us important insights into the benefits social tools bring and the barriers developers may face using development tools and communication channels to participate.

3.3 Bridging the methodological divide

Clearly there are advantages and disadvantages to the choice of research methods and certain tradeoffs need to be made. Siegmund et al. note: “There is an inherent trade-off in empirical research: Do we want observations that we can fully explain, but with a limited generalizability, or do we want results that are applicable to a variety of circumstances, but where we cannot reliably explain underlying factors and relationships? Due to the options’ different objectives, we cannot choose both.” McGrath also describes tradeoffs in terms of the precision of the data that can be collected, the realism of the setting studied, and the collected evidence’s generalizability to the possible population of actors.

Our research approach has been to follow a pragmatic ap-
proach by using mixed methods \[3\] where we let our re-
search questions (which may be exploratory, explanatory or con-
firmatory) guide the selection and order of the appropri-
ate research method or methods \[4\]. In our studies of com-
munication and social channels, the methods we have used in-
clude: mining and analysis of software artifacts, ethnog-
raphic observations, and interviews and surveys.

The role of theory in our studies varies greatly depending on
the stage of our research \[5\]. In the case when we have an
existing (but perhaps preliminary) theory, the theory guides
which variables should be measured. In the case where a
theory is emerging, we may use it to categorize the data.
We have found the need to use multiple methods to assist in
theory development.

Too often researchers divide research methods into qualita-
tive and quantitative camps. Thankfully most research
methods involve the collection of both kinds of data, al-
though a method may favor one kind of data over the other.
In the case of mining, the data is typically quantitative in
nature (but not always, e.g. email communications), and in
the case of participant studies in the field, the data may be
predominantly qualitative, but again not always. For the
large-scale surveys we have conducted (some with over 2000
developers), many of our questions were quantitative (e.g.,
when we probed on the number of developers they collabo-
rated with).

4. TOWARDS A PARTICIPATORY
RESEARCH CULTURE
We have found that social media is causing a paradigm shift
in software engineering (just as it has in domains such as
politics and journalism), and likewise believe that it may
result in changes in our research community.

During the course of our research, we have found it beneficial
to use social media to study social media use in software en-
gineering, and to disseminate our research findings. Under-
standing the culture and nuances of social media language
enabled us to reach out to and connect with study partic-
ipants. In our most recent study, we attracted over 2000
responses to a long and time consuming survey by inform-
ing our participants that our results would be openly shared
as we have done with other studies—e.g., we tweeted and
blogged\[^5\] about the results from our study of Twitter \[15\]
and received excellent feedback that helped us validate our
findings and add additional insights to our research results.
Our blog saw thousands of views in a single day and dozens
of comments from real practitioners. These channels provide
a way for us to validate our findings at a speed and scale
not possible before. As our collaborator, Gousios recently
blogged\[^6\]: “The benefit is mutual: developers learn about ex-
iting results while our research is getting spread. My blog post
about the results of the ICSE 2015 paper is by far the most
read one in my blog (around 8k views now). I really doubt
that that many people read my paper.” Furthermore, we can
engage in a more in-depth discussion with participants pro-
viding us with a deeper understanding of the phenomena we
study as well as revealing new research questions.

Many software engineering researchers have already estab-
lished excellent social media literacy skills. We have used
our social graph to gain research data and insights, and to
form alliances with developers and collaborations with other
researchers. Just as some software engineers consider Twit-
ter to be an essential tool in keeping up to date on technology
developments \[23\], we also find it a useful avenue for dissemi-
ating our research and learning about related research in
our field. It also becomes a useful backchannel for discussing
research as it is being presented at conferences.

However, there are many challenges and risks from using so-
cial tools in our research community. We mentioned the need
for researchers to develop improved literacy skills, otherwise
they may miss out on research developments. Another risk
is that easier access to participants through public channels
(such as GitHub) may lead to us inadvertently spamming
our participants. We need to take care to reach out to them
when we have very good reasons to do so—one way to ensure
this is to pilot smaller studies and seek feedback on prelimi-
inary findings before reaching out to a wider audience. In our
studies, we make sure to explain the purpose of the study
and its benefits, and not to send invites to the same people
we used for previous studies.

Another issue we need to be aware of is to ensure that we do
not somehow cause harm to our participants. We refer to the
now famous Facebook study that manipulated users’ feeds to
study emotional impact: “Having written and designed this
experiment myself, I can tell you that our goal was never
to upset anyone. I can understand why some people have
concerns about it, and my coauthors and I are very sorry for
the way the paper described the research and any anxiety it
caused. In hindsight, the research benefits of the paper may
not have justified all of this anxiety.”\[^6\]

Lastly, our future work should consider if our interactions
with our research participants have an impact on their prac-
tices when we share and discuss our results with them. Al-
though, we have not intended it so far, our research may be
approaching an action research methodology \[8\].

In conclusion, we suggest that social media can have a trans-
formative impact on software engineering research—through
social media, researchers have the opportunity to influence
and guide the industry. We further propose that researchers
can benefit from the use of social media to help in sharing
and disseminating research results with one another and in
forming collaborations with others. We look forward to dis-
cussing the benefits and challenges of social media use for
both developers and researchers in software engineering.

5. ADDITIONAL AUTHORS
Alexey Zagalsky (University of Victoria, BC, Canada
alexeyza@uvic.ca) and Leif Singer (University of Victoria,
BC, Canada leif.singer@uvic.ca).

\[^{http://blog.leif.me/2013/11/}
how-software-developers-use-twitter/\[^{http://www.gousios.gr/blog/}
Scaling-qualitative-research/\[^{https://www.facebook.com/akramer/posts/}
l01b2095f150867796\]
6. REFERENCES


