How Communication Channels Shape a Participatory Culture in Software Development

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

Leif Singer
University of Victoria
Victoria, BC, Canada
lsinger@uvic.ca

Fernando Figueira Filho
Universidade Federal do Rio Grande do Norte, Natal, Brazil
fernando@dimap.ufrn.br

Alexey Zagalsky
University of Victoria
Victoria, BC, Canada
alexeyza@uvic.ca

Daniel M. German
University of Victoria
Victoria, BC, Canada
dmg@uvic.ca

***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, giving rise to a wide range of socially-enabled communication channels and social media that developers use to support their activities. A participatory culture of software development is emerging in which developers want to engage with, learn from, and co-create software with other developers. However, the interplay of these channels, as well as the opportunities and challenges they may create when used together, are not yet well understood.**

In this paper we report on a large-scale survey conducted with 1,516 GitHub users. We describe which 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 developers.

I. Introduction

Over the past few decades, 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. Project boundaries blur not just in terms of their architecture and how they are used, but also in terms of how they are authored, as many developers nowadays contribute to multiple projects. Indeed, we see a participatory culture [1] forming within many development communities of practice [2] where developers want to engage with, learn from and co-create with other developers. Many developers care not just 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' needs to communicate and collaborate with others, many development tools nowadays are integrated with or supplemented with communication channels and social media [4] (e.g., email, chat, or micro-blogging 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 participation in development projects. We see that the collaborative and participatory nature of software development continues to evolve, shape and be shaped by tools that support communication and the social aspects of development activities within communities of practice [5]. We use the general term *communication channel* to refer to both 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). However, not much is known about the impact this participatory culture may have on software development practices nor on the velocity and quality of the developed software.

In this paper, we study how the choice of communication channels shapes the activities that developers partake in within a participatory culture of development, as well as explore the challenges they may face. We report on a large-scale survey conducted with developers that work on either collaborative or community-based development projects. We study the population of developers that use the GitHub social coding website. Through this survey, we first want to understand the demographics of the developers participating in this community and we aim to understand which channels and tools these developers use to support learning, discovery, and collaboration with others. We learn which communication channels these developers find essential to their work, and gain an understanding of the challenges they face. Using these insights, we make recommendations to developers as well as to tool designers on how to use and improve communication and social tools for developers.

This paper is a continuation of previous work [6], which included only brief results from our survey. Here, we describe the survey in detail and provide detailed analyses of the survey results leading to an initial descriptive theory of the role of communication channels in supporting software development activities within a participatory development ecosystem.

II. Background

We begin with an overview on communities of practice and communication channels in software development, illustrating the interplay between them.

A. Communities of Practice in Software Development

Communities of Practice are groups of people connected by the similarity in their activities [7]. Such communities can be found in many domains, amongst them software development.

Community members do not have to be spatially or socially connected, but solve similar problems. They learn from each another through processes like apprenticeship or mentoring. While previously the line between professional work and hobbyist development in open source seemed relatively clear-cut, it is becoming more blurred as some companies increasingly rely on open source projects and sometimes even have their staff contribute to them. At the same time, developers start considering themselves as part of a broader community of like-minded individuals that learn from and teach one another.

Since software development lends itself to distributed or remote work—collaborators need not be in the same office, city, or time zone [8]—these communities arise on a global scale and are mostly connected through tools that incorporate social aspects, helping developers connect 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.

B. The Importance of Media in Software Development

Software development activities are supported by development environments, debuggers, source code forges, version control systems, and bug trackers. Other tools that play an essential role in collaborative development include project management tools and communication channels such as chat [9], mailing lists [10] or micro-blogging services [11]. They also support knowledge management activities that are central to the success and longevity of a large software development endeavor.

Media—i.e., development tools and communication channels—play a critical role in how externalized and tacit knowledge is formed, shared, manipulated, and captured. Development tools and communication channels become an extension of the programmer [12] through which they can extend and distribute their cognition to develop, refine, and share knowledge. We previously [6] provided 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*. Our review shows how development tools have been adding more social aspects, which has also led to an increase in knowledge that is stored in communities and social networks.

C. The Rise of The Social Programmer

Developers these days 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 contribute with them. They rely on tools to participate effectively in these social networks, although sometimes they also face hurdles in participating and in staying up to date. The rise of the social programmer [13], [6] and the ways that communities of developers make use of tools that are increasingly social has led to the emergence of a highly participatory culture of software development. Jenkins [1] defines a participatory culture as one that *provides strong support for and lowers barriers to sharing; facilitates informal mentorship; and values social connections between members*.

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 as well as provide guidance for how to use the media effectively both by individual developers but also by teams and communities. In the next section, we describe a large-scale survey we designed to start our investigation on this topic.

III. Methodology: The Developer Survey

Our overarching research goal is to understand how communication channels and social media support a broad set of knowledge activities within a participatory culture of software development. To meet this goal, we designed and conducted a survey to learn how developers use tools to support their knowledge activities, to learn which media channels are important to them, and to determine which challenges they may face. Our research questions are phrased as follows:

RQ1 *Who is the social programmer* that participates in these communities?

RQ2 *Which channels* do these developers use to support development activities?

RQ3 *What challenges* do developers face using an ecosystem of communication channels to support their activities?

We downloaded account data for the 7,500 most recently active users who had also made their email address public. To indicate activity, we used the 20 events defined by the GitHub API v3[1]. 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[2]. We emailed our survey to 7,000 active GitHub users during November and December of 2013, and 1,516 developers responded (21% response rate).

In our survey[3], we first inquired about the **demographics** of the developers. We then inquired about **communication channel** use for a set of 11 **development activities**. These activities were as follows: *staying up to date* with technologies, practices and tools, *finding answers*, *learning* and improving skills, *discovering*, *connecting with other developers*, getting and providing *feedback*, *publishing* development activites, *watching* other developers' activities, *displaying* skills and accomplishments, *assessing* other developers, and *coordinating* with other developers.

[1] https://developer.github.com/v3/activity/events/types/

[2] https://octoverse.github.com/

[3] http://thechiselgroup.org/2013/11/19/how-do-you-develop-software/

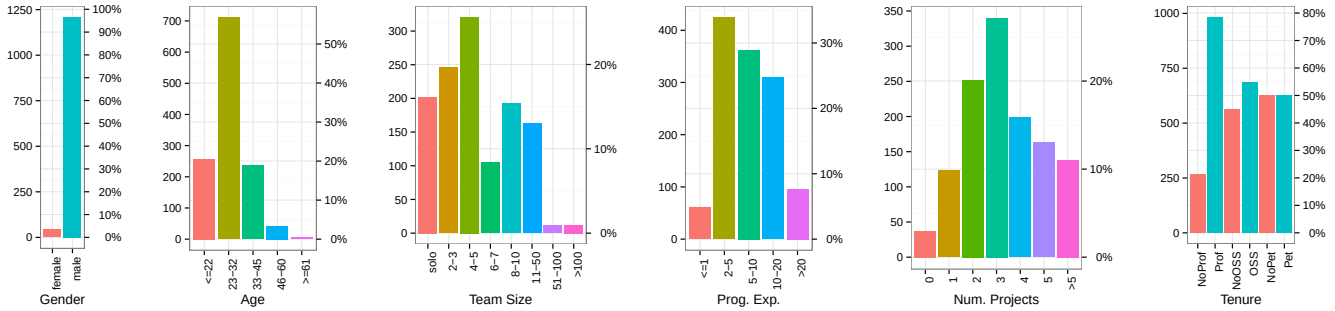


Fig. 1. Demographics of the population of programmers that answered the survey (those recently active on GitHub with public activity).

The survey questions about activities all followed the same form whereby we used a matrix of social channels that the survey respondent could select to indicate it was used for the corresponding activity (see [3]). The social channels specified in the matrix were determined in part from our own knowledge as developers, as well as feedback from fellow developers, but they were also refined using the research literature and were further refined through the survey design pilots. We included an "Other" option to elicit channels we did not consider. We further aimed to understand the **challenges** developers may face using social channels.

The survey instrument is one of the contributions of this paper; its source code can be found in a repository on GitHub[4]. This will allow others to replicate and build upon our work. We describe the analysis approach we followed as we present each research question, and refer to the limitations of the study in the discussion section of the paper. Due to space constraints, we have published additional findings, as well as provide links to the evidence from our findings, online at[5]. We dedicate more space in the paper to our third research question, as the insights gained from answering this question are more actionable in terms of making recommendations to programmers and to tool designers.

IV. 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 about demographic information such as gender, age, their geographical location, programming experience, the programming languages, technologies and tools they use, the number of projects they participate in, whether they program professionally, and about the sizes of project teams they had worked with. The answers to some of these preliminary survey questions are summarized in Fig. 1[5].

Gender: The overwhelming majority of our respondents identified as male—only 3.4% said they were female. However, it is possible that other respondents were female but did not wish to be identified as such[6].

Age: 20.5% of respondents said they were between 23 and 32, representing the largest age group in our survey and showing a strong bias towards relatively young developers. In fact, 77.4% said they were 32 or younger—3.7% were older than 45, and only 0.04% were older than 60.

Team Size: Team size was slightly more evenly distributed. Only 1.8% of respondents said they had worked in teams of more than 50 members. We found a slight bias towards smaller teams, with 61.5% having worked on teams of 5 or less, and with 16.2% saying they had only worked on projects where they were the sole member.

Programming Experience: In terms of experience, responses varied. Only 4.7% had less than 2 years of experience. 28.9% had worked as a developer for 5 to 10 years, and 24.4% had done so for more than 10 years.

Number of Projects: Most survey respondents (89.0%) had worked on 5 or less projects so far. Most developers had experience on 2 (20.1%), 3 (27.1%), or 4 (15.9%) projects.

Professionalism: Most respondents were *professional* software developers (79%); 55% considered themselves open source developers, and 50% worked on pet projects.

Table I depicts the Spearman correlation table between the different factors surveyed. A moderate correlation was found between age and programming experience (0.56), while there were several mild correlations between whether they are professional developers, whether they have pet projects, number of projects, team size and the number of channels they stated they have used. We found a mild negative correlation between developers that do open source and developers who are professional programmers and age (i.e., OSS developers are less likely to be professional and are more likely to be young).

V. 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. Table II shows which channels developers tend to use to support different activities[5]. We have grouped channels according to *analog*, *digital* and *social+digital* in this table. We can see from this table that there is a heavy reliance

[4]https://github.com/thechiselgroup/devsurvey
[5]Our technical report shows more details on these results
[6]http://meta.stackoverflow.com/a/281304

TABLE I

SPEARMAN CORRELATION TABLE FOR THE DIFFERENT DEMOGRAPHIC FACTORS IN THE SURVEY, THE NUMBER OF CHANNELS THEY USE, AND THEIR RESPONSES REGARDING PRIVACY, OVERWHELMED AND DISTRACTED. *** CORRESPONDS TO $p < 0.001$, ** FOR $p < 0.01$, * FOR $p < 0.05$

	Age	Gender	Tenure Prof	Tenure Pet	Tenure Oss	Num. projects	Team Size	Prog. Exp	Channels Used	Privacy	Over-whelmed
Age											
Gender	-0.02										
Tenure Prof	**0.31*****	-0.01									
Tenure Pet	0.02	**-0.07****	0.03								
Tenure Oss	**-0.16*****	-0.03	**-0.26*****	**0.33*****							
Num. Projects	0.05	**-0.07***	**0.21*****	**0.20*****	0.05						
Team Size	**0.07****	-0.02	**0.22*****	**0.07***	**-0.08****	**0.21*****					
Prog. Exp.	**0.56*****	**-0.08****	**0.29*****	**0.21*****	0.04	**0.20*****	**0.13*****				
Channels Used	-0.01	-0.01	**0.14*****	**0.19*****	**0.12*****	**0.22*****	**0.13*****	**0.09****			
Privacy	-0.05	0.05	-0.03	-0.05	**-0.08****	-0.05	-0.01	-0.04	0.01		
Overwhelmed	**0.07***	**0.07***	0.05	**-0.08****	**-0.09****	**-0.09****	-0.03	-0.06*	0.04	**0.20*****	
Distracted	0.04	0.01	0.03	-0.01	-0.02	0.01	0.02	**0.07****	-0.02	**0.18*****	**0.24*****

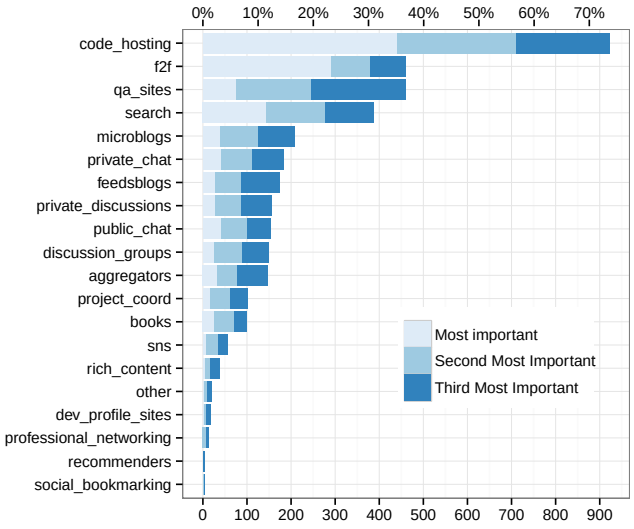


Fig. 2. Number of responses per channel indicating the importance of each.

on communication channels that support social features. On average developers indicated they use 11.7 channels across all activities, with median 12 and quartiles of $[9, 14]$ (see Figure B.1 in [5]). For each activity, we also asked developers to indicate other channels they may use that we did not list in the survey (see Table C.1 in [5]).

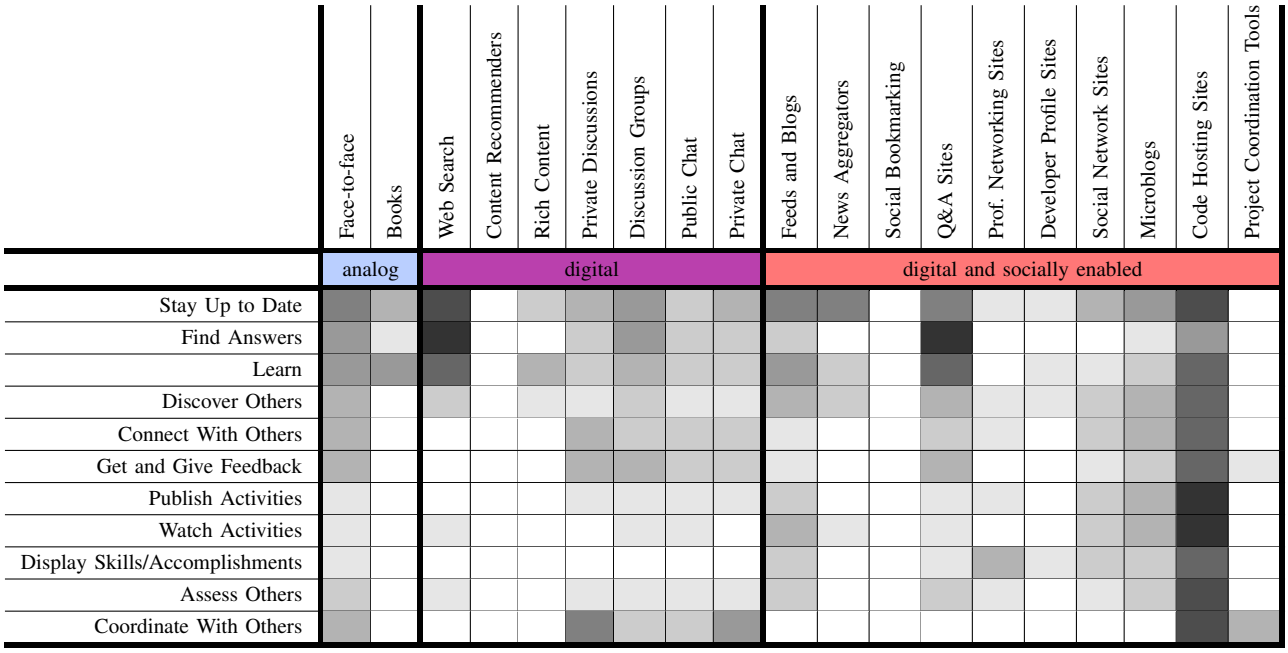
In the survey, we also asked developers to choose the three most important channels that they use to support their software engineering activities (overall). The results are shown in Fig. 2. It is not surprising that *code hosting sites* is the most frequent response (73% chose it) as we reached respondents via Github and code hosting sites serve an important role providing version control, issue tracking, and several means of communication between developers. *Face-to-Face* and *Question and Answers 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 with other developers. This finding resonates with previous works on the impact of distance in collaborative settings (e.g., [14], [15]). We know from other research [16] that *Question & Answers sites* play a prominent role in the activities of the developers, but we do not know if developers use it 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 information needs of developers [17], [18].

VI. CHALLENGES DEVELOPERS FACE USING COMMUNICATION CHANNELS

Our third research question **(RQ3)** inquires about the challenges developers face using communication channels. From our previous work [3], [11], [6], we were already aware that developers face challenges related to distractions, privacy and being overwhelmed by communication chatter using social media channels. Consequently in the survey, we asked specifically whether the respondents experienced these concerns. We report the results from these Likert style questions in Fig. 3. 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. We calculated correlations for the answers to each of these three challenges with age, gender and the number of tools developers used and found no significant correlations (see Table I). We anticipated that age may influence their responses in terms of privacy concerns, but no such correlation was found.

We further asked the respondents to share with us any **additional challenges** they face through an open-ended text question. Over 350 respondents either elaborated on the challenges mentioned above, or informed us about other challenges they face. We coded, sorted, clustered and then categorized these reported challenges using an open coding and iterative clustering technique. The main categories of challenges 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



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

TABLE II
ACTIVITIES AND THE CHANNELS DEVELOPERS IN OUR SURVEY USE FOR THEM.

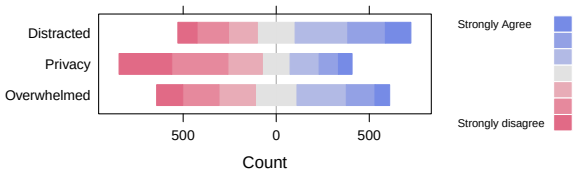


Fig. 3. Frequency of responses to Likert questions probing on developer challenges with DISTRACTION, PRIVACY, FEELING OVERWHELMED.

We report these challenges, as well as share some of the strategies respondents suggested using to address these challenges below. We use **bolded** text to indicate codes assigned to quotes and we provide representative participant quotes (shown in italics and linked to each participant using P#). Two of us independently coded the challenges in two passes. In the case where agreement was lacking, we followed a conservative approach by omitting the code for that response. 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 and space permitting, we discuss and link the reported findings to relevant literature. Further details (linking categories to findings to codes, and counts of codes and additional quotes can be found in[5]).

A. Developer issues

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

Distractions and interruptions from communication channels negatively impact developer productivity:

Survey respondents elaborated how social and communication media that is at their finger tips could be a **distraction** or could impact their productivity through **interruptions** [19] (see also Fig. 3). P165 mentions: *"Social Networking websites like Facebook are the worst ingredients for good concentration in general."* And P541 describes how it can be easy 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."*

Meanwhile, P1336 shared how he avoided distractions and interruptions by consciously deciding **when to use certain communication channels**: *"I turn off most communication tools at the right times (ie when I'm not in need of feedback or help). I'll still use GitHub for finding resources and a private messaging tool, HipChat or email, for quick questions."*

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

Keeping up with new technologies was a concern. P625 sums this up: *"Staying cutting edge is a never ending task"* and several respondents discussed not knowing which channel to watch, as P1564 mentions: *"Knowing where the activity is. Some days, hackernews 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."* The challenge of keeping up emerged in our previous work that investigated how developers use Twitter [11]. P706 discusses how keeping up with technology

is easier with open source because of their ability to use 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 its open source help is plenty and readily available, not so much for internal tools."*

Keeping up with activities on projects was also a reported challenge [20], and tools could be further improved as P109 shares: *"In a big project (WebKit, Mozilla, etc) it can be hard to filter for only ongoing work that is relevant. Most legacy UI's are terrible (bugzilla) and new ones (Github) lack features for large-scale development."* Keeping up with projects relates to how developers collaborate on projects, discussed next.

B. Collaboration and coordination hurdles

Respondents talked about challenges they face managing and coordinating their projects. As noted earlier the vast majority of the respondents contribute to two or more projects and also contribute to professional projects (see Fig. 1).

Sharing and explaining code lack adequate tool support:

Developers had difficulties **sharing code** and **explaining code**. P1416 explains: *"many communications tools (email, IM, etc) are not especially good for talking about code. Generally in any given conversation I'll end up using several tools, eg IRC + a paste-bin (GitHub Gists) to effectively communicate ideas."* Indeed P445 is enthusiastic about collaboration tool support except for the aspect of explaining 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 in getting **feedback** about their own activities, especially for **proprietary projects** that can't use social tools, as P706 mentions: *"Getting quick feedback for internal technology because you cannot ask on stack overflow"*. Sometimes the challenge of getting feedback can be due to the size of the project (community) rather than the channel, as P751 explains: *"It's difficult to share small new projects that aren't very far along and get feedback."*

Collaborative coding activities need improved tool support:

Our survey concerned communication tools, but P1154 shared with us that coding tools are not adequate to support **collaborative coding** activities as follows: *"We can currently edit a document collaboratively in google doc. If we can have a IDE / tool like that for coding too, that would be useful."* There are some browser based IDEs that indeed provide this support, such as Nitrous.io and Cloud9, that are starting to see increased adoption [21]. Such tools may also address the challenges of explaining code and getting real time feedback.

C. Barriers to community building and participation

Over half of the survey respondents contribute to open source projects many of which are "pet" projects (see Fig. 1). We can also anticipate that most developers rely on one or more open source projects or community authored resources such as Stack Overflow. It is thus not surprising that many respondents found it challenging to participate in or to find others to participate on community based projects.

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

Challenges relating to geographic, economic or demographic factors emerged from the survey. Different **time zones** interfering with their work was mentioned, 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 [5]). As P31 mentions: *"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."*

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

A few developers also discussed challenges in **finding developers** to collaborate with or **convincing others to participate**. For example, P1285 complains: *"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 discusses 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."*

D. Social and human communication challenges

Many respondents specifically mentioned struggling with communication or people issues.

Miscommunications on text-based channels are common:

Issues from **miscommunications** on **text-based** channels were frequently mentioned. As P385 mentions: *"With the increasing amount of communication being done with social tools and IMs, Chat, the amount of misunderstanding and bad incomplete briefs grows in the same rate."* P126 describes

how text based communication takes effort: *"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."* P1331 agreed with this and mentions how slang can introduce even more ambiguity.

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

Many respondents shared the opinion that *"Nothing replaces face to face communication."* as **miscommunications** occur more frequently. Other developers discussed how **face-to-face** is needed for some activities such as code review and for explaining the big picture underlying a design, as P319 mentions: *"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"* and P1241 adds: *"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."*

Others shared with us that there are tools that can support face-to-face like interactions, as P206 told us: *"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."*

People are challenging, no matter which channels are used:

Poor attitude and a lack of willingness to collaborate are issues no matter which tools are used. As P264 sums up *"[Tools] still don't solve the difficult people problems."*. P532 further explains: *"Tools facilitate good processes and interactions between individuals who are willing to collaborate and co-operate. They don't make people willing to co-operate 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."*

The social transparency [22], [23] of the channels introduced other issues [24]. 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. P302 shared: *"The biggest thing I fear in my work is that I'll say something that is not 100% technically accurate or could be misinterpreted. Other developers are utterly merciless, and I have thin skin. Whenever I post something on HN or StackOverflow I find that I feel anxious that someone will tear me a new one over some oversight in my analysis."* P643 also discusses the potentially negative impact of these feelings of intimidation as follows: *"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)."*

E. Communication channel affordances, literacy and friction

Developers need to consider **channel affordances**:

Different channels provide different kinds of affordances as Daft and Lengel's Media Richness Theory describes [25]. Many developers recognize the strengths and weaknesses of different channels and recognize tensions between **private vs. public** channels, **synchronous vs. asynchronous** communication, **ephemeral vs. archival** channel properties, **anonymous vs. identified** participation, and support for different communication types such as **textual vs verbal**, and **face-to-face** based conversations.

Developers shared with us that **no single tool fits all** developers' needs nor suits all stakeholders. Different stakeholders have very different needs and so different channel affordances will not suit everyone, as P160 tells us: *"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. The makes overall coordination extremely difficult"*

Finding the right channel to support **communication with users** was also a challenge. P1528 describes how keeping track of communication from users (and other developers) is difficult: *"Users/Developers tends to report bugs or asks for new features with email and forums posts which can be tricky to keep track off."* To address this, some projects promote user reporting of issues on GitHub, but P394 explains 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:

Knowing how to use development tools effectively is a requisite skill for developers, but **literacy** with communication channels and understanding the different channel affordances is also key for developers that need to collaborate, exchange information or network with others across their communities. As P1005 describes: *"The main challenge for me is the interaction with people who are not literate enough to use the tools I consider standard."* P1600 also discusses how Git (the underlying version control tool 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."* While P1385 mentions how using GitHub itself or even IRC can even 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's similar issues (like how to submit patches) although documentation is improving."*

Respondents discussed challenges having to **learn** new tools, as P404 describes: *"The biggest challenge from 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."* Each new channel may furthermore require acquiring a different vocabulary, as P526 explains: *"If you don't know the right [vocabulary] or technology you want to use, the [dialogue] usually ends quickly."* P981 mentions that some of these tools are difficult to learn because of a **lack of documentation**.

Literacy, however, relates to not just knowing how to use a particular channel, but when to use it (and when not to use it according to their needs, 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, as P919 exclaims: *"Never have enough screens."* This concern about screens is not surprising given the number of channels and tools developers use. Other tools, despite being highly popular among developers, suffer usability issues. P1279 admonishes: *"Hacker News dominates and is terrible."*

Vendor lock-in was also an issue that was mentioned by developers, not surprising since many of the social tools developers use are proprietary. P334 discusses this as follows: *"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."* Also projects face issues when privacy policies change with proprietary tools, P681 told us: *"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)."*

Many challenges arose from **poor channel integration**, such as having to deal with identity management. P1430 explains the many issues that 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** (i.e., that there were simply too many channels to monitor or to choose from). But newer communication channels address this integration problem, as P980 tells us: *"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."*

Poor or scattered **adoption** of tools can furthermore introduce friction, especially when many tools may be available, as P694 explains: *"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 stands out as an obvious leader."* And getting agreement about communication tools is also difficult, as P422 tells us that his challenge is: *"Convincing other developers on a project to all use the same communication tool for that project."* The problems are exacerbated by globally distributed teams, P380 explains: *"I live internationally and work on globally disparate project teams, so G11N 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..."*

But it is not just poor adoption that introduces friction, the **way the tools are used** can be an issue. As P253 explains: *"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 successfully figure this out as P783 tells us: *"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."*

F. Content and knowledge management

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

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

Information fragmentation results from the use of so many channels, as P1401 says: *"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."* Fragmentation also occurs because of the inconsistent or poor **adoption** of particular channels (as discussed above). P1250 suggests how to 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 very challenging to find what some termed the signal in the **noise**. The 'explosion' of channels has lead to an increase in volume and duplicate information posted on multiple channels. This is particularly a challenge for developers working on multiple project where different tools are used, as P815 mentions: *"The variety of tools,*

and the need to switch context and tool set between various subprojects, adds a lot of cognitive overhead." There is also a fear of missing important information, as P917 explains: *"Too many channels mean that needed or interesting information disappears, and going through all of channels you mentioned is impossible in limited time."* 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, as P1189 describes: *"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."*

The diversity and velocity of information makes it hard for developers to **keep up with new technologies**. As P1409 puts 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 [11] but the availability of many different news sites and aggregators means they are inundated with content, as P1144 told us they affect productivity: *"The News overload via Aggregators (Hackernews, Reddit, Digg, Slashdot, ...) affects productivity. I don't use too much of social networking (facebook or twitter) as they are a huge time-sink and sheer noise as far as technical development work is concerned."*

A few respondents shared with us the strategies they use to avoid being overwhelmed by the quantity of information they may receive. P232 suggests: *"Use of numerous tools may be overwhelming. It is usually assumed that it is better to use less tools, and increase the direct communication frequency between developers using face-to-face or chat."* and P692 further suggests to just **unplug**: *"Sometimes it helps to have a day of development where you unplug in the internet."*

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

In addition to having too much information, many developers described how they were concerned with the **quality** of the information. As P846 mentions: *"Judging the reliability and credibility of sources can be a challenge as information changes quickly and isn't always correct."* There was especially concern with social sites, as P95 describes: *"I sometimes feel 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."* Developers were also concerned when they would find contradicting or inconsistent information, P257 explains: *"Information is not consistent across IRC, mailing lists, blog posts, etc even within the same project especially re: new features and development changes."*

There were also specific complaints that information could sometimes be **obsolete**, as P492 describes: *"Technologies are moving so fast, and most of the content on the Internet could be outdated quickly. It's hard sometime to filter those outdated information."* Or that posted information could be **spam**, as P671 tells us: *"User right management is crap. Email notification control is very limited which leads to either being spammed or missing everything."* Developers also described how it can be difficult to find content on **niche** technologies and that understanding the **history** of the information created was hard to acquire.

VII. Discussion

Through this survey, we have started to investigate how a complex ecosystem of communication channels shapes and challenges a participatory development culture. Firstly, we discovered the **characteristics of the programmers** that contribute openly to projects hosted on the Github social coding site. We then considered the different **activities** developers care about, such as collaborative authoring, learning and sharing with each others, networking, and keeping up to date with technologies and project activities, and we inquired about the **communication channels** developers use for each of these activities. We also determined which channels were most **important** for development activities and probed through an open-ended question about the **challenges** developers face using an ecosystem of communication channels.

Although the survey was time consuming to fill out, we saw a 21% response rate. Indeed many of the developers added additional comments about how they were happy to see and contribute to this research, and were themselves curious about the channels other developers use and the challenges others experience. Although our data and findings from the questions concerning demographics and channel usage are interesting, the data we received about the challenges they face was the most interesting. These challenges reveal that although the modern social tools developers can use, many developers care about how tools impact their work practices.

We used a **survey** instrument to study this ecosystem, because a survey allowed us to reach a broad population of developers. The survey design followed several phases of design, as it is challenging to uncover the landscape of tools developers use for a broad set of activities. Our final design was a good compromise in terms of the information we gathered with developer time. The survey inclusion criteria suggests participant bias towards social code hosting systems, however, this population was the focus of our study, and we do not claim generazability of our results. In the presentation of the challenges, we rely heavily on the developers' own words to bring credibility to our findings. In the appendix to our companion technical report, we provide counts and additional quotes, however, we stress that the counts are not valid indicators of the importance of the challenges that were shared through an optional manner. Future work is needed to understand more about the prevalence of the challenges unearthed, as well as reveal strategies that developers follow to address these challenges.

As a continuation of this study, we anticipate that interviews with developers can shed more light on the *strategies* developers use to mitigate the challenges they face. Additionally, this study 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), while the

themes that emerged in this study will help to form a new version of the survey[4]. Lastly, our future work will allow us to further develop 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 for tool improvements.

VIII. Conclusions

Communication channels shape and challenge the participatory culture in software development. However, the reverse is also true, and not much is understood about the impact of the participatory culture on software development practices and the communication channels developers use. This has far more impactful implications on other knowledge workers as well; Software developers are referred to as the *knowledge worker prototype*, as they are not only the first to use and shape the tools and channels, but also have far lower barriers to build and tweak these tools [26]. In that case, it won't be surprising if these challenges will propagate to other domains as well.

References

[1] H. Jenkins, *Confronting the challenges of participatory culture: Media education for the 21st century*. Mit Press, 2009.

[2] E. C. Wenger and W. M. Snyder, "Communities of practice: The organizational frontier," *Harvard business review*, 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 *Proceedings of the 2013 conference on Computer supported cooperative work*. ACM, 2013, pp. 103–116.

[4] M. C. et al. (2012) The social economy: Unlocking value and productivity through social technologies. http://www.mckinsey.com/insights/high_tech_telecoms_internet/the_social_economy.

[5] F. Lanubile. (2013) Social software as key enabler of collaborative development environments. 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. of the 36th Intl. Conf. on Software Engineering, Future of Software Engineering*, ser. FOSE 2014. New York, NY, USA: ACM, 2014, pp. 100–116. [Online]. Available: http://doi.acm.org/10.1145/2593882.2593887

[7] J. Lave and E. Wenger, *Situated learning: Legitimate peripheral participation*. Cambridge University Press, 1991.

[8] J. Fried and D. H. Hansson, *Remote: Office Not Required*. Ebury Digital, 2013.

[9] 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 *Mining Software Repositories, 2009. MSR '09. 6th IEEE International Working Conference on*, May 2009, pp. 107–110.

[10] A. Guzzi, A. Bacchelli, M. Lanza, M. Pinzger, and A. v. Deursen, "Communication in open source software development mailing lists," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, ser. MSR '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 277–286. [Online]. Available: http://dl.acm.org/citation.cfm?id=2487085.2487139

[11] L. Singer, F. Figueira Filho, and M.-A. Storey, "Software engineering at the speed of light: How developers stay current using twitter," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 211–221. [Online]. Available: http://doi.acm.org/10.1145/2568225.2568305

[12] M. McLuhan and Q. Fiore, *The medium is the message*, New York, 1967.

[13] C. Treude, F. Figueira Filho, B. Cleary, and M.-A. Storey, "Programming in a socially networked world: the evolution of the social programmer," *The Future of Collaborative Software Development*, pp. 1–3, 2012.

[14] 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

[15] 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

[16] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, "Design lessons from the fastest q&a site in the west," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '11. New York, NY, USA: ACM, 2011, pp. 2857–2866. [Online]. Available: http://doi.acm.org/10.1145/1978942.1979366

[17] A. J. Ko, R. DeLine, and G. Venolia, "Information needs in collocated software development teams," in *Proceedings of the 29th International Conference on Software Engineering*, ser. ICSE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 344–353. [Online]. Available: http://dx.doi.org/10.1109/ICSE.2007.45

[18] A. Begel and T. Zimmermann, "Analyze this! 145 questions for data scientists in software engineering," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 12–23. [Online]. Available: http://doi.acm.org/10.1145/2568225.2568233

[19] X. Wang, S. Ye, and H. H. Teo, "Effects of interruptions on creative thinking," 2014.

[20] O. Baysal, R. Holmes, and M. W. Godfrey, "No issue left behind: reducing information overload in issue tracking," *Under review*, 2014.

[21] M. Goldman, G. Little, and R. C. Miller, "Real-time collaborative coding in a web ide," in *Proc. 24th annual ACM symposium User interface software and technology*. ACM, 2011, pp. 155–164.

[22] H. C. Stuart, L. Dabbish, S. Kiesler, P. Kinnaird, and R. Kang, "Social transparency in networked information exchange: A theoretical framework," in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, ser. CSCW '12. New York, NY, USA: ACM, 2012, pp. 451–460. [Online]. Available: http://doi.acm.org/10.1145/2145204.2145275

[23] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb, "Social coding in github: Transparency and collaboration in an open software repository," in *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, ser. CSCW '12. New York, NY, USA: ACM, 2012, pp. 1277–1286. [Online]. Available: http://doi.acm.org/10.1145/2145204.2145396

[24] 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 *Proceedings of the 18th ACM conference on Computer supported cooperative work & social computing*, 2015, pp. 1–13.

[25] R. L. Daft and R. H. Lengel, "Organizational information requirements, media richness and structural design," *Management science*, vol. 32, no. 5, pp. 554–571, 1986.

[26] A. Kelly, *Changing Software Development: Learning to Become Agile*. John Wiley & Sons, 2008.

APPENDIX A
RADAR CHARTS FROM THE DEVELOPER SURVEY

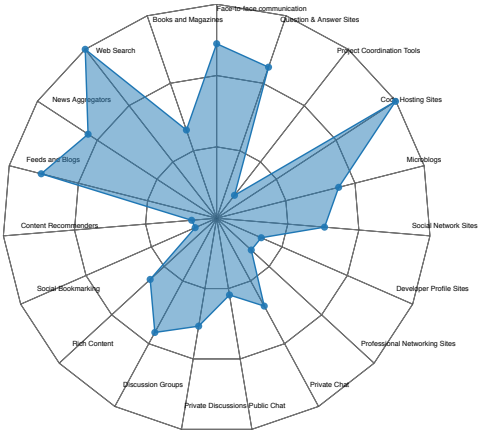


Fig. A.1. The following help me stay UP TO DATE about technologies, practices and tools for software development.

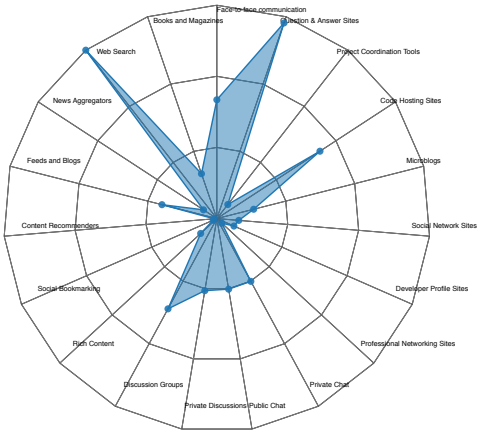


Fig. A.2. The following help me FIND ANSWERS to technical questions.

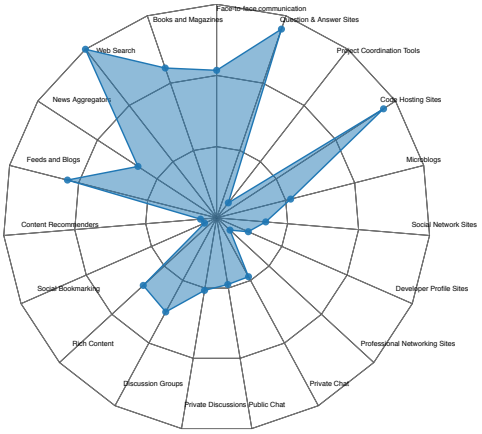


Fig. A.3. The following help me LEARN and improve my skills.

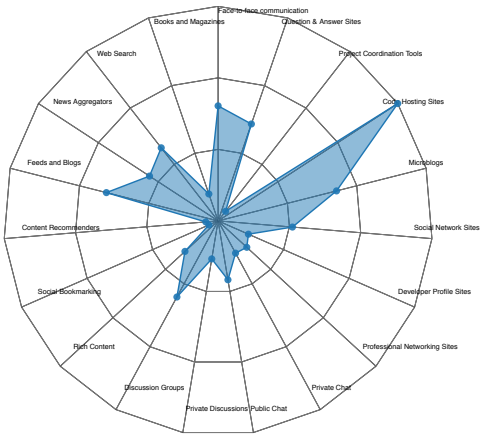


Fig. A.4. The following help me DISCOVER interesting developers.

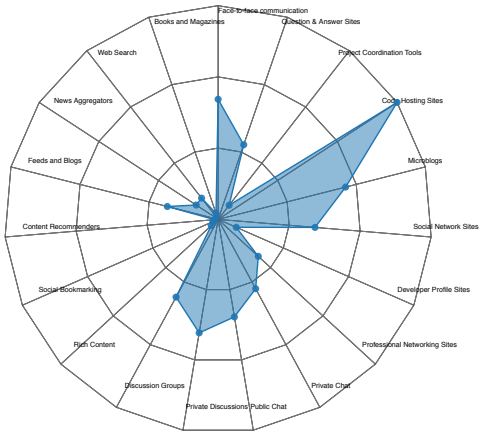


Fig. A.5. The following help me CONNECT with interesting developers.

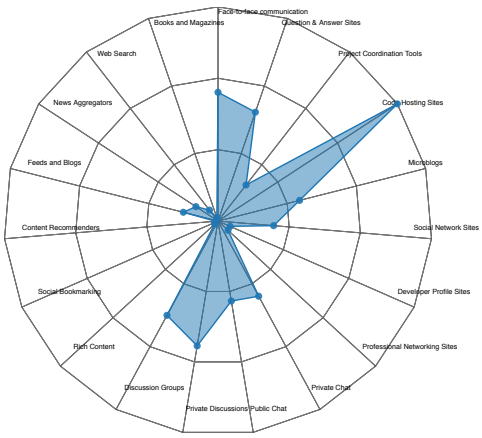


Fig. A.6. The following are useful for getting and giving FEEDBACK.

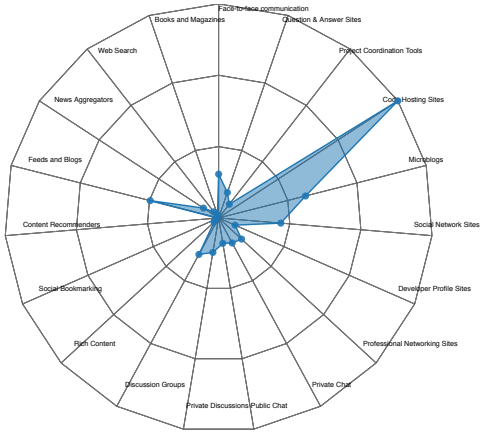


Fig. A.7. I use the following to PUBLISH my development activities.

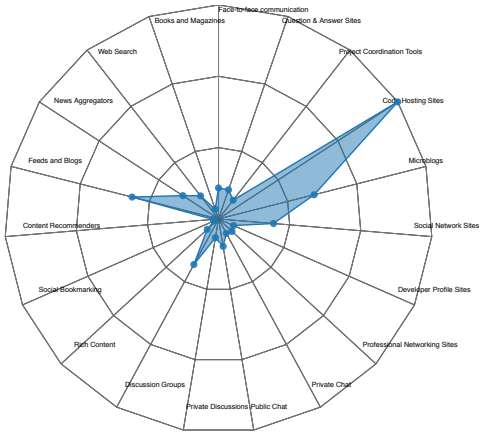


Fig. A.8. I use the following to WATCH other developers' activities.

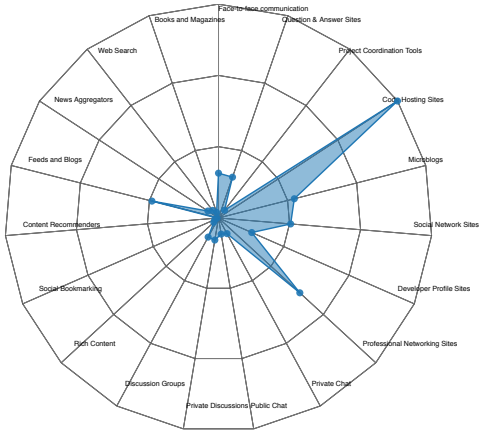


Fig. A.9. I use the following to display my SKILLS/ACCOMPLISHMENTS.

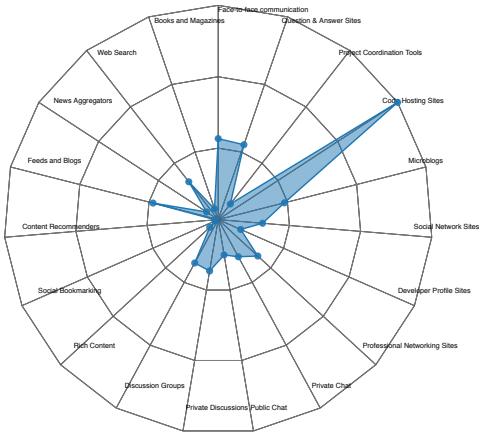


Fig. A.10. I use the following to ASSESS other developers.

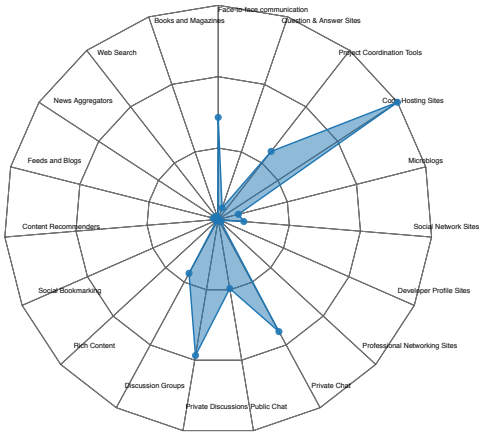


Fig. A.11. I use the following tools to COORDINATE with other developers when I am participating on projects.

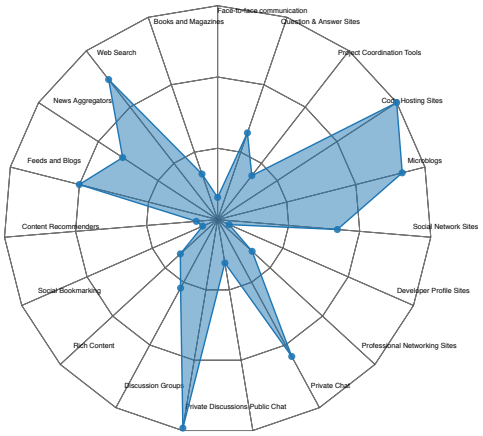


Fig. A.12. I use the following tools on a MOBILE DEVICE (smartphone, tablet) for development-related activities.

APPENDIX B
ADDITIONAL FIGURES

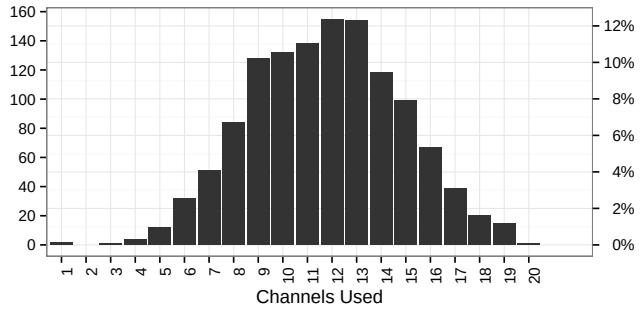


Fig. B.1. Histogram of number of channels selected by each developer.

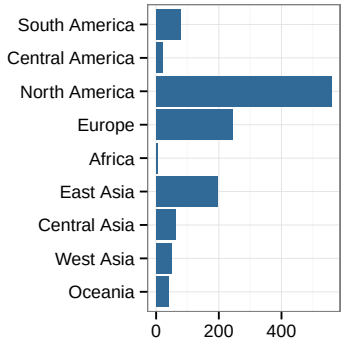


Fig. B.2. Geographic location of the population of programmers that answered the survey.

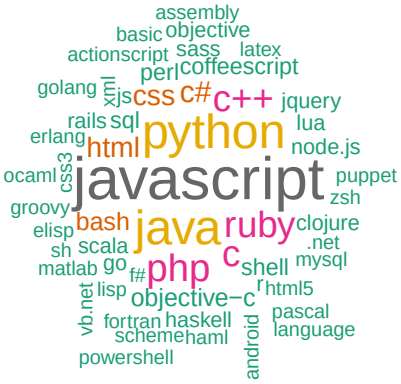


Fig. B.3. Word cloud of the programming languages used by the population of programmers that answered the survey.

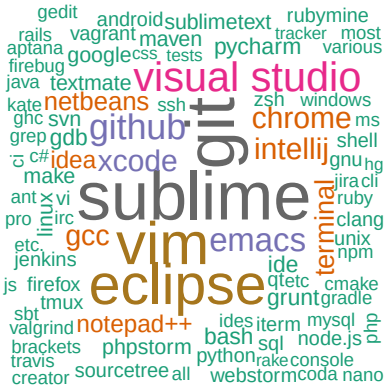


Fig. B.4. Word cloud of the tools and frameworks used by the population of programmers that answered the survey.

APPENDIX C
ADDITIONAL TABLES

Activity	Channels
Keeping up to date	events and meetups [17], software documentation [3], research papers [3], formal education [2], MOOCs [2]
Finding answers	software documentation [11], research papers [3]
Learning	events and meetups [17], software documentation [13], educational sites and MOOCs [18], tutorials [7], research papers [6], code reviews [4]
Discovering developers	headhunters [16], recruiting sites [9]
Connecting with developers	events and meetups [29], programming competitions [2]
Getting and giving feedback	events and meetups [9], code reviews [4], issue trackers [4]
Publishing activities	personal blogs and websites [28], conferences [6]
Watching activities	events and meetups [3], personal blog and websites [2]
Displaying skills	personal blogs and websites [57], resumes [6], events and meetups [4]
Assessing others	personal blogs and websites [2], source code[2]
Coordinating with others	conference calls [9], cloud-based services (e.g., Dropbox, Google Drive) [7]

TABLE C.1
OTHER CHANNELS REPORTED IN THE SURVEY.