How Social and Communication Channels Shape and Challenge a Participatory Culture in Software Development

Margaret-Anne Storey, Alexey Zagalsky, Fernando Figueira Filho, Leif Singer, and Daniel M. German

Abstract—Software developers use many different communication tools and channels in their work. The diversity of these tools has dramatically increased over the past decade and developers now have access to a wide range of socially enabled communication channels and social media to support their activities. The availability of such social tools is leading to a participatory culture of software development, where developers want to engage with, learn from, and co-create software with other developers. However, the interplay of these social channels, as well as the opportunities and challenges they may create when used together within this participatory development culture are not yet well understood. In this paper, we report on a large-scale survey conducted with 1,449 GitHub users. We discuss the channels these developers find essential to their work and gain an understanding of the challenges they face using them. Our findings lay the empirical foundation for providing recommendations to developers and tool designers on how to use and improve tools for software developers.

Index Terms—Participatory culture, communication, social media, CSCW, software engineering

1 Introduction

Software development has transitioned from a predominantly solo activity of developing standalone programs, to a highly distributed and collaborative approach that depends on or contributes to large and complex software ecosystems. Many developers now contribute to multiple projects, and as a result, project boundaries blur, not just in terms of their architecture and how they are used, but also in terms of how they are authored. Developers want to engage with, learn from, and co-create with other developers, forming a participatory culture [1] within many development-related communities of practice [2]. Many developers not only care about the code they need to write, but also about the skills they acquire [3], the contributions they make, and the connections they establish with other developers. These activities, in turn, lead to more collaborative software development opportunities.

To support developers' collaboration and communication needs, modern development tools are often integrated with or supplemented by communication channels and social media [4] (e.g., email, chat, or microblogging services). The rich and varied ecosystems of tools that developers use help them discover important technological trends, co-create with other developers, and learn new skills. Furthermore, these social tools foster creativity, promote engagement, and encourage participation in development projects. We see that the collaborative and participatory nature of software development continues to evolve, shape, and be shaped by communication channels that are used by development-related communities of practice [5]. We use the general term *communication channel* to refer to traditional communication channels (e.g., telephone, in-person interactions) as well as social features that may be standalone or integrated with other development tools (e.g., email, chat, and forums).

Within a community of practice, software is the combination of externalized knowledge (e.g., code, documentation, history of activities) as well as the tacit knowledge that resides in community members' heads (e.g., experience of when to use an API, or design constraints that are not written down). Communication channels and development tools support developers in forming and sharing both externalized and tacit knowledge in a highly collaborative manner. However, not much is known about the impact this participatory culture may have on software development practices, velocity, and software quality.

In this paper, we investigate how the choice of communication channels shapes developers' activities within a participatory culture of development, as well as explore the challenges they encounter. We report on a large-scale survey with developers that contribute to either collaborative or community-based development projects on the popular GitHub code hosting site. We wanted to uncover the demographics of the developers participating in this community, and we aimed to understand what channels and tools these developers use to support learning, discovery, and collaboration with others. Our survey revealed the communication channels these developers find essential to their work, and we gained an understanding of the challenges they face. These insights have led to several recommendations on how to use and improve communication and social tools

- *M.-A. Storey, A. Zagalsky, L. Singer, and D.M. German are with the University of Victoria, Victoria, BC V8P 5C2, Canada. E-mail: {mstorey, alexeyza, lsinger, dmg}@uvic.ca.*
- *F. Figueira Filho is with Universidade Federal do Rio Grande do Norte, Natal 59078-970, Brazil. E-mail: fernando@dimap.ufrn.br.*

Manuscript received 17 July 2015; revised 22 May 2016; accepted 1 June 2016. Date of publication 0 . 0000; date of current version 0 . 0000.
Recommended for acceptance by H. Sharp.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSE.2016.2584053

for developers. Our work investigates the following research questions:

RQ1 *Who is the social programmer* that participates in these communities?
RQ2 *What communication channels* do these developers use to support development activities?
RQ3 What communication channels are the most *important* to developers and *why*?
RQ4 *What challenges* do developers face using an ecosystem of communication channels to support their activities?

In previous work [6] we briefly described the first iteration of our survey (conducted in 2013) with some initial high-level results. This paper describes our survey in detail and provides in-depth analyses of the results from two deployments of the survey, conducted at the end of 2013 and again at the end of 2014. This work has formed an initial descriptive theory of the role communication channels play in supporting software development activities within a participatory development ecosystem.

2 Background

We begin with an overview of communities of practice and communication channels in software development, illustrating the interplay between them. The ongoing formation and evolution of these channels brings numerous challenges, both to the individual software developer and to the community as a whole.

2.1 Communities of Practice in Software Development

Communities of practice are groups of people connected by the similarity of their activities [2]. Such communities can be found in many domains, including software development. Community members do not have to be spatially or socially connected, but they solve similar problems and learn from one another through processes like apprenticeship or mentoring. Members advance through a process called legitimate peripheral participation: novices watch passively and then take on peripheral activities that are not vital, but nevertheless provide value to the community.

For example, in open source development, a potential contributor may start by only reading discussions and reporting defects. Over time, these contributors learn community conventions and move closer to the core group of experts. Developers may start fixing bugs and progress to the point where they can add their own features. They might gain commit rights, and at some point become involved in strategic project decisions. This phenomenon has also been observed by Crowston et al. [7] and Pham et al. [8].

We see that developers consider themselves to be part of a broader community of like-minded individuals that learn from and teach one another. In open source projects, professionals and hobbyists contribute to the same projects and interact in the same communities—some companies that rely on open source projects may have their own staff contribute to them.

Since software development lends itself to distributed or remote work—collaborators need not be in the same office, city, or time zone [9]—developer communities arise on a global scale and are often connected through tools that incorporate social aspects to help developers make contact with and learn from each other. In a sense, socially enabled tools and media can be considered catalysts to the formation of global, virtual software development communities of practice.

2.2 The Importance of Media in Software Development

When we think of software development tools, our first thoughts often concern development environments, debuggers, source code forges, version control systems, and bug trackers. But as Naur [10] emphasizes, software is much more than the code being developed. Software also involves the immediate knowledge in developers' minds and the documentation accompanying the code. Thus, other tools that play an essential role in collaborative development include project management tools and communication channels such as mailing lists [11], micro-blogging services [12] and chat [13], [14]. These tools support knowledge management activities that are central to the success and longevity of large software projects.

Naur also argues that programmer knowledge transcends documentation in three primary ways: it helps relate the software back to the real world, it helps explain why each part of the software is what it is, and lastly, it allows for modification of the software while maintaining a mapping to real-world aspects. Furthermore, Naur notes that certain software development activities (e.g., maintenance and continued support) are dependent on knowledge which is **distributed** among the members of the development group. Thus, in this paper, we look at software development as an activity that creates two types of knowledge—tacit and externalized—and we explore how developers use media to communicate and share this information.

Likewise, Scacchi's open source studies refer to the importance of "software informalisms", which he says are "information resources and artifacts that participants use to describe, proscribe, or prescribe what's happening in a OSSD project." [15] Such informal narratives are captured and related in a myriad of online Web-based communication artifacts and documentation resources (such as emails, blogs, wikis, IRC). Together, these resources comprise a distributed knowledge base that continually evolves as participants gain knowledge about the systems they develop and use.

Media—i.e., development tools and communication channels—play a critical role in how externalized and tacit knowledge is formed, shared, manipulated, and captured. These tools and channels become an extension of the programmer [16], helping them extend and distribute their cognition to develop, refine, and share knowledge. We previously [6] reported on an in-depth survey of how tools and channels play an essential role in communicating knowledge that is:

- captured in developers' heads;
- externalized in tools;
- stored in community knowledge resources; or
- captured in developer networks.

Fig. 1 provides a historical perspective of how different tools support the communication of these types of knowledge during software development. This simplified view—developed in our previous research—reveals how

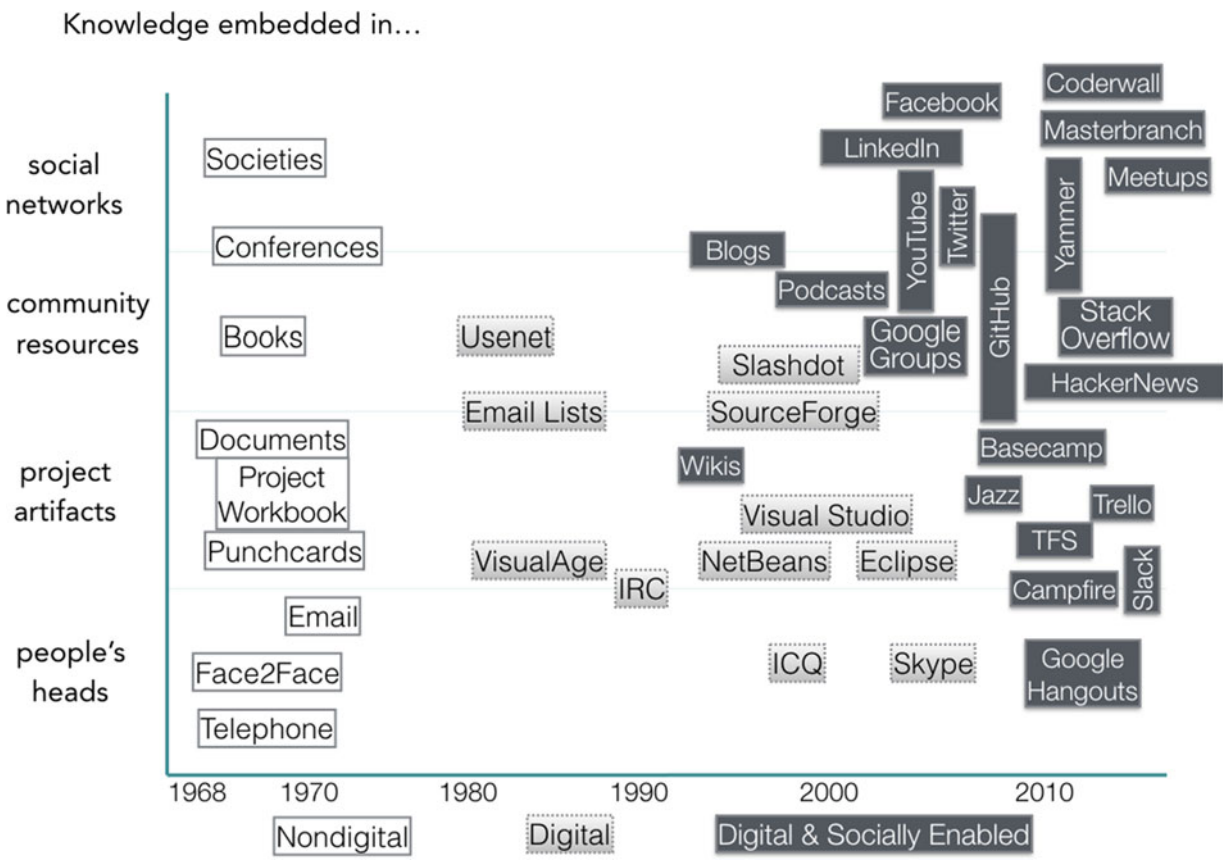


Fig. 1. Development communication channels over time and how they support the transfer of different kinds of developer knowledge.

development tools and media have evolved from a non-digital form to a digital form, eventually becoming infused with social features. We see a recent trend towards the use of social media channels and the embedding of social features in development tools. Other researchers have also noted an increase in the number of tools (in particular, social tools) adopted by software developers [17], [18], [19].

In Fig. 1, we distinguish tools that support the communication and capture of knowledge in developers' heads from knowledge that is externalized through development artifacts and tools, and from knowledge that is embedded within community resources or social networks. The inclusion of more social aspects over time has led to an increase of knowledge that is stored in community resources and social networks. Our previous work [6] provided an overview of the research that has been conducted on social channels in software development. Giuffrida and Dittrich [20] elaborated on this topic further by providing a systematic mapping study on research that has investigated the use of social software in global software development.

2.3 The Rise of the Social Programmer

Developers are becoming increasingly social and rely on their social networks to keep up to date, to find projects to contribute to, and to find others to collaborate with. They rely on tools to help them participate effectively in these social networks, although sometimes they also face hurdles in participating and staying up to date. The rise of the social programmer [6], [21] and the ways that communities of developers make use of increasingly social tools have led to the emergence of a highly participatory culture of software development. Jenkins [1] defines a participatory culture as one that:

- lowers barriers to participation;
- provides strong support for sharing;
- facilitates informal mentorship;
- has its members believe their contributions matter; and
- values social connections and what others think.

While this framework helps us better understand how developers work in this new context, we do not have a good understanding of how particular combinations of channels and tools shape and are shaped by communities of developers. We also do not adequately understand which channels support which knowledge activities, and whether individual developers face challenges using such a complex ecosystem of tools while contributing to potentially many different communities and projects. Achieving a deeper understanding of how media shape this participatory culture will guide tool designers and provide recommendations for how individual developers, teams, and communities should use the media effectively. In the next section, we describe a large-scale survey we designed to investigate this topic.

3 Methodology: The Developer Survey

Our overarching research goal was to understand how communication channels and social media support a broad set of knowledge activities within a participatory culture of software development. To help realize this goal, we designed and conducted a survey to learn how developers use tools to

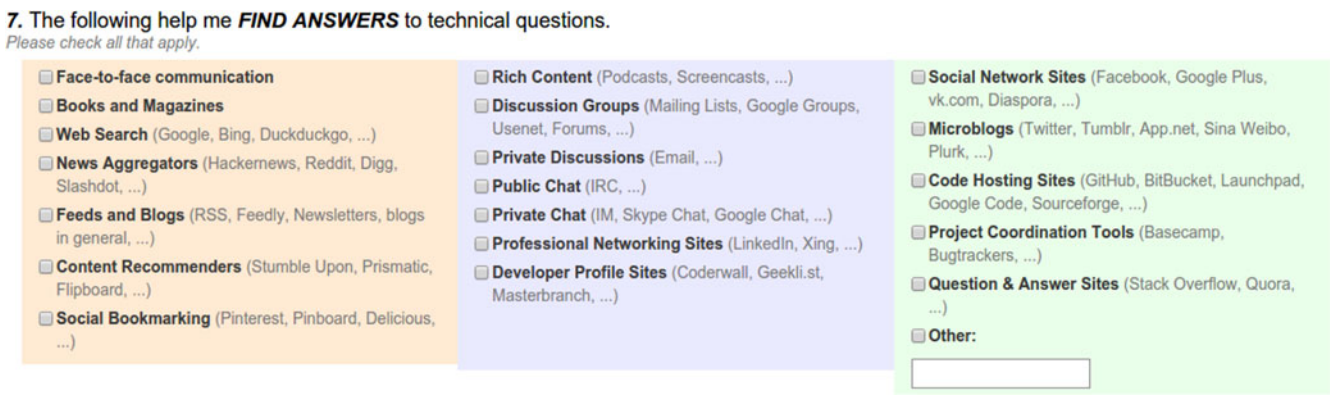


Fig. 2. The channel matrix designed for the survey.

support their knowledge activities, what media channels are important to them, and what challenges they face.

We deployed the same survey during two different time periods: at the end of 2013 and at the end of 2014. For both deployments, we downloaded account data for the most recently active GitHub users with public email addresses. To indicate activity, we used the 25 event types defined by the GitHub API v3.[1] Most of these events concern development tasks such as committing code, creating repositories, and creating issues, but there are also more general events related to following users and watching repositories. To find developers for our survey, we used the GitHub Archive[2] to query public events happening on GitHub. Therefore, our findings are limited to this population. We sorted events by their timestamp and excluded users who did not have public email addresses at the time we sent our invitation emails. For the second iteration, we also ignored users we had emailed in the first iteration. We focused on this population of developers because GitHub is currently the most widely used social coding platform by developers who contribute to one or more collaborative development projects in an open manner.[3]

We emailed our survey to 7,000 active GitHub users during November and December of 2013, and to 2,000 active GitHub users in December of 2014. 1,492/332 developers responded to the two instances of the survey in 2013 and 2014, respectively (21 and 16 percent response rates). The only statistical difference between the two deployments was an increase in the number of women (from 3.5 to 6.3 percent, $\rho = 0.042$). We combined the responses from both surveys and ignored incomplete ones, resulting in 1,449 survey responses.

Our survey followed several iterations of design and was based on an in-depth review of the existing literature on software engineering as well as related literature on knowledge work. In the survey,[4] we first inquired about the developers' **demographics**. We then inquired about **communication channel** use for a set of 11 **development activities**. The set of activities was informed by our review of the literature that examines tool and communication channel use by software developers [6]. These activities, which go beyond finer grained development and project management activities, were as follows:

- **STAY UP TO DATE** about technologies, practices, and tools for software development
- **FIND ANSWERS** to technical questions
- **LEARN** and improve skills
- **DISCOVER** interesting developers
- **CONNECT** with interesting developers
- **GET** and **GIVE FEEDBACK**
- **PUBLISH** development activities
- **WATCH** other developers' activities
- **DISPLAY** my **SKILLS/ACCOMPLISHMENTS**
- **ASSESS** other developers
- **COORDINATE** with other developers when participating on projects

The survey questions relating to activities all followed the same form: we used a matrix of options that the survey respondents could select from to indicate that an item was used for the corresponding activity (see Fig. 2 for an example question about activity and channel use). The social channels specified in the matrix were determined from our own knowledge as developers, as well as through feedback from fellow developers. The channels were refined using the research literature and through piloted surveys. We included an "Other" option to elicit channels we did not consider. We asked developers to rank the **most important tools and channels** they used to support development activities and explain why those tools were important.

We also aimed to understand what **challenges** developers face using social channels, probing about privacy, interruptions, and feeling overwhelmed, as these were concerns that came up in earlier studies conducted with adopters and non-adopters of social media (e.g., Singer et al. [12]).

The survey instrument is one of the contributions of this paper and its source code can be found in a repository on GitHub.[5] This will allow others to replicate our survey and build upon our work. We describe our analysis approach as we present each research question, and refer to the limitations of the study in the Discussion section of the paper.

In the following sections, we present the results from our analysis of the survey responses to answer our four research questions.

1. https://developer.github.com/v3/activity/events/types/
2. https://www.githubarchive.org
3. https://octoverse.github.com/
4. http://thechiselgroup.org/2013/11/19/how-do-you-develop-software/

5. https://github.com/thechiselgroup/devsurvey

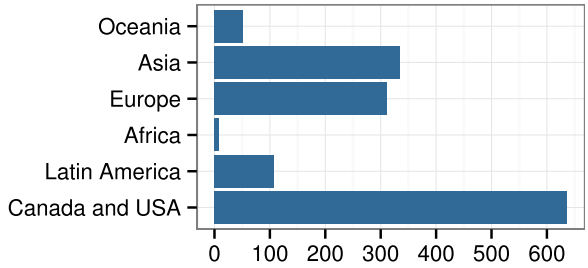


Fig. 3. Demographics of the programmers that answered the survey (those recently active on GitHub with public activity).

4 Social Programmer Demographics

For our first research question (**RQ1**), we wished to characterize the *modern social programmer* that openly participates on projects hosted on the GitHub social coding platform.

We asked our survey respondents to provide demographic information such as gender, age, geographical location, programming experience, the programming languages they use, the number of projects they participate in, whether they program professionally, and the size of the project teams they have worked with. The answers to these questions are summarized in Figs. 3, 4, and 5.

Geographic Location: Our survey was successful in attracting respondents from all over the world: 43.4 percent from North America, 24.2 percent from Asia, 21.1 percent from Europe, 7.1 percent from South or Central America, and 4.1 percent from Africa or Oceania. It is notable that there were more respondents from Asia than Europe: 143 respondents originated in China, making it the second most frequent country of origin after the United States, which had 547 respondents. Canada was third with 90 respondents.

Gender: The overwhelming majority of our respondents identified as male—only 3.9 percent said they were female. However, it is possible that other respondents were female but did not wish to be identified as such.[6]

Age: 56.7 percent of respondents said they were between 23 and 32 years of age (so-called millennials), representing the largest age group in our survey and showing a strong bias towards relatively young developers. In fact, 77.9 percent said they were 32 or younger. 3.7 percent were older than 45 and only 0.4 percent were older than 60.

Team Size: Team size was slightly more evenly distributed. Only 1.8 percent of respondents said they had worked in teams of more than 50 members. We found a slight bias towards smaller teams, with 61.5 percent having worked on teams of five members or less and 16.2 percent saying they had only worked on projects where they were the sole member.

Programming Experience: In terms of experience, responses varied. Only 5.1 percent had 1 year of experience or less. 33.5 percent had worked as a developer for 2 to 5 years, 29.1 percent for 5 to 10 years, 24.4 percent for 10 to 20 years, and only 7.6 percent for more than 20 years.

Number of Projects: The majority of our survey respondents (88.9 percent) had worked on 5 projects or less and most had experience working on 2 (21.5 percent), 3 (27.7 percent), or 4 (15.7 percent) projects.

6. http://meta.stackoverflow.com/a/281304

Professionalism: Most respondents were *professional* software developers (78 percent). 54 percent considered themselves open source developers and 51 percent worked on pet projects.

Programming Languages: Fig. 5 shows a word cloud of programming languages used by the participants, while Table 1 shows the most popular languages. The three most popular languages included JavaScript (61.9 percent), Python (44.6 percent), and Java (41.5 percent). This may indicate that at least 60 percent of our respondents develop for the Web.

Table 2 shows the results of testing independence between the different factors surveyed. Our respondents provided three types of answers: categorical (including dichotomous), such as whether the respondent was a professional programmer; ordinal, such as how concerned they were about their privacy; and numeric, such as the number of different channels used. This forced us to use different tests of independence for each pair of factors: for pairs of categorical factors, we used chi-square; for pairs of one categorical and one ordinal or numeric, we used Kruskall-Wallis; and for pairs of two ordinal or numeric, we used Spearman correlations. We highlight some of the differences below.

Regarding age, we found a moderate positive correlation between age and programming experience ($\rho = 0.56$, $p \ll 0.001$); also, older programmers are more likely to work on professional projects (H=160.63, df=1, $p \ll 0.001$) and less likely to work on open source projects (H=34.87, df=1, $p \ll 0.001$). However, there is almost no correlation between age and team size ($\rho = 0.09$, $p = 0.001$).

Regarding gender, we found that female programmers are less likely to have professional experience (H=21.55, df=4, $p \ll 0.001$) and pet projects (H=10.6, df=2, $\phi = 0.08$, $p = 0.05$), and they work on fewer projects than their male counterparts (H=14.79, df=6, $p = 0.022$).

People working in larger teams are more likely to be professional programmers (H=65.05, df=1, $p \ll 0.001$). There is very little (if any) positive correlation with both the number of projects they are members of ($\rho = 0.18$, $p \ll 0.001$) and the different channels they use ($\rho = 0.18$, $p \ll 0.001$).

When a person is a professional programmer, it is less likely they will work on open source projects (H=102.6, df=1, $\phi = 0.27$). However, a person that has a pet project is more likely to also be involved in open source projects (H=177.4, df=1, $\phi = 0.35$).

Regarding the different number of channels a person uses, there is very little (if any) positive correlation with team size ($\rho = 0.18$, $p \ll 0.001$) and the number of projects they belong to ($\rho = 0.18$, $p \ll 0.001$).

5 Communication Channels Developers Use to Support Development Activities

To answer our second question (**RQ2**), we asked the respondents to indicate which channels they use for a variety of software development activities. As mentioned, these activities were determined through a literature review and from our previous research.

On average, developers indicated they use 11.7 channels across all activities, with a median of 12 and quartiles of $[9, 14]$ (see Fig. 6 for the distribution of channels used by survey respondents).

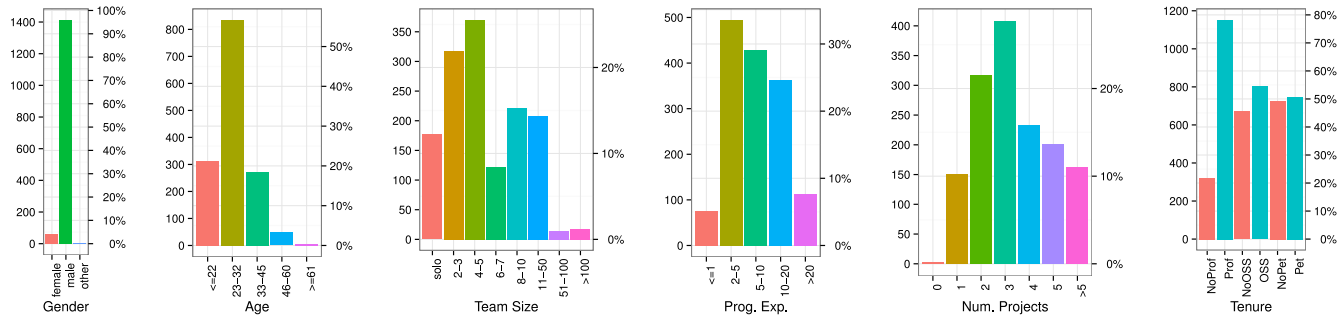


Fig. 4. Demographics of the programmers that answered the survey (those recently active on GitHub with public activity).

Table 4 shows an overview of the channels developers said they use to support different activities. In this table, we grouped the channels according to *analog*, *digital*, and *social +digital* (see our discussion for Fig. 1). This table highlights that there appears to be more reliance on communication channels that support social features. For each activity, we generated a radar graph showing the distribution of responses about channel use (see the companion website[7]).

For each activity, we also asked developers to indicate other channels they might use that were not listed in the survey (see Table 3). There are a number of channels that we did not specify or did not specify clearly enough in our survey, including events and meetups, software documentation, and personal blogs and Websites. The respondents also mentioned research papers, headhunters, recruiting Websites, and conference calls as important ways of supporting their activities.

Although Table 4 paints a high-level picture of which channels tend to support a set of development activities, a limitation with this data is that we only know that developers use the selected channels for supporting a given activity—we don't know how important these channels are. Consequently, we asked the respondents to tell us which three channels are the most important across all activities and why. We report these results in the next section.

6 Software Developers' Most Important Communication Channels

Beyond mapping the channels developers use as part of their software development activities, it is also important to understand *why* these channels are important to them. For this purpose, as well as to answer **RQ3**, we asked developers to indicate the top three channels that are important to them and why each one is important. Fig. 7 shows the number of responses given per channel.

In the following, we provide more insights into why certain channels were perceived as important to the developer respondents. We also provide quotes from specific participants, indicated by P#.

Code hosting sites allow for better team collaboration, group awareness, and project coordination. The ability to share one's code on the Web has lowered the barriers to entry by making source code easily accessible: *"All levels of users and employees know how to use it: The hard-core developers use the command-line-based tools, and the 'end users' just use the Web interface, without feeling overwhelmed."* [P64]

7. http://thechiselgroup.github.io/channel-study/

Face-to-face interactions were also deemed very important by our survey respondents. Developers can receive rapid feedback from their co-workers which facilitates talking through complex problems, discussing ideas, and making design decisions: *"Nothing beats being able to sit one-on-one and talk through a topic, plan out a design or just converse while coding. This is also my favorite way to learn from an instructor because of the ability to ask as many questions as possible and have an open conversation."* [P319] Some respondents reported using videoconferencing as a way to mimic co-located interactions with other developers.

Q&A sites offer a quick way to debug issues while providing access to high-quality answers: *"Almost any question that I have, I can get an answer through these sites."* [P635]

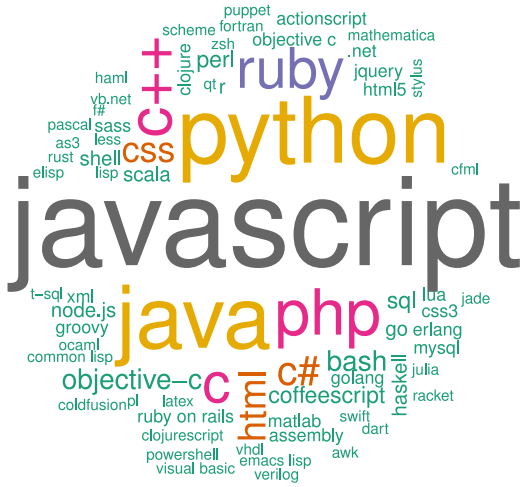


Fig. 5. Word cloud of the programming languages used by the programmers that answered the survey.

TABLE 1
Most Popular Languages

Language	Frequency	Percentage
JavaScript	912	61.9
Python	657	44.6
Java	611	41.5
PHP	411	27.9
C++	383	26.0
C	351	23.8
Ruby	341	23.1
C#	213	14.5
HTML	194	13.1

TABLE 2
Test of Independence between the Different Demographic Factors in the Survey, the Number of Channels They Use, and Their Responses Regarding Privacy, Feeling Overwhelmed, and Being Distracted

	Age	Gender	TeamSize	Prog. Exp.	Tenure Prof	Tenure Pet	Tenure OSS	Number Projects
Gender	kw: 2,2.58							
Team size	sp: **0.09*****	kw: 7,3.61						
Prog. Exp.	sp: **0.56*****	kw: **4,21.55*****	sp: **0.11*****					
Tenure Prof.	kw: **1,160.63*****	cs: 2,3.8,0.05	kw: **1,65.05*****	kw: **1,128.13*****				
Tenure Pet	kw: 1,0.49	cs: **2,10.6,0.08****	kw: **1,4.24***	kw: **1,55.39*****	cs: 1,0.4,0.02			
Tenure OSS	kw: **1,34.87*****	cs: 2,4.7,0.06	kw: **1,8.35****	kw: **1,4.55***	cs: **1,102.6,0.27*****	cs: **1,177.4,0.35*****		
Number Projects	sp: **0.05***	kw: **6,14.79***	sp: **0.18*****	sp: **0.19*****	kw: **6,84.80*****	kw: **6,58.26*****	kw: **6,18.39****	
Channels Used	sp: 0.03	kw: 19,27.17	sp: **0.18*****	sp: 0.05	kw: **19,63.91*****	kw: **19,32.58***	kw: 19,26.27	sp: **0.18*****

Each value is preceded by the name of the test used followed by its results: kw *represents Kruskall-Wallis (degrees of freedom, χ^2 value),* cs *represents Chi-square (degrees of freedom, χ^2 value and Cramer's ϕ), and* sp *represents Spearman correlation (r value). Values in bold represent where there the two factors appear not to be independent with $p < 0.05$, specifically *** corresponds to $p < 0.001$, ** for $p < 0.01$, * for $p < 0.05$.*

Other respondents mentioned additional uses of Q&A sites, including learning from code examples and getting feedback from experts.

Search is an essential tool for finding information: *"Good for finding the initial direction; also [...] to learn something new."* [P484] It also provides quick access to software documentation and supports problem solving. Many respondents reported using search engines as the entry point for finding answers on Q&A sites.

Microblogs provide just-in-time awareness of the latest advancements and updates in the development community: *"Allows me to get up-to-date information on topics I'm interested in—conferences, new releases, new articles/books, etc."* [P95] They were also considered important for getting feedback from other developers and for nurturing relationships with like-minded people.

Private chats (e.g., IM, Skype chat, Google chat) are essential tools for supporting team communication and collaboration through a single channel: *"It provides a single channel to digest and discuss everything that is going on with the team."* [P415] Many survey respondents felt that private chats are the closest replacement for face-to-face interactions when quick feedback is needed and team members are geographically distributed.

Feeds and blogs provide the most up-to-date information on development practices and technologies: *"By following several feeds, one can find out how veterans use a tool/technology/language... And it's easier to know the trends"*. [P1016] Blogs encapsulate a more personalized view on a given topic and are an important channel for documenting techniques while sharing specific coding tips and tweaks that can be used by other developers.

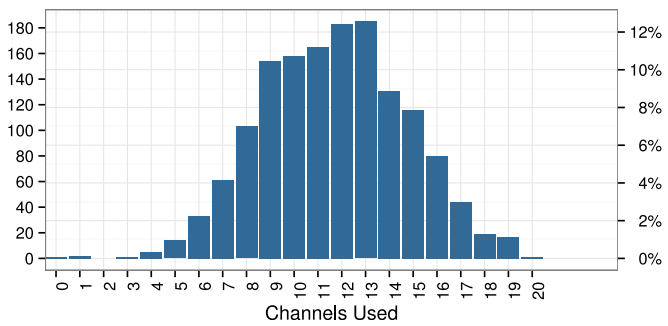


Fig. 6. Histogram of number of channels selected by each developer.

Private discussions (e.g., email) support communication across virtually every platform and among different stakeholders (e.g., customers and users). They are a convenient channel for disseminating information to large groups (e.g., mailing lists) while keeping conversations private and persistent for later retrieval: *"This is how you get to communicate privately and can have proof for a later stage."* [P1041]

TABLE 3
Other Channels Reported in the Survey

Activity	Channels
Keeping up to date	events and meetups [18], software documentation [3], research papers [3], formal education [2], MOOCs [2]
Finding answers	software documentation [11], research papers [3]
Learning	educational sites and MOOCs [20], events and meetups [19], software documentation [13], tutorials [7], research papers [6], code reviews [4]
Discovering developers	headhunters [16], recruiting sites [12]
Connecting with developers	events and meetups [35], programming competitions [2], formal education [2]
Getting and giving feedback	events and meetups [9], code reviews [4], issue trackers [4]
Publishing activities	personal blogs and Websites [32], conferences [7]
Watching activities	events and meetups [4], personal blogs and Websites [3]
Displaying skills	personal blogs and Websites [64], resumes [6], events and meetups [5]
Assessing others	personal blogs and Websites [2], source code[2]
Coordinating with others	conference calls [10], cloud-based services (e.g., Dropbox, Google Drive) [8]

The values indicate the number of times the channel was mentioned by respondents for the corresponding activity.

TABLE 4
Channels Used by Our Respondents and the Activities They Support

	Face-to-face	Books	Web Search	Content Recommenders	Rich Content	Private Discussions	Discussion Groups	Public Chat	Private Chat	Feeds and Blogs	News Aggregators	Social Bookmarking	Q&A Sites	Prof. Networking Sites	Developer Profile Sites	Social Network Sites	Microblogs	Code Hosting Sites	Project Coordination Tools
	analog		digital							digital and socially enabled									
Stay Up to Date																			
Find Answers																			
Learn																			
Discover Others																			
Connect with Others																			
Get and Give Feedback																			
Publish Activities																			
Watch Activities																			
Display Skills/Accomplishments																			
Assess Others																			
Coordinate with Others																			

Legend: 0-10% | 10-20% | 20-30% | 30-40% | 40-50% | 50-60% | 60-70% | 70-80% | 80-90% | 90-100%

(Percentage of survey respondents mentioning a channel being used for an activity.)

Public chats (e.g., IRC) have the advantage of enabling communication among developers and users of a particular software project. By being public, anyone with an interest can join in and have a conversation with project maintainers: *"Gives me direct access to the people who write my tools, and gives me direct access to people who are using things I've written"* [P191]. Public chats also enable discussions and faster feedback among team members, even if they are distributed around the world: *"As a team spreads across the world, we use IRC to preview most of our concepts before any code is written."* [P731]

Discussion groups (e.g., mailing lists, Google groups, forums) support mass communication and coordination among people scattered across large and geographically distributed groups: *"We are a physics collaboration of 3,000 people, spread all over the world. Internal discussion groups are essential for coordination on all subjects, including software development."* [P365] Respondents also reported the usefulness of discussion groups for gathering customer feedback: *"Because it's where I find my customers' opinions and ideas."* [P130]

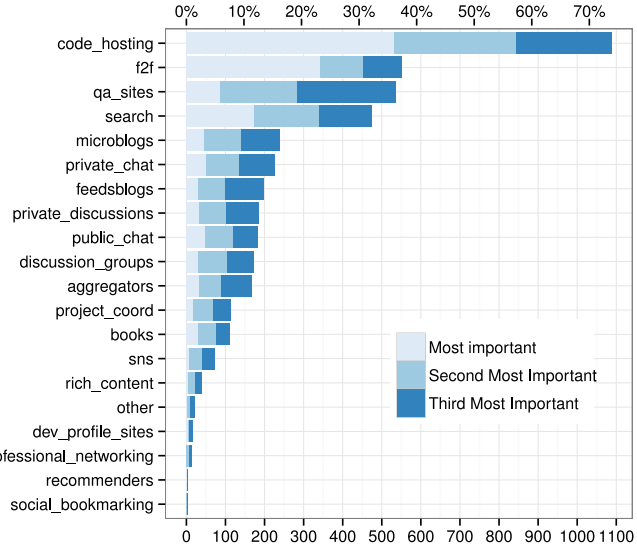


Fig. 7. Number of responses per channel indicating the importance of each channel.

Aggregators (e.g., Reddit) are socially curated channels focused on new trends. They provide access to crowdsourced content that has been filtered and collated by others, allowing for developers to stay up to date with the latest technologies without active participation. As one respondent put it, aggregators are *"[. . .] roughly the heartbeat of the current software dev industry. If a technology is worth talking about, it will be talked about."* [P1419] The value of aggregators is closely associated with the value of their supporting communities, and survey respondents appreciated that aggregators allow them to interact with like-minded developers and get their feedback: *"[Hacker News] is the most welcoming community I have ever seen. [. . .] You can interact with anyone (if they have public email) and the content quality is top notch."* [P1126]

Project coordination tools increase group awareness of current tasks and issues and provide a means for tracking progress and discussing next steps: *"Permits tracking in-progress work as well as receiving feedback. Essential for distributed teams."* [P105] These tools improve the transparency of a project's activities, increasing progress visibility not only among team members but also among clients: *"Helps us coordinate large tasks bases, especially when reporting back to clients."* [P486]

Books were indicated by some of our survey respondents as a cohesive and progressive way for learning about a topic: *"[They make] learning much easier than the hunt and peck method of digging through sites on the net."* [P1189] Another subtle but crucial advantage of books is that they are

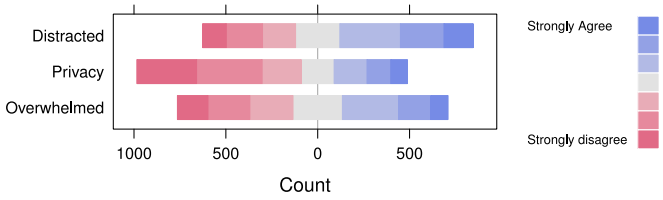


Fig. 8. Frequency of responses to Likert questions probing on developer challenges with **distraction, privacy, feeling overwhelmed**.

"distraction free and generally better thought through and considered." [P1319] Developers can gain in-depth and focused understanding about specific topics, while avoiding being distracted by the noise of concurrent information.

Social network sites increase awareness of the community and help developers disseminate information from other channels in various ways: *"Because most often they function as the entry point to more relevant information published on blogs, newspapers, books, etc."* [P224] In addition, developers can reach potential users more easily, which is essential for gathering feedback: *"We have a group for Android Development Testers in Google Plus where we can post things we want tested and receive almost immediate feedback."* [P1240]

Rich content such as screencasts and podcasts provide learning materials and communicate the state of the art in technologies, tools, and practices for software development. Developers are able to consume content while commuting or performing other tasks. One survey respondent highlighted yet another interesting aspect of learning using rich content: *"I'm a visual and audible learner. Seeing and hearing others makes learning better."* [P1354]

7 The Challenges Developers Face Using Communication Channels

Our fourth research question (**RQ4**) inquires about the challenges developers face using communication channels. Our previous work [3], [6], [12] revealed that developers face challenges related to distractions, privacy, and feeling overwhelmed by communication chatter when using social media channels. Consequently, our survey specifically asked whether the respondents experienced these concerns. We show the results from these Likert-style questions in Fig. 8. We note that privacy is not a big concern for everyone, whereas being interrupted and feeling overwhelmed by communication traffic are issues for more developers.

To investigate if there were any relationships between these three factors (Privacy, Distraction, Overwhelmed) and their demographics, we performed a more in-depth analysis of the responses. Table 5 shows the test of independence between whether a person feels their privacy is affected or not and if they feel overwhelmed or distracted by their use of communication channels, as well as the different demographic factors of our respondents.

We anticipated that age might influence responses in terms of privacy concerns, but no factor shows a statistically significant relationship with the Privacy factor. A similar result was found regarding the Distraction factor, where the only statistically significant result was that there is very little correlation (if any) with programming experience: $\rho = 0.07$, $p = 0.008$. The Overwhelmed factor was found to have a very low correlation (if any) to: age ($\rho = 0.07$, $p = 0.014$), programming experience ($\rho = -0.07$, $p = 0.012$), and number of projects ($\rho = -0.07$, $p = 0.004$). It was also found that people who work on open source projects feel slightly more overwhelmed than people who do not (H=19.89, df=6, $p = 0.005$). We believe these results show a lack of evidence that the developers who worry about privacy, feel overwhelmed, or feel distracted belong to any specific type of group (as reported in the survey). Nonetheless, it is notable that there is a modest positive correlation between the three factors (with $p \ll 0.001$): people who worry about their privacy feel overwhelmed ($\rho = 0.21$) and distracted ($\rho = 0.19$), and those who feel distracted also feel overwhelmed ($\rho = 0.24$).

We also asked the respondents to share any **additional challenges** they face through an open-ended text question. 432 respondents (356 to the 2013 survey, 76 to the 2014 deployment) either elaborated on the challenges mentioned above or informed us about other challenges they experience. A wide variety of challenges were reported and we coded, sorted, grouped, and then categorized them using an open coding and iterative clustering technique. The main categories that emerged from our analysis are as follows:

- Developer issues
- Collaboration and coordination hurdles
- Barriers to community building and participation
- Social and human communication challenges
- Communication channel affordances, literacy, and friction
- Content and knowledge management concerns

Postman, who extensively studied the use of media in communities of practice, refers to a **media ecology** and suggests we undertake the study of "entire environments, their structure, content and impact on people." [22] We note that the categories of challenges we found mirror the main areas of study suggested by Postman. We report these challenges below.

TABLE 5
Test of Independence between the Different Demographic Factors and Whether Respondents Feel Worried about Privacy, Feel Overwhelmed, or Are Distracted by Their Use of Communication Channels

	Age	Gender	Team Size	Prog. Exp.	Tenure Prof	Tenure Pet	Tenure OSS	Number Projects	Channels Used	Privacy	Over-whelmed
Privacy	sp: -0.04	kw: 6,8.26	sp: 0.01	sp: -0.04	kw: 6,10.69	kw: 6,11.39	kw: 6,8.97	sp: -0.04	sp: 0.03		
Over-whelmed	sp: **0.07****	kw: 6,9.36	sp: -0.01	sp: **-0.07***	kw: 6,11.71	kw: 6,12.57	kw: **6,19.89****	sp: **-0.07****	sp: **0.08****	sp: **0.21*****	
Distracted	sp: 0.04	kw: 6,4.94	sp: 0.01	sp: **0.07****	kw: 6,5.49	kw: 6,8.35	kw: 6,7.54	sp: 0.01	sp: 0.05	sp: **0.19*****	sp: **0.24*****

Each value is preceded by the name of the test used followed by its results: kw *represents Kruskall-Wallis (degrees of freedom, χ^2 value),* cs *represents Chi-square (degrees of freedom, χ^2 value and Cramer's ϕ), and* sp *represents Spearman correlation (r value). Values in bold represent when the two factors appear not to be independent with $p < 0.05$, specifically *** corresponds to $p < 0.001$, ** for $p < 0.01$, * for $p < 0.05$.*

Developer Issues:
- *Distractions and interruptions from communication channels negatively impact developer productivity*
  - Distractions 38*
  - Interruptions 11*
- *Keeping up with new technologies and project activities can be challenging, but social tools help*
  - Keeping up with new technologies 9
  - Keeping up with activities on projects 8

Collaboration and coordination hurdles:
- *Sharing and explaining code lack adequate tool support*
  - Sharing code 4
  - Explaining code 7
- *Getting feedback on development activities is challenging*
  - Getting feedback 8
  - Proprietary projects 3
- *Collaborative coding activities need improved tool support*
  - Collaborative coding 3

Barriers to community building and participation:
- *Geographic, cultural and economic factors pose participation barriers*
  - Time zones 14
  - Access to the Internet 3
  - Language barriers 20
- *Despite social channels, finding developers to participate is difficult*
  - Finding right people 9
  - Convincing others (to participate) 3

Social and human communication challenges:
- *Miscommunications on text-based channels are common*
  - Miscommunications 24
- *For many developers, face-to-face communication is best*
  - (Not) Face-to-face 24
- *People are challenging, no matter which channels are used*
  - Poor attitude 13
  - Intimidated 12

Communication channel affordances, literacy and friction:
- *Developers need to consider channel affordances*
  - Private vs. public 10*
  - Synchronous vs. asynchronous 10
  - Ephemeral vs. archival 3
  - Anonymous vs. identified 1
  - Text-based vs. verbal 7
  - Face to face (vs. not) see above
  - No one tool fits all 14
  - Communication with users 11
- *Developers need to be literate with communication channels*
  - Literacy 22
  - Lack of documentation 9
  - Learning tools 10
- *Communication channel friction can obstruct participation*
  - Tool friction 23
  - Search is inadequate 16
  - Poor mobile support 5
  - Vendor lock-in concern 6
  - Notification issues 5
  - Poor channel integration 8
  - Channel overload 36
  - Poor adoption by others 21

Content and knowledge management:
- *Use of many channels leads to information fragmentation*
  - Information fragmentation 15
- *The quantity of communicated information is overwhelming*
  - Quantity 11*
  - (Finding the signal in the) noise 19*
- *The quality of communicated information is hard to evaluate*
  - Quality 29
  - Obsolete information 8
  - Spam 4
  - Niche technologies 4
  - History of information missing 4

Strategies:
- *Developers used a variety of strategies to address their challenges*
  - Deciding when to use particular channels 3
  - Deciding which channels to use and how 6
  - Encouraging others to use tools 1
  - Unplugging 3

Fig. 9. This shows the categories, codes, and counts of each code occurrence in the additional challenges shared by participants. Note that some participants shared multiple challenges in the Other field. The **categories of challenges we derived are shown in bold text**; the *findings that emerged from the analysis are shown in italics*, followed by the codes and counts in normal font. Codes marked with an * indicate challenges the participants already indicated in the closed question (see Fig. 8). Strategies shared are described in the Discussion section of the paper.

A few participants also noted strategies they use to address the challenges they face, which we describe in the Discussion section of the paper. In the following, we provide representative participant quotes (shown in italics and linked to each participant using P#), and we use **bolded** text to indicate codes we assigned to quotes. Note that some responses from the survey were coded with multiple codes. The main challenges that emerged from our analysis are shown in boxes. When appropriate, we discuss and link the reported findings to relevant literature. We summarize the categories of challenges found, the main findings that emerged, and the codes with their respective counts in Fig. 9. Further details (linking categories to findings to codes, and counts of codes and additional quotes) can be found on the companion Website.[8]

However, we stress that counting the coded challenges could be misleading—only some participants took the time to share this information with us after an already long survey, and thus they may have selected which challenges to share with us in an *ad-hoc* manner. Nevertheless, for concerns that were mentioned numerous times, the counts may help us identify challenges that may be more prevalent and warrant further investigation—we share these counts in hopes of provoking future research. Two researchers independently coded these challenges, iterating several times to reach agreement on the codes derived and how they were applied. When agreement was not reached, the researchers followed a conservative approach and omitted applying those codes to the survey responses.

8. http://thechiselgroup.github.io/channel-study/

7.1 Developer Issues

Some of the challenges reported concerned the developers themselves or specific development activities.

Distractions and interruptions from communication channels negatively impact developer productivity:

Survey respondents spoke of how the social and communication media at their finger tips can be a **distraction** or can negatively impact their productivity through **interruptions** [23] (see also Fig. 8). As P165 mentioned, *"Social Networking Websites like Facebook are the worst ingredients for good concentration in general."* P541 described how easy it is to be drawn into unnecessary work: *"If I spend a lot of time talking with other developers about the best way to do things or reading articles on social sites, I end up constantly refactoring or optimizing code instead of making progress toward the functional requirements of the project."*

P320 discussed how notifications and emails can be both a cause for distraction and a waste of time: *"Notifications: when used in a moderate way, it is fine, but when overused, it is a distraction for developers. Emails: too many emails from Project*

Coordination Tools can easily waste 15-30 minutes only to go through them all in the morning, especially when I'm involved in more than a few projects simultaneously."

Keeping up with new technologies and project activities can be challenging, but social tools help:

Keeping up with new technologies was a concern for our respondents: *"Staying cutting edge is a never-ending task."* [P625] Several respondents discussed not knowing which channels to watch: *"Knowing where the activity is. Some days, Hacker News might be the best place to follow. Another day, Twitter might be. Another day, GitHub might be where I should look. Another day, it might be a site or network I'm not even aware of."* [P1564] The challenge of keeping up also emerged in our previous work that investigated how developers use Twitter [12]. P706 discussed how they found it easier to keep up with technology in open source because of their use of social tools: *"Staying up to date with new company internal technology because it is not on Twitter [or] Hacker News. I guess there is a theme here: if it's open source, help is plenty and readily available, not so much for internal tools."*

Tools for **keeping up with activities on projects** were seen as inadequate and in need of improvement: *"In a big project (WebKit, Mozilla, etc.) it can be hard to filter for only ongoing work that is relevant. Most legacy UIs are terrible (Bugzilla) and new ones (GitHub) lack features for large-scale development."* [P109] While Baysal et al. [24] also noted this challenge, other respondents described how some modern tools address it: *"We use HipChat (kind of a private IRC) with HUBOT that watches our GitHub activity. It's wonderful because our entire team can be instantly notified about who's doing what on which repository, and we're all in communication via mobile and desktop with the same feed."* [P892] Keeping up with projects also relates to how developers collaborate on projects, which is discussed next.

7.2 Collaboration and Coordination Hurdles

Respondents spoke of the challenges they face managing and coordinating their projects. As noted earlier, the vast majority of respondents said they had contributed to two or more projects, including professional projects (see Fig. 4).

Many of the challenges shared with us did not relate to the use of communication channels but rather **poor project documentation**, a **lack of requirements management**, **poor standards adherence**, and **managing software licenses**. Respondents also mentioned challenges specific to project coordination, such as *work distribution*, *workflow friction*, *scheduling hurdles*, and the *lack of a roadmap*: *"A lot of developers I know spend more time planning and debugging the workflow, rather than developing."[P49]*

Organizational constraints were also discussed, such as how outside communication was discouraged in one organization, and how there was a reliance on proprietary services in certain projects. However, some respondents did talk about the lack of tool support for collaborative coding and coordination activities, which is discussed next.

There is a lack of adequate tool support for **sharing and explaining code**:

Developers had difficulties **sharing code** and **explaining code** using their existing tools. P1416 discussed this issue and how he dealt with it: *"Many communication tools (email, IM, etc.) are not especially good for talking about code. Generally in any given conversation I'll end up using several tools, e.g., IRC + a paste-bin (GitHub Gists), to effectively communicate ideas."* P445 was enthusiastic about collaboration tool support except when needing to explain code: *"The biggest challenge in soft-dev for me is four-fold: communicating the idea (Hangout), managing the idea (Trello), logging the implemented idea (GitHub), and explaining the implemented idea with the team (Nitrous.io). The first three solutions are pretty solid. It's the fact you can't always sit right next to someone and show them the code and explain how everything works that is the most challenging part. Cloud9, Koding, Nitrous, etc. are all trying to solve the last problem."*

Getting feedback on development activities is challenging:

Developers also faced challenges getting **feedback** about their own activities, especially for **proprietary projects** that can't use social tools: *"Getting quick feedback for internal technology because you cannot ask on Stack Overflow."* [P706] Sometimes the challenge of getting feedback can be due to the size of the project (community) rather than the channel: *"It's difficult to share small new projects that aren't very far along and get feedback."* [P751]

Collaborative coding activities need improved tool support:

Our survey concerned communication tools, but P1154 spoke of how coding tools do not adequately support **collaborative coding** activities: *"Live collaborative coding tools. For example, we can currently edit a document collaboratively in Google Docs. If we can have an IDE/tool like that for coding too, that would be useful."* Some Browser-based IDEs now provide this support and are starting to see increased adoption [25], such as Nitrous.io and Cloud9. Such tools may also address the challenges of explaining code and getting real-time feedback (as mentioned above).

7.3 Barriers to Community Building and Participation

Over half of the survey respondents said they contributed to open source projects, many of which were "pet" projects (see Fig. 4). But many respondents experienced difficulties participating in or finding others to join community-based projects, as we discuss next.

Geographic, cultural, and economic factors can pose barriers to participation through social channels:

The survey exposed challenges related to geographic, economic, and demographic factors. Respondents mentioned that different **time zones** interfered with their work, as well as other issues such as poor **access to the Internet** due to economic or political reasons. **Language barriers** were also a common concern as many of the respondents were from non-English speaking countries (see Fig. 3): *"The majority of development-related communication I do is primarily written—IRC, chat,*

email, forums, microblogging, blogging, etc. Considering that the developers I work with come from a variety of nationalities and cultural backgrounds, the intent of communication is often hard to impart or judge, which can lead to misunderstanding." [P31] These challenges may be reduced in the future as modern tools such as Slack incorporate the notion of time zones, while Skype now offers multilingual support.

Despite using social channels, **finding developers** to participate can be difficult:

A few developers also discussed challenges concerning **finding developers** to collaborate with or **convincing others to participate**. As P1285 mentioned, *"I used to think that publishing application code with an open source license would attract collaborators with an interest in using/improving that application, but now I feel like most users of code hosting sites are more interested in collaborating on tools they can incorporate into their own projects, and almost no one is interested in working on application code. I guess the challenge here is convincing developers that your application is interesting even if your code is not."* But P556 discussed how social tools make it easier to reach people that could participate: *"When I was first contributing to an open source video game project on Google Code, it was hard to get in touch with one or more of the core developers of that project because there was no right place/tool to do that. The same project moved to GitHub recently, and now I feel more comfortable because I can simply make a pull request to the project."*

7.4 Social and Human Communication Challenges

Many respondents specifically mentioned struggling with communication or people issues. We discuss a selection of the most prominent themes below.

Miscommunication on text-based channels is common:

Respondents frequently mentioned experiencing issues due to **miscommunication** on **text-based** channels, which P385 discussed: *"With the increasing amount of communication being done with social tools and IMs, chat, the amount of misunderstanding and bad, incomplete briefs grows [at] the same rate."* P126 described how much more effort text-based communication requires: *"When you write, context and expression is lost. There have been so many times when something I wrote did not come across to the other person as intended. This causes problems all the time. You must be careful and spend time on the words and phrases you use when communicating in text. If you can't pick up the phone, then use IM, and as a last resort, email."* P1331 agreed with this and mentioned how slang can introduce even more ambiguity.

Text-based miscommunication also relates to the **language barriers** mentioned above. As P31 lamented, *"The lone shooter that misunderstands you and begins shooting at you. [I] have been called an arrogant dickhead once and have also been informed that I was dictating somebody something when I actually tried to SUGGEST something. So, language is also a challenge because I am a Dane."*

For many developers, **face-to-face** communication is best:

Many respondents shared the opinion that *"nothing replaces face-to-face communication"* when trying to avoid **miscommunications**. As P136 explained, *"Without face-to-face communication, misunderstandings happen more often."* Other developers discussed how **face-to-face** communication is needed for activities such as code review and for explaining the big picture underlying a design. As P319 mentioned, *"When working apart people aren't always at their computer or responding to messages, and in the case of code review they have to summarize their thoughts into a few paragraphs. In person it's much easier to convey a thought, have a discussion and come to a resolution."* P1241 added, *"Clarity of intent (it can be hard to get a point across through text-based media)—it can be difficult to see the big picture, or even the pieces, without talking face to face."*

P176 mentioned that to address such issues, some companies move their staff on-site: *"Often collaboration tools are still not as good as face-to-face communication. Many software companies have moved to having all of their staff on-site full-time, because the communication is just better. Especially when two or more developers are collaborating on the same code in real time (pair programming) or 'whiteboarding' on a design. . .being there in person is just different."*

Others shared with us that there are tools that can support face-to-face-like interactions: *"Often times it is difficult to get ideas across in written communication. This is where tools such as Google Hangout and Skype can be beneficial, but they are not always an option."* [P206] On the other hand, not everyone wants face-to-face interactions. P737 mentioned their main challenge was *"other developers that insist on using face-to-face communication exclusively."*

People are challenging, no matter what channels are used:

Poor attitude and a lack of willingness to collaborate are issues no matter what tools are used: *"[Tools] still don't solve the difficult people problems."* [P264] As P532 explained, *"Tools facilitate good processes and interactions between individuals who are willing to collaborate and cooperate. They don't make people willing to cooperate in the first place; in these situations they actually get in the way of identifying the root problems and dealing with them. People can hide behind GitHub better than they can in person."*

P421 mentioned that the benefits of using these tools can be achieved by ignoring some of these people issues: *"Personal restraint [with] people who are mean (or, dicks). Otherwise, amazing area for learning and sharing."* P468 explained that although tools bring opportunities for transparency and collaboration, getting others to buy in can be tricky: *"The biggest challenge is getting other devs to be open both with their work and to new ideas."*

The social transparency [26], [27] of the channels introduced other issues [28]. Developers told us how they felt **intimidated**, either because they were worried that their own contributions or skills were not good enough, or that others may not react well to their contributions. As P302 told us, *"The biggest thing I fear in my work is that I'll say something that is not 100 percent technically accurate or could be misinterpreted. Other developers are utterly merciless, and I have thin skin. Whenever I post something on HN or Stack Overflow, I find that I feel anxious that someone will tear me a new one over some oversight in my analysis."* P643 discussed how feeling intimidated can lead to repressed interactions within the

community: *"Lots of people communicate less than they otherwise would for fear of looking stupid to peers who are assessing them in a colossal global meritocracy. Very unhealthy. I suspect it contributes to the high prevalence of anxiety and depression in IT (in combination with often sedentary lifestyles)."*

7.5 Communication Channel Affordances, Literacy, and Friction

Here we consider challenges that relate closely to the properties of the communication channels used, such as the affordances of the channels, literacy needs, the impact of the ways developers use (or misuse) the channels, and the friction the channels may pose to development activities.

Developers need to consider **channel affordances:**

Different channels provide different kinds of affordances, as Daft and Lengel's Media Richness Theory describes [29]. But developers don't always consciously think about these affordances when choosing which channels to use.

For example, communicating using face-to-face and verbal channels is normally **ephemeral as opposed to archival** and may be more suited to a smaller **group size**, but on the other hand, *"voice communication is much quicker, but it is not easily transcribed and it is difficult to use with more than 3 (if not 2) people."* [P1224]

Developers also frequently referred to a tension between **synchronous versus asynchronous** communication channels. P927 elaborated why asynchronous is sometimes preferred: *"I try to use asynchronous communication so [I] can decide the time when to communicate and [I won't] be disturbed when [I'm] programming or learning! On the other side I have the [ability] to talk face to face to the other developers in the team, which is the most effective way to learn, coordinate d. But it is not asynchronous."* However, as mentioned, the use of synchronous channels can lead to **interruptions** and **misunderstandings**.

Developers experienced tension when having to choose between **private versus public** communication channels: *"Many developers host their code projects on blogs, etc. Sometimes the only way to communicate with the author is to post a comment, since they do not disclose their email address. Also, on forums, of say, a particular library—e.g., openCV forums. The forum posters seem so professional and experienced, I tend to avoid posting on the forum to avoid embarrassment (lol)."* [P1036] Some respondents reacted negatively to the transparent nature of social networking tools over traditional communication media such as email: *"Very few new developers are learning how to use old tools such as email and are tending toward using social networks and other tools. It can be frustrating when you want to collaborate and they insist on proprietary software or technology, or insist on using privacy-invasive tools such as Skype or Facebook."* [P897]

Some also experienced confusion using channels where communication can be private or public: *"I am sometimes frustrated that there are too many places to do the same thing, especially when I answer a question privately and want to instead make that answer public."* [P137] In contrast, P1399 shunned the use of private channels: *"People complaining about their privacy tend to slow down development."*

Another affordance touched on by P40 relates to the **fanfare** of the communication channels used: *"Sending urgent or major information is hard, too: how to highlight information sent to others when they are free to ignore it?"*

Developers shared with us that **no single tool fits all** developers' needs nor suits all stakeholders. Different stakeholders have very different needs, therefore, different channel affordances will not suit everyone: *"Different groups in the entire team often prefer to use different tools. For example, the coders will use GitHub, but a project manager and the testing team will use Basecamp. This makes overall coordination extremely difficult."* [P160]

Finding the right channel to support **communication with users** is also a challenge. P1528 described how difficult it can be to keep track of communication from users (and other developers): *"Users/developers tend to report bugs or ask for new features with email and forum posts which can be tricky to keep track of."* To address this, some projects promote user reporting of issues on GitHub, but P394 explained the potential disadvantage with this approach: *"... it's also a lot of work to separate real, confirmed issues that we create from the tons of not-always-useful stuff that our users create. I think the nail in the coffin was when a user closed one of our bugs."* Moreover, P700 did not appreciate any communication with users and disliked the way social channels create opportunities for users to contact them: *"Users of my open source software often feel entitled to free technical support. Because it's so easy to reach me, they can be a nuisance sometimes."*

Developers need to be **literate** with communication channels:

Inadequate communication channel **literacy** and a poor understanding of channel affordances can lead to challenges for developers wishing to collaborate, exchange information, or network with others across their communities. As P1005 said, *"The main challenge for me is the interaction with people who are not literate enough to use the tools I consider standard."* P1600 discussed how Git (the underlying version control system for GitHub) can be challenging to use: *"Also, Git is a critical collaboration tool, but it is not well understood by many of the programmers I interact with."* Developers recognize that learning these tools is a challenge. P1385 mentioned how using GitHub itself, or even IRC, can seem difficult: *"I still don't understand how to do simple things in IRC and often don't bother because of the perceived effort involved—much easier to post on Stack Overflow. With tools like GitHub, there are similar issues (like how to submit patches) although documentation is improving."*

Respondents discussed the challenges around having to **learn** new tools. P404 felt that *"the biggest challenge [about using] social tools during development is when a new one is adopted into the mix; the learning curve associated with a new tool eats time unless the program is intuitive and pointed."* P1315 emphasized how these channels require different communication skills than when communicating face to face: *"Basically a very different way to communicate compared to face-to-face comm. It simply has to be learned."* Furthermore, each new channel may necessitate the acquisition of a different vocabulary. P526 felt that *"if you don't know the right [vocabulary] or technology you want to use, the [dialog] usually ends quickly."* P981 mentioned that some of these tools are difficult to learn because of a **lack of documentation**.

Literacy, however, is not just about knowing how to use a particular channel, but when to use it and when not to use it (depending on the need, e.g., to avoid **interruptions**).

Communication channel **friction** can obstruct participatory development activities:

Technical issues introduced **friction** for developers, such as the **lack of mobile support**, tools crashing, poor support for search, annoying **notifications**, and hardware limitations. P919 claimed that they *"never have enough screens."* This concern about monitors is not surprising given the number of channels and tools developers use. Other tools, despite being highly popular among developers, suffer usability issues: *"Hacker News dominates and is terrible."* [P1279]

Vendor lock-in was also an issue that was mentioned by developers, and not surprisingly since many of the social tools developers use are proprietary. P334 discussed this issue at length: *"Much of the value that we [are] responsible for is now kept safe and maintained by a third party. It's risky... If eclipse.org, stackoverflow.com or GitHub goes away, our team(s) would suffer severe damages. I would love to be able to have my own weekly backup of the social interactions that take place in a format that is machine processable such that in the event of total failure I could migrate our history of communication to another host or another technology."* Projects also face issues when a proprietary tool's privacy policies change: *"Having to shift platforms when a company's policies change (e.g., Facebook's altering of privacy settings, SourceForge's adware, Google's pushing Plus)."* [P681]

Developers also mentioned **usability issues** getting in their way—that collaboration tools and social tools can introduce friction even though they bring benefits: *"Code review and collaboration tools (Asana, Trello) I mostly find to be necessary to some extent but generally very annoying. Seems like one more thing. Maybe they're necessary evils, but they seem to get in the way a lot."* [P1255]

Many challenges arise due to **poor channel integration**, such as having to deal with identity management. P1430 explained the many issues that a lack of integration brings: *"Poor integration between them and an overabundance of options. There are a lot of tools out there, but it's hard to put them together into a cohesive workflow. Especially when participating in a lot of open source projects, every one has a different set of overlapping but different tools. Another problem is identity management. With personal projects and work projects, I'd like to be able to manage them separately but without the inconvenience of maintaining separate accounts."* Poor integration makes it difficult to monitor multiple channels, but developers also complained about **channel overload**—there are simply too many channels to choose from and follow.

Poor or scattered tool **adoption** can also introduce friction, especially when there are many tools available: *"There are too many variants of things like project management tools, time-trackers, issue-trackers. . .it's hard to get a team to agree on tools, and none of these stand out as an obvious leader."* [P694] And getting agreement on communication tools is also difficult. P422 experienced difficulties *"convincing other developers on a project to all use the same communication tool for [the] project."* The problems are exacerbated by globally distributed teams: *"I live internationally and work on globally disparate project teams, so G11N [globalization] is a big deal. Finding a consistently-used platform for such a large and varied group of developers is also a challenge, as some only do IRC, others only Google Groups, others only email."* [P380]

7.6 Content and Knowledge Management

Developers face challenges with information fragmentation and with the *"quality/quantity of information available."* [P1242]

Use of many channels leads to **information fragmentation:**

Information fragmentation results from the use of too many channels: *"One of the things that bugs me most is multiple mediums. At any given moment I can get a chat, an email, a text, or whatever—wish it was more streamlined."* [P1401] Fragmentation also occurs because of the inconsistent or poor **adoption** of particular channels (as discussed above). P1250 suggested that one could address this by **encouraging others to use the same tools** or by using tools that integrate communicated information from multiple channels: *"Making sure everyone else you're working with also uses the tools. One of the biggest issues with fragmentation of the communication options is that there are so many different ways to communicate that it's harder to find it all in one place. Important communications get lost; Key people don't see them; They can't be retrieved by a single search tool. Companies such as Slack are attempting to solve this problem, but it has a long way to go."*

The **quantity** of communicated information is overwhelming:

Developers found it challenging to find the "signal in the **noise**"—the "explosion" of available channels has led to an increase in volume and duplicate information posted in multiple locations. This is particularly difficult for developers working on multiple projects where different tools are used: *"The variety of tools, and the need to switch context and tool set between various sub-projects, adds a lot of cognitive overhead."* [P815] There is also a fear of missing important information: *"Too many channels means that needed or interesting information disappears, and going through all of the channels you mentioned is impossible in limited time."* [P917] Although **poor channel integration** that leads to **information fragmentation** is one issue, the channels themselves further promote an increase in the quantity of communication posts to attend to: *"I feel that social tools largely present information in fragments, with many different approaches and styles and agendas, which makes it time consuming to stitch together a working knowledge of technologies I'm learning."* [P1189]

The diversity and velocity of information makes it hard for developers to **keep up with new technologies**. As P1409 put it, *"There definitely is information overload. People think I'm joking when I mention the 'javascript framework of the day'."* Developers try to stay up to date on these new technologies [12], but the availability of so many different news sites and aggregators means they are inundated with content. P1144 talked about how these tools affect productivity: *"The news overload via Aggregators (Hacker News, Reddit, Digg, Slashdot, . . .) affects productivity. I don't use too much social networking (Facebook or Twitter) as they are a huge time-sink and sheer noise as far as technical development work is concerned."*

The **quality** of communicated information is hard to evaluate:

In addition to receiving too much information, many developers described their concerns with the **quality** of information: *"Judging the reliability and credibility of sources can be a challenge as information changes quickly and isn't always correct."* [P846] There was a particular concern with the quality of content on social sites: *"I sometimes feel the lack of quality content on social networks, q&a sites—especially when it comes to incompetent answers to questions I ask. So the challenge is to filter the information you get from all of the sources."* [P95] Developers were also concerned by discoveries of contradicting or inconsistent information.

There were also complaints that some information may be **obsolete**: *"Technologies are moving so fast, and most of the content on the Internet could be outdated quickly. It's sometimes hard to filter that outdated information."* [P492] Sometimes the channels developers use are subject to **spam**. As P750 told us, *"Recruiters keep spamming me through GitHub or Stack Exchange looking for Web developers."* Developers also described how it can be difficult to find content on **niche** technologies and that it was hard to acquire and understand the **history** of the information created.

8 Discussion

Through our survey, we investigated how a complex ecosystem of communication channels shapes and challenges a participatory development culture.

We first discovered **characteristics of the programmers** that participate in and contribute to projects hosted on the GitHub social coding site. We now examine how the reported respondent characteristics may indicate a lack of participant diversity, as well as the implications that arise from this.

Our survey asked respondents about their **participatory development activities**. Previous research has focused on development activities, but we discuss why it is important to also consider non-development activities (such as networking and learning) in terms of understanding future tool needs.

Next, we look at the complex **ecology of communication tools** that our survey revealed. We compare our findings about the benefits and challenges of using these tools to existing research about communication tool use in global software and open source development. We also share some **recommendations** that emerged directly from the survey responses to address challenges developers experience using a complex communication and social tool ecology.

Finally, we discuss some of the **limitations** of our study and propose **future research** directions.

8.1 Characteristics of the Programmers Surveyed

Our survey was answered by developers that contribute to publicly hosted software projects on GitHub. When we designed our survey, we expected that this population may be skewed towards younger, male, North American developers, and that we might see differences across the demographics in the number and types of tools used and their perceived benefits and challenges. Our expectations were somewhat met—the respondents were skewed in the manner we expected—but our analysis did not reveal differences in the tools used or how they were used across the varied developer demographics.

We were surprised that only 3.9 percent of our respondents said they were female. We expected this number to be higher as a recent survey of more than 2,000 FLOSS contributors indicated that 10 percent were female [30]. Our statistic is, however, more in line with earlier surveys that indicated females accounted for 1–5 percent of open source participants [31]. Combined, these results may indicate an ongoing or increasing lack of gender diversity in the FLOSS community. This lack of gender diversity may go beyond FLOSS as many of our survey participants were professional developers. A lack of diversity has recently been shown to negatively impact productivity in FLOSS projects [32], but we are also concerned that skill development and networking benefits gained from participating in FLOSS or publicly hosted projects may be harder to achieve for certain groups of developers. Further research is sorely needed to investigate if and how communication and social channels can be improved to reduce the diversity gap in software development.

Another interesting but somewhat expected characteristic about our respondents relates to their age: 77.9 percent said they were 32 or younger, so-called millennials. Other surveys with FLOSS developers also show similar distributions [30], [32]. We expected that older developers may not use the same set of tools or as many tools as younger developers, but we did not see much of a difference, likely because our survey was biased towards developers that embrace social tools. We hope to repeat the survey with a different population of developers. Our survey also showed that older programmers are more likely to work on professional projects and less likely to work on open source projects. Meanwhile, we found that when a person is a professional programmer, it is less likely they will contribute to open source. One hypothesis is that some software organizations may discourage open source participation among their employees, either directly or indirectly. Future work is needed to investigate this hypothesis.

8.2 Beyond Coding: Understanding Tool Needs for Participatory Development

Previous research into tool needs has tended to focus on development activities rather than the broader set of activities that are the hallmark of a participatory development culture [6]. In our survey, we considered how communication and social tools are used to support a number of different **activities** that developers care about, such as learning and sharing with others, networking, and keeping up to date with new technologies and project activities. The participatory activities we inquired about were inspired by Jenkins' definition of a participatory culture [1], but our previous literature review [6] revealed that this list of activities is also important to developers. However, it is possible that this set of activities is not complete and that further research is needed to understand the full set of participatory development activities that need to be supported through communication and social tools.

We feel it is important to understand the broader activities that developers care about so that our future tools and guidance on work practices can support these needs.

Although developers care about code quality and velocity, they also recognize that they need to continuously learn and network to create opportunities. Ultimately, this should help improve the work they do on current as well as future projects.

8.3 Towards Understanding the Ecology of Communication Channels Developers Use in a Participatory Culture

Our survey helped paint a picture of the complex and broad ecology of tools that developers use. We were also able to determine which tools were deemed to be the most **important** for participatory development activities: code hosting sites, face-to-face interactions, Q&A sites, and search engines (Fig. 7).

Given we requested participants via GitHub, it is not surprising that the majority of respondents said **code hosting sites** were the most important (73 percent chose it). Code hosting sites serve an important role in providing version control, issue tracking, and several means of communication between developers.

Face-to-face and **Question & Answer (Q&A) sites** were practically tied in second place. The importance of **Face-to-face** implies that, even in a rich environment of electronic communication channels, developers still have a strong preference towards communicating in person. This finding resonates with previous works on the impact of distance in collaborative settings (e.g., [33], [34]). Although Face-to-face is recognized as the "richest" channel for communication [35], we were still surprised that so many developers said it was important. A high percentage of respondents said they worked on distributed open source projects where co-location is likely impossible—we suspect that many developers thought of Skype or Google Hangouts as a Face-to-face channel.

We know from other research [36] that **Q&A sites** play a prominent role in developers' activities, but we do not know if developers use them as a communication channel with other developers (bidirectional communication) or mostly as a resource where they can get quick answers to questions that they have (unidirectional communication). We also suspect that the goal of **Search** (the fourth most important channel) may be related to the goal of **Q&A sites**, as answering technical questions is one of the most important developer information needs [37], [38] and resources such as Stack Overflow are designed to be reached through Search tools.

Giuffrida and Dittrich [20] reported on the usage of social software in software engineering projects and in distributed teams through a systematic mapping study. They discussed the use of instant messaging for reducing communication barriers between remote collaborators. In a related work, Dittrich and Giuffrida [14] explored the role of instant messaging in a global software development project and found that IM not only supports communication among distributed team members, but also provides a means to build trust and social relationships among co-workers. In our survey, **Private (e.g., Skype chat) and Public (e.g., IRC) chats** were deemed as the most important channels by nearly 15 percent of our survey respondents (6th and 9th positions, respectively; see Fig. 7), which indicates the importance of chat tools for supporting development activities, especially regarding informal communication.

Giuffrida and Dittrich [20] also mapped studies on the use of blogs and microblogs. In our survey, **Microblogs** were deemed as the most important channel by over 200 respondents (nearly 20 percent chose it), thus earning the 5th position, while **Blogs** ranked 7th. Our qualitative analysis showed that both of them help increase awareness of the most up-to-date developments in developer communities. It is important to note that the vast majority of papers found in Giuffrida and Dittrich's systematic mapping refer to the use of communication channels in enterprise settings. As the demographics of our survey indicate, our survey population is more mixed, including professional, open source, and hobbyist developers. Further research is needed to explore the interplay between private and public software development when it comes to communication channel usage.

It is also important to note that most studies that have explored how social tools are used in software development studied just one or two communication or social tools [6], [20]. One exception is a short survey conducted by Black et al. in 2010 where they found that several social media tools were used to support development work [18]. Another key exception is the work by Turner et al. [17] where they studied the "workplace communication ecology" in a small company of about 50 participants using surveys in 2008 and 2009, followed by interviews with 23 participants. They reported on clusters of tools used, as well as the strengths and weaknesses from the various channels. Since the employees were co-located, not surprisingly, Face-to-face was the most preferred communication channel.

The ecology of tools that developers use is interesting to study because we see developers using multiple tools for the same activity, as well as using more tools over time. Turner et al. [17] also found an increase in the number of tools used as far back as 2009. We stress that this increasing reliance on a complex constellation of tools brings several challenges, as discussed in the findings above and as Giuffrida et al. presented in their mapping study [20].

Next, we discuss preliminary recommendations for practitioners wanting to address some of these challenges, but we also call on the research community to study these issues further and to suggest new processes or tools to address the increasing participatory needs of software developers.

8.4 Recommendations for Practitioners Choosing Tools

In our survey, we probed through an open-ended question about the **challenges** developers face using an ecosystem of communication channels. These challenges point to a number of recommendations that may be helpful to other developers. In the following, we formulate some recommendations for developers that rely on a number of communication channels while engaging with a participatory culture of software development. Our recommendations are partly based on our literature review as well as on strategies the survey respondents reported using to address the various challenges they

experienced. We coded and categorized these shared strategies, as shown in Fig. 9.

We stress, however, that future research is needed to validate these recommendations and that this set of recommendations is likely not complete—discovering them was not one of our research goals, and thus our survey did not explicitly attempt to elicit such recommendations.

Recommendation 1: Be aware of **channel affordances** and choose tools accordingly.

There is a vast array of communication channels that today's developers (and other knowledge workers) can use (see Fig. 1). Particular channels offer different affordances, as described by the *"Media Richness"* theory [33] and as elaborated in Section 7.5 of this paper. For example, some channels offer more immediacy for communication (e.g., face-to-face) while asynchronous channels such as email offer a chance for deeper reflection before having to respond. Other channels, although immediate, may introduce distractions.

In previous work, Treude et al. [39] investigated the properties of different documentation channels in software projects and found that different channels had different benefits and drawbacks. For example, they found that blogs were seen to generate more fanfare than wikis and were more suitable for posting important announcements that should not be missed, whereas wikis were easier to change. Similarly, Calefato et al. [40] discuss the appropriateness of different communication media to support distributed requirements engineering.

Developers may not always be consciously aware of channel affordances and the trade-offs between them. However, they need to learn to recognize the strengths and weaknesses of different channels and to recognize tensions between **private versus public** channels, **synchronous versus asynchronous** communication, **ephemeral versus archival** channel properties, **anonymous versus identified** participation, and support for different communication types, such as **textual versus verbal** versus **face-to-face** conversations.

Recommendation 2: Define a **communication covenant** with project members.

To enhance distributed work, Olson and Olson [33] suggest that teams create a *"communication covenant"* to prescribe which channels should be used for different kinds of communication within a team. Similarly, Giuffrida and Dittrich [19] conceptualize the role of communication channels in helping to establish persistent coordination mechanisms among team members. Indeed, many successful open source projects recommend which channels to use, such as how the Angular project specifies which channels should be used for different activities.[9] P253 felt that collaborators should not just agree on tools, but also agree on how they must be used: *"The tool matters less than how people use it. Biggest problem is people not using tools the way it was agreed upon."* Although some project teams do figure this out without a formal covenant: *"Small autonomous projects/teams who have a fairly mutual understanding of what communication/collaboration tools they want to use to achieve their needs/goals tend to experience little communication friction, I find."* [P783]

Giuffrida et al. [20] also suggest that groups need to pay attention to how tools are socially negotiated. Social protocols and tools not only need to be decided upon initially, but also adopted and adapted by people over time, thus being socially shared, modified, and appropriated [19]. Furthermore, other researchers have noted that there is a need to establish practices on how to use social software in development contexts [20].

Rrecommendation 3: **Think lean** when adopting new tools.

A common challenge reported by our respondents was **channel overload**, as discussed above. Although there are many possible channels for developers to choose from, using too many will lead to people feeling overwhelmed from having to manage so many different tools and will also increase the chance that information is fragmented across channels. P232 suggested a way to address this: *"The use of numerous tools may be overwhelming. It is usually assumed that it is better to use fewer tools, and increase the direct communication frequency between developers using face-to-face or chat."*

Recommendation 4: Stay abreast of the **latest tools** that may improve development productivity and team work.

This recommendation may seem to contradict the last recommendation to use fewer tools, but many of the more recent tools (such as Slack) aggregate communication from different tools through one channel. We learned from our respondents that **no one tool fits all** needs: *"There are too many sources of communication to monitor, I have been trying to use tools like HipChat and Flowdock to get a more unified communication channel."* [P980] Likewise, new tools may emerge that address other challenges and they should be considered for adoption.

Recommendation 5: Take the time to **learn** how to use the channels most effectively.

As discussed above, tool **literacy** is considered to be very important. P488 described how poor literacy with team members can lead to frustrations: *"Lesser-skilled developers (typically designer-developers) sometimes struggle to use the common tools like Git/GitHub, screen sharing, text-based communication and to configure their own development environment. This [means those] collaborating remotely get bogged down in stupid troubleshooting sessions."* P729 described how important it is to develop skills that make the most of particular channels and avoid challenges such as **noise**: *"Developing filter skills to pick out the important things from the noise."*

Recommendation 6: Know when to **unplug.**

Some respondents described how they unplug from the Internet or from specific communication channels to allow them to focus and to avoid **interruptions** and **distractions**. P1336 shared how they consciously decide **when to use certain communication channels**: *"I turn off most communication tools at the right times (i.e., when I'm not in need of feedback or help). I'll still use GitHub for*

9. https://github.com/angular/angular.js/blob/master/CONTRIBUTING.md

finding resources and a private messaging tool, HipChat or e-mail for quick questions." Similarly, P534 described using a command line tool to avoid distractions in the Browser: *"If you have to go to a Web browser there is a 10 percent chance you'll be distracted. I use the project 'howdoi' to get answers from Stack Overflow on the command line so I can stay out of the browser and keep focus."*

P825 discussed how it is important to be mindful about how one feels: *"Biggest challenge is meta–e.g., *noticing* when I'm feeling overwhelmed or distracted and adjusting to adapt (e.g., closing IRC, taking a twitter hiatus, etc.)"*. P646 recommended that *"one needs to exercise self control when using these tools, otherwise it's easy to end up spending more time on them than needed."* P692 went one step further, suggesting that *"sometimes it helps to have a day of development where you unplug [the] Internet."*

8.5 Limitations

Studying the participatory culture of software development is challenging because it involves understanding the *tools* and *communication channels* developers use, the *content* generated through these channels, the *developers* themselves and their perspectives, the development *activities* and *actions* supported by the tools, and the *interplay* between all of these aspects. Through our survey, we aimed to focus our investigation on the communication channels developers use to support their participatory development activities, as well as developers' perspectives concerning the use of these communication tools.

We opted to use a *survey* instrument to reach a broad population of developers. The survey was developed through several phases of design and pilot studies as we needed to compromise between developer time and the amount of information we gathered. The survey inclusion criteria suggest participant bias towards social code hosting systems, however, this population was the focus of our study. Therefore, we do not claim generalizability of our results to all developers. Our survey and the underlying source code are available online.

At least 60 percent of our respondents work in Web development. Our results resonate with an increase in the popularity of Web technologies and programming languages such as JavaScript and CSS.[10] However, our findings may also be biased towards development practices that are most commonly found in Web development projects.

Despite their length, the 2013/2014 surveys had 21 percent/16 percent response rates, respectively. Many developers told us they were happy to contribute to this research as they were also curious about the channels other developers use and the challenges others experience. Since the survey was long, the respondents may have suffered from fatigue and may have selected fewer channels for activities further into the survey, and the earlier responses to questions may have influenced later responses. We considered randomizing the order of the questions, but we opted for a more logical order as we felt developers would find it easier to answer questions in this manner.

Another limitation (that we recognized as we conducted our study) is that the tools developers use are changing faster than we can study! For example, Slack was not widely adopted at the start of our survey, but is now used by a great many developers.

Although our findings from the demographic and channel usage questions are insightful, the data we received about the challenges developers face was the most interesting as we learned how tools may benefit but also negatively impact developer work practices. To offset bias during coding, we recruited an additional independent coder to analyze our data. When the coders did not agree, we did not apply the code. We stress that the counts we report may not be an accurate indicator of the importance of the additional challenges as this was an optional open-ended question posed at the end of a long survey. In the presentation of the challenges, we rely heavily on the developers' own words to bring credibility to our findings. In Fig. 9, we provide counts for each code and further expand this table in the companion Website[11] with additional quotes. Future work is needed, however, to determine the importance of these challenges, as well as reveal additional strategies to address these challenges and to validate the recommendations we provided above.

Finally, we emphasize that our study is just one step in a larger effort towards forming a theory of knowledge work in a participatory culture of development, as we discuss next.

8.6 Future Work

Our study paves the way for the development of theories concerning how developers use tools and suggests ways that tools may be improved. The software engineering community has witnessed a major paradigm shift in recent years in how developers communicate and participate in each other's projects and in each other's learning. There is a great need to study emergent software practices as well as the tool constellations that modern developers use.

To date, we have collected data from two large-scale surveys (in 2013 and 2014), but we intend to deploy the same in future years (with some changes to account for newer communication channels) as we wish to understand how the use of communication channels by developers may evolve over time. But surveys are limited in the kinds of data and insights they can provide.

As a continuation of this study, we anticipate that future interviews with developers would shed more light on the *strategies* they use to mitigate the challenges we revealed in our survey. Observations may also uncover other challenges as well as how developers are using new tools (such as Screen Hero,[12] which supports collaborative screen sharing) and how they use homegrown tools.

Additionally, these studies should be extended to commercial projects—i.e., non-open source projects—with different constraints and restrictions (e.g., being forced to use specific tools). The themes that emerged in this study will help form a new version of the survey.[13]

10. https://github.com/blog/2047-language-trends-on-github

11. http://thechiselgroup.github.io/channel-study/

12. https://screenhero.com/

13. http://thechiselgroup.org/2013/11/19/how-do-you-develop-software/

Our future work will allow us to continue developing a descriptive theory on how different channel affordances may shape participatory development activities. Our hope is that this theory can be used to help developers and tool designers anticipate the benefits and challenges certain combinations of tools may bring, as well as reveal new opportunities to improve tools and work practices.

Currently, the vast majority of the tools developers adopt and rely on are developed by industry. We suggest that researchers may be able to play a bigger role in tool design by understanding the implications of the tools that are used and then revealing ways they may be improved. For example, tools that offer aggregation mechanisms may address the information fragmentation issues, or there may be ways that tools can be improved to address cultural barriers.

9 Conclusions

While this study is part of ongoing research, it presents several key contributions: the **survey instrument**; **demographics** of the social programmer; which **channels developers use** to support their participatory development activities, and which ones are most **important** to them; and the **challenges** developers face using a broad spectrum of tools while engaging in participatory development work practices. We also provide a number of preliminary **recommendations** that developers may follow to address the challenges we presented.

Finally, communication channels shape and challenge the participatory culture in software development. However, the reverse is also true: not much is understood about the impact of the participatory culture on software development practices and the communication channels developers use. We believe this research also has implications on other knowledge workers—software developers are referred to as the *knowledge worker prototype* as they are not only the first to use and shape tools and channels, but also have far lower barriers to build and tweak them [41]. It won't be surprising if the challenges and opportunities that emerged from our study propagate to other domains as well.

Acknowledgments

The authors would like to thank the respondents for taking so much time to answer the survey, as well as Cassandra Petrachenko for her assistance with coding the data and editing this paper.

References

[1] H. Jenkins, *Confronting the Challenges of Participatory Culture: Media Education for the 21st Century*. Cambridge, MA, USA: Mit Press, 2009.

[2] E. C. Wenger and W. M. Snyder, "Communities of practice: The organizational frontier," *Harvard Business Rev.*, vol. 78, no. 1, pp. 139–146, 2000.

[3] L. Singer, F. Figueira Filho, B. Cleary, C. Treude, M.-A. Storey, and K. Schneider, "Mutual assessment in the social programmer ecosystem: An empirical investigation of developer profile aggregators," in *Proc. Conf. Comput. Support. Coop. Work*, 2013, pp. 103–116.

[4] M.Chui, et al., *The social economy: Unlocking value and productivity through social technologies*, 2012. [Online]. Available: http://www.mckinsey.com/insights/high_tech_telecoms_internet/the_social_economy

[5] F. Lanubile, *Social software as key enabler of collaborative development environments*. 2013. [Online]. Available: http://www.slideshare.net/lanubile/lanubilesse2013-25350287

[6] M.-A. Storey, L. Singer, B. Cleary, F. Figueira Filho, and A. Zagalsky, "The (r)evolution of social media in software engineering," in *Proc. 36th Int. Conf. Softw. Eng. Future Softw. Eng.*, 2014, pp. 100–116. [Online]. Available: http://doi.acm.org/10.1145/2593882.2593887

[7] K. Crowston, K. Wei, Q. Li, and J. Howison, "Core and periphery in free/libre and open source software team communications," in *Proc. 39th Annu. Hawaii Int. Conf. Syst. Sci.*, vol. 6, Jan. 2006, pp. 118a–118a.

[8] R. Pham, L. Singer, O. Liskin, F. Figueira Filho, and K. Schneider, "Creating a shared understanding of testing culture on a social coding site," in *Proc. 35th Int. Conf. Softw. Eng.*, 2013, pp. 112–121.

[9] J. Fried and D. H. Hansson, *Remote: Office Not Required*. London, U.K.: Ebury Digital, 2013.

[10] P. Naur, "Programming as theory building," *Microprocess. Microprogr.*, vol. 15, no. 5, pp. 253–261, 1985.

[11] A. Guzzi, A. Bacchelli, M. Lanza, M. Pinzger, and A. V. Deursen, "Communication in open source software development mailing lists," in *Proc. 10th Working Conf. Min. Softw. Repositories*, 2013, pp. 277–286. [Online]. Available: http://dl.acm.org/citation.cfm?id=2487085.2487139

[12] L. Singer, F. Figueira Filho, and M.-A. Storey, "Software engineering at the speed of light: How developers stay current using Twitter," in *Proc. 36th Int. Conf. Softw. Eng.*, 2014, pp. 211–221. [Online]. Available: http://doi.acm.org/10.1145/2568225.2568305

[13] E. Shihab, Z. M. Jiang, and A. Hassan, "On the use of internet relay chat (IRC) meetings by developers of the GNOME GTK+ project," in *Proc. IEEE 6th Int. Working Conf. Min. Softw. Repositories*, May2009, pp. 107–110.

[14] Y. Dittrich and R. Giuffrida, "Exploring the role of instant messaging in a global software development project," in *Proc. IEEE 6th Int. Conf. Global Softw. Eng.*, Aug. 2011, pp. 103–112.

[15] W. Scacchi, "Understanding the requirements for developing open source software systems," in *Proc. IEE Softw.*, vol. 149, no. 1, pp. 24–39, Feb. 2002.

[16] M. McLuhan and Q. Fiore, *The Medium Is the Message*. Penguin Books: New York, NY USA 1967.

[17] T. Turner, P. Qvarfordt, J. T. Biehl, G. Golovchinsky, and M. Back, "Exploring the workplace communication ecology," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2010, pp. 841–850.

[18] S. Black, R. Harrison, and M. Baldwin, "A survey of social media use in software systems development," in *Proc. 1st Workshop Web 2.0 Softw. Eng.*, 2010, pp. 1–5.

[19] R. Giuffrida and Y. Dittrich, "A conceptual framework to study the role of communication through social software for coordination in globally-distributed software teams," *Inf. Softw. Technol.*, vol. 63, pp. 11–30, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S095058491500049X

[20] R. Giuffrida and Y. Dittrich, "Empirical studies on the use of social software in global software development a systematic mapping study," *Inf. Softw. Technol.*, vol. 55, no. 7, pp. 1143–1164, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584913000153

[21] C. Treude, F. Figueira Filho, B. Cleary, and M.-A. Storey, "Programming in a socially networked world: The evolution of the social programmer," in *Proc. Workshop Future Collaborat. Softw. Develop.*, 2012, pp. 1–3.

[22] N. Postman, "The humanism of media ecology," in *Proc. Media Ecology Assoc.*, vol. 1, 2000, pp. 10–16.

[23] C. Parnin and S. Rugaber, " Resumption strategies for interrupted programming tasks," *Softw. Quality J.*, vol. 19, no. 1, pp. 5–34, 2011. [Online]. Available: http://dx.doi.org/10.1007/s11219–010-9104-9

[24] O. Baysal, R. Holmes, and M. W. Godfrey, "No issue left behind: Reducing information overload in issue tracking," in *Proc. 22nd ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, 2014, pp. 666–677. [Online]. Available: http://doi.acm.org/10.1145/2635868.2635887

[25] M. Goldman, G. Little, and R. C. Miller, "Real-time collaborative coding in a web IDE," in *Proc. 24th Annu. ACM Symp. User Interface Softw. Technol.*, 2011, pp. 155–164.

[26] H. C. Stuart, L. Dabbish, S. Kiesler, P. Kinnaird, and R. Kang, "Social transparency in networked information exchange: A theoretical framework," in *Proc. ACM Conf. Comput. Support. Coop. Work*, 2012, pp. 451–460. [Online]. Available: http://doi.acm.org/10.1145/2145204.2145275

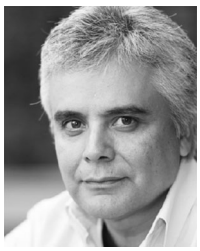
[27] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in GitHub: Transparency and collaboration in an open software repository," in *Proc. ACM Conf. Comput. Support. Coop. Work*, 2012, pp. 1277–1286. [Online]. Available: http://doi.acm.org/10.1145/2145204.2145396
[28] I. Steinmacher, T. U. Conte, M. Gerosa, and D. Redmiles, "Social barriers faced by newcomers placing their first contribution in open source software projects," in *Proc. 18th ACM Conf. Comput. Support. Coop. Work Social Comput.*, 2015, pp. 1379–1392.
[29] R. L. Daft and R. H. Lengel, "Organizational information requirements, media richness and structural design," *Manage. Sci.*, vol. 32, no. 5, pp. 554–571, 1986.
[30] G. Robles, L. Arjona Reina, A. Serebrenik, B. Vasilescu, and J. M. González-Barahona, "FLOSS 2013: A survey dataset about free software contributors: Challenges for curating, sharing, and combining," in *Proc. 11th Working Conf. Min. Softw. Repositories*, 2014, pp. 396–399. [Online]. Available: http://doi.acm.org/10.1145/2597073.2597129
[31] P. A. David and J. S. Shapiro, "Community-based production of open-source software: What do we know about the developers who participate?" *Inf. Econ. Policy*, vol. 20, no. 4, pp. 364–398, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167624508000553
[32] B. Vasilescu, et al., "Gender and tenure diversity in GitHub teams," in *Proc. 33rd Annu. ACM Conf. Hum. Factors Comput. Syst.*, 2015, pp. 3789–3798. [Online]. Available: http://doi.acm.org/10.1145/2702123.2702549
[33] G. M. Olson and J. S. Olson, "Distance matters," *Hum.-Comput. Interact.*, vol. 15, no. 2, pp. 139–178, Sep. 2000. [Online]. Available: http://dx.doi.org/10.1207/S15327051HCI1523_4
[34] P. Bjørn, M. Esbensen, R. E. Jensen, and S. Matthiesen, "Does distance still matter? revisiting the CSCW fundamentals on distributed collaboration," *ACM Trans. Comput.-Hum. Interact.*, vol. 21, no. 5, pp. 27:1–27:26, Nov. 2014. [Online]. Available: http://doi.acm.org/10.1145/2670534
[35] A. R. Dennis and S. T. Kinney, "Testing media richness theory in the new media: The effects of cues, feedback, and task equivocality," *Inf. Syst. Res.*, vol. 9, no. 3, pp. 256–274, 1998.
[36] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, "Design lessons from the fastest Q&A site in the west," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2011, pp. 2857–2866. [Online]. Available: http://doi.acm.org/10.1145/1978942.1979366
[37] A. J. Ko, R. DeLine, and G. Venolia, "Information needs in collocated software development teams," in *Proc. 29th Int. Conf. Softw. Eng.*, 2007, pp. 344–353. [Online]. Available: http://dx.doi.org/10.1109/ICSE.2007.45
[38] A. Begel and T. Zimmermann, "Analyze this! 145 questions for data scientists in software engineering," in *Proc. 36th Int. Conf. Softw. Eng.*, 2014, pp. 12–23. [Online]. Available: http://doi.acm.org/10.1145/2568225.2568233
[39] C. Treude and M.-A. Storey, "Effective communication of software development knowledge through community portals," in *Proc. 19th ACM SIGSOFT Symp. 13th Eur. Conf. Found. Softw. Eng.*, 2011, pp. 91–101. [Online]. Available: http://doi.acm.org/10.1145/2025113.2025129
[40] F. Calefato, D. Damian, and F. Lanubile, "Computer-mediated communication to support distributed requirements elicitations and negotiations tasks," *Empir. Softw. Eng.*, vol. 17, no. 6, pp. 640–674, 2012.
[41] A. Kelly, *Changing Software Development: Learning to Become Agile*. New York, NY, USA: Wiley, 2008.

Margaret-Anne Storey is a professor of computer science and the director of software engineering program at the University of Victoria. She holds a Canada Research Chair in Human and Social Aspects of Software Engineering. Her research goal is to understand how technology can help people explore, understand, and share complex information and knowledge. She evaluates and applies 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.

Alexey Zagalsky received the bachelor's and master's degrees in computer science from Tel Aviv University, Israel. He is working toward the PhD degree under the guidance of Margaret-Anne Storey from the University of Victoria. He focuses on software engineering, studying the interplay between developers, tools, their activities, and how all of it affects collaboration and communication. He is also interested in human-computer interaction, human aspects in software engineering, and computer supported collaborative learning. He has published papers in a variety of conferences, including ICSE, CSCW, and MSR, and his current research aims to form a theory of knowledge in software engineering.

Fernando Figueira Filho is a professor of software engineering at the Federal University of Rio Grande do Norte. His research interests include software engineering and (CSCW), focusing on sociotechnical systems, and helping design software that empowers people to cooperate more effectively and creatively.

Leif Singer received the PhD in computer science from Leibniz Universität Hannover, Germany. He is a jack of all trades at Automattic Inc., where he talks to users, plans iterations, ships software, and analyzes data to make WordPress.com better. He is also an affiliate researcher with the University of Victoria. His reasearch focused on how software developers collaborate, how people use tools (among them computers) to achieve their goals, how we learn, and how all of those can be made better.



Daniel M. German is a professor at the Department of Computer Science, University of Victoria, where he does research in the areas of mining software repositories, open source software engineering, and intellectual property.