The (R)Evolution of Social Media in Software Engineering

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

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

Brendan Cleary
University of Victoria
Victoria, BC, Canada
bcleary@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

ABSTRACT

Software developers rely on media to communicate, learn, collaborate, and coordinate with others. Recently, *social media* has dramatically changed the landscape of software engineering, challenging some old assumptions about how developers learn and work with one another. We see the rise of the *social programmer* who actively participates in online communities and openly contributes to the creation of a large body of *crowdsourced socio-technical content*.

In this paper, we examine the past, present, and future roles of social media in software engineering. We provide a review of research that examines the use of different media channels in software engineering from 1968 to the present day. We also provide preliminary results from a large survey with developers that actively use social media to understand how they communicate and collaborate, and to gain insights into the challenges they face. We find that while this particular population values social media, traditional channels, such as face-to-face communication, are still considered crucial. We synthesize findings from our historical review and survey to propose a roadmap for future research on this topic. Finally, we discuss implications for research methods as we argue that social media is poised to bring about a paradigm shift in software engineering research.

Categories and Subject Descriptors

H.5.3 [**Group and Organization Interfaces**]: Computer-supported collaborative work

General Terms

Human Factors

Keywords

Social Media, Software Engineering, Collaboration.

FOSE'14, May 31 – June 7, 2014, Hyderabad, India
ACM 978-1-4503-2865-4/14/05
http://dx.doi.org/10.1145/2593882.2593887

1. INTRODUCTION

A few years after the term "software engineering" was coined at a 1968 NATO conference [49], Weinberg published his acclaimed book on the "Psychology of Computer Programming." [78] Weinberg emphasizes that although programming is an activity that can be performed alone, it relies on extensive social activities as developers will often have to ask others for help.

The need to collaborate and interact with others has only increased over time. Even solitary developers need to interact directly or indirectly with others to learn, to understand requirements and to seek feedback on their creations. In addition to face-to-face communication, developers use many channels to interact. Communication media, such as the telephone, email, chat, bug tracking tools, and code hosting sites, play a pivotal role in collaborative software engineering activities, thus helping to shape and form co-located and distributed online communities. However, media channels have different affordances for communication, coordination, collaboration and knowledge flow.

In just the past decade, the world has seen a massive and widespread adoption of social media—media which supports many-to-many communication through social networks. The speed and scale of adoption of social media such as Facebook and Twitter is unprecedented in the history of technology adoption [12]. These tools alone have had a profound and disruptive impact on many domains, most notably journalism and politics. Likewise, GitHub[1] (a social coding site) and Stack Overflow[2] (a question and answer Website) have become part of the standard toolset for many software engineers in just the last few years.

In this paper, we consider the impacts of social media on software engineering activities. We propose that social media in software engineering is contributing to a paradigm shift in three significant ways:

1. The rise of the social programmer that actively participates in online development communities;
2. A rapid increase in the creation and diffusion of technologies and crowdsourced content; and
3. The formation of ecosystems around content, technology, media, and developers.

[1] `https://github.com`
[2] `http://stackoverflow.com`

This paper showcases a research roadmap that outlines opportunities and challenges arising from the use of social media in software engineering. We include a history of media use in software engineering, from the very first days of software engineering in the late 1960s, to the present day use of social media, preceded by a conceptual background on the role of media in communities. The paper concludes with a discussion on the impact social media is having on software engineering research directions and methods.

2. BACKGROUND

In this section, we provide background on communities of practice and the role media plays in shaping those communities. We then examine how social media has contributed to a participatory culture within communities of practice. Finally, we discuss communities of practice in software engineering.

2.1 Role of Media in Communities of Practice

According to Wenger [79], **communities of practice** arise when "*groups of people who share a concern or a passion for something they do and learn how to do it better as they interact regularly.*"

Communities of practice support ongoing learning [37] for core and peripheral members, and act as a "*living curriculum for the apprentice*" with long term members in the community sharing practices, resources and tools with each other and with newcomers. People outside the community recognize that the community exists and value the skills and relationships members have. Social scientists study communities of practice to discern how people learn from one another and to understand successful relationships and practices. Such insights can be helpful in building new communities and sharing work practices.

Media play an essential role in supporting communication, collaboration and coordination activities within a community of practice. McLuhan proposed that media shapes society and culture—he defines media as "*ways of communicating that are extensions of a human being's senses.*" [46] He describes a history of media starting with the **tribal era**, where speech shared in a group was a unifying act leading to enthusiasm and spontaneity. This gave way to the **literary era** with the development of writing which could be referred to multiple times and reflected on independently. The literary era was succeeded by the **print era**, where the Gutenberg printing press facilitated mass reproductions, allowing people to consume the content in isolation. The introduction of instant communications brought about by the telegraph, telephone, radio, TV, information technology and Internet led to the **electronic era**. The constant and immediate contact the electronic era afforded across boundaries led to what McLuhan coined the **global village**.

As new forms of media emerge, McLuhan suggests we consider how these new channels **enhance** pre-existing forms of media, how they may make other forms of media **obsolete**, how they may **retrieve** affordances from older forms of media and what they may **reverse** (or flip) into if taken to the extreme. Moreover, McLuhan emphasized that media determine who can participate, the scale of information that can be distributed, how fast it is delivered and how it will be presented. For example, mass media channels prevalent in the early days of the electronic era, such as radio, TV, newspapers and websites, support one-to-many spectator style communication, often leading to hierarchical structures in organizations and society. Postman, who also studied media extensively, refers to a **media ecology** and suggests we undertake "*the study of environments: their structure, content and impact on people.*" [58] Thus, Postman reminds us that when media is studied, it is important to consider the full landscape of the channels used, their underlying context and the impacts on all stakeholders.

2.2 Social Media and Participatory Cultures

The development of the World Wide Web and cheap access to Internet communication (e.g., email and online chat) has greatly impacted communities of practice. The biggest changes have occurred in the past 5 - 10 years with the creation of a different kind of media: **social media**. They support enhanced communications that spread faster and with wider reach. Social media (e.g., Facebook, Twitter) are cheaper and easier to use than traditional mass media, with many social media channels supporting a many-to-many distribution mechanism. There are fewer barriers to participation, and merely interacting with the media is an implicit form of participation (e.g., clicking on a link reinforces the value of that link). Social media often result in increased transparency, they are self-reinforcing in terms of participation and bring with them new forms of value. Social media are recognized as a disruptive force in many domains [1] and they have been increasingly adopted in corporate settings.

Jenkins et al. [30] observed that social media bring about a participatory culture with the following attributes:

- Low barriers to artistic expression and engagement
- Strong support for creating and sharing one's creations with others
- Informal mentorship—what is known by the experienced is passed along to novices
- Members believe their contributions matter
- Members feel some degree of social connection and care what others think about their creations

This new participatory culture calls for a new form of literacy—social media literacy skills that are ways of interacting with a larger community of practice in contrast to individual skills used for personal expression through mass media.

Although many see the recent widespread adoption of social media as a major change in human culture, Tom Standage [67] speculates that social media is a retrieval of the previous two-way horizontal models of communication that relied on social connections rather than the one-way vertical communication that is typical of mass media. Standage provides "social media" examples from ancient times, such as from the days of the Romans where slaves (inexpensive to their owners) were used for sending, copying and replying to letters between friends and family. He suggests that the mass media era was a relatively short-lived anomaly in human culture and that social media is more of the norm. One criterion he gives for media to be deemed "social" is that they must be relatively inexpensive and easy to use.

Standage also reminds us of the "coffee houses" from the 17th and 18th centuries where people in the UK gathered

to discuss and read letters of news, a precursor to the mass-produced printed news. Coffee house articles were composed of other news, similar to today's news aggregators or Pinterest. Coffee houses were an "alluring environment" where serendipity was commonplace as newsletters contained all sorts of re-posted information. It is interesting to note that the number of copies / reprints / repurposed content indicated the success of a message, just as the number of retweets / reblogs / repins may be measures of value today.

McLuhan also highlighted the participatory nature of media, but he did not witness the current phenomenon of social media. For the purpose of distinguishing this more recent phenomenon from the preceding electronic era, we refer to this recent time period as the **social era**. What sets the social era apart from the electronic era is the inexpensive and low barrier to publish, as well as the rapidly spreading peer-to-peer, large-scale communications made possible by social media.

2.3 A Participatory Culture in Software Engineering

Communities of practice are prevalent in software engineering, and we can see them forming to share ideas about software architecture, testing, requirements and around specific technologies. They have been the topic of much research, particularly in the open source realm which relies on intense communication. With the advancement of remote systems and the wider availability of the Internet in the early 80s, free and open source software systems were developed by small groups of developers. However, it wasn't until 1993 that Linux broke the mold using what Raymond [59] refers to as a "Bazaar" model of programming, an approach which accepted contributions from anyone and adopted a mantra of "release early, release often." Communities of practice naturally formed around free and open source projects, with communication tools such as email and version control playing an important role in the community formation. The open source movement further bears all the hallmarks of a participatory culture, with lower barriers to entry, strong support for co-creation of artifacts, mentorship opportunities and appreciation of social relationships.

A participatory culture cannot be found in open source projects alone. Global software development [27] and collaborative software development [81] have become increasingly popular modes due to relatively inexpensive and readily available communication tools, such as email, chat, bug tracking and version control. Whitehead [80] notes that the two threads of appropriation of novel communication tools and model-based development together distinguish how collaboration in software engineering results in unique collaboration challenges and requirements. Seven years ago, he acknowledged a trend of Web-based collaboration tools that might support distributed teams better—a scenario that has become very real, as attested by the recent interest in remote work supported by asynchronous communications via Web-based tools (cf. e.g., Fried and Heinemeier Hansson [20]). However, the increase in interest in remote work and the popularity of recommendations on the topic show that distance still matters [50].

Nowadays, we see continuing widespread adoption of social media and social features in coding tools. Throughout the emergence of this participatory development culture, media has played a pivotal role. Next, we review how the use of different media has evolved within software engineering.

3. COMMUNICATION MEDIA IN SOFTWARE ENGINEERING: A RETROSPECTIVE

An important role of media in any socio-technical endeavor is the transfer of knowledge between stakeholders. This facilitates individual learning and expression, as well as coordination and collaboration between members in a community. Wasko et al. [77] distinguish different theories of knowledge: **knowledge embedded in people** that may be tacit or embodied within people's heads, with its exchange typically done one-on-one or in small group interactions; **knowledge as object** that exists in artifacts and can be accessed independently from any human being; and **knowledge as public good** that is "socially generated, maintained and exchanged within emergent communities of practice." We use these three types of knowledge to categorize how media support the flow of knowledge in software engineering. We also add a fourth type of knowledge that has recently become relevant in software engineering: **knowledge about people and social networks**.

Our discussion of the channels for each of these perspectives focuses on how the various channels evolved and influenced one another over time (cf. Fig. 1). Some channels overlap multiple perspectives. The timeline of media use in software engineering is further discussed at the end of this section (cf. Section 3.5).

3.1 Communicating Knowledge Embedded in People's Heads

In the early history of software development, the main communication channel was **face-to-face**, as most groups and teams at that time were co-located. Furthermore, reliance on other members was not that significant as most programs written in the 1960s and early 1970s tended to be small [64]. Face-to-face interaction was essential to support learning and problem solving, to build common ground [13] and to support collaborative system design and development. Face-to-face is often espoused to be the best way to exchange tacit knowledge, i.e., knowledge that resides in programmers' heads. Face-to-face interactions are still a mainstay of communication in software projects (cf. results of our survey, Sec. 4.2)—however, many projects today are developed by engineers that may have never met face-to-face and never will. In some cases, video chat tools such as **Skype** or **Google Hangouts** are used as a substitute.

The **telephone** was also an important medium in supporting the early days of collaboration in software engineering. De Marco and Lister [16] say the following about the phone in their 1987 book: *"The telephone is here to stay. You can't get rid of it, nor would you probably want to."* However, they worried about it causing interruptions.

De Marco and Lister also discussed the importance of

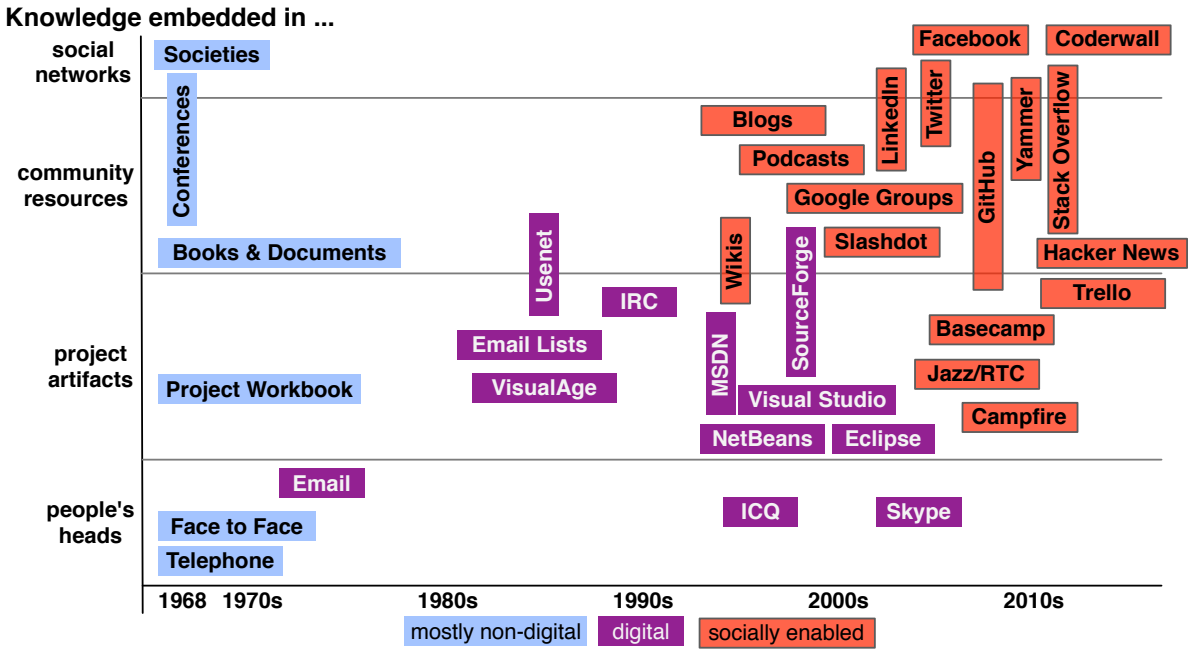

Figure 1: Media channels over time and how they support the transfer of developer knowledge. Note that some channels overlap multiple types of knowledge communication.

email as a communication channel between developers. When comparing email to the telephone, they remarked: *"The big difference between a phone call and an electronic mail message is that the phone call interrupts and the e-mail does not. The trick isn't in the technology; it is in the changing of habits."* This tension between synchronous and asynchronous communication channels—and even workflows—is garnering renewed attention now that remote work and open source development models are being experimented with at young software companies (cf. e.g., Holman [29] about workflows at GitHub).

Instant messaging is also used by many developers to support one-on-one discussions and exchange knowledge. One such tool is ICQ. It was developed in 1996 and is still in use today with more recent versions of ICQ including support for video messaging and social networking. Many developers also make use of **text-based group chat systems** for communicating about development, such as IRC. Handel et al. studied a customized version of an IRC chat tool and found that globally distributed developers predominantly used it for technical discussions [26], but they also found that adoption was inconsistent across development teams [28]. Tools that are popular today include the Web-based archivable Campfire[3] or HipChat[4].

3.2 Communicating Knowledge Embodied in Project Artifacts

By project artifacts, we refer to communication that specifically discusses code, documentation, test cases, requirements, designs and tasks. There are numerous tools used to communicate about these artifacts—we discuss a small sample to demonstrate their breadth and scope.

In his landmark book *The Mythical Man Month*, Brooks [8, p. 74] discussed the role of a **project workbook** as a comprehensive way to externalize critical knowledge about the development of the OS/360 system in 1965. The project workbook was used to document system knowledge, track all project activities, including rationale for design decisions and change information across versions. But Brooks noted that after six months, the workbook was five feet thick and inserting new pages and adding margin notes to maintain it was onerous. Introducing microfiche helped with the size issue, but highlighting and commenting became impossible.

The workbook was eventually replaced with more sophisticated tools, notably **IDEs** (Integrated Development Environments), **online hyperlinked documentation**, **project forges**, **version control systems**, **bug trackers** and **project management** tools. Over time, these different tools incorporated various communication media and social features to support collaborative and distributed interactions.

In 1995, Ward Cunningham designed **wikis**—notably the WikiWikiWeb—as a medium for collaboratively editing software documentation [40]. From this technology, Wikipedia emerged in 2001. Wikis were innovative because they allowed authors to easily link between internal pages and include text for pages that did not yet exist [40]. Wikis have been used to support defect tracking, documentation, requirements tracking, test case management and for the creation of project portals [41]. Wikis are also used frequently in global software development [36] and remain integrated in collaborative and social coding sites [35].

In the free and open source world, **SourceForge**[5] has been widely used since 1999, and offers support for hosting projects and version control. A recent study by Guzzi et

[3] `https://campfirenow.com`
[4] `https://www.hipchat.com`
[5] `http://sourceforge.net`

al. [24] showed most of the communication about development issues occurred through the code repository discussion feature rather than email.

GitHub, launched in 2008, markets itself as a social code hosting site. GitHub uses Git, a distributed version control system for large-scale collaboration. For projects and teams, GitHub fosters collaboration through several awareness features, integration with external tools and support for asynchronous workflows (cf. e.g., Pham et al. [57]). It has recently overtaken SourceForge as the largest code hosting site[6].

In 2010, Treude et al. [71] investigated how a **community portal** for IBM's **Jazz** development environment is used to provide awareness of ongoing activities in a hybrid (open/closed) software development community. Jazz was designed as an IDE with collaboration in mind, and its **dashboards** and **activity feeds** played a major role in maintaining awareness—a necessary feature to support effective coordination.

Email lists have played an ongoing role in keeping members up to date with project activities. They have been used as a channel to disseminate commit logs from software repositories [23], supporting project awareness and coordination. They have also been used for asynchronous code review in open source projects by sending small patches to members for review [63, 62].

Gutwin et al. [23] studied communication in three open source projects and found that email lists supported information seeking, dissemination of project knowledge, as well as developer activities and project discussion. They further found that **chat**—in this case, IRC—was used for informal communication about project artifacts. Although not archivable, important aspects of those discussions would be siphoned off to the email lists. Modern team- and project-oriented chat systems used by development teams support the archiving of content and offer project support (such as Campfire and Gitter[7]).

Gutwin et al. also noted that important information about code was sometimes fragmented across the three channels used in those projects (email lists, chat and commit logs). Their study found that it was often difficult to ensure that information was read by the right people in a timely fashion. We suspect that fragmented communication may be an even bigger concern today given the increase in the number of communication channels used by many developers.

3.3 Communicating Knowledge Socially Constructed in Community Resources

Usenet was developed in 1980 as a "worldwide distributed Internet discussion system" and soon supported the flow of knowledge about technologies and projects across team, project, and community members. Users could read and post threaded messages, and posts were referred to as "news" belonging to specific categories or "newsgroups". It was a precursor to **forums** that are more widely used in recent years. Usenet differed from other **bulletin boards** at the time as it did not rely on a central server and a dedicated administrator: it was distributed among a network of servers that stored and forwarded messages to one another using "news feeds". But in other ways, it resembled the asynchronous text-based discussions on bulletin boards.

Wasko et al. [77], in 2000, studied how Usenet played a role in knowledge management in three development communities of practice. In their survey, they found various reasons for participation: tangible returns such as the speed in getting multiple answers to questions, staying up to date, interaction with the community, reciprocity, enjoyment, as well as enhanced reputation.

Web-based archiving of Usenet posts began in 1995 at Deja News, which was acquired in 2001 by Google; the archives are now accessible by Google Groups. However, **Google Groups** is more than a gateway to Usenet archives—it provides discussion groups, mainly used as forums and mailing lists for software developers, and serves as a collaboration tool for global software engineering [36].

Many of the features in Usenet can now be seen in more modern media—most notably, **Stack Overflow**, a question and answer Website. Stack Overflow was created in 2008 and has experienced rapid uptake. Even though Stack Overflow has many parallels to Usenet, it differs in several important ways. Firstly, there is moderation of both questions and answers, which improves the trustworthiness and value of the content. Secondly, Stack Overflow has a gamification [17] aspect with reputation scores and the ability to earn new powers through participation, which may encourage involvement through intrinsic and extrinsic motivation. Finally, the Stack Overflow community responds very quickly: over 92% of questions are answered within a median time of 11 minutes [42]. Recently, Stack Overflow has been studied by several researchers [42, 2, 14, 83] and is rapidly growing into a formidable documentation resource. E.g., Parnin et al. [54] find that it provides very good coverage for documentation on open source APIs.

Stack Overflow and many of the other channels mentioned in this section use a feature that was somewhat of a poster child for Web 2.0: **Folksonomies** [56], also known as collaborative tagging or social bookmarking, comprise a user-generated system of classifying and organizing online content into different categories by using metadata such as tags. What began with Delicious and Flickr in 2003 and 2004, respectively, soon spread to many other tools and Websites. In previous work, we studied the potential for **social tagging** in the IDE [68] and the use of tags for work item management in Jazz [70].

Blogs are another important community-based knowledge resource used in software engineering. First used in 1994 and widely adopted by 1999, with blogs, everyone can broadcast. They are frequently used by developers to document "how-to" information, to discuss the release of new features and to support requirements engineering [53]. Moreover, some practitioners advocate that every developer should have a blog[8], arguing that blog posts help exchange technical knowledge among a larger audience than email messages. Pagano and Maalej [52] examined both blogs and commit messages and found a relationship between blogging and commit behavior. Parnin and Treude [55] found that blogs play an effective role in documenting APIs: by analyzing the Google results

[6] http://redmonk.com/sograd y/2011/06/02/blackduck-webinar

[7] https://gitter.im

[8] http://bit.ly/Every-Developer-Needs-a-Blog

for API calls of the jQuery API, they found that 87.9% of the API methods were covered by blogs, mainly featuring tutorials and personal experiences about those API methods.

Audio and video **podcasts** are a medium closely related to blogs and used by software developers in a similar fashion: for learning [33], keeping up to date with the latest trends and technologies [82], for (job) training and as how-to guides. Podcast use in higher education has increased since 2005 [9, 6] and has been shown to be effective [33].

Social news Websites are another medium experiencing a recent surge in popularity. Many of these sites [75] and aggregators allow developers to disseminate knowledge, discover new software and keep up to date. Some of the most popular among developers are Digg[9], reddit[10], and Hacker News[11]. Lampe and Resnick [34] analyzed Slashdot[12], a precursor to modern news Websites, and found that the basic concept of distributed moderation works well. However, their analysis revealed that it often takes a long time to identify especially good comments, that incorrect moderation activities are often not reversed, and that non top-level comments and comments with low starting scores did not receive as much consideration from moderators as other comments did.

Lerman [39] examined the incentives that drive user participation on Digg and found competition with other community members, social acceptance, and internal factors. Findings also indicated that user participation is non-uniformly distributed with a few top users doing a large fraction of the work. Gilbert [21] studied Reddit votes and found a widespread underprovision of votes on Reddit, where half of the most popular links were overlooked the first time they were submitted, thus jeopardizing its main purpose.

The importance of news Websites like Hacker News for software projects is beyond aggregating news and keeping up to date, as reaching the top of a site can provide valuable feedback and help the growth of a project's users, contributors and the community as a whole. These sites are a specific form of *social navigation*, which was first defined by Dourish and Chalmers [18] as movement from one item to another when provoked as an artifact of the activity of another or a group of others.

Microblogging is another channel that plays an increasingly important role in curating community knowledge. Twitter, the first microblogging tool and one of the most popular social media channels, was created in 2006 as a way to share short messages with people in a small group. The idea was to share inconsequential ephemeral information, but it has become an important medium in many domains. Twitter is seeing significant adoption in software engineering.

Several researchers have investigated how software developers and software projects make use of Twitter [7, 76], finding that Twitter is used to communicate issues, documentation, to advertise blog posts to its community, as well as to solicit contributions from users. In a recent study [66], we found that developers who adopted Twitter use it to filter and curate the vast amount of technical information available to them. We found it brought benefits in terms of awareness, learning, and relationship building. Developers who feel that Twitter benefits them rely on a variety of strategies for posting and reading Twitter content. Twitter shares some commonalities with Usenet, in that people join Twitter to discover information, to stay up to date and to be part of the community. Yet, Twitter is also quite different from newsgroups. With Twitter, users follow individuals, but create the community they wish to be a part of themselves. Twitter also includes a few signals that can help assess content and people, such as a user's number of followers or the number of retweets for an individual tweet.

Social media is not only a disruptive force for developers, but also for enterprises. However, applying social media concepts in enterprise environments raises various concerns (e.g., privacy, security, distractions, ownership, etc...), which has led to the emergence of Enterprise Social Media [38]. These are social media adjusted for use in enterprise contexts that try to mitigate some of the concerns. **Yammer**[13] is a microblogging service much like Twitter, but designed for corporate use. Zhang et al. [84] investigated how employees of a large enterprise use Yammer and how its usage differs from Twitter. They found that employees use Yammer for news about groups and less for posting content about themselves. Dullemond et al. [19] also studied the use of a microblogging tool within a small distributed software company where moods are communicated explicitly as part of posts.

3.4 Communicating Knowledge About Developers Through Social Networks

Sites that are primarily focused around social networking features are becoming very popular in development communities—just as Facebook has been very popular with the general population. This kind of media is new in software engineering, although implicit social networks have been present in other media channels, such as through newsgroup or mailing list affiliations. However, with these new media, the affiliation is one of the primary features of the medium, and as such, is made visible. **Facebook** is used by some developers to support programming interactions, such as local meet-ups. Sites such as **LinkedIn** are more focused on professional connections, but Barzilay et al. [2] explored the practice of example usage in software development by observing software developers' exchanges in LinkedIn discussion groups.

There are also specific sites designed to showcase developer activities and skills, such as **Masterbranch**[14] and **Coderwall**[15], while many other developer tools now also integrate social networking features, such as **GitHub**. Below, we summarize some recent studies that have studied the impact of social networking on software engineering.

Dabbish et al. [15] conducted a study of GitHub users. They found that the visibility of developer activities and profiles on GitHub influences a project's success by motivating others to contribute. Developers further decide

[9] http://digg.com
[10] http://www.reddit.com
[11] https://news.ycombinator.com
[12] http://slashdot.org
[13] https://www.yammer.com
[14] https://masterbranch.com
[15] https://coderwall.com

which projects are worth contributing to by determining which projects have frequent contributions or contributions from high status developers [72]. High status developers are those that have already done interesting work.

In a similar study, Marlow et al. [44] explored how users on GitHub form impressions of others through GitHub's transparency. They found that developers assess each other to learn about projects and their status. These impressions guide future developer interactions and influence how much one would trust contributions from someone they had not interacted with before. Pham et al. [57] interviewed GitHub users with regard to testing practices and found that when assessing contributions from others, project owners behave differently based on how much they trust the contributor. Unknown developers' code would be more thoroughly scrutinized than code from trusted people.

Marlow and Dabbish [43] studied how employers assess developers based on their GitHub profiles, which provide cues about their activities, skills, motivations and values. These cues were regarded as more reliable than resumes. Employers who were interviewed by Marlow et al. said that open source contributions indicated that a developer had the "right" set of values and was not in software development for purely financial reasons. Also, contributions to high status projects were said to demonstrate "some level of proficiency."

Singer et al. [65] conducted a similar study, but investigated social media profiles of developers in general. This included GitHub, profile aggregators such as Coderwall and Masterbranch, Twitter, Stack Overflow, and other sites. They found that profiles can be hard to understand or evaluate for some recruiters. To assess the real merit of a developer's public activities, one would always need to look at actual artifacts, such as code, tests, documentation or discussions. Developers also assessed recruiters and companies, trying to gauge their authenticity and suitability as colleagues. Most developers and recruiters were aware that the absence of signals does not mean much—a developer with few or no public activities might just be a private person or work exclusively on closed source projects.

Capiluppi et al. [10] further argue that information that helps assess others will allow for more merit-based and less credential-based evaluation of potential candidates, possibly leveling the playing field for those who don't have access to traditional education but are skilled and desirable developers.

Together with Begel and Bosch, we interviewed representatives of GitHub, the **Microsoft Developer Network (MSDN)**, **Stack Exchange** and **TopCoder** about the role of social networking in their companies [3]. All four sites provide features to increase the visibility of developers and developer activities. On GitHub, one can follow particularly high status developers and watch their activities. MSDN and Stack Exchange have explicit gamification elements to assist in assessment and discovery of content, and to support strong engagement. Interestingly, not having explicit connections between users was a conscious decision for Stack Exchange: the company feels users should concentrate on the content, and status should not play into a question or answer being up-voted. The Stack Exchange interviewee mentioned that the hardest part about building the Stack Exchange Network was not creating the software that runs it—the challenge was building the community, its mechanisms, and the processes that support it. Similarly, the TopCoder[16] representative also mentioned that building a functioning community was the most difficult aspect in building the business. Both GitHub and Stack Overflow (the most notable of the Q&A sites Stack Exchange hosts) relied on high status developers to seed their communities.

In summary, the pervasive use of public social media by developers makes the participants and their relationships more transparent to software engineering researchers who want to understand how methods and tools are used in practice. While the mining of software repositories has given us access to software products for a relatively long time, the recent accessibility of developers' interactions and their relationships can provide more qualitative insights than had previously been available. Research so far has indicated that social networking features influence which projects developers participate in, impact project success, and play a role in developer assessment and recruitment. Gamification elements add feedback and improve trust in developer contributions. We believe studies on social networking and their associated gamification aspects will become more prevalent in the future of software engineering research, adding to these preliminary insights.

3.5 Summary and Discussion

The timeline (see Figure 1) shows an approximation of when some popular tools and media channels were adopted or innovated by software engineers. The timeline gives a rather simplified view of what is really a very complex history of how media channels evolved and were appropriated in software engineering. The channels are organized using the different types of knowledge exchanged over time (knowledge in developer heads, knowledge embedded in project artifacts, knowledge embedded in community resources and knowledge about or communicated through social networks). Note that some tools overlap multiple types of knowledge, and a single tool may encompass multiple social features within them. E.g., IDEs such as Jazz support Web portals, dashboards, tagging and newsfeeds; Github, while primarily a code hosting site, has social networking features. Consequently, it is not always possible to distinguish a tool for a channel from a feature of a channel: e.g. tagging is a feature of many channels, but it is also a primary activity on sites such as Pinterest or Delicious.

Over time, we see the emergence of channels that are increasingly socially enabled, and a rise in the adoption of channels for exchanging community knowledge and social networking content. The timeline highlights how we have entered a **social era** in terms of media use in software engineering. We loosely categorize tools as 1) non-digital, 2) digital or 3) digital and socially-enabled. Distinguishing digital from non-digital is trivial, but characterizing tools as socially-enabled is not as clear cut. This is particularly the case for tools (e.g., SourceForge) that have evolved over time with the addition of social or social networking features. By social, we refer to features that promote and afford participation through the transparency of identity (e.g., user profiles), content and interactions [15]. This transparency further leads to social signals that themselves

[16] `http://www.topcoder.com`

Non-digital
Face-to-face, Books and Magazines
Digital
Web Search, Content Recommenders, Rich Content, Private Discussions, Discussion Groups, Public Chat, Private Chat
Digital and socially enabled
Feeds and Blogs, News Aggregators, Social Bookmarking, Question & Answer Sites, Professional Networking Sites, Developer Profile Sites, Social Network Sites, Microblogs, Code Hosting Sites, Project Coordination Tools

Table 1: The channels respondents were able to choose from in our survey. The survey included examples for each channel.

become useful content (e.g., the number of votes associated with an answer or the number of followers a user has).

We can expect to see even more software development tools become socially enabled by the inclusion of features such as commenting, tagging, like buttons and the ability to follow others. But adding these features to a tool is not sufficient to promote community participation. As the interviews with Stack Overflow and TopCoder have shown, building software for communities is much easier than building the communities themselves. It may be more likely that new social tools that we can't yet anticipate may lead to a disruptive change in how developers work.

The adoption of social media is relatively new and has taken some parts of the industry and research community by surprise. Although we have some insightful research findings about the implications of this paradigm shift on software engineering, we lack insights on how various media channels are used in combination to support software engineering activities. That is, we need more understanding about the landscape of social media use within software engineering ecosystems before we lay out a roadmap for future work.

4. DEVELOPER SURVEY 2013: EXPLORING MEDIA USE

As discussed, there has been some research on how the recent generation of developers use media to support software engineering. But most of these studies considered how just one or two channels impact software development activities. One exception is a pilot survey by Black et al. [5] conducted in 2010 which asked about the use of a broad set of channels in software engineering. To try to understand how the newer generation of developers use a broad range of social media channels, and thus to anticipate important opportunities and challenges for a roadmap for future work, we conducted a survey with members of a thriving participatory community through GitHub.

4.1 Method

We emailed our survey[17] to 7,000 active GitHub users in November and December 2013. 1,516 developers responded (21% response rate). We asked them about demographic

[17]URL to the survey: `http://thechiselgroup.org/2013/11/19/how-do-you-develop-software`

Channel	Reasons for Importance
Code Hosting	Provides the state-of-the-art tools and lowers the barriers for collaboration. 1018
F2F	Eases team communication, facilitates problem solving and knowledge sharing. 512
Q&A	Provides rapid access to answers and code examples produced by the crowd. 496
Web Search	Immediate access to vast amounts of information, supporting learning and problem solving. 429
Microblogging	Allows for keeping up with new technologies, practices and people. 221
Private chat	Single point of coordination for remote teams, supporting real-time collaboration. 194
Feeds and blogs	Provides personalized information for developers to keep up with latest practices and technologies. 190

Table 2: The top channels mentioned and why they are important (bars show # of respondents selecting that channel as most important, bar color indicates analog, digital or social channels).

information such as gender, age, and location, their programming experiences, technologies and tools they use, and asked them about the number of projects they participate in, whether they program professionally, and about the sizes of project teams they had worked with.

Findings from our literature review and from earlier studies we conducted prompted us to find out which media channels developers use for activities such as **staying up to date**, **learning**, **connecting with other developers** or **coordinating**. For each of those tasks, we asked them to select the channels they use for them. Respondents were able to select from those shown in the columns in Table 1. We asked the respondents to indicate the three most important channels that they use to support their software engineering activities, and to tell us why they selected those channels. We also asked respondents how **overwhelmed** they feel when using the tools selected, whether they have **privacy** concerns when sharing information on those channels, whether they suffer from **distractions**, and what other **challenges** they may experience.

Limitations: We recognize that our population of respondents is biased towards developers that firstly use GitHub, and secondly that they are more likely to use social tools since GitHub is also a social tool by design. That said, this is the population we wished to study in our survey. From the survey responses we were able to determine that developers on GitHub further tend to develop programs for the Web as opposed to systems

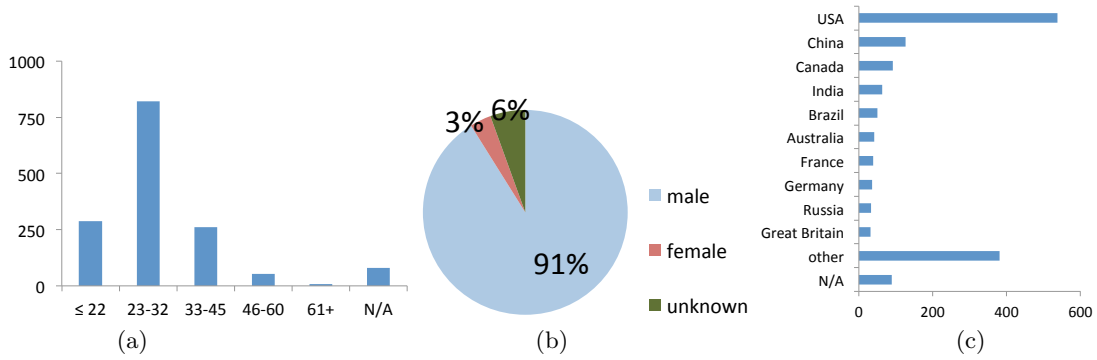

Figure 2: Background of the respondents: (a) age, (b) gender and (c) location

programming, adding another bias to the responses. Some of the questions may have been ambiguous especially to non-native English speakers. Finally, we only present some preliminary findings in this paper; future work may reveal additional insights.

4.2 Preliminary Findings

An in-depth analysis of the survey findings is forthcoming; in the meantime we provide preliminary data to help us understand the kinds of media channels this population uses to communicate and collaborate. Figure 2 shows an overview of the respondents' backgrounds. Figure 2(a) shows the age distribution, which is strongly skewed towards relatively young developers. Figure 2(b) shows that the overwhelming majority of our respondents were male—only 3% identified themselves as female, while 6% did not answer this question. Finally, Fig. 2(c) shows that many respondents were from the USA, but there was also a wide distribution among other unspecified countries.

Calculated over all survey questions that asked about channel usage, our respondents said they use a median of 12 unique channels (mean: 11.55, min: 1, max: 21). While 25% of the respondents use between 14 and 21 unique channels. Table 2 shows the top channels mentioned by respondents and summarizes the main reasons for importance that were given. The bars indicate how many developers named the channel as one of their three most important ones for development. While code hosting sites may be in the lead due to the GitHub-centric population we surveyed, it is surprising that face-to-face communications are still considered very important. Q&A sites are considered almost equally important.

Survey respondents considered ***Code Hosting Sites*** the most important channel, as they provide **state-of-the-art tools** and **lower the barriers of entry** for developers willing to **share their work** with others and **get feedback**. Respondents also valued being able to access other developers' posts to **learn about new technologies and practices** in software engineering.

Despite the fast pace of evolution in communication technologies and tools, we found that ***face-to-face*** communication is still considered the best channel for supporting **team communication** and **effective collaboration**. Olson and Olson [50] emphasized the role of synchronous interactions in providing rapid feedback among team members and in supporting design and collaborative problem solving.

Q&A sites such as Stack Overflow and ***Web search*** engines (e.g., Google), play a crucial role in providing **access to a vast amount of information** produced by the crowd. ***Microblogging*** adds another layer of social interaction by supporting **relationships, discovery, and awareness of people, trends, and practices**.

Private chat provides an important alternative to face-to-face interactions as it enables **real-time collaboration** and **a single point for coordination**, but still falls short at supporting features that are present in collocated interactions, such as spatiality of reference and shared local contexts. ***Feeds and Blogs*** are a particularly important medium for conveying **personalized information** and for **communicating developments and ideas** about software engineering concerns, practices, and technologies.

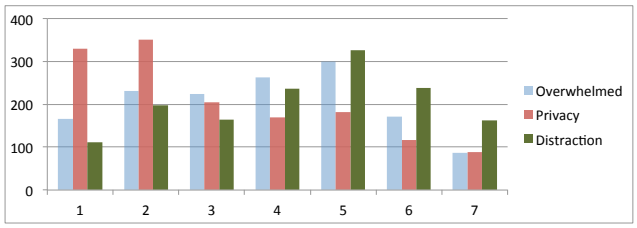

Figure 3: Results from the Likert-type scale questions (where 1 was strongly disagree and 7 was strongly agree) about three different challenges.

Figure 3 shows a summary of responses to the Likert-type scale questions asking whether developers are **overwhelmed** by the information communicated on the channels, whether they are concerned about **privacy**, and whether they are **distracted** by the channels they use. The majority of developers seem to feel overwhelmed and find that social tools can be a distraction, while privacy appears to be of less concern.

Table 3 provides qualitative data on the challenges most often mentioned in the survey, all of which will be discussed in more detail in Sec. 5. From the perspective of individual developers, **media literacy** was often mentioned as an important challenge. Developers have to learn the conventions and technical nuances of new media that appear from time to time and have to learn how to use them effectively with their teams—who may not always adopt the new channels.

	Developers
Literacy	*"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."*
	Communities
Anti-social behavior	*"Misinformation is easy to communicate and propagate. People can be rude or obnoxious on social mediums, distracting from a discussion. The asynchronous nature of social media interaction can often lead to missed information or incomplete contexts for understanding information."*
	Content
Trusting content	*"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."*
	Ecosystem of Media Channels
Vendor lock-in	*"I worry that we are relying on many of these 'free' services, which in the end are not free – they simply have a different payment model (that appears to change)."*

Table 3: Some key challenges reported in our survey.

In communities, **anti-social behavior** was mentioned as a constant threat to smooth interaction. Many media have either a low bandwidth or are asynchronous, so misunderstandings are more likely to happen. The perceived anonymity on the Internet may also play a role.

Regarding knowledge, knowing when and whether to **trust content** is regularly an issue. While sites like Stack Overflow have voting mechanisms in place that help surface the high quality content, niche technologies might simply have an adoption rate that is still too low to yield reliable trust signals. Other sites may lack explicit signals altogether, so developers are forced to interpret surrogate signals based on their experience and intuition.

From the perspective of many interrelated media channels as an ecosystem, several respondents were concerned about **vendor lock-in**. Developers are by now aware that free services will always have a high potential for downsides, be it possible discontinuation or marketing-related changes to a business model.

4.3 Discussion

On average, our respondents indicated that they use 12 channels to support their software engineering activities. We see that some recent newcomers in terms of communication channels used in software engineering see widespread adoption, and that those tools are also among the most important as selected by the respondents.

Although our survey respondents are those that work on projects mediated through GitHub, face-to-face interactions were also rated very highly. In future work, we will analyze the results from the survey examining possible correlations between channels used and developer factors. A preliminary interactive visualization—a work in progress—of the results can be viewed online[18]. We use some of these initial results and combine them with our retrospective of software engineering media research from Sec. 3 to build a roadmap for future work.

5. FUTURE OF SOCIAL MEDIA IN SOFTWARE ENGINEERING

We build on existing research into social media use in software engineering (Sec. 3) and the preliminary findings of our survey (Sec. 4) to develop a roadmap for future work. We pay attention to McLuhan's guidance (see Sec. 2.1) as he suggests we consider not just how media enhances, replaces and retrieves elements of what has come before, but also that we pay attention to what it reverses into, if taken to the extreme. We consider opportunities and challenges that relate to the following aspects (following Postman, see Sec. 2.1):

- the **developers**;
- the **communities** the developers belong to;
- the **content** and **knowledge** that is created by the developer community; and
- the **ecosystem of media channels** that shape the community and content exchanged.

5.1 The Emergence of the Social Programmer

Developers participate in communities of practice because they wish to directly contribute to the community's public good or to form relationships with other developers. They also participate for more personal reasons—in particular, to improve their technical skills and positively influence their career prospects. However, there are challenges in managing interruptions and acquiring social media literacy skills.

5.1.1 A Living Laboratory for Learning

Before the innovations of the Internet and social media, most learning occurred in the classroom, through self training or through training on small teams [78]. These days, developers acquire extensive skill-sets by participating in online communities that showcase current technology trends. This was particularly evident in the studies on Twitter (cf. 3.3). Networking allows developers to acquire diverse skills by connecting with different groups and communities that suit their learning needs or interests. **Novices** can seek out like-minded developers as learning partners or mentors. Social coding sites that support open source projects, such as GitHub and Twitter (e.g., through the #pairwithme tag on Twitter), provide mechanisms to make those connections.

Sites such as GitHub provide rich opportunities for use in **university courses**—Arie van Deursen discusses how he successfully used GitHub in a software architecture course[19]. Furthermore, **online learning environments**, such as Khan Academy and Coursera, provide additional mechanisms to foster self-paced learning in online communities. Jenkins reminds us that *"appropriation enters education when learners are encouraged to dissect, transform, sample or remix existing cultural materials."* [30]

[18] `http://fose2014.thechiselgroup.org`

[19] `http://avandeursen.com/2013/12/30/teaching-software-architecture-with-github`

We already see the effects of this kind of encouragement in social coding sites. Social sites are likely to revolutionize how software engineering is taught in the classroom.

Future work should consider how developers use tools such as GitHub and Stack Overflow in conjunction with other channels to support their ongoing learning, as well as how their use impacts the quality of the software they build. From an enterprise perspective, how can organizations support this mode of learning and balance learning with short term productivity goals? Are developers joining the best communities to support their learning needs, and are they choosing to learn the most suitable development technologies and learning those in an effective way? Are they using their time effectively?

5.1.2 Expanding Career Opportunities

As we discussed in Sec. 3.4, understanding the impact of **transparent participation on career development** is a current research focus. The participatory development communities that we see forming through tools such as Twitter, GitHub and Stack Overflow, as well as developer profile aggregator sites such as Masterbranch and Coderwall, provide further mechanisms for developers to advertise their skills and experience, as well as for employers and recruiters to seek out suitable employees or team members. However, there is still much to learn about these mechanisms, as one developer responded to our survey: *"It can be challenging to understand which tools to invest time in, and how to use them to best promote oneself to further one's career."* This phenomenon is relatively new and its impact on developer behaviors and team churn have yet to be studied in depth.

5.1.3 Maintaining a State of Flow

DeMarco and Lister in their Peopleware book discuss the importance of maintaining a sense of flow during development [16]. Flow refers to a psychological concept of being fully immersed in a task, where the participant barely notices the passage of time. Achieving a sense of flow requires a balance along an anxiety-boredom continuum. If the task is too easy, the individual will be bored, and if too difficult, they will be stressed. DeMarco and Lister worried about the telephone or email interrupting developers' sense of flow. Interrupting flow is potentially even more of a concern given the many channels developers use today (see Sec. 4). Many of the survey respondents discussed feeling overwhelmed and distracted by the vast number of **notifications** they received and felt compelled to respond to notifications. Although it is often possible to customize notifications, every channel does this differently. On the positive side, gaining quick access to information may improve flow by **reducing anxiety**. More research is needed to understand how developers use these channels and why some developers struggle while others succeed at finding a **sense of rhythm**.

5.1.4 Social Media Literacy for Developers

As some of our survey respondents told us, learning how to use social media effectively can be a challenge (cf. Sec.4). But if a user has the right skills, social media can promote critical thinking [12] and social networking can improve a developer's ability to search for, synthesize and disseminate information [30]. The respondents in our recent Twitter study shared strategies they use for curating their social graph and improving the value they receive from using Twitter (see Sec. 3.3). Effective use of social media can also reduce unnecessary communication. We heard from expert developers that they explicitly post answers to commonly asked questions on blogs or answer questions on Stack Overflow to reduce their email volume.

Jenkins suggests a set of social media skills for users that include [30]: **transmedia navigation**, the "ability to follow the flow of stories and information across multiple modalities"; **appropriation**, the "ability to meaningfully sample and remix media content"; **multi-tasking**, the "ability to scan one's environment and shift focus to salient details as needed"; and **collective intelligence**, the "ability to pool knowledge and compare notes with others towards a common goal." Soft skills are typically not taught in university, but are learned "on the job" or by participation within communities of practice. Do Jenkins' recommendations for social media literacy also apply to software development?

5.2 Participatory Development Cultures

Social technologies have enabled participatory cultures to flourish beyond the constraints of time, space and group [12], and we can clearly see this influence on the global software development community. There has already been some research on communities of practice in software development (see Sec. 3). Here we propose future work on increasing the scale of who collaborates, and identifying new practices that are being shaped and enabled by social technologies. We also raise the importance of being aware of the potential for anti-social behaviors and other barriers to participation.

5.2.1 Increasing the Size of the Crowd

Social media exposes new customers, partners and users to one another, and social tools like GitHub are designed with participation in mind. Although GitHub is designed to encourage participation, **non-technical users** still reportedly struggle with some of its features. Effective **technical** and **social media literacy** skills will help facilitate participation, but strong **leadership** is also important [12].

The software industry further benefits from crowdsourcing (participation and feedback from a large number of users or stakeholders) across customer care, testing [32], documentation [54] and requirements gathering [51]. Research into some of these relatively new phenomena indicates a lack of tool support for crowdsourcing activities in software engineering [54, 51].

Startup companies understand the importance of releasing "minimally viable products" early to gain iterative and valuable customer feedback [61]. But which **channels** are the most effective for reaching users and gaining their feedback and participation? What are the **barriers** that may block or turn some potential collaborators away? What kinds of new **tools** can entice, improve and capture crowd-based participation in software engineering? How do we ensure crowd diversity, as a "smart crowd" depends on having diverse skills and viewpoints [69]?

5.2.2 Discovering Next, Not Just Best Practices

Our Developer Survey revealed that shared knowledge and information might be spread across too many channels and can be easily missed or impossible to find again (especially with ephemeral channels). Survey respondents recognized the need to use multiple channels but were frustrated that others were reluctant to use the same channels or used them inappropriately (e.g., tweeting rather than emailing). The issue of private versus public posting also led to knowledge fragmentation. However, missing out on some discussions is not a new problem: from the earliest days of software engineering, conversations around the water cooler always meant some participants were not privy to all relevant discussions. Nevertheless, articulating and sharing communication **best practices** would alleviate some of the issues from **knowledge fragmentation across multiple channels**.

But following best practices may not be enough [12]. As new tools are rapidly adopted, appropriated and innovated by developers, **new practices** will quickly follow. The emergence of new practices is driven by both the transparency and speed of technology diffusion. For example, we were able to observe how developers learn **testing practices** through the transparency of GitHub [57], as well as how social technologies support the **matching of talent to task** in the case of peer review [63]. Kazman and Chen observed that some crowdsourced development can be described using the metaphor of a "metropolis" [32]. What other **new development models** and practices are emerging from this participatory culture?

5.2.3 Knowing When Social Becomes Anti-Social

The transparency of social media may be highly effective at encouraging shared creativity, but the same transparency can also be intimidating to more private or shy individuals. One concern that was raised in our survey is the risk of **ambiguous communication** even when the parties speak the same language. Many expressed that face-to-face communication is still the preferred mode of working when possible.

Many of the Developer Survey respondents also expressed worries about being **personally attacked** if they say something inappropriately, advise poorly or inadequately explain something. In such situations, communication on social channels may alienate people, leaving them more reluctant to share. Social media also makes it easy to communicate directly with participating stakeholders, but sometimes the communication may be **invasive** (e.g., users asking too many questions, or unsolicited contact from recruiters). Another issue is the poor articulation of **community expectations** concerning when communication should be acknowledged or addressed.

There are other long-standing differences that may lead to complex social issues. Exclusion from a community may result from poor **natural language** proficiency (English is the language of choice in many channels), **time zone** differences, **cultural** issues and belonging to certain **minorities**. E.g., female participation tends to be low on many of the more transparent social channels [74]—we also saw this in our survey (cf. Sec.4). Understanding how these differences can be addressed is paramount.

5.3 Software Knowledge as Public Good

Social media is the underlying fuel for collaborative development and crowdsourcing of content in software engineering. The content created in a community of practice is referred to as "public good" by Wasko et al. [77], and in this case, such content refers to not just **code**, but also **documentation**, as well as **content about developers** (their skills, accomplishments and activities). Social media further transforms communications into content [12]. In this section, we consider how this content is a source of big data which can be analyzed for making predictions and recovering traceability in software projects. We also look at the opportunities that arise from the speed of communication and technology diffusion made possible by the Internet and social media. Finally, we consider issues with finding salient information from a mass of communication channels, as well as issues with trust.

5.3.1 Mining Social Media in Software Development

Mining of software repositories has been an active area of research in the past decade[20]. That research has also considered other communication channels, most notably email archives as email lists are frequently used for exchanging software repository artifacts [47, 48, 4]. There are even more analysis opportunities of the more recently adopted social media channels used in software engineering (e.g., Twitter and Stack Overflow). Tools that integrate multiple channels, such as GitHub, make it easier to relate information that was previously stored in silos and had to be linked to facilitate analysis. Many researchers are using these richer resources to support new lines of research inquiry about development (e.g., the MSR challenge for 2014 is based on GitHub projects[21]).

The output from these analyses show potential for improved **predictions** about software success, failure and reliability, improved **traceability**, monitoring support for project health and adoption, and finally, insights about developers' own activities [65]. Aggregations of communicated information also show promise at forming **documentation** and learning resources [55, 54]. At the same time, this newly available data brings with it new challenges that require the attention of researchers [31].

5.3.2 Software Engineering at the Speed of Light

The adoption of social media has occurred at a much faster rate than any previous communication technology [12]. These channels facilitate technology diffusion that self-accelerates adoption. This rapid diffusion may lead to a competitive edge for some (faster access to new technologies and practices, ongoing feedback, and access to collaborators and users [66]). But what effects does this have on **software quality**? How does **diffusion** and **discovery** occur across an ecosystem of channels? Which **combinations of channels** and work practices should be used or avoided? How can organizations and developers effectively **stay up to date** and learn in such a rapidly changing landscape? What challenges will faster technology diffusion and adoption bring in the future in terms of **maintenance and evolution**?

[20] `http://msrconf.org`

[21] `http://2014.msrconf.org/challenge.php`

5.3.3 Finding the Signal in the Noise

Many-to-many communication that occurs through multiple channels inevitably leads to a mass of information distributed within a community. Many respondents to our survey reported feeling overwhelmed by the amount of information and had trouble finding or discovering salient information. The developers that are social media literate and have carefully curated their social graph seemed to suffer less from this problem. Finding documentation that relates to a particular technology version was also a reported problem, particularly on GitHub due to what has been termed "forking hell"[22]: as GitHub makes it very easy to fork a project and some of a project's many forks could include crucial improvements not yet merged back into the main project, it can be hard sometimes to find the ideal fork to use. Improved **search tools** and **recommenders** could assist with this challenge[23].

5.3.4 Trusting Content

A concern that was raised numerous times in the Developer Survey is that of assessing the quality of the content available on social media channels. In terms of developer skills, there were concerns that some developers may falsely **embellish their skills**. For crowdsourced documentation, although widely thought to be powerful, there were concerns that it may be **out of date**. There were also concerns with having to understand multiple conflicting documentation resources and which source to trust, especially if there was insufficient information on the people posting the documentation. Although there are mechanisms for showing trustworthiness (e.g., voting up on Stack Overflow), perhaps relating these to a developer's social graph (i.e., improved traceability) may improve one's confidence on the trustworthiness of content. However, some of the **gamification** mechanisms that are used on some sites (such as on Stack Overflow) may lead to content that is not as useful or authentic if the size of the crowd curating it is rather small.

5.4 Improving the Social Media Ecosystem for Software Engineering

One of the most interesting findings from the Developer Survey is the number of channels some developers use today. Respondents reported using an average of 12 channels, with 25% using between 14 and 21 channels. Recently we see the development and adoption of channels that are highly specialized (e.g., Stack Overflow for Q&A), and yet we also see the integration of multiple channels or features in a single tool (such as the case with GitHub). In this section, we report on some opportunities for new channels and features that would be more applicable to software developers, as well as discuss challenges that arise from relying on proprietary channels and the issues that may arise if social media are overused or pushed to the extreme.

[22] `http://bit.ly/dada-beatnik-forking-hell` and `http://zbowling.github.io/blog/2011/11/25/github/`

[23] There are workshops on both of these topics at ICSE 2014, see `http://2014.icse-conferences.org/workshops`

5.4.1 Social Media Channels for Developers

In our survey, some respondents indicated poor support for explaining design concepts or algorithms in the code. Several other respondents also mentioned issues when the tools they use lack integration with programming artifacts. This lack of support caused friction when switching between tools and led to poor traceability between discussions and software artifacts. Some channels are specifically designed with development in mind (e.g., Stack Overflow and Hipchat), but other channels lack facilities for sharing and discussing code. To reduce friction from switching between tools and integrated content, some researchers have proposed integrating **social features in the IDE** (e.g., integrating microblogging in the IDE [25, 60]). At the same time, there is momentum to migrate the **IDE towards the Web** [73, 22].

Whatever the form of improved integration, we anticipate that the coding tools and environments developers use will become more social. Understanding the **affordances** that the different features can offer to software developers is important, but we also need to develop **theories** about ideal combinations of tools. Achieving an effective media ecology, according to McLuhan *"means arranging various media to help each other so they won't cancel each other out, to buttress one medium with another."* Likewise, in software development we need research to understand how to package or integrate the best tools for developers of the future so that they not only have the most appropriate communication tools available, but also that the tools together address the knowledge fragmentation issues that may occur.

5.4.2 Reducing Risks from Vendor Lock-in

Another concern raised in our survey is that **mission-critical knowledge** can become "trapped" in **proprietary tools** that are outside developers' control. If such proprietary sites go down (or disappear completely), this knowledge could become inaccessible or lost. Nicholas Carr further discusses the problem of relying on online information resources when he questions *"is Google making us stupid?"* [11]. We found from the survey that developers are heavily reliant on Google as it directly links to other resources, such as Stack Overflow and Twitter. Many of our respondents talked about the importance of these latter resources and their concern that the community assets embedded in them might not always be accessible. Although there are data dumps available, those require effort to use. As new channels are adopted, what will happen to content stored on old channels? The same lock-in concern also applies to the valuable **profile information** within these and other proprietary sites that developers rely on for their portfolios.

5.4.3 Exploring the Dark Side of Social Media

As McLuhan reminds us, every media can have negative effects when taken to the extreme or abused. Therefore, what are the downsides of using social media in software engineering? While technology diffusion may seem like a positive outcome of social media use, is the **mass of technology niches**, such as special purpose programming languages, components and tools, that social media supports ultimately good for the industry, or does it lead to maintenance issues in the future? And is there a risk of

the same channels being used to diffuse **harmful content**, such as spam or information that will harm security?

Some survey respondents discussed a need for improving **bounded transparency** in social media channels. Communications that should be private or anonymous are posted publicly, and vice versa. For example, such a **context collapse** [45] could negatively influence future job opportunities. Several survey respondents mentioned their desire for electronic channels that are truly ephemeral (whereas much communication in current social channels forms a **digital tattoo**). Unfortunately, many channels either lack support for bounding transparency, or if they do have support, some users do not know how to use those privacy features effectively. Some survey respondents expressed fear of misusing channels as they were unsure how publicly or widespread their posted information would be diffused.

Another tradeoff arises when users make use of free services in exchange for their personal information. **Identity linking** across networks (which may or may not be desirable) is also a challenge in how people use social media in knowledge work [12]. We anticipated that developers would be concerned by these and other **privacy issues**, but the answers to our survey revealed that the majority are not as concerned as we had thought.

Finally, it is important to note that many developers do not openly participate in online communities because they do not have a need to participate. Scott Hanselman labels such developers as *"Dark Matter Developers"*[24]. But perhaps dark matter developers use many of the resources (that is, "lurk" there), but don't leave a trace. Does the community miss out when they do not participate?

6. IMPLICATIONS FOR RESEARCHERS

"It is the framework which changes with each new technology and not just the picture within the frame." [46]

McLuhan reminds us that when media or technology change, we must adapt how we study that technology and the questions we ask. In the case of our research, we found it was necessary to **study social media using social media**. Understanding the culture and nuances of social media language enabled us to first reach out to and connect with study participants, and then understand what they had to tell us.

We attracted approximately 1,500 responses to our survey (over 21% response rate) by reaching out to GitHub developers[25], and by informing our participants that our results would be openly shared as we have done with other studies—e.g., we tweeted and blogged about the results from our study of Twitter [66] and received excellent **feedback** that helped us validate our findings and add additional insights[26]. The GitHub participants we connected with enthusiastically answered our survey and gave us excellent insights, even with a rather long and tedious survey (see Sec. 4). One respondent mentioned they felt they were contributing to their own well-being by participating in our research: *"Good to see a survey on this topic. It is wonderful to be part of a global developer movement and have the entire world of developers helping each other."*

Chui et al. note that researchers are generally slower to adopt social media than knowledge workers [12]. This may put some researchers at a disadvantage as we speculate that social media is poised to cause a paradigm shift in some aspects of research, particularly in research that involves socio-technical aspects of software engineering. One opportunity that is however evident for researchers is that there is an increase in **openly available data** that researchers can analyze and use to publish.

Many software engineering researchers have already established excellent social media literacy skills. They use their social graph to gain research data and insights, and to form **alliances with developers** and **collaborations with other researchers**.

Furthermore, blogging about research results can lead to thousands of views in a day, gaining public feedback from real practitioners[27]. Thus, we suggest that social media can have a **transformative impact** on software engineering research, as researchers have the opportunity through social media to influence and guide the industry. As a research community we should ask ourselves whether we want to focus our energies on describing what is happening, predict what happens next, or strive to influence the future.

7. CONCLUSIONS

From the very early days of software engineering, software developers have used, innovated and adopted media to increase their social interactions with other developers. The social interactions that were possible dramatically changed with the innovation of the Internet, and communities of practice quickly emerged that went beyond co-located developers or small networks connected by leased lines.

The more recent innovations in social media have led to yet another paradigm shift in software development, with highly tuned participatory development cultures contributing to crowdsourced content and supported by media that has become increasingly more social. Social features are being layered on top of traditional channels, but we also see the galloping and widespread adoption of new social tools for developers, such as GitHub, Stack Overflow and developer social networking tools.

The most recent generation of developers, in particular millennials[28], are both used to and expect collaboration. This newer generation of developers are adept at using social media for communicating, coordinating with others, and for learning. They are by nature open, transparent and expect to share. They are tightly coupled to their devices and to their content, and they tend to care more about their public image than their corporate or company-internal image. It is also anticipated that millennials will change their jobs many more times than older generations, and they have fostered the ability to learn new technologies easily. Through the widespread

[24] `http://www.hanselman.com/blog/DarkMatterDevelopersTheUnseen99.aspx`

[25] We used the email addresses they had published on GitHub

[26] `http://blog.leif.me/2013/11/how-software-developers-use-twitter`

[27] `http://blog.ninlabs.com/2012/05/crowd-documentation` and `http://blog.leif.me/2013/11/how-software-developers-use-twitter/`

[28] `http://en.wikipedia.org/wiki/Millennials`

adoption and appropriation of social media by this generation, we see a participatory culture emerging in software engineering.

Tools such as GitHub lower the barriers to joining and contributing to development communities, and developers do so because they care about social connections with other developers. They wish to mentor or be mentored by others and they believe their contributions matter. Participation in these communities provides opportunities for improving their skills, and they know how to use their social connections to discover and learn about emerging technical trends.

The number and scale of teams and communities that developers belong to has radically increased since the early days, and the breadth of media channels they use to participate in those communities has likewise increased. The participatory development culture has become a network of tightly coupled ecosystems consisting of developers, communities of practice, shared content and media channels. The adoption of social media and the resulting paradigm shift this has caused opens up many interesting and exciting future research opportunities.

8. ACKNOWLEDGMENTS

We thank Cassandra Petrachenko for editing support, Daniel German for stimulating discussions and feedback on our paper, and Jim Herbsleb, Elisabetta Di Nitto and Alfonso Fuggetta for detailed suggestions. Finally, we thank the survey respondents for sharing their insights.

9. REFERENCES

[1] A. Archambault and J. Grudin. A longitudinal study of facebook, linkedin, and twitter use. In *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI '12)*, pages 2741–2750, NY, USA, 2012. ACM.

[2] O. Barzilay, O. Hazzan, and A. Yehudai. Using social media to study the diversity of example usage among professional developers. In *Proc. 19th ACM SIGSOFT Symposium and the 13th European Conf. Foundations of Software Engineering*, ESEC/FSE '11, pages 472–475, New York, USA, 2011. ACM.

[3] A. Begel, J. Bosch, and M.-A. Storey. Social networking meets software development: Perspectives from github, msdn, stack exchange, and topcoder. *Software, IEEE*, 30(1):52–66, 2013.

[4] C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan. Mining email social networks. In *Proc. 2006 Int. Workshop Mining Software Repositories*, MSR '06, pages 137–143, New York, USA, 2006. ACM.

[5] S. Black, R. Harrison, and M. Baldwin. A survey of social media use in software systems development. In *Proc. 1st Workshop Web 2.0 for Software Engineering*, pages 1–5. ACM, 2010.

[6] S. B. Bongey, G. Cizadlo, and L. Kalnbach. Explorations in course-casting: Podcasts in higher education. *Campus-wide information systems*, 23(5):350–367, 2006.

[7] G. Bougie, J. Starke, M.-A. Storey, and D. M. German. Towards understanding twitter use in software engineering: Preliminary findings, ongoing challenges and future questions. In *Proc. 2Nd Int. Workshop Web 2.0 for Software Engineering*, Web2SE '11, pages 31–36, New York, USA, 2011. ACM.

[8] F. Brooks. *The Mythical Man-Month: Essays on Software Engineering*. Addison-Wesley, 1975.

[9] G. Campbell. Podcasting in education. *Educause Review*, 40(6):32–47, 2005.

[10] A. Capiluppi, A. Serebrenik, and L. Singer. Assessing technical candidates on the social web. *Software, IEEE*, 30(1):45–51, 2013.

[11] N. Carr. *The shallows: What the Internet is doing to our brains*. WW Norton & Company, 2011.

[12] M. Chui, J. Manyika, J. Bughin, R. Dobbs, C. Roxburgh, H. Sarrazin, G. Sands, and M. Westergren. The social economy: Unlocking value and productivity through social technologies. `http://www.mckinsey.com/insights/high_tech_telecoms_internet/the_social_economy`, July 2012.

[13] H. H. Clark and S. E. Brennan. *Grounding in Communication*, chapter 7, pages 127–149. American Psychological Association, 1991.

[14] B. Cleary, C. Gomez, M.-A. Storey, L. Singer, and C. Treude. Analyzing the friendliness of exchanges in an online software developer community. In *6th Int. Workshop Cooperative and Human Aspects of Software Engineering (CHASE2013)*, pages 159–160, 2013.

[15] L. Dabbish, C. Stuart, J. Tsay, and J. Herbsleb. Social coding in GitHub: transparency and collaboration in an open software repository. In *Proc. ACM 2012 Conf. Comput. Supported Cooperative Work*, pages 1277–1286. ACM, 2012.

[16] T. DeMarco and T. Lister. *Peopleware: Productive Projects and Teams*. Addison-Wesley, 1987.

[17] S. Deterding, M. Sicart, L. Nacke, K. O'Hara, and D. Dixon. Gamification. using game-design elements in non-gaming contexts. In *PART 2—Proc. 2011 annual conf. extended abstracts Human factors in computing systems*, pages 2425–2428. ACM, 2011.

[18] P. Dourish and M. Chalmers. Running out of space: Models of information navigation. In *Proc. of HCI '94*, Glasgow, Scotland, 1994. ACM Press.

[19] K. Dullemond, B. van Gameren, M.-A. Storey, and A. van Deursen. Fixing the "out of sight out of mind" problem: One year of mood-based microblogging in a distributed software team. In *Proc. 10th Working Conf. Mining Software Repositories*, MSR '13, pages 267–276, Piscataway, NJ, USA, 2013. IEEE Press.

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

[21] E. Gilbert. Widespread underprovision on reddit. In *Proc. 2013 Conf. Comput. Supported Cooperative Work*, CSCW '13, pages 803–808, New York, USA, 2013. ACM.

[22] 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*, pages 155–164. ACM, 2011.

[23] C. Gutwin, R. Penner, and K. Schneider. Group awareness in distributed software development. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW '04, pages 72–81, New York, NY, USA, 2004. ACM.

[24] A. Guzzi, A. Bacchelli, M. Lanza, M. Pinzger, and A. van Deursen. Communication in open source software development mailing lists. In *Proc. 10th Int. Workshop Mining Software Repositories*, pages 277–286. IEEE Press, 2013.

[25] A. Guzzi, M. Pinzger, and A. van Deursen. Combining micro-blogging and ide interactions to support developers in their quests. In *Software Maintenance (ICSM), 2010 IEEE Int. Conf. Soft. Maintenance*, pages 1–5, 2010.

[26] M. Handel and J. D. Herbsleb. What is chat doing in the workplace? In *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work*, CSCW '02, pages 1–10, New York, NY, USA, 2002. ACM.

[27] J. Herbsleb and D. Moitra. Global software development. *Software, IEEE*, 18(2):16–20, 2001.

[28] J. D. Herbsleb, D. L. Atkins, D. G. Boyer, M. Handel, and T. A. Finholt. Introducing instant messaging and chat in the workplace. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '02, pages 171–178, New York, NY, USA, 2002. ACM.

[29] Z. Holman. How github works: Be asynchronous. `http://zachholman.com/posts/how-github-works-asynchronous`, 2011.

[30] H. Jenkins, K. Clinton, R. Purushotma, A. J. Robison, and M. Weigel. Confronting the challenges of participatory culture: Media education for the 21st century. `http://digitallearning.macfound.org/atf/cf/%7B7E45C7E0-A3E0-4B89-AC9C-E807E1B0AE4E%7D/JENKINS_WHITE_PAPER.PDF`, 2006.

[31] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian. The promises and perils of mining github. In *2014 11th Working Conference on Mining Software Repositories (MSR) (to appear)*, 2014.

[32] R. Kazman and H.-M. Chen. The metropolis model a new logic for development of crowdsourced systems. *Communications of the ACM*, 52(7):76–84, July 2009.

[33] S. Lakhal, H. Khechine, and D. Pascot. Evaluation of the effectiveness of podcasting in teaching and learning. In *World Conf. E-Learning in Corporate, Government, Healthcare, and Higher Educ.*, volume 2007, pages 6181–6188, 2007.

[34] C. Lampe and P. Resnick. Slash(dot) and burn: Distributed moderation in a large online conversation space. In *Proc. SIGCHI Conf. Human Factors in Computing Systems*, CHI '04, pages 543–550, New York, USA, 2004. ACM.

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

[36] F. Lanubile, C. Ebert, R. Prikladnicki, and A. Vizcaino. Collaboration tools for global software engineering. *IEEE Software*, 27(2):52–55, 2010.

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

[38] P. M. Leonardi, M. Huysman, and C. Steinfield. Enterprise social media: Definition, history, and prospects for the study of social technologies in organizations. *Journal of Computer-Mediated Communication*, 19(1):1–19, 2013.

[39] K. Lerman. User participation in social media: Digg study. In *Web Intelligence and Intelligent Agent Technology Workshops, 2007 IEEE/WIC/ACM Int. Conf.*, pages 255–258, 2007.

[40] B. Leuf and W. Cunningham. *The Wiki Way: Quick Collaboration on the Web*. Addison-Wesley Professional, 2001.

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

[42] 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. Human Factors in Computing Systems*, CHI '11, pages 2857–2866, New York, USA, 2011. ACM.

[43] J. Marlow and L. Dabbish. Activity traces and signals in software developer recruitment and hiring. In *Proc. 2013 Conf. Comput. Supported Cooperative Work*, CSCW '13, pages 145–156, NY, USA, 2013. ACM.

[44] J. Marlow, L. Dabbish, and J. Herbsleb. Impression formation in online peer production: Activity traces and personal profiles in github. In *Proc. 2013 Conf. Comput. Supported Cooperative Work*, CSCW '13, pages 117–128, New York, USA, 2013. ACM.

[45] A. E. Marwick and D. Boyd. I tweet honestly, i tweet passionately: Twitter users, context collapse, and the imagined audience. *New Media & Society*, 13(1):114–133, 2011.

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

[47] A. Mockus, R. T. Fielding, and J. D. Herbsleb. A case study of open source software development: the apache server. In *Software Engineering, 2000. Proc. 2000 Int. Conf.*, pages 263–272, 2000.

[48] A. Mockus, R. T. Fielding, and J. D. Herbsleb. Two case studies of open source software development: Apache and mozilla. *ACM Trans. Softw. Eng. Methodol.*, 11(3):309–346, July 2002.

[49] P. Naur and B. Randell, editors. *Software Engineering: Report of a Conference Sponsored by the NATO Science Committee*, Garmisch, Germany, Oct. 1968. NATO.

[50] G. M. Olson and J. S. Olson. Distance matters. *Hum.-Comput. Interact.*, 15(2):139–178, Sept. 2000.

[51] D. Pagano and B. Brügge. User involvement in software evolution practice: A case study. In *Proc. 2013 Int. Conf. Software Engineering*, ICSE '13, pages 953–962, Piscataway, NJ, USA, 2013. IEEE Press.

[52] D. Pagano and W. Maalej. How do developers blog?: An exploratory study. In *Proceedings of the 8th Working Conference on Mining Software Repositories*, MSR '11, pages 123–132, New York, NY, USA, 2011. ACM.

[53] S. Park and F. Maurer. The role of blogging in generating a software product vision. In *Cooperative and Human Aspects on Software Engineering, 2009. CHASE '09. ICSE Workshop*, pages 74–77, 2009.

[54] C. Parnin, C. Treude, L. Grammel, and M.-A. Storey.

Crowd documentation: Exploring the coverage and the dynamics of api discussions on stack overflow. Technical Report GIT-CS-12-05, Georgia Tech, 2012.

[55] C. Parnin, C. Treude, and M.-A. Storey. Blogging developer knowledge: Motivations, challenges, and future directions. In *Program Comprehension (ICPC), 2013 IEEE 21st International Conference on*, pages 211–214, May 2013.

[56] I. Peters. *Folksonomies: Indexing and Retrieval in Web 2.0*. De Gruyter, 2009.

[57] 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. 2013 Int. Conf. Software Engineering*, ICSE '13, pages 112–121, Piscataway, NJ, USA, 2013. IEEE Press.

[58] N. Postman. The reformed english curriculum. In A. C. Eurich, editor, *High school 1980: the shape of the future in American secondary education*. Pitman Pub. Corp., 1970.

[59] E. S. Raymond. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly Media, 2001.

[60] W. Reinhardt. Communication is the key - support durable knowledge sharing in software engineering by microblogging. In *Software Engineering (Workshops)*, volume 150 of *LNI*, pages 329–340. GI, 2009.

[61] E. Ries. *The Lean Startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. Random House Digital, Inc., 2011.

[62] P. C. Rigby, B. Cleary, F. Painchaud, M.-A. Storey, and D. M. German. Contemporary peer review in action: Lessons from open source development. *IEEE Software*, 29(6):56–61, 2012.

[63] P. C. Rigby and M.-A. Storey. Understanding broadcast based peer review on open source software projects. In *Proc. 33rd Int. Conf. Software Engineering*, ICSE '11, pages 541–550, New York, USA, 2011. ACM.

[64] M. Shaw. Three patterns that help explain the development of software engineering. In *Dagstuhl Seminar 9635 on History of Software Engineering*, pages 52–56, 1996.

[65] 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. 2013 Conf. Comput. Supported Cooperative Work*, CSCW '13, pages 103–116, NY, USA, 2013. ACM.

[66] 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 2014 International Conference on Software Engineering (to appear)*, 2014.

[67] T. Standage. *Writing on the Wall: Social Media-The First 2,000 Years*. Bloomsbury Publishing, 2013.

[68] M.-A. Storey, L.-T. Cheng, I. Bull, and P. C. Rigby. Waypointing and social tagging to support program navigation. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, pages 1367–1372, New York, USA, 2006. ACM.

[69] J. Surowiecki. *The Wisdom of Crowds*. Doubleday; Anchor, 2005.

[70] C. Treude and M.-A. Storey. How tagging helps bridge the gap between social and technical aspects in software development. In *Software Engineering, 2009. ICSE 2009. IEEE 31st Int. Conf.*, pages 12–22, 2009.

[71] C. Treude and M.-A. Storey. Awareness 2.0: Staying aware of projects, developers and tasks using dashboards and feeds. In *Proc. 32Nd ACM/IEEE Int. Conf. Software Engineering*, volume 1 of *ICSE '10*, pages 365–374, New York, USA, 2010. ACM.

[72] J. Tsay, L. Dabbish, and J. D. Herbsleb. Social media in transparent work environments. In *Cooperative and Human Aspects of Software Engineering (CHASE), 2013 6th Int. Workshop*, pages 65–72, 2013.

[73] A. van Deursen, A. Mesbah, B. Cornelissen, A. Zaidman, M. Pinzger, and A. Guzzi. Adinda: a knowledgeable, browser-based ide. In *Proc. 32nd ACM/IEEE Int. Conf. Software Engineering-Volume 2*, pages 203–206. ACM, 2010.

[74] B. Vasilescu, A. Capiluppi, and A. Serebrenik. Gender, representation and online participation: A quantitative study of stackoverflow. In *Int. Conf. Social Informatics. ASE*, 2012.

[75] D. M. Virasoro, P. Leonard, and M. Weal. An analysis of social news websites. In *Proc. ACM WebSci'11*. ACM, June 2011.

[76] X. Wang, I. Kuzmickaja, K.-J. Stol, P. Abrahamsson, and B. Fitzgerald. Microblogging in open source software development: The case of drupal using twitter. *IEEE Software*, 99(PrePrints):1, 2013.

[77] M. M. Wasko and S. Faraj. "It is what one does": why people participate and help others in electronic communities of practice. *The Journal of Strategic Inform. Systems*, 9(2-3):155 – 173, 2000.

[78] G. Weinberg. *The Psychology of Computer Programming*. Van Nostrand-Reinhold, 1971.

[79] E. C. Wenger and W. M. Snyder. Communities of practice: The organizational frontier. *Harvard business review*, 78(1):139–146, 2000.

[80] J. Whitehead. Collaboration in software engineering: A roadmap. In *2007 Future of Software Engineering (FOSE '07)*, pages 214–225, Washington, DC, USA, 2007. IEEE Computer Society.

[81] J. Whitehead, I. Mistrík, J. Grundy, and A. van der Hoek. *Collaborative Software Engineering: Concepts and Techniques*, chapter 1, pages 1–30. Springer Berlin Heidelberg, 2010.

[82] D. W. Wilson. Monitoring technology trends with podcasts, rss and twitter. *Library Hi Tech News*, 25(10):8–22, 2008.

[83] A. Zagalsky, O. Barzilay, and A. Yehudai. Example overflow: Using social media for code recommendation. In *Recommendation Systems for Software Engineering (RSSE), 2012 3rd Int. Workshop*, pages 38–42, 2012.

[84] J. Zhang, Y. Qu, J. Cody, and Y. Wu. A case study of micro-blogging in the enterprise: use, value, and related issues. In *Proc. SIGCHI Conf. Human Factors in Computing Systems*, pages 123–132. ACM, 2010.