Knowledge Building in Software Developer Communities

by

Alexey Zagalsky
B.Sc., Tel Aviv University, 2009
M.Sc., Tel Aviv University, 2013

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

Knowledge Building in Software Developer Communities

by

Alexey Zagalsky
B.Sc., Tel Aviv University, 2009
M.Sc., Tel Aviv University, 2013

Supervisory Committee

ABSTRACT

Software development has become a cognitive and collaborative knowledge-based endeavor where developers and organizations, faced with a variety of challenges and an increased demand for extensive knowledge support, push the boundaries of existing tools and work practices. Researchers and industry professionals have spent years studying collaborative work and communication media, however, the landscape of social media is rapidly changing. Thus, instead of trying to model the use of specific technologies and communication media, I seek to model the knowledge-building process itself. Doing so will not only allow us to understand specific tool and communication media use, but whole ecosystems of technologies and their impact on software development and knowledge work, revealing aspects not only unique to specific tools, but also aspects about the combination of technologies.

In this dissertation, I describe the empirical studies I conducted aimed to understand social and communication media use in software development and knowledge curation within developer communities. An important part of the thesis is an additional qualitative meta-synthesis of these studies. My meta-analysis has led to a model of software development as a knowledge building process, and a theoretical framework: I describe this newly formed framework and how it is grounded in empirical work, and demonstrate how my primary studies led to its creation. My conceptualization of knowledge building withing software development and the proposed framework provide the research community with the means to pursue a deeper understanding of software development and contemporary knowledge work. I believe that this framework can serve as a basis for a theory of knowledge building in software development, shedding light on knowledge flow, knowledge productivity, and knowledge management.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor **Margaret-Anne (Peggy) Storey** for her mentorship, positive and enthusiastic approach, and her immense passion and openness. Peggy's approach provided an environment of support that helped me flourished. Even more so, the freedom and encouragement I received to pursue *all* the research directions and side-projects I desired were invaluable. I'm deeply thankful for her guidance and inspiration that made me the researcher I am today.

I would also like to thank my supervisory committee, **Arie van Deursen and Leif Singer**, for their insights and guidance through my research journey, and for challenging me to go further. I have thoroughly enjoyed every single one of our discussions, and I couldn't have asked for a better committee or more supportive mentors. I'm also very grateful to **James D. Herbsleb** who served as my external examiner. I highly appreciate the intriguing discussion we had and the feedback he provided, both have been very helpful in finalizing this thesis.

I am thankful to **Daniel German** for sharing a passion for and interest in the technical aspects of my research. I am deeply thankful for our extensive discussion about Stack Overflow, GitHub, and academic life in general. It has been my pleasure working with him and I hope to continue doing so in the future.

To all the current and former **members of the CHISEL research group** at the University of Victoria I extend my gratitude and cherish our friendship: Leif Singer, Christoph Treude, Carlos Gómez, David Rusk, Elena Voyloshnikova, Laura MacLeod, Bo Fu, Eric Verbeek, Carly Lebeuf, Courtney Bornholdt, Eirini Kalliamvakou, Jorin Weatherston, Matthieu Foucault, Omar Elazhary, Huihui (Nora) Huang, Maryi Arciniegas Méndez, Maria (Tania) Ferman Guerra, Ying Wang, Ian Bull, Brendon Earl, Joseph (Noel) Feliciano, Neil Ernst, and Cassandra Petrachenko. I have greatly enjoyed being surrounded by such amazing and interesting people. A special thanks to **Carly** for the great discussions and advice she had for me, and for keeping me on track with the thesis writing ("thesis buddies"). **Eirini** for providing helpful advice and brainstorming over the scattered pieces of my research when it was needed most. Her help was timely and invaluable, and I was lucky to share an office with her. **Cassie** for helping with *pretty much everything*, and most notably improving my writing skills.

A big thank you also goes to **Wendy Beggs** and **Nancy Chan**, our graduate and financial secretaries, that were extremely helpful and made the administrative side of things as smooth as possible.

During my PhD, I was fortunate to collaborate and work with many extremely talented people. A big thank you goes to all **my co-authors and collaborators** outside the CHISEL group: Daniel German, Fernando Figueira Filho, Bin Lin, Germán Poo-Caamaño, Allyson Hadwin, Alexander Serebrenik, Gail Murphy, Per Runeson, Fabio Calefato, and Moritz Beller. I have learned much from these collaborations and it gave me an opportunity to work with exceptionally talented and interesting people. A special thanks goes to **Arie van Deursen**, **Andy Zaidman**, and **Alberto Bacchelli** for hosting me during my visit to the TU Delft Software Engineering Research Group, and to **Filippo Lanubile**, **Fabio Calefato**, and **Nicole Novielli** for hosting me during my visit to the University of Bari.

A special thank you goes to **Ohad Barzilay** and **Amiram Yehudai**. Their mentorship and guidance have transcended my masters and remain with me until today. The inspiration I had from Amiram and Ohad, have influenced how I saw myself as a PhD student and as a researcher, and I strive to pay it forward to others around me. Amiram and Ohad have inspired me, provided me with opportunities to make mistakes and grow, and have given me invaluable advice.

Lastly, I would like to thank my **parents**, **brother**, and **close friends**, for believing in me and supporting me along the way. While being far away, you managed to always be there for me—encouraging and recharging me with new energies and strength. This work would not have been possible without your love, support, and encouragement.

DEDICATION

To my parents, Luba and Misha, for their endless love and support.
I'm especially grateful to my father who sparked and nourished my interest and passion for technology and engineering.

Preface

"I had leaned and climbed forward like Alice through the looking-glass. I had no idea just how deep the rabbit hole would go."

– Simon Pegg, 2011

Technology and software are shaping and "fueling" our modern world, progressively expanding and empowering our human abilities to communicate and work collaboratively. However, it's a double-edged sword—the software we build and the tools and processes we use to build it, are becoming increasingly more complicated. At the same time, we struggle to understand the software development process (on all of its aspects) or how to make it better.

When I set on this journey, I was looking to understand how social media use by developers shapes the software development process. But early on, it became apparent that just looking at the social and communication media developers use is not going to be enough. We needed to take into account the context in which they use the channel (e.g., for what task/activity, types of content, etc), the reasons for using the channel (e.g., provides cognitive support, overcomes challenges), the unit of scale (i.e., individual, group, or community), and how it may be affecting development practices and activities. Thus, I decided to view this from a knowledge transfer perspective. Not long after, it was clear to me that we needed a knowledge theory within software engineering—a theory that would link and explain the relationships between these different components. In my candidacy exam, a two-year mark into my PhD, I argued that *"studying only the tools and channels can only provide a narrow perspective. We believe that software development has evolved... into a Participatory Process—A knowledge building process which is characterized by the (1)* ***knowledge activities*** *and* ***actions****, (2)* ***stakeholder roles****, and (3) is enabled by socially enhanced* ***tools and communication channels****"*.

When I explained this to my colleagues and other researchers, many agreed but warned me that this task is perhaps too ambitious for a PhD time frame. Nonetheless, I was determined. By the three-year mark of my PhD, I needed to begin combining my research work and form a thesis. But, there were still unexplored directions (e.g., roles of developers) and missing pieces (e.g., what is knowledge?) that would help form a theory. Slowly, I began to realize that this perhaps was too large of a task. As a result, I changed the scope and direction of my research topic: A socio-technical view of knowledge curation within software development. I was no longer trying to form a theory.

Interestingly, while focusing on the new direction, knowledge curation within developer communities, my research work continued to provide broader insights on the knowledge transfer process and to reinforce the way we modeled the software development process (as part of the knowledge theory). After conducting studies on different aspects of software engineering and many months spent examining existing theories, I began to connect the pieces. Ironically, I ended up forming a knowledge framework for software development after all. This is not a full theory yet, but it is a basis for one, and I can continue to extend it after my PhD.

In this dissertation, I describe the resulting knowledge framework and the studies used to form it.

Part I

General Introduction

Chapter 1

Introduction

“No one knows everything, everyone knows something, all knowledge resides in collaborative social networks.”

– adapted from Pierre Levy, 1997

The computer revolution, the microprocessor, and the Internet have paved the way for the rise of software. In 2011, Marc Andreessen hypothesized that ”software is eating the world” due to the increasing dependence on software. Technology and software have invaded and overtaken a multitude of industries and organizations, subsequently interweaving with and disrupting conventional work practices, and as a result, have pushed the boundaries of work, collaboration, and communication. Blackler [14] cautioned that *“it would be a mistake to regard the new generation of information and communication technologies as neutral tools that can merely be grafted onto existing work systems.”*

As software systems become more pervasive and the systems themselves become more interconnected, the development of software evolves into a distributed, cognitive and collaborative knowledge-based endeavor. This knowledge evolution is further fueled by the social transformations of the 21st century, which has introduced new communication channels and socially-enabled tools (e.g., GitHub, Stack Overflow, Slack) that dictate the flow of knowledge, and shape how developers work. As a result, software developers find themselves at the forefront of knowledge work [77], often being the first to adopt new tools and practices, face challenges , and be tasked with adapting and shaping them. Researchers and industry professionals have invested tremendous effort in investigating software development practices, processes, the tools developers use, and the artifacts they create. However, software is not only code, it is also the combined knowledge within teams and the organizations [105]. Without understanding the knowledge building process itself, our understanding of *'what software is'* and *'how software is built'* are and will remain limited.

As part of my PhD and Master's studies, I have been studying communication and social media use in software development at length. I gained insights on how developers use specific communication channels such as Stack Overflow and GitHub, how different channels support different knowledge sharing activities, and how a developer community makes use of and is challenged by its communication media. However, instead of trying to model the use of specific technologies and communication media (some are rapidly changing), I seek to **model the knowledge-building process itself**. Doing so will not only allow us to understand the use of specific tool and communication media, but whole ecosystems of technologies and their impact on software development and knowledge work, revealing aspects not only unique to specific tools, but also aspects about the combination of technologies.

In this thesis, I describe my studies on social and communication media use in software development and studies on knowledge curation within software developer communities. An important part of the thesis is an additional qualitative meta-synthesis of these studies. For this purpose, I model software development as a knowledge building process, and undertake an *across-study conceptualization* to form a theoretical framework. I describe this newly formed framework and how it is grounded in empirical work, and demonstrate how our primary studies led to its creation. This phase of my work aimed to identify underlying conceptual relations that were not necessarily explicitly expressed in the findings, and to form a theoretical groundwork for a knowledge building theory within software engineering. Theoretical models and frameworks help researchers go beyond answering *'what'* or *'which'* empirical patterns may be observed, but rather they help to understand and explain the *'why'* of empirical findings [154, 143]. This proposed framework provides the research community with the means to pursue a deeper understanding of software development and contemporary knowledge work. I believe that this framework can serve as a basis for a theory of knowledge building in software development, shedding light on and benefiting knowledge flow, knowledge productivity, and knowledge management.

1.1 Research Goal and Scope

The overarching goal of this research is to model the knowledge building process in software development; it is scoped on knowledge transfer, the knowledge activities software developers perform, and the communication media that facilitate them.

The research I present and reason about in this dissertation did not start off as an explicit study of knowledge building, but rather the initial aim was to understand the impact social and communication media may have on software development. I realized that with the influence of social media, modern software development has changed. These media are becoming embedded in the software development process and seem to: (1) influence developers activities by changing them or creating new ones; and (2) expose development processes which previously were taken for granted, ignored, or misunderstood. To capture this influence, I studied communication channels and social media use in software development at length, applying multiple empirical software engineering research methods. As I progressed with these studies, it also became apparent that knowledge building was a fundamental part of developing software. Therefore, the need to model software development as a knowledge building process arose gradually and ended up forming the direction I followed with subsequent studies, as well as having shaped the results and interpretations I present in the thesis.

Each of the empirical studies I present in this dissertation has its own *study-level* research questions. Yet, as I progressed with the individual studies, each one enriched my understanding of the different processes and tools I was studying in software development. This allowed me to reason and abstract from the directly observed study results, with two outcomes: first, it informed the research questions I asked in subsequent studies, and second, it shaped the *thesis-level* research questions I address and discuss in this dissertation. Below, I describe the rationale behind the thesis research questions and how they link to the individual studies.

1.2 Rationale Behind the Thesis Research Questions

Previous studies within the software engineering community (and some of my own previous work) examined how developers use specific communication channels and socially enabled tools such as Twitter [142], Stack Overflow [165, 10], and GitHub [27]. These studies focused on one channel at a time, and revealed why developers adopt these channels (e.g., improving awareness, supporting learning), the challenges they face (e.g., information overload), and their coping strategies (e.g., content and network pruning).

Building on that work and aiming to get a fuller picture, I first conducted an empirical study [155] on the role and interplay of the wide landscape of social and communication media developers use (described in Chapter 4). Through the study's research questions, I sought to understand the characteristics of the "social programmers" that participate in online communities, and identify what communication channels developers use to support their activities, the communication channels they find most important, and the challenges they face when using a whole ecosystem of tools. When taken together, these findings help us understand **how social channels and tools affect collaborative software development, and more specifically, the way they affect knowledge building**; this is the first *thesis-level* research question this dissertation showcases.

Besides the study's findings on tool use by developers, there was an additional insight: the study revealed a highly complex picture of the modern software development process in today's media-rich development environments. It became clear that to better understand the software development process, one needs to account for more than strictly the communication media and their use—we also need to account for the activities developers carry out, the artifacts they create and share, the roles they take on, and the assemblages they participate in. The need to consider and understand all these aspects pointed to how valuable a descriptive theoretical construct can be in enhancing our understanding of the software development process, and that a non-trivial part of this process relates to knowledge construction. Thereby, **I set out towards modeling the software development process and the creation of a theoretical framework to help understand knowledge transfer in software development**.

This goal required three layers of work: (1) gather information through additional studies to give coverage and depth for aspects beyond media channel use; (2) leverage knowledge-related theoretical constructs (from other models and theories) to inform and expand my understanding of the phenomenon; and (3) synthesize the findings and my enriched understanding into a theoretical construct, and use that as a lens to reflect on the gathered empirical results.

In the next empirical study [191], I moved from a *global view* of social media use in software development to a *focused view* on how the R developer community uses two specific media channels to curate and share knowledge: Stack Overflow and the R-help mailing list. In this case, the *study-level* research questions focused on identifying the types of knowledge artifacts R developers created and shared, as well as the developers' participation behavior between the two channels (described in Chapter 5).

When classifying the types of information developers shared, I identified different knowledge creation modes on each channel. I also observed challenges that were previously identified in the first empirical study, but now I was able to see their impact on the community and its knowledge sharing activities. As an example, Stack Overflow's gamified design provides incentives for individual knowledge contributions, but also hinders collaborative knowledge sharing and community knowledge creation (as exemplified in the number of low quality, unanswered, duplicate, or zero-score questions on Stack Overflow). The answers to this study's research questions helped me understand **how knowledge is constructed and curated in a developer community**; that is the second *thesis-level* research question the dissertation discusses.

While conducting the above empirical studies, I also participated in a variety of additional studies with my colleagues. In parallel to my own findings, I was exposed to further aspects of social media impact on knowledge transfer within the software development process, which also helped inform my *thesis-level* research questions.

- My studies of GitHub in an educational context [190, 44] revealed how social media platforms can shape workflows and knowledge flow, and introduced me to the role and impact of communities of practice.
- The studies on regulation theory within software development [6, 7] exposed me to important dimensions of collaboration—behavior, cognition, and motivation—and provided insights about sub-processes that govern activities (e.g., Task Understanding, Enacting) at different levels (self-, co-, shared). I believe this study encouraged me to not only focus on understanding the tools and channels developers use, but also to explore the theoretical underpinnings of why certain tools and practices are used, and to consider collaboration at a meta-cognitive level.
- The studies on Bots [157, 87, 88] allowed me to reflect on their role in knowledge sharing within software development. Often, the Bot perspective has challenged my view and understanding of what knowledge is and how knowledge flows

within software development, and I came to realize that Bots are extremely versatile in terms of the types of knowledge they can transfer.

Together with the empirical studies described in this thesis, these findings have added to the *richness* of my understanding and provided valuable *descriptive* insights on the building and transfer of knowledge within software development. Section 7.5 includes a more detailed description of how these additional studies have shaped the work presented in this thesis.

As my final step towards the creation of a theoretical construct, I synthesized the empirical findings and insights. For this, I conducted a meta-synthesis study: a review of the earlier studies, where instead of trying to model the use of specific communication media, the aim was to model the knowledge-building process itself. The meta-synthesis provided insights about **how knowledge is transferred as part of the software engineering process**; this is the third *thesis-level* research question I discuss in the dissertation.

In summary, through my empirical studies I formed a mental model of *how software is built* (early iterations of the mental model are shown in Appendix C), and later through the meta-synthesis study I formed **a theoretical framework of Knowledge Building in Software Development**. This framework builds on directly observed results from my own empirical studies, and is also informed by existing literature and the other studies I participated in. Due to this, *the framework is capable of providing deeper insights than the individual components that it was derived from.* Consequently, I have revisited core findings of my research work, and by using the knowledge framework as a lens, I have achieved a deeper understanding of **'why'** these observations and patterns were happening. I elaborate on the use of the technique and demonstrate the additional insights it generated in Chapter 6.

1.3 Contributions

This research work makes the following overarching contributions:

Empirical studies of social and communication media use in software development communities.

In this thesis, I describe the empirical studies I've conducted on the use of social and communication media in software developer communities. These studies have revealed valuable insights about the impact of communication channel use on developers and the software development process, and how different channels support different knowledge sharing activities. The findings of these studies helped us form the basis for a knowledge building framework in software engineering.

A theoretical framework of knowledge building in software development.

A key contribution of this thesis is the emerging theoretical framework. The knowledge framework aims to provide researchers with the 'theoretical mechanisms' needed to understand and articulate knowledge transfer within software development processes and organizations—thus, leading to a better understanding of software development itself. This framework is grounded in our empirical work and has been demonstrated to extend our findings.

A heuristic analysis instrument for practitioners.

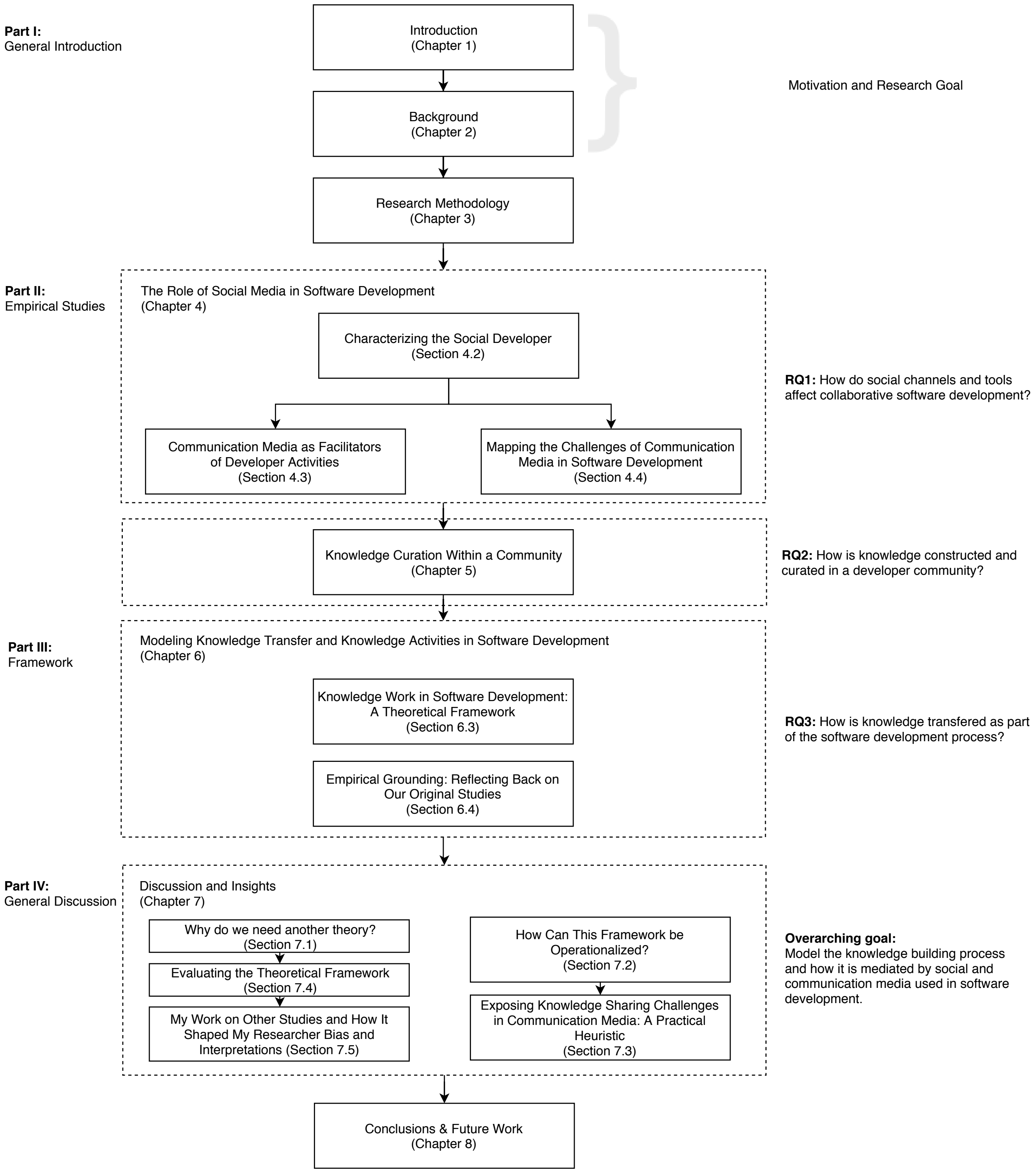
To help organizations and development teams choose and design their communication infrastructure, I describe an inspection method in the form of heuristic analysis for revealing and mapping knowledge sharing challenges when using social media and communication channels. This heuristic is given in the form of 25 questions that are designed to be used as a reflective and guiding tool.

1.4 Thesis Outline

The thesis is structured in eight chapters. The following provides an outline of each chapter, and shows how content from these chapters relates to papers that were published as part of the thesis. Figure 1.1 shows a high-level outline of the thesis.

Part I: General Introduction

Content: After introducing the topic and subject matter of this thesis in Chapter 1, a review on background and related work is given in Chapter 2. The background describes the modern landscape of social media and tools used by developers, and its ever-changing and complex nature. Additionally, it motivates the need to understand the impact of social media on developer activities, practices, and community participation. Then, Chapter 3 describes the overarching methodological choices and the



Part I:
General Introduction
Introduction (Chapter 1)
Background (Chapter 2)
Motivation and Research Goal
Research Methodology (Chapter 3)
Part II:
Empirical Studies
The Role of Social Media in Software Development (Chapter 4)
Characterizing the Social Developer (Section 4.2)
Communication Media as Facilitators of Developer Activities (Section 4.3)
Mapping the Challenges of Communication Media in Software Development (Section 4.4)
RQ1: How do social channels and tools affect collaborative software development?
Knowledge Curation Within a Community (Chapter 5)
RQ2: How is knowledge constructed and curated in a developer community?
Part III:
Framework
Modeling Knowledge Transfer and Knowledge Activities in Software Development (Chapter 6)
Knowledge Work in Software Development: A Theoretical Framework (Section 6.3)
Empirical Grounding: Reflecting Back on Our Original Studies (Section 6.4)
RQ3: How is knowledge transfered as part of the software development process?
Part IV:
General Discussion
Discussion and Insights (Chapter 7)
Why do we need another theory? (Section 7.1)
Evaluating the Theoretical Framework (Section 7.4)
My Work on Other Studies and How It Shaped My Researcher Bias and Interpretations (Section 7.5)
How Can This Framework be Operationalized? (Section 7.2)
Exposing Knowledge Sharing Challenges in Communication Media: A Practical Heuristic (Section 7.3)
Overarching goal:
Model the knowledge building process and how it is mediated by social and communication media used in software development.
Conclusions & Future Work (Chapter 8)

Figure 1.1: An outline of this dissertation.

rationale behind the studies that comprise this thesis. It begins with a description of my epistemological worldview and my research goal, which dictated the research approach and research design I followed. In this chapter, I describe how the research presented in this thesis combines both theoretical and empirical work.

Publications: The history of social media and communication tools (presented in Chapter 2) was published as part of a roadmap paper in the International Conference on Software Engineering (ICSE 2014) [156]. It was produced in collaboration with Margaret-Anne Storey, Leif Singer, Fernando Figueira Filho, and Brendan Cleary.

Part II: Empirical Studies

Content: Chapter 4 describes my research work on the role of social media in software development. Through a large-scale survey with 1,449 developers, I explored how developers use social and communication media to support their development activities. In this study, I focused on the interplay of these channels, and the opportunities and challenges they introduce. Our findings showed that developers use a plethora of communication media to support their development activities, and further emphasize that developers engage in essential non-coding activities, such as learning and keeping up to date. Code hosting sites, face-to-face conversations, question & answer sites, and web search were the top most important channels described in the survey. However, other channels were also deemed important by the participating developers. The most commonly cited reasons were the channel's support of group awareness, collaboration, allocation and retrieval of information, and its ability to enhance dissemination or consumption of information (i.e., cognitive support). This work helped us form an initial mental model of knowledge transfer in software development—connecting developers, communication media, and their shared artifacts and activities (see Appendix C). Additionally, it helped establish a preliminary taxonomy of knowledge activities for software developers (summarized in Fig. 4.5). Later, I refined these knowledge activities and formed a knowledge building framework (described in Chapter 6).

Informed by these findings and insights, our goal for the following study was to understand the knowledge curation process at a community level (described in Chapter 5). We used a mixed methods exploratory case study methodology. We began by empirically comparing how knowledge, specifically knowledge in question-and-answer form, is sought, shared, and curated on the two primary channels for sharing knowledge within the R community: a mailing list and Stack Overflow. Our findings indicated that there were two different approaches for constructing knowledge: *participatory knowledge construction*, where members cooperate and complement each other's contributions, and *crowd knowledge construction*, where members work towards the same objective but not necessarily together. We observed that knowledge transfer through Stack Overflow was done in a more crowdsourced manner, while knowledge transfer through the R-help mailing list was usually in a participatory manner. We then were able to explore the behavior and participation patterns of community members on the R-help mailing list and Stack Overflow. This has revealed a reduction in growth of active participation on Stack Overflow, i.e., the number of new questions with an overall positive score has started to decrease over time, perhaps indicating that the R community is maturing as a community and moving from knowledge creation to knowledge curation. These findings show promise for applicability to other similar systems and assemblages, e.g., companies adopting an internal Stack Overflow or communities that plan to use private groups on Stack Overflow.

Publications: Our study on understanding how social and communication media affect and disrupt software development (Chapter 4) was published in the Transactions on Software Engineering (TSE) journal [155]. It was an extension of our earlier work that was published at the International Conference on Software Engineering (ICSE 2014) [156]. These studies were performed in collaboration with Margaret-Anne Storey, Leif Singer, Daniel M. German, Fernando Figueira Filho, and Brendan Cleary. Our study on knowledge curation (Chapter 5) and its extension were published at the International Conference on Mining Software Repositories (MSR 2016) [192] and in the Journal of Empirical Software Engineering (EMSE) [191]. These studies were performed in collaboration with Daniel M. German, Margaret-Anne Storey, Carlos Gómez Teshima, and Germán Poo-Caamaño.

Part III: A Theoretical Knowledge Building Framework

Content: This part presents a theoretical framework of knowledge building in software development that is an outcome of an *across-study conceptualization*. Chapter 6 describes a qualitative meta-synthesis of two former studies: (1) the study on how social and communication media affect and disrupt software development (Chapter 4); and (2) the study about knowledge curation within the R community (Chapter 5). By applying a bottom-up approach consisting of a gradual and iterative analysis, and an interpretive synthesis of empirical data from these two studies, I formed a knowledge framework. This framework builds on existing concepts and models of knowledge work and CSCW, and on the empirical findings from the research work described in chapters 4–5. This framework is grounded in our empirical work and has been demonstrated to extend our findings.

Part IV: General Discussion

Content: The last part of the thesis presents insights from my work and provides a discussion on the formed theoretical framework. I begin this chapter by reflecting on why we need another theory in software engineering. To support my arguments, I summarize existing theories used in software engineering that are relevant to the subject mater of the thesis. Then, I discuss how the proposed knowledge framework can be operationalized, and provide an actionable instrument for practitioners in the form of a *heuristic analysis* for revealing and mapping knowledge sharing challenges when using social media and communication channels (Section 7.3). Subsequently, I evaluate the framework along two main criteria—credibility and applicability—and reflect on the additional factors that have shaped my researcher bias as a result of *additional studies I conducted or have been a part of* (that are **not** part of this thesis). Lastly, in Chapter 8, I conclude my thesis and discuss future work.

Chapter 2

Background

"No century in recorded history has experienced so many social transformations and such radical ones as the twentieth century."

– Peter F. Drucker, 1995

Social and communication media has seen a rapid growth and widespread adoption in the past decade. In fact, the speed and scale of social media adoption, such as Facebook and Twitter, is unprecedented in the history of technology adoption [19]. This proliferation of communication media has led to a paradigm shift in how people communicate, collaborate, and work. It has affected all forms of knowledge work, but perhaps its biggest impact has been on software development.

The rate of technology and communication media change in software development is bewildering. Relatively new communication media, such as GitHub and Stack Overflow (both just released in 2008), have quickly become an essential part of the standard toolset for many software engineers. In order to better understand this phenomena, we first need to take a look at the history of communication media use in software engineering [156].

Shanon [138] defined a communication channel as *"merely the medium used to transmit the signal from transmitter to receiver"*. While Rogers [130, p. 17] defined it as *"the means by which messages get from one individual to another"*. For the purpose of this thesis, I define a communication channel as follows:

Definition: A communication channel is the means by which information flows from transmitter to receiver(s).

2.1 The 'Wild West' of Communication Channels: A Brief History of Media Use in Software Development

In the early history of software development, the main communication channel was **face-to-face** interactions, 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 [139]. Face-to-face interaction was essential to support learning and problem solving, to build common ground [20], and to support collaborative system design and development. Face-to-face interactions are still a mainstay of communication in software projects, however, the increase of remote work [47] and distributed teams means that many developers nowadays may never meet face to face. In these cases, video chat tools such as **Skype** or **Google Hangouts** are used as a substitute.

The next medium to be adopted in the workplace was the **telephone**, which was important in supporting the early days of collaboration in software development. In 1987, De Marco and Lister [30] proclaimed that *"the telephone is here to stay. You can't get rid of it, nor would you probably want to."* However, they also understood it can be a source of interruptions. To illustrate the issue, they compared the telephone to **email**. *"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 dichotomy between synchronous and asynchronous communication channels—and even workflows—is garnering renewed attention now that remote work and open source development models are being more readily adopted by software companies.

An extension of email is the **mailing list**, which plays an ongoing role in keeping community members up to date with project activities. They have been used as a channel to disseminate commit logs from software repositories [59], 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 [126, 125]. A study by Gutwin *et al.* [59] showed that mailing lists support information seeking and dissemination of project knowledge, developer activities, and project discussion. On the other hand, the same study indicated that important information about code was sometimes fragmented across different channels (mailing lists, chat, and commit logs) within the studied open source projects. They found that it was often difficult to ensure that information was read by the right people in a timely fashion.

Capable of being used synchronously or asynchronously, it is not surprising that **text-based communication channels for private chat** and **instant messaging** apps have become widely used by software teams. Developers use these channels for coordinating tasks, sharing work artifacts, and communicating about their work. Initially, developers used general purpose instant messaging apps such as ICQ, AIM, and Skype. However, with time, more specialized chat apps and development tools embedded with instant messaging capabilities have replaced those (e.g., Gitter).

For supporting team interactions, developers use **text-based communication channels for group chat**. Internet Relay Chat (**IRC**) was perhaps one of the earliest group chat platforms used for work by software developers. Its golden era was between the 1990s and early 2000s, but starting in 2003, it has been gradually superseded by more modern group chat platforms. In 2002, researchers conducted empirical studies to investigate how IRC is used by distributed teams and found that globally distributed developers predominantly used it for technical discussions [64], however, its adoption was inconsistent across development teams [68]. Gutwin *et al.* [59] found that IRC was used for informal communication about project artifacts in open source projects, where important aspects of the non-archived IRC discussions were siphoned off to the project's mailing list. Interestingly, while IRC has been replaced, its design and features can still be seen in modern chat platforms. Some of the better known examples of modern group chat and project-oriented communication platforms used by developers are: Hipchat (released in 2010, replaced by Stride in 2017, and bought by Slack in 2018[1]), Campfire (released in 2006 and replaced by Basecamp in 2014), Slack (released in 2013), Microsoft Teams (released in 2017), and Telegram (released in 2013).

Wikis are another commonly used type of communication channel in software teams, primarily for collaborative knowledge sharing. In 1995, Ward Cunningham designed wikis as a medium for collaboratively editing software documentation [91]. Wikis were innovative because they allowed authors to easily link between internal pages and include text for pages that *did not yet exist* [91]. Wikis have been used to support defect tracking, documentation, requirements tracking, test case management, and are used for project portals [95]. Wikis are also used frequently in global software development [84], and remain integrated in collaborative and social coding

[1]zapier.com/blog/slack-versus-hipchat

sites [83].

Beyond collaborating on software documentation, developers make use of additional communication media to communicate and collaborate on other project artifacts. In 1975, Brooks [18, p. 74] described how they used a **project workbook** to document system knowledge and track all project activities, including rationale for design decisions and change information across versions. Nowadays, developer teams no longer use a physical workbook, and instead have replaced it 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.

For collaborating on code artifacts, developers use social coding hubs such as **SourceForge** (launched in 1999 but lost its popularity to GitHub in 2011), **GitHub** (launched in 2008), **BitBucket** (launched 2008), and **Gitlab** (launched in 2011). For instance, when examining communication in an open source project's mailing list between 2001 and 2012, Guzzi *et al.* [60] noticed that most of the communication about development issues occurred through the code repository discussion feature rather than email. These platforms were primarily designed to offer support for hosting projects and to provide revision control. However, modern social platforms such as GitHub go beyond that and serve as community collaboration hubs. They foster collaboration through various awareness features (e.g., **dashboards** and **activity feeds**), provide integration with external tools, and support asynchronous workflows (cf. e.g., Pham *et al.* [117]).

For facilitating the exchange of questions and answers and community discussions, developers initially used communication channels such as **Usenet** (developed in 1980, archiving of posts began in 1995), **bulletin boards**, and **Google groups**. However, over time these were replaced by **forums**, **news feeds**, and **Q&A platforms** such as **Stack Overflow**, **Quora**, and **Reddit**. Nonetheless, many of the features in Usenet can now be seen in more modern media such as Stack Overflow.

Stack Overflow was created in 2008 and has experienced rapid uptake. Even though it has many parallels to Usenet, Stack Overflow 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 aspect [32] 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 [97]. Stack Overflow has been extensively studied by the research community[2] and is rapidly growing into a formidable documentation resource. For example, Parnin *et al.* [115] found that it provides very good coverage for documentation on open source APIs.

Another medium that has evolved from the early bulletin boards are **social news websites** and **news feeds**, which have been experiencing a recent surge in popularity. Many of these sites [171] and aggregators allow developers to disseminate knowledge, discover new software, and keep up to date. Some of the most popular among developers are **Digg** (launched in 2004 but lost its popularity in 2010), **Reddit** (launched in 2005), and **Hacker News** (launched in 2007, was inspired by early Reddit communities). The importance of these news websites is beyond aggregating news and keeping up to date, as being mentioned at the top of the news site can provide valuable feedback and help the growth of a project's users, contributors, and the community as a whole. Moreover, these media provide a specific form of *social navigation* [34] and foster *serendipitous discovery.* Lampe and Resnick [81] analyzed Slashdot, 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 those with low starting scores do not receive as much consideration from moderators as other comments do.

Blogs, **microblogging**, and **podcasts** are an important community-based knowledge resource used in software development. The unique value of blogs (first used in 1994 and widely adopted by 1999) is that everyone can broadcast information. **Blogs** are frequently used by developers to document "how-to" information, to discuss the release of new features, and to support requirements engineering [113]. Parnin and Treude [114] found that blogs play an effective role in documenting APIs. Closely related to blogs and used in a similar fashion by developers are **podcasts** (either audio or video). Developers use podcasts for learning [80], keeping up to date with the latest trends and technologies [185], and for (job) training and as how-to guides. **Microblogging** also plays an increasingly important role in curating community

[2]https://meta.stackexchange.com/questions/134495/academic-papers-using-stack-exchange-data

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 has also seen significant adoption in software development. Studies [16, 177] showed that Twitter is used to communicate issues, documentation, to advertise blog posts to the community, and to solicit contributions from users. Developers who adopted Twitter used it to filter and curate the vast amount of technical information available to them [142]. Singer *et al.* [142] found that developers who felt that Twitter benefited them described benefits in terms of awareness, learning, and relationship building.

This brief history shows the ever-changing, continuously evolving, and complex ecosystem of social and communication media used by developers and other knowledge workers. It is in this landscape that we seek to understand the impact of social media on developer activities, practices, and community participation.

2.2 Not Just Many Channels: Social Media's Expanding Cognitive Support

Not just the number of channels developers use is changing, but the needs these channels address, the cognitive support they provide, and how they support modern work have also been changing. Social media are primarily seen as communication mechanisms used to facilitate the creation and sharing of information between people. However, social media also provide invaluable cognitive support. For example, they can help people navigate complex systems and networks, help with decision making, help memorize and recall information (e.g., reminders about important tasks), or help by automating trivial and repetitive tasks. *"The power of the unaided mind is highly overrated. Without external aids, deep, sustained reasoning is difficult. Human intelligence is highly flexible and adaptive, superb at inventing procedures and objects that overcome its own limits. The real powers come from devising external aids that enhance cognitive abilities"* [111]. In essence, cognitive support is the assistance that external aids (artifacts, tools, and technology) provide to humans in their thinking and problem solving processes [176]. Let's examine this aspect of communication media.

Early social media (e.g., face-to-face interaction, telephone) focused on facilitating local communication between individual workers. These channels were used as tools of communication and coordination, but provided no cognitive support on their own. Over time, technology accommodated for the rising need for remote work and communication, and channels such as instant messaging, video chat, and email were adopted—channels that still focused on communication between individuals, but brought support for **individual cognition**. For example, a conversation history feature facilitates the developer's external memory.

In work environments, developer teams also began to use group-supporting media for communication and collaboration. The CSCW community refers to these as *groupware*, which stands for *"computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment"* [41]. Groupware comes in many forms. From the channels mentioned in the previous section, examples of groupware include email, mailing lists, team wikis, version control systems, project management tools, IRC, and other group chat channels. Additional examples of groupware include shared calendars, shared document storage, group meeting spaces, address books, and shared task lists (e.g., Trello). These channels are capable of providing more **extended individual cognitive support** [176] (e.g., reduce mental effort, improve developer knowledge, make cognitively difficult problems easier, support reflective thinking) and enable **group cognition support** [149, 176] (e.g., assist in group knowledge building, externalize shared understanding, help manage group memory).

A well-recognized model for discussing groupware is the 3C Model described by Ellis *et al.* [41] in 1991, which consists of three key areas that require attention when studying collaborative work: *Communication*, *Collaboration*, and *Coordination*. In this model, *Communication* refers to the exchange of knowledge within a group and allows for the coordination of group tasks. *Coordination* refers to the awareness of and agreements made regarding tasks to be completed through team interactions, as well as any overhead (e.g., planning) that is necessary for the *Collaboration* effort itself [41]. Later, Gerosa *et al.* [51] proposed extending the 3C Model to also include *Awareness*: *"an understanding of the activities of others, which provides a context for your own activity"* [33]. Group communication media can be categorized by using these dimensions—an example by Sauter *et al.* [134] is shown in Fig. 2.1. However, even with these models, gaining an understanding of collaboration in software development is challenging. Developers engage in mindful processes of regulation to deter-

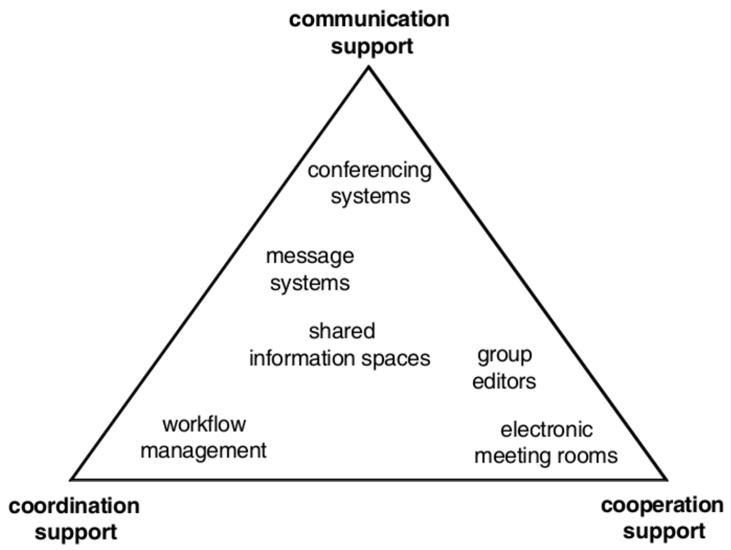


Figure 2.1: Categorizing groupware by using the dimensions of the 3C model (source: Sauter *et al.* [134]).

mine what tasks they need to complete and who should be involved, what their goals are relative to those tasks, how they should meet their goals, what domain knowledge needs to be manipulated, and why they use a particular approach or tool. Software engineering also involves *dynamic informal learning* where participants, guided by their various interests, engage in task coordination and the co-construction of knowledge. For this reason, I helped compose a *Model of Regulation* to capture how individuals self-regulate their tasks, knowledge and motivation, how they regulate one another, and how they achieve a socially shared understanding of project goals and tasks [6, 7]. This work articulated how computer-based tools can be used to support self-, co-, and shared regulation, and described the different categories of regulation tool support (structuring support, mirroring support, awareness tools, and guidance systems). Figure 2.2 shows an example of how modern social media used by developers (Trello, WakaTime, Codealike, GitHub) support reflective thinking and group cognition.

While groupware focused on communications between members of small groups, a new paradigm of global participation has emerged and formed social media. The bursting dot-com bubble of 2001 marked a turning point for the web and the beginning of social software. Coined by Tim O'Reilly and Dale Dougherty [112], Web 2.0 was the term used to refer to the socially enabled media and tools that prolif-

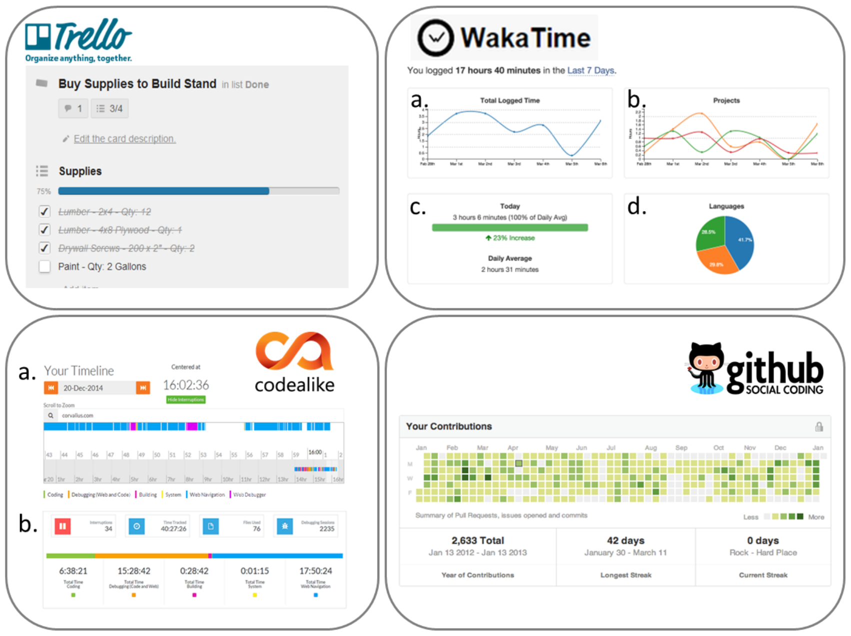


Figure 2.2: Examples of modern social media used by developers that incorporate mirroring support. These tools reflect individual or collective actions by summarizing data: Trello's progress bar; WakaTime graphs show a) total logged time, b) time dedicated to different projects, c) today's logged time, and d) distribution of programming languages; Codealike visualizations show a) distribution of activities and b) statistics and total time per activity; and GitHub's frequency of contributions timeline matrix. (Source: Arciniegas-Mendez *et al.* [7])

erated since then—nowadays, Web 2.0 media are referred to as *'social media'*. The most important concept of Web 2.0 is participation, which is primarily made possible by lowering the barriers to entry. The proliferation of socially enabled tools and participatory media has led to the formation of global, virtual software development communities of practice, where groups of people are connected by the similarity of their activities [184]. Community members do not have to be spatially or socially connected, but they solve similar problems and learn from one another through processes like apprenticeship, mentoring, and legitimate peripheral participation.

Some examples of modern social media channels that developers use include public chat messaging systems (e.g., Slack), social coding hubs (e.g., GitHub), activity feeds and dashboards, forums, news feeds, Q&A websites, blogs, podcasts, and microblogging platforms such as Twitter. These channels address a variety of needs. For instance, they help facilitate discussions about day-to-day activities, provide ways for

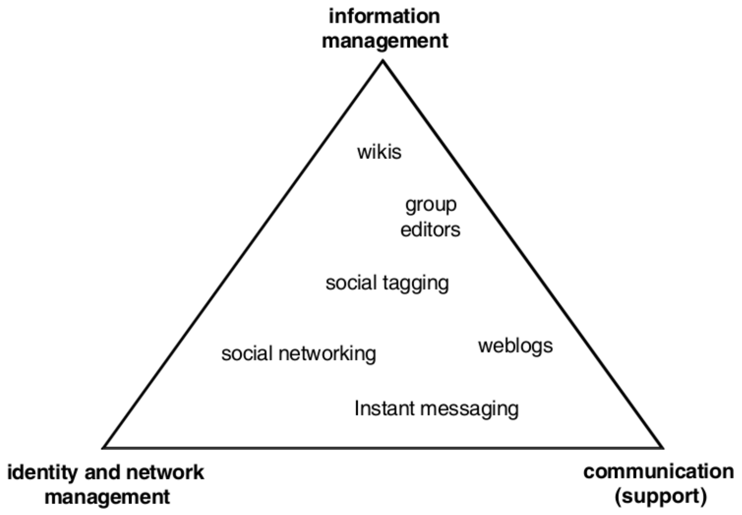


Figure 2.3: Social software triangle (source: Koch [78]).

developers to signal others [168], and allow people to broadcast and monitor information [142]. Following the style of the 3C triangle shown in Fig. 2.1, Koch [78] illustrated how social media can be positioned within a triangle of three core social concepts: communication support, information management, and identity and network management (see Fig. 2.3). In terms of cognitive support, these channels further extend the support they provide to include **community cognition** (e.g., built on membership and participation, facilitates socially constructed knowledge, focuses on problem-domain cognitive tasks).

In a sea of options, media choice is not the simple, intuitively obvious process it may appear to be at first glance. In fact, even in 1980-1990 when media options were much more limited (face-to-face interactions, letters, telephone calls, emails, memos, and bulletins), researchers strived to understand organizational media choice. For example, one study found that higher performing managers matched the equivocality of the message with the richness of the communication medium [167] (rich channels for more equivocal content and lean channels for less equivocal content). However, counter to earlier media richness studies [31, 127], more recent studies have not shown that matching media richness to task equivocality would improve performance. In an attempt to better understand media choice and media use, in particular *the media's ability to effect a change in a person's understanding of information*, Robert and Dennis [127] examined media richness from a cognitive perspective. They found that the use of rich media high in social presence (e.g., face-to-face interaction, formal group meetings) induces increased motivation but decreases the ability to process information, while the use of lean media low in social presence (e.g., email, fax, memo) induces decreased motivation but increases the ability to process information. From a cognitive perspective, this formed a paradox: rich media high in social presence simultaneously acts to both improve and impair performance. This encourages us to consider the cognitive foundations underpinning social interaction and strive to understand the impact of social media on the knowledge building process in developer communities.

Chapter 3

Research Methodology

"Scientific objectivity is not the absence of initial bias. It is attained by frank confession of it."

– Mortimer J. Adler, 1940

In this chapter, I discuss the overarching methodological choices and the rationale behind the studies that comprise this thesis. I begin with describing my epistemological worldview, and then discuss my research approach and research design. Note that each case's methodology is described in detail in the corresponding chapter later in the thesis.

3.1 My Worldview

There are different schools of thought dedicated to philosophical stances and their definitions. As a result, several terms are used to refer to a person's worldviews. Cresswell [25] uses the term **worldview**, while Lincoln and Guba [58] call it a **paradigm**. Crotty [26] refers to it as an **epistemology**, and others talks about broadly conceived research methodologies. These terms are used interchangeably and refer to a subset of a person's philosophical beliefs. Guba [57, p. 17] defined them as *"a basic set of beliefs that guide action"*. In other words, worldviews are philosophical orientations, and by using these beliefs about the world and the nature of research, researchers introduce them into their studies. These worldviews are shaped by the discipline area, beliefs, and past research experiences of the researcher. The types of beliefs held by individual researchers will influence their selection of qualitative, quantitative, or mixed-methods approaches in their research [24, 25]. However, the philosophical ideas and assumptions behind a given research product may not always be made explicit [145], yet they have an influence on the methodological choices made. To address this and help readers of this thesis better understand my approach and my research rationale, I identify and discuss my philosophical stance and worldview [25].

My own worldview, which has influenced the methodological choices behind my research, is that of a *constructivist* (also referred to as a *social constructivist*). Creswell defined it as follows:

> *"Social constructivists believe that individuals seek understanding of the world in which they live and work. Individuals develop subjective meanings of their experiences meanings directed toward certain objects or things. These meanings are varied and multiple, leading the researcher to look for the complexity of views rather than narrowing meanings into a few categories or ideas. The goal of the research is to rely as much as possible on the participants views of the situation being studied. Thus, constructivist researchers often address the processes of interaction among individuals. They also focus on the specific contexts in which people live and work in order to understand the historical and cultural settings of the participants. Researchers recognize that their own backgrounds shape their interpretation, and they position themselves in the research to acknowledge how their interpretation flows from their personal, cultural, and historical experiences. The researchers intent is to make sense of (or interpret) the meanings others have about the world. Rather than starting with a theory (as in postpositivism), inquirers generate or inductively develop a theory or pattern of meaning."* [25, p. 8]

Crotty [26] explains that constructivists see reality as *"contingent upon human practices, being constructed in and out of interaction between human beings and their world, and developed and transmitted within an essentially social context."* Thus, our understanding of the world comes from observations and understanding of human behaviors and social interactions. Another important aspect of constructivism is that it promotes theory generation. As opposed to a positivist worldview where researchers typically start with a theory, constructivists tend to generate or inductively develop patterns of meaning that eventually can lead to a theory.

3.2 Research Goal

My overarching goal for the research work described in this thesis is to *model the knowledge building process in software development* in order to better understand how communication and social media affect (and perhaps disrupt) software development. In Chapters 1 and 2, I motivated and demonstrated the need to pursue this goal. Moreover, the investigation of this topic is timely and significant for both the software engineering research community and industry.

In today's interconnected and software-dominated world, industry professionals and researchers care deeply about helping development teams be more productive, improve existing work processes, and cope with increasing demand and scale. Consequently, the research community has long studied communication media and its potential benefits to supporting modern work. Findings from these studies have produced valuable insights, however, these studies mostly focused on investigating individual mediums (i.e., in isolation from other communication tools or developer activities), and typically set the unit of analysis to be the medium itself.

In contrast, my goal is to consider *the ecosystem* of social and communication media , the interplay between the media channels, and their impact on developers, developer communities, and developer activities and practices. Understanding the role of social and communication media in software development is, of course, a complex problem. First, it requires one to map and understand *what* media channels developers use—outlining the ecosystem of social media and communication tools used in software development. It is also necessary to capture the context and rationale behind the reasons *why* these communication channels and tools are used. Then, it requires one to understand *what* purpose each channel serves and *what* development activities it supports. These research questions are not easily answered in isolation or without a deeper understanding of the underpinning processes of knowledge work.

I strive not only to understand the *'what'* aspects of the phenomena, but also the *'why'* and *'how'*. This requires designs and methods for contextual understanding of a plethora of perspectives [25] that are typically associated with qualitative research designs. For this reason, I have followed qualitative and mixed-methods designs and strategies in my line of inquiry.

3.3 Research Approach

The research presented in this thesis combines both theoretical and empirical work. Both of these research levels are vital: the theoretical level aims to describe natural phenomena in general terms, and the empirical level aims to describe phenomena observed in the real world. The choice between theoretical or empirical work is shaped by the researcher's worldview. Moreover, *the way* these scientific *"worlds"* are combined and interleaved characterizes and describes the conducted research type and the research approach taken [96]; e.g., a researcher can start either by deducing properties of the real world from a theory (following a deductive model), or by forming and generalizing theories based on observations (following an inductive model). Mackay and Fayard [96] show examples of how different types of research can be characterized by their research process. As a way to illustrate my research journey, I use Mackay and Fayard's framework and present a high-level view of my research process in Figure 3.1. Note that this is a simplified view of my research process. Its goal is to illustrate the high-level connections between the primary parts of my thesis, and how my theoretical and empirical work was combined. However, this illustration doesn't show all the connections, nor the iterative and emerging nature of my work. For the sake of clarity, these will be described in detail in chapter 6.

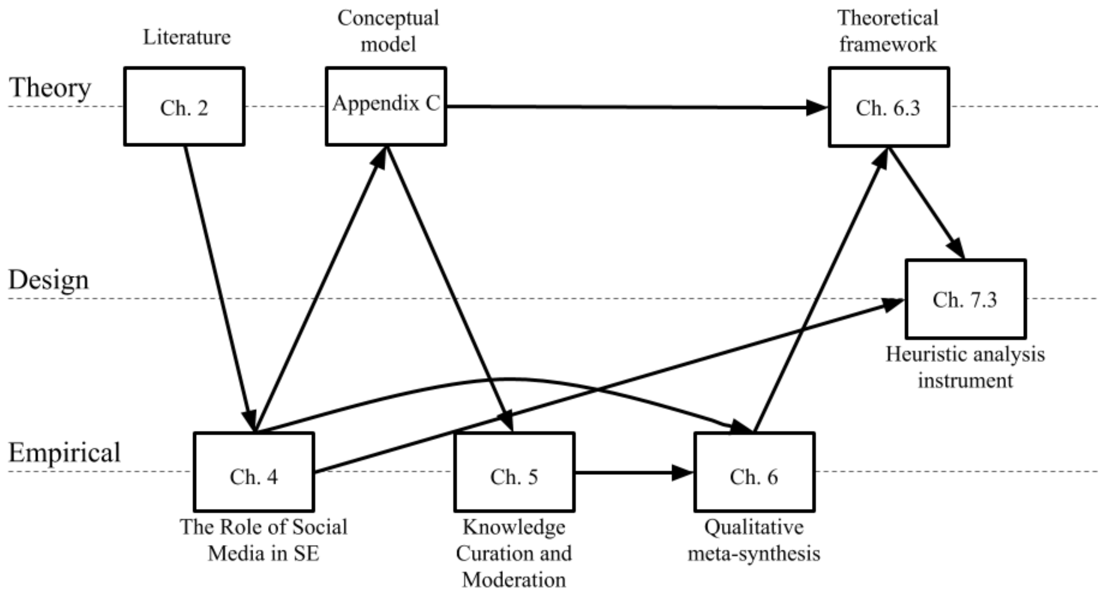

Theory
Design
Empirical
Literature
Ch. 2
Conceptual model
Appendix C
Theoretical framework
Ch. 6.3
Ch. 7.3
Heuristic analysis instrument
Ch. 4
The Role of Social Media in SE
Ch. 5
Knowledge Curation and Moderation
Ch. 6
Qualitative meta-synthesis

Figure 3.1: A high-level view of my research process. Each component is annotated with the corresponding chapter number.

3.4 Research Methods and Design

Crotty [26] suggests that, in describing research, we need to put considerable effort into answering two main questions: what methods will we be employing in the research, and how do we justify this choice and use of methods. I have answered the latter by describing my worldview and research goal, and now I will address the former.

Given my worldview and research goal, I have chosen qualitative and mixed-methods modes of inquiry. In the studies that comprise this thesis, I have used a selection of methodologies and strategies [99, 25]:

- **Qualitative survey** – a sample survey strategy (Chapter 4)
- **Mixed-methods exploratory case study** – a field study strategy (Chapter 5)
- **Qualitative meta-synthesis** – a formal theory strategy (Chapter 6)

Research methods involve the forms of data collection, analysis, and interpretation for studies. Usually with qualitative methods, researchers allow behaviors and perspectives to emerge, and the data collection and analysis methods reflect that. As I discuss in detail in the following chapters, I collected data through participant surveys and mining interaction data from archives.

3.5 Researcher Location

There is a responsibility on the researcher to the group and community being studied, because the perspective a researcher has impacts the knowledge produced about that group. In the studies described in this thesis, I have been a *participant-observer* within the communities under study, and a *user* of the tools and technologies investigated. In traditional ethnographic terms [49, 100, 66, 55, 104], I have simultaneously been an *insider* and an *outsider*[1].

The reason I identify myself as an *insider* is because I am a software developer, and I have used and continue to use most of the modern tools and communication media that we investigated (e.g., Slack, GitHub, Stack Overflow, Twitter). For example, we use Slack as the main communication medium in our research lab, so I'm familiar with all of its features and capabilities (and have experienced some of the pains associated

[1]In anthropology and the social and behavioral sciences, these terms are often referred to as *Emic* (insider) and *Etic* (outsider).

with using it, e.g., information overload). Moreover, I am a member in many of the social media communities we studied (e.g., Hacker News, Stack Overflow, GitHub). The 'insider' viewpoint granted me a deeper understanding and allowed me to record richer information about the phenomena. However, it also shaped my bias on these tools and communities.

Similarly, I identify myself as an *outsider* because I am neither a direct member of the studied communities nor had I any impact on our sample set. For example, despite being a user on Stack Overflow and familiar with R, I was not a member of the R developer community on Stack Overflow. This, allowed me to mitigate the risk of *reactivity*, i.e., reducing my influence on what is being observed.

I believe that a combination and a balance of both perspectives, the insider and the outsider, is needed to understand the impact of socio-technical systems on existing processes and behaviors. A researcher needs to *"move back and forth between involvement and detachment"* [120, 187]. It is imperative that the researcher involve themselves, because a close familiarity with the subject(s) helps provide a deeper understanding of the studied culture, and helps prevent cultural biases during data interpretation [160]. At the same time, the researcher *"must always remember her/his primary role as a researcher and remain detached enough to collect and analyze data relevant to the problem under investigation"* [9].

Part II

Empirical Studies

Chapter 4

Understanding How Social and Communication Media Affect and Disrupt Software Development

"We shape our tools, and thereafter our tools shape us."

– John M. Culkin, 1967

The rich and varied ecosystem of communication media and tools that developers use has grown significantly in the last decade (as illustrated in Chapter 2). Developers habitually use and rely on communication channels such as Stack Overflow, GitHub, email, forums, Wikis, Jira, Trello, and other tools that incorporate or supplemented by communication capabilities. These channels support the developers' work and collaboration needs, help reduce coordination barriers (e.g., geographical, temporal, and organizational), and blur organizational boundaries. They help developers discover important technological trends, co-create with other developers, and learn new skills. Furthermore, these social tools foster creativity, promote engagement, and encourage participation in development projects. However, not much is known about the impact of communication media adoption and use on software development practices, velocity, and software quality.

In this study, we wanted to determine the activities developers do as part of their work (e.g., learning, discovery, and collaboration), understand what channels and tools developers use to support their activities, and understand how these channels support or hinder their work. Specifically, this study has been been guided by the following research questions:

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

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

RQ3 What communication channels are the most *important* to developers and *why*?

RQ4 *What challenges* do developers face using an ecosystem of communication channels to support their activities?

To achieve our research goal, we have deployed and conducted a large-scale survey of software developers that participate in communities of practice and most likely to adopt social and communication media for their work. In two rounds of surveys, we surveyed developers that are active on GitHub: once at the end of 2013 (with 1,492 responses), and once at the end of 2014 (with 332 responses). Ignoring incomplete submissions, we had a total of 1,449 survey responses.

In the survey, we inquired about developers' communication channel use for a preliminary set of 11 development activities: staying up to date, finding answers, learning, discovering, connecting, getting and giving feedback, publishing activities, watching other developers' activities, displaying skills and accomplishments, assessing other developers, and coordinating. The survey also included questions on which channels developers find most important for supporting their work, and asked them to describe the challenges they faced.

By using the collected data, we mapped and associated different communication channels to each activity. Our results show that developers use a plethora of communication media to support their development activities, and our findings further emphasize that developers engage in non-coding yet essential activities, such as learning and keeping up to date. Code hosting sites, face-to-face, question & answer sites, and web search were the top most important channels described in the survey. However, other channels were also deemed important by the participating developers. The most commonly cited reasons were the channel's support of group awareness, collaboration, allocation and retrieval of information, and its ability to enhance dissemination or consumption of information.

This work has helped us form an initial mental model of knowledge transfer in software development (see Appendix C) and established a preliminary taxonomy of knowledge activities for software developers (more on this in Chapter 6).

4.1 Methodology

Our overarching research goal is to understand how communication channels and social media affect and disrupt software development. To help realize this goal, we designed and conducted an online **survey** to learn how developers use communication media to support their knowledge activities, what media channels are important to them, and what challenges they face.

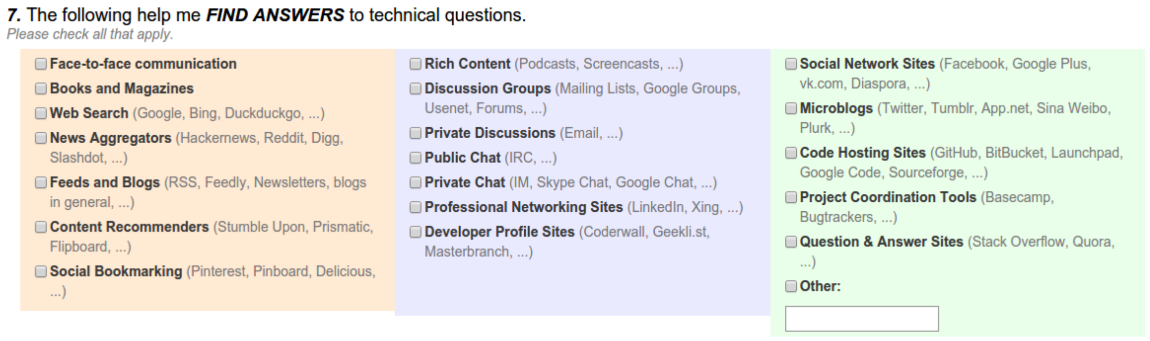
The design of the survey included several iterations and was based on an in-depth review of the existing literature on software engineering as well as related literature on knowledge work. In the survey[1], we first inquired about the developers' **demographics**. We then inquired about **communication channel use** for a set of 11 **development activities**. The set of activities was informed by our review of the literature that examines tool and communication channel use by software developers [156]. These activities, which go beyond finer grained development and project management activities, were as follows:

1. **Stay up to date** about technologies, practices, and tools for software development
2. **Find answers** to technical questions
3. **Learn** and improve skills
4. **Discover** interesting developers
5. **Connect** with interesting developers
6. **Get** and **give feedback**
7. **Publish** development activities
8. **Watch** other developers' activities
9. **Display** my **skills/accomplishments**
10. **Assess** other developers
11. **Coordinate** with other developers when participating on projects

The survey questions relating to activities all used the same structure: each of these questions provided a matrix of channel options, where respondents could select all the channels that apply to indicate that they were used for the corresponding activity (see Fig. 4.1 for an example of an activity question). The social and communication channels specified in the matrix were selected from our own knowledge as developers, as well as through feedback from fellow developers. The channels were refined using the research literature and through piloted surveys. We provided an

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

option to select "Other" in order to elicit channels we did not consider. Additionally, we asked developers to rank the **most important tools and channels** they used to support development activities and explain why those tools were important.


7. The following help me ***FIND ANSWERS*** to technical questions.
Please check all that apply.

- ☐ **Face-to-face communication**
- ☐ **Books and Magazines**
- ☐ **Web Search** (Google, Bing, Duckduckgo, ...)
- ☐ **News Aggregators** (Hackernews, Reddit, Digg, Slashdot, ...)
- ☐ **Feeds and Blogs** (RSS, Feedly, Newsletters, blogs in general, ...)
- ☐ **Content Recommenders** (Stumble Upon, Prismatic, Flipboard, ...)
- ☐ **Social Bookmarking** (Pinterest, Pinboard, Delicious, ...)
- ☐ **Rich Content** (Podcasts, Screencasts, ...)
- ☐ **Discussion Groups** (Mailing Lists, Google Groups, Usenet, Forums, ...)
- ☐ **Private Discussions** (Email, ...)
- ☐ **Public Chat** (IRC, ...)
- ☐ **Private Chat** (IM, Skype Chat, Google Chat, ...)
- ☐ **Professional Networking Sites** (LinkedIn, Xing, ...)
- ☐ **Developer Profile Sites** (Coderwall, Geekli.st, Masterbranch, ...)
- ☐ **Social Network Sites** (Facebook, Google Plus, vk.com, Diaspora, ...)
- ☐ **Microblogs** (Twitter, Tumblr, App.net, Sina Weibo, Plurk, ...)
- ☐ **Code Hosting Sites** (GitHub, BitBucket, Launchpad, Google Code, Sourceforge, ...)
- ☐ **Project Coordination Tools** (Basecamp, Bugtrackers, ...)
- ☐ **Question & Answer Sites** (Stack Overflow, Quora, ...)
- ☐ **Other:**

Figure 4.1: An example of the channel matrix we used to inquire about communication media used for each of the 11 development activities. We designed the channel matrix for the survey.

We also aimed to understand what **challenges** developers face using social channels, probing about privacy, interruptions, and feeling overwhelmed, as these were concerns that came up in earlier studies conducted with adopters and non-adopters of social media (e.g. Singer *et al.* [142]). The survey instrument itself is one of the contributions of this study and its source code can be found in a repository on GitHub[2]. This allows others to replicate our survey and build upon our work.

We deployed the survey in two iterations: we emailed a survey invitation to 7,000 active GitHub users during November and December of 2013, and to 2,000 active GitHub users in December of 2014. To find developers for our survey, we downloaded account data for the most recently active GitHub users with public email addresses, and used GitHub Archive[3] to query public events on GitHub. As an indicator for level of activity, we used the 25 event types defined by the GitHub API v3[4]. Most of these events concern development tasks such as committing code, creating repositories, and creating issues, but there are also more general events related to following users and watching repositories. We sorted events by their timestamp and excluded users who did not have public email addresses at the time we sent our invitation emails. For the second iteration, we also ignored users we had emailed in the first iteration. We

[2]`https://github.com/thechiselgroup/devsurvey`
[3]`https://www.githubarchive.org`
[4]`https://developer.github.com/v3/activity/events/types`

focused on this population of developers because GitHub is currently the most widely used social coding platform by developers who contribute to one or more collaborative development projects in an open manner[5].

1,492 and 332 developers responded to the two instances of the survey in 2013 and 2014, respectively (21% and 16% response rates). The only statistical difference between the two deployments was an increase in the number of women (from 3.5% to 6.3%, $\rho = 0.042$). We combined the responses from both surveys and ignored incomplete ones, resulting in a total of 1,449 survey responses.

Next, we describe our findings and answer our research questions as we illustrate the prevalence of social and communication media use among software developers.

4.2 Characterizing The Social Developer

First, we wanted to get a sense of the demographics of the developers that are active on social channels like GitHub. In the first part of the survey, we asked participants to provide demographic information about their gender, age, geographical location, programming experience, the programming languages they use, the number of projects they participate in, whether they program professionally, and the size of the project teams they have worked with. Figures 4.2 and 4.3 show a summary of the developer demographics.

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

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

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

[5]`https://octoverse.github.com`
[6]`http://meta.stackoverflow.com/a/281304`

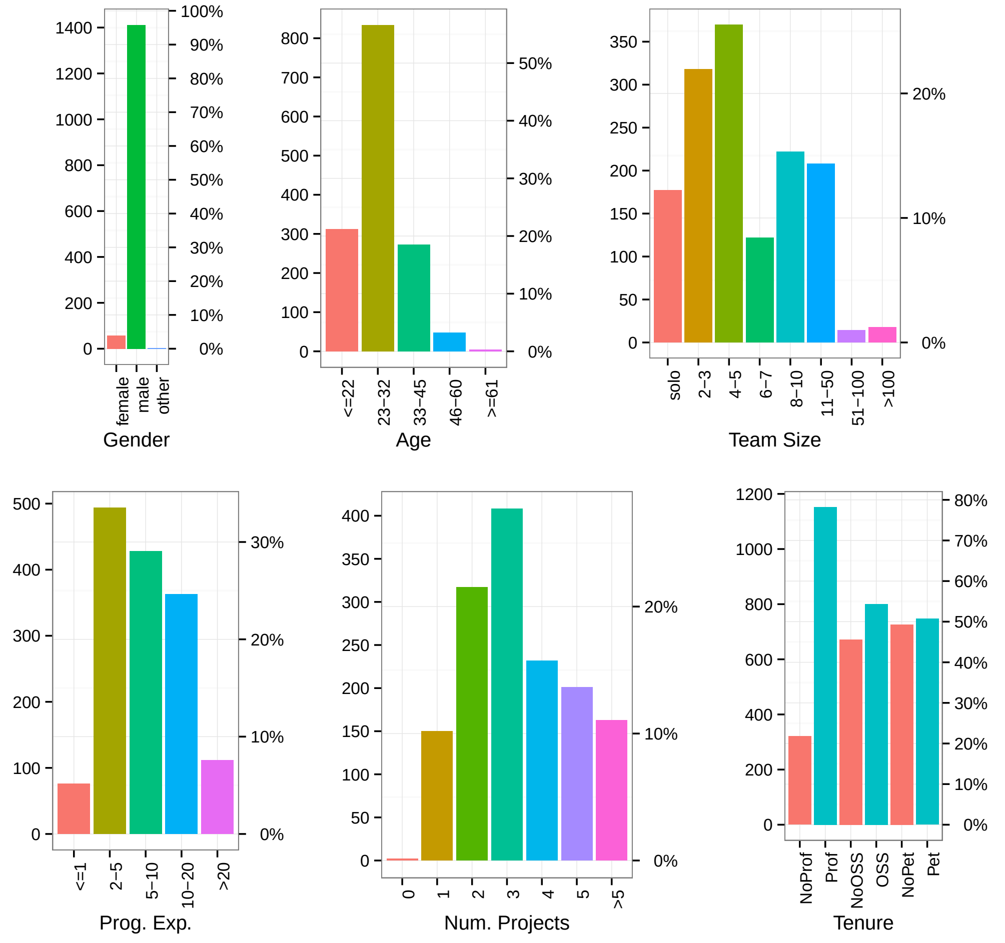


Figure 4.2: Demographics of the programmers that answered the survey (those recently active on GitHub with public activity).

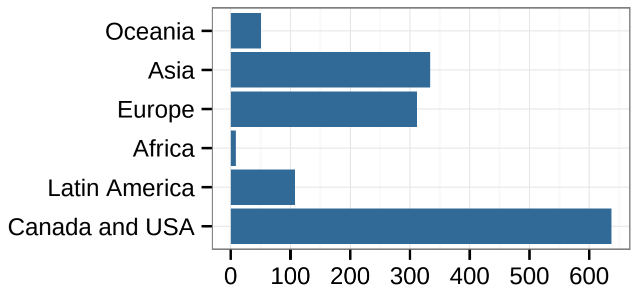


Figure 4.3: Geographical location of the developers that participated in the survey.

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

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

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

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

Programming languages: Table 4.1 shows the most popular programming languages. The three most popular languages included JavaScript (61.9%), Python (44.6%), and Java (41.5%). This may indicate that at least 60% of our respondents develop for the Web.

Table 4.1: Most popular languages used by developers that participated in our survey.

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

RQ1: Who is the social programmer that participates in these communities?
Finding: Our results show that modern social developers come from diverse geographic locations, yet there is a skew towards professional, younger developers (millennials), who identify themselves as males, and work in smaller teams (up to 5 members). We saw no evidence of differences in the channels used or how they were used across the varied developer demographics.

4.3 Communication Media as Facilitators of Developer Activities

Software developers use a large assortment of social and communication media. Our findings showed that on average, developers indicated they use 11.7 channels across all activities, with a median of 12 and quartiles of $[9, 14]$ (see Fig. 4.4 for the distribution of channels used by survey respondents).

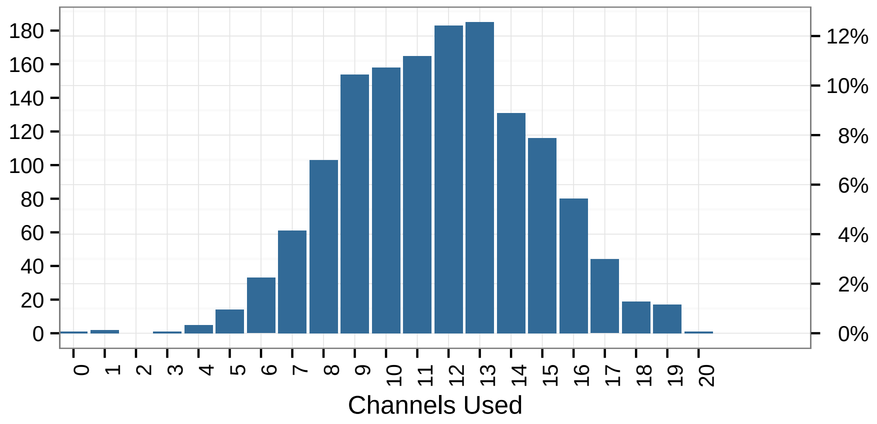


Figure 4.4: A histogram of the number of communication channels developers use.

Beyond revealing that developers rely and make use of a high number of communication channels, our results also mapped the specific channels developers use to support different activities. Figure 4.5 shows an overview of the channels developers use to support each of their activities. For example, the results show that developers primarily **find answers** through **Question & Answer sites** and **Web search**, while for **publishing their activities** developers primarily use **Code hosting sites**. For a more detailed view, we also present the channel use distribution for each activity via radar charts (see Fig. 4.6). However, in addition to the social and communication channels we asked about, developers participating in our survey have also indicated other channels (e.g., events and meetups, software documentation, and personal blogs and Websites), which we haven't asked directly about. We aggregated these additional channels in Table 4.2.

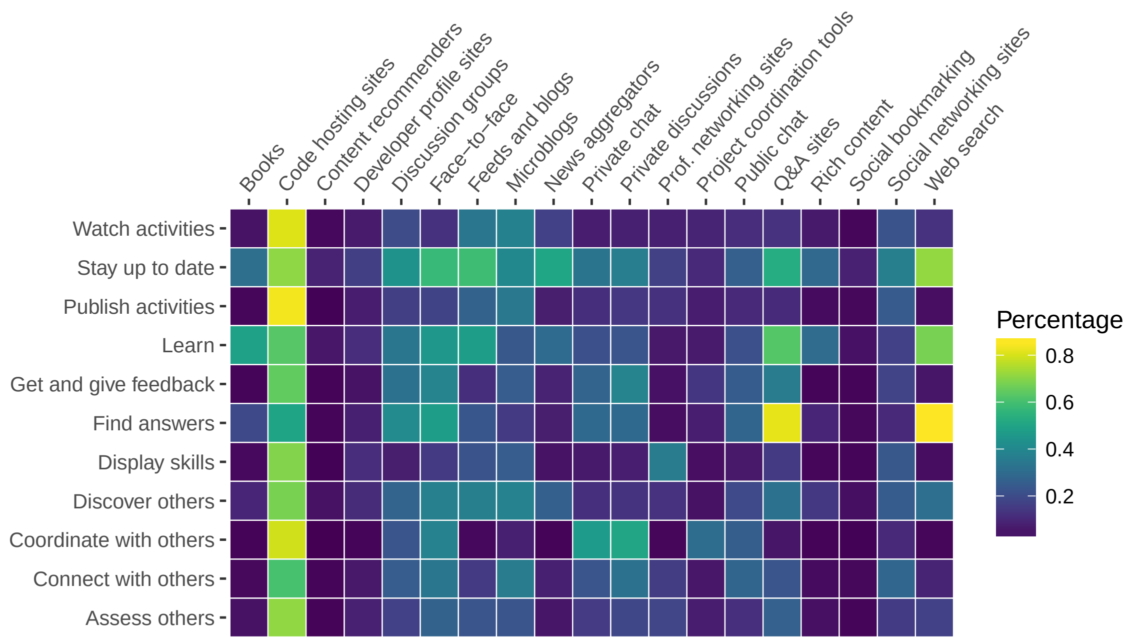
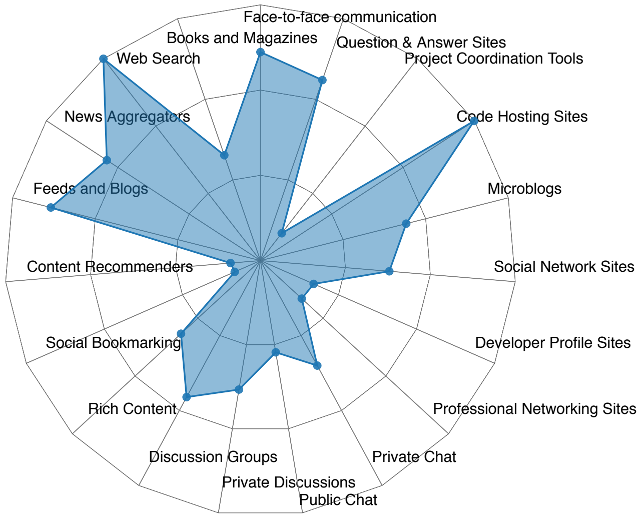


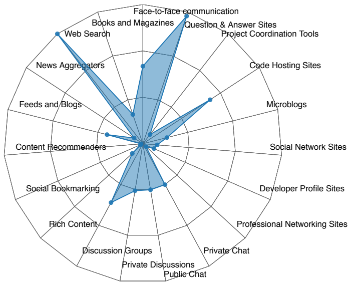
Figure 4.5: Channels used by software developers and the activities they support. Results are based on a survey with 1,449 developers [155].

Table 4.2: Other channels reported in the survey. The values indicate the number of times the channel was mentioned by respondents for the corresponding activity.

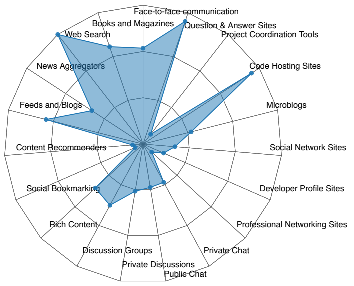
Activity	Channels
Keeping up to date	events and meetups (18), software documentation (3), research papers (3), formal education (2), MOOCs (2)
Finding answers	software documentation (11), research papers (3)
Learning	educational sites and MOOCs (20), events and meetups (19), software documentation (13),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



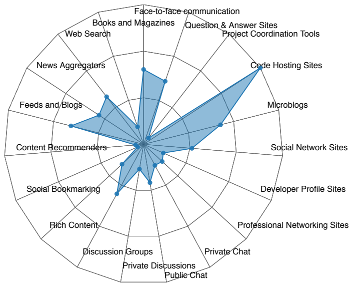
(a) Stay up to date



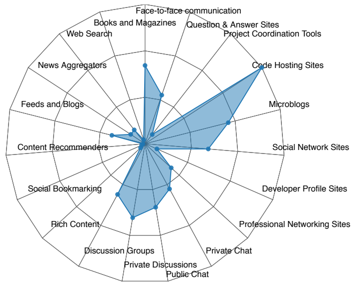
(b) Find answers



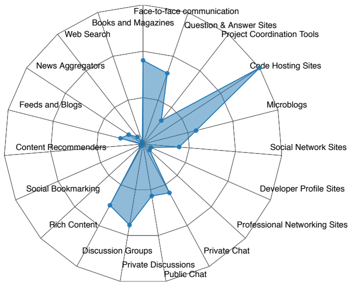
(c) Learn



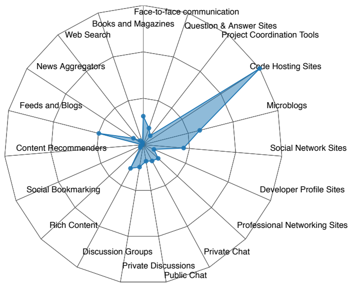
(d) Discover others



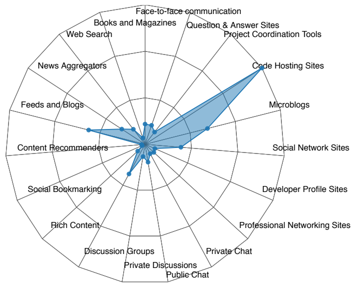
(e) Connect with others



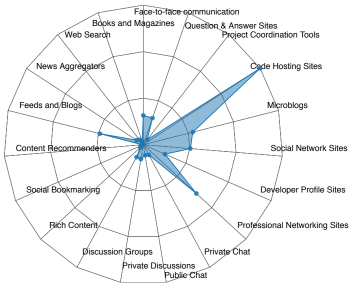
(f) Get and give feedback



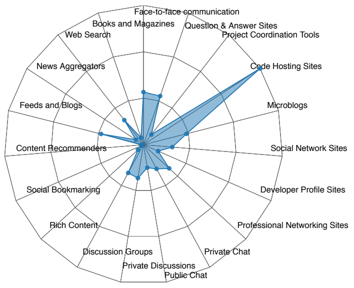
(g) Publish activities



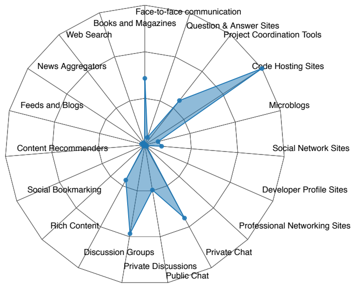
(h) Watch activities



(i) Display skills



(j) Assess others



(k) Coordinate with others

Figure 4.6: Channel use per activity shown in the form of radar charts.

RQ2: What communication channels do these developers use to support development activities?
Finding: Software developers use a large assortment of communication media: face-to-face, books and magazines, web search, news aggregators, feeds and blogs, content recommenders, social bookmarking, podcasts, discussion groups, public and private chat tools, professional networking sites, developer profile sites, social networking sites, code hosting sites, microblogs, project coordination tools, and Q&A sites. Moreover, developers use other, less obvious channels, such as events and meetups, documentation, and personal blogs and Websites. In terms of activities, developers engage not only in coding activities but also in other important activities, such as learning and keeping up to date.

Although the findings above show a high level overview of the channels developers tend to use for supporting specific activities, a limitation with that data is that we only know that developers use the selected channels for supporting a given activity—we don't know how important these channels are. Consequently, we asked the respondents to indicate the top three channels that are important to them and why each one is important. Figure 4.7 shows the number of responses given per channel.

Moreover, the open-ended survey responses shed light into *why* certain channels were perceived as important. We analyzed and summarized the survey responses, and extracted the reasons for each channel's importance. We demonstrate these reasons through quotes from specific participants, indicated by P#.

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

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

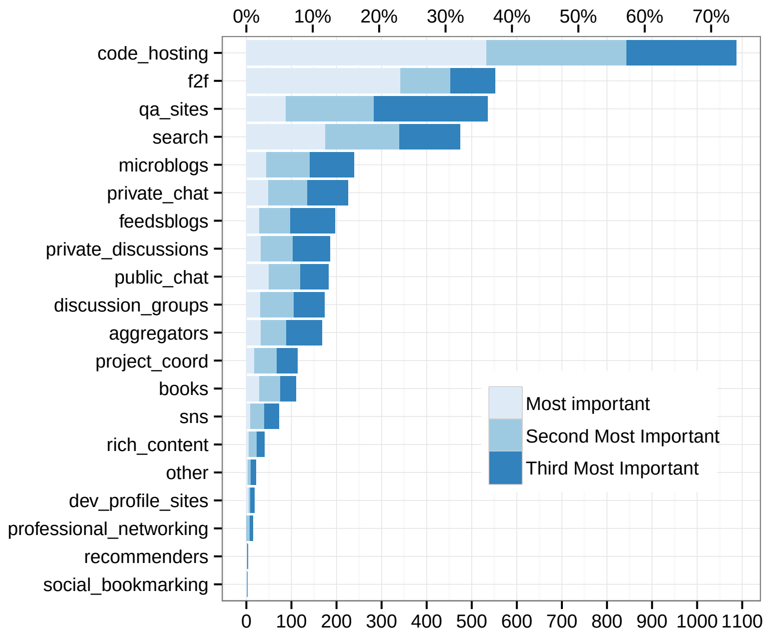


Figure 4.7: Number of responses per channel indicating the importance of each channel.

co-located interactions with other developers.

Q&A sites offer a quick way to debug issues while providing access to high-quality answers: *"Almost any question that I have, I can get an answer through these sites."* [P635] Other respondents mentioned additional uses of Q&A sites, including learning from code examples and getting feedback from experts.

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

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

Private chats (e.g., IM, Skype chat, Google chat) are essential tools for supporting team communication and collaboration through a single channel: *"It provides a single channel to digest and discuss everything that is going on with the team."*

[P415] Many survey respondents felt that private chats are the closest replacement for face-to-face interactions when quick feedback is needed and team members are geographically distributed.

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

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

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

Discussion groups (e.g., mailing lists, Google groups, forums) support mass communication and coordination among people scattered across large and geographically distributed groups: *"We are a physics collaboration of 3000 people, spread all over the world. Internal discussion groups are essential for coordination on all subjects, including software development."* [P365] Respondents also reported the usefulness of discussion groups for gathering customer feedback: *"Because it's where I find my customers' opinions and ideas."* [P130]

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

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

Books were indicated by some of our survey respondents as a cohesive and progressive way for learning about a topic: *"[They make] learning much easier than the hunt and peck method of digging through sites on the net."* [P1189] Another subtle but crucial advantage of books is that they are *"distraction free and generally better thought through and considered."* [P1319] Developers can gain in-depth and focused understanding about specific topics, while avoiding being distracted by the noise of concurrent information.

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

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

RQ3: What communication channels are the most important to developers and why?
Finding: Developers have a spectrum of channels they find important for supporting their activities. The top channels that were classified as most important are: Code hosting sites, face-to-face, Q&A sites, and Web search. However, other channels were also deemed important by the participating developers. The most commonly cited reasons were the channel's support of group awareness, collaboration, allocation and retrieval of information, and its ability to enhance dissemination or consumption of information.

4.4 Mapping The Challenges of Communication Media in Software Development

Modern social media and communication channels are changing the way software is developed and maintained. These channels foster digital habitats, support communities of practice, and allow organizations to operate at a competitive velocity and scale. Social media tools such as Slack, GitHub, Stack Overflow, and chatbots bridge technical, organizational, and geographical gaps, thereby reducing boundaries between the organization and its external environment, and getting stakeholders more involved in the development process (e.g., developers, designers, users, suppliers). However, the adoption and combination of these channels disrupt conventional processes and have inherent challenges, resulting in underutilization, productivity loss, and sub-optimal knowledge flow.

> *"It would be a mistake to regard the new generation of information and communication technologies as neutral tools that can merely be grafted onto existing work systems...These technologies have been found to disrupt conventional practices. The way the technologies intimately interlace with the minutiae of everyday practices is exposing processes which, previously were taken for granted, ignored, or misunderstood."*
>
> – Frank Blackler, 1995

Thus in this study, we were interested to also inquire about the challenges developers face using communication channels. Previous work [140, 142, 156] revealed that developers face challenges related to distractions, privacy, and feeling overwhelmed by communication chatter when using social media channels. Hence, in the survey we explicitly asked if developers experienced these challenges. Our results show that privacy is not a big concern for everyone, whereas being interrupted and feeling overwhelmed are more prominent challenges among the developers (Figure 4.8 shows the results of these Likert-style questions).

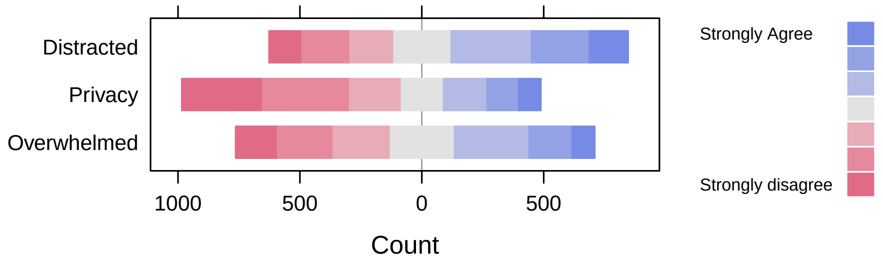


Figure 4.8: Frequency of responses to Likert questions probing on developer challenges with DISTRACTION, PRIVACY, FEELING OVERWHELMED.

Table 4.3: Test of independence between the different demographic factors and whether respondents feel worried about privacy, feel overwhelmed, or are distracted by their use of communication channels. Each value is preceded by the name of the test used followed by its results: *kw* represents Kruskall-Wallis (degrees of freedom, χ^2 value), and *sp* represents Spearman correlation (r value). Values in bold represent when the two factors appear not to be independent with $p < 0.05$, specifically *** corresponds to $p < 0.001$, ** for $p < 0.01$, * for $p < 0.05$

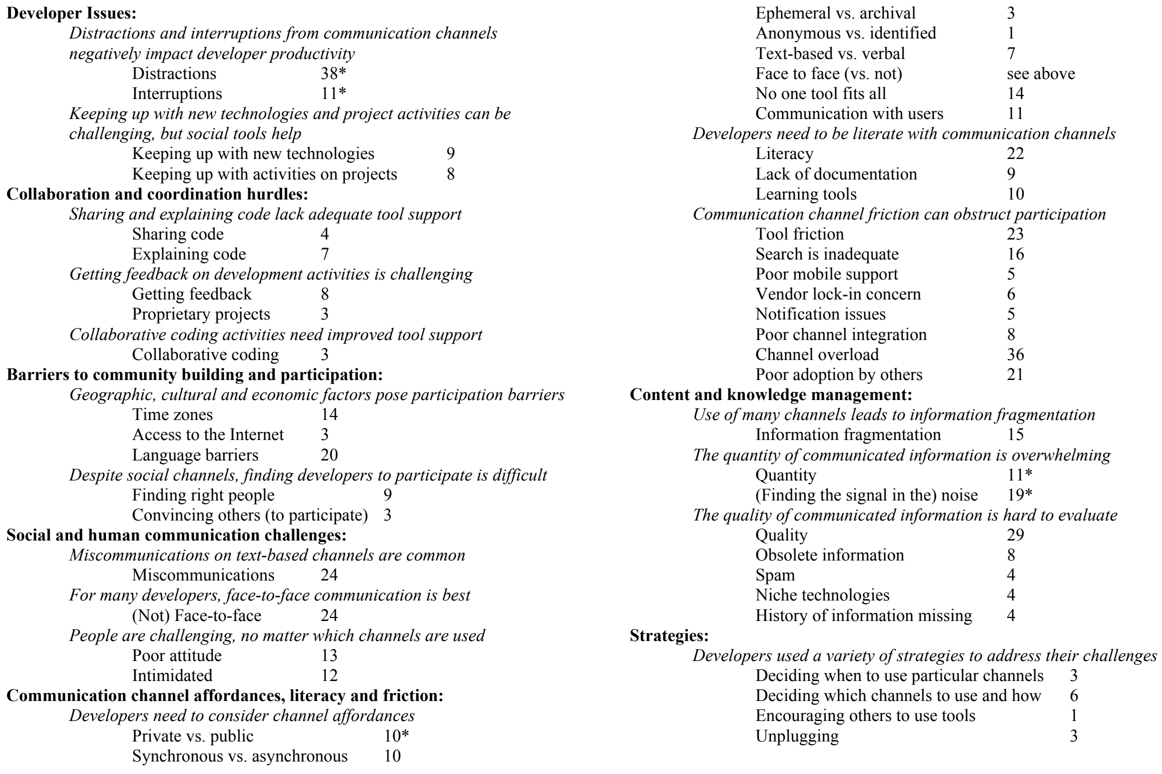
	Age	Gender	Team Size	Prog. Exp.	Tenure Prof	Tenure Pet	Tenure OSS	Number Projects	Channels Used	Privacy	Over-whelmed
Privacy	sp: -0.04	kw: 6,8.26	sp: 0.01	sp: -0.04	kw: 6,10.69	kw: 6,11.39	kw: 6,8.97	sp: - 0.04	sp: 0.03		
Over-whelmed	sp: **0.07****	kw: 6,9.36	sp: - 0.01	sp: **-0.07***	kw: 6,11.71	kw: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

To investigate if there were any relationships between these three factors (Privacy, Distraction, Overwhelmed) and the participants' demographics, we performed a more in-depth analysis of the responses. Table 4.3 shows the test of independence between whether a person feels their privacy is affected or not and if they feel overwhelmed or distracted by their use of communication channels, as well as the different demographic factors of our respondents. We anticipated that age might influence responses in terms of privacy concerns, but no factor shows a statistically significant relationship with the Privacy factor. A similar result was found regarding the Distraction factor, where the only statistically significant result was that there is very little correlation (if any) with programming experience: $\rho = 0.07$, $p = 0.008$. The Overwhelmed factor was found to have a very low correlation (if any) to: age ($\rho = 0.07$, $p = 0.014$), programming experience ($\rho = -0.07$, $p = 0.012$), and number of projects ($\rho = -0.07$, $p = 0.004$). It was also found that people who work on open source projects feel slightly more overwhelmed than people who do not (H=19.89, df=6, $p = 0.005$). We believe these results show a lack of evidence that the developers who worry about privacy, feel overwhelmed, or feel distracted belong to any specific type of group (as reported in the survey). Nonetheless, it is notable that there is a modest positive correlation between the three factors (with $p \ll 0.001$): people who worry about their privacy feel overwhelmed ($\rho = 0.21$) and distracted ($\rho = 0.19$), and those who feel distracted also feel overwhelmed ($\rho = 0.24$).

Beyond these three factors, the rich survey responses to an open-ended question on challenges helped us identify and categorize different challenges developers face. Figure 4.9 shows a high level overview of the challenges we found. Full details on the challenges are provided by Storey *et al.* [155]. Later in the thesis (Chapter 7), I follow up on and operationalize these challenges—I provide a practical inspection method in the form of a *heuristic analysis* for revealing and mapping knowledge sharing challenges when using social media and communication channels.

RQ4: What challenges do developers face using an ecosystem of communication channels to support their activities?

Finding: Developers struggle to handle interruptions, and to effectively collaborate with others. They experience coordination, communication, and social challenges, and face barriers to community building and participation. Communication channel affordances, literacy, and friction further add to these challenges. Moreover, the use of many communication channels leads to knowledge fragmentation, overwhelmingly large quantities of information, and difficulties in evaluating the quality of information.



Developer Issues:			
	Distractions and interruptions from communication channels negatively impact developer productivity		
		Distractions	38*
		Interruptions	11*
	Keeping up with new technologies and project activities can be challenging, but social tools help		
		Keeping up with new technologies	9
		Keeping up with activities on projects	8
Collaboration and coordination hurdles:			
	Sharing and explaining code lack adequate tool support		
		Sharing code	4
		Explaining code	7
	Getting feedback on development activities is challenging		
		Getting feedback	8
		Proprietary projects	3
	Collaborative coding activities need improved tool support		
		Collaborative coding	3
Barriers to community building and participation:			
	Geographic, cultural and economic factors pose participation barriers		
		Time zones	14
		Access to the Internet	3
		Language barriers	20
	Despite social channels, finding developers to participate is difficult		
		Finding right people	9
		Convincing others (to participate)	3
Social and human communication challenges:			
	Miscommunications on text-based channels are common		
		Miscommunications	24
	For many developers, face-to-face communication is best		
		(Not) Face-to-face	24
	People are challenging, no matter which channels are used		
		Poor attitude	13
		Intimidated	12
Communication channel affordances, literacy and friction:			
	Developers need to consider channel affordances		
		Private vs. public	10*
		Synchronous vs. asynchronous	10
		Ephemeral vs. archival	3
		Anonymous vs. identified	1
		Text-based vs. verbal	7
		Face to face (vs. not)	see above
		No one tool fits all	14
		Communication with users	11
	Developers need to be literate with communication channels		
		Literacy	22
		Lack of documentation	9
		Learning tools	10
	Communication channel friction can obstruct participation		
		Tool friction	23
		Search is inadequate	16
		Poor mobile support	5
		Vendor lock-in concern	6
		Notification issues	5
		Poor channel integration	8
		Channel overload	36
		Poor adoption by others	21
Content and knowledge management:			
	Use of many channels leads to information fragmentation		
		Information fragmentation	15
	The quantity of communicated information is overwhelming		
		Quantity	11*
		(Finding the signal in the) noise	19*
	The quality of communicated information is hard to evaluate		
		Quality	29
		Obsolete information	8
		Spam	4
		Niche technologies	4
		History of information missing	4
Strategies:			
	Developers used a variety of strategies to address their challenges		
		Deciding when to use particular channels	3
		Deciding which channels to use and how	6
		Encouraging others to use tools	1
		Unplugging	3

Figure 4.9: The categories, codes, and counts of each code occurrence in the participant responses to the open-ended challenges question (source: Storey *et al.* [155]). Codes marked with an * indicate challenges the participants already indicated in the closed question. Note that some participants shared multiple challenges, and even though we provide code counts, we caution that counting the coded challenges could be misleading—only some participants took the time to share this information with us after an already long survey, and thus they may have selected which challenges to share with us in an *ad-hoc* manner. Nevertheless, for concerns that were mentioned numerous times, the counts may help us identify challenges that may be more prevalent and warrant further investigation—we share these counts in hopes of provoking future research.

Chapter 5

Knowledge Curation Within a Community

"Quality, relevant content can't be spotted by an algorithm. You can't subscribe to it. You need people - actual human beings - to create or curate it."

– Kristina Halvorson, 2009

The emergence and adoption of socially enabled tools and channels (e.g., GitHub, Stack Overflow, mailing lists) has fostered the formation of large *communities of practice* where users share a common interest, such as programming languages, frameworks, and tools [156]. These communities rely on many different communication channels, but little is known about how they create, share, and curate knowledge using such channels.

One prominent community of practice is the group that has formed in support of the R programming language, an open source project without commercial backing that relies heavily on its rapidly growing and highly heterogeneous software development community. The R community plays an important role in diffusing the R language: users have access to numerous resources for learning the language and receiving help, such as mailing lists, blogs, books, online and offline courses, and question & answer sites (e.g., Stack Overflow). While the R community benefits from this vast and rich corpus of knowledge, it also drives the creation and curation of the information.

Without a single entity directing and controlling it, the R language has grown organically from its community. Similar to other communities of practice, knowledge is exchanged and curated in many communication channels, and two particular communication channels are at the center of this process: the *R-help mailing list* and *Stack Overflow*. The R-help mailing list was created to assist those using the language, and while Stack Overflow is not specifically oriented towards R, its section dedicated to R (the R tag) has grown rapidly[1].

Stack Overflow has revolutionized the way programmers seek knowledge [92, 170], assuming the role of a capable "expert on call" that is able—and willing—to answer questions of any level of difficulty about any programming technology (R included). Stack Overflow's gamification features normally guarantee that enthusiastic experts will answer questions, often within minutes of being posted [97]. Equally important is the ability of Stack Overflow's users to curate the knowledge being created, making sure that the best answers surface to the top and become a valuable asset to those seeking an answer now or in the future. Stack Overflow has become a popular and effective tool for creating, curating, and exchanging knowledge, including knowledge about the R language.

One would expect that the traffic on the R-help mailing list would begin to fizzle as Stack Overflow popularity increased. If Stack Overflow is so effective at matching those who seek knowledge with those that have it, doesn't that obviate most of the need for the R-help mailing list? Yet that does not appear to be the case as the R-help mailing list has maintained a steady level of activity, implying that it is still an important resource for the R community. In fact, it appears as if the mailing list and Stack Overflow complement each other.

There are obvious inherent differences between both communication channels. On the one hand, mailing lists unite users by subscription, creating a tight community, but their content lacks organization, except for the natural structure provided by email metadata (e.g., subjects, threading, authors, dates), and they are not optimized for long-term storage and retrieval. On the other hand, Stack Overflow's community is not as tight as the R-help mailing list community, but the channel is optimized for the curation and long-term storage of knowledge. However, little is known about the differences in how people use both communication channels, such as how the types of questions and answers sought in one channel compare to the other, why users choose one channel over the other, why some users participate in both channels, and how

[1]http://www.r-bloggers.com/r-is-the-fastest-growing-language-on-stackoverflow/

participants perceive each communication channel.

In this study, we first focus on characterizing knowledge artifacts. We empirically compare how knowledge, specifically knowledge manifested as questions and answers, is sought, shared, and curated on both the R-help mailing list and Stack Overflow. We then build on these findings and focus on the knowledge curation process in the R community. We examine the participation patterns and behavior of users in both sub-communities, and seek to learn more about community's prolific members and knowledge curators. Our research employed a mixed methods *exploratory case study* methodology to answer the following research questions:

RQ1: What types of knowledge artifacts are shared on Stack Overflow and the R-help mailing list within the R community?

RQ2: How is the knowledge constructed on Stack Overflow and the R-help mailing list?

RQ3: Why do users post to a particular channel and why do some post to both channels?

RQ4: How do users participate on both channels over time?

RQ5: Are there significant differences in participation activity between community users?

By mining archival data, we identified and categorized the main types of knowledge artifacts found on the R-help mailing list and in Stack Overflow (RQ1). The emerging categories form a *typology* (see Table 5.2) that allows researchers to study and characterize Q&A knowledge dissemination within a community of practice. We used this typology to study how knowledge is constructed and shared on Stack Overflow and the R-help mailing list. We found that these channels support two distinct approaches for constructing knowledge—*participatory knowledge construction* and *crowd knowledge construction*—however, each channel supports them differently (RQ2). Our findings indicate that participatory knowledge construction is more prevalent on the R-help mailing list, while crowd knowledge construction is more prevalent on Stack Overflow.

We found that some contributors are active on both channels. As a result, we conducted a survey to investigate the benefits they gain by doing so (RQ3). But beyond that, we wanted to examine how participation differs between Stack Overflow and the R-help mailing list over time and how long users participate on the two channels (RQ4). Additionally, we wanted to understand the behavior and *participation patterns* of the contributing users (RQ5). We focused on several sets of contributors:

those who rarely contribute, the top contributors, and those who contribute to both channels. Our results show that a great majority of participants are fleeting and a small number of individuals are responsible for most answers. Furthermore, our findings indicate that both channels are reaching maturity: for the R-help mailing list, this means a steady flow of questions; for Stack Overflow, there is a continuous decrease of new questions with a positive score (number of positive votes minus negative votes), hinting to the fact that, as time progresses, the most sought after questions have already been asked.

The findings we report and discuss in this chapter show how channel affordances and community rules (e.g., topic restriction and gamification) influence knowledge construction and curation. This has implications on other open source projects or companies and should be considered by those that are thinking of using Stack Overflow instead of or in addition to email for knowledge sharing. This information can also help guide which behaviour patterns project or community leaders should monitor over time.

5.1 Background

The R project[2] was born in 1993 as a free and open source programming language and software environment for statistical computing, bioinformatics, and graphics [72]. R's popularity has continuously increased over the years: in 2016, IEEE Spectrum ranked it as the 6th most popular language[3].

The R community is composed of:

1. *R-core*, a team of 20 software developers that maintain and evolve the R language; and
2. *Periphery*, which includes everyone else (language users and package developers).

The R community is an eclectic open source community that goes beyond software development and includes biologists and statisticians with no or limited programming experience. Its entire history of mailing list communication is archived and publicly available. The R community has also been the subject of extensive research in community evolution [50, 169] and the interplay between channels [170].

Our study focused on the analysis of Stack Overflow and the R-help mailing list,

[2]https://www.r-project.org/

[3]http://spectrum.ieee.org/computing/software/the-2017-top-programming-languages

two channels in the R community. We chose them because they are the main channels that provide Q&A support to the community.

5.1.1 The R-help Mailing List

There are several mailing lists to help R community users solve programming problems with the R language: *R-help*, *R-package-devel*, *R-devel*, *R-packages*, *R-announce*, and *Bioconductor*. However, R-help is the main mailing list for discussing problems and solutions using R. Other messages are also encouraged, such as documentation, benchmarks, examples, and announcements.

The R-help mailing list used to be the main communication channel for asking and answering questions within the R community, but a significant number of users migrated to Stack Overflow [170]. Despite the reduced number of users, the R-help mailing list is still very active—on average, a subscriber may receive approximately 25 emails a day (as of October 2016).

5.1.2 Stack Overflow

In contrast to the R-help mailing list, Stack Overflow incorporates a rich visual and user-friendly interface with social media and gamification features. The social aspect of the website improves participation and provides strong support for creating and sharing knowledge as well as encouraging informal mentorship [74, 156]. Meanwhile, Stack Overflow's gamification features provide reputation points and badges to reward user participation and earn them points that enable functionality inside the site. It has been reported that Stack Overflow's gamification mechanisms boost participation [169] and enable mutual assessment [141] between developers who ask and answer questions.

5.1.3 Stack Overflow vs. Mailing Lists

Software development is a knowledge-building process [105]. Due to the emergence of socially enabled tools and channels and the formation of communities of practice [156], it is important to understand how knowledge is created and shared within these communities. In this study, we focus on knowledge in the form of questions and answers within the R community.

Other researchers have also examined communities using Stack Overflow and R-help. As part of a study on the transition to gamified environments, Vasilescu [169] examined the popularity of Stack Exchange (including the Stack Overflow R tag) and mailing lists within the R community. He found that the number of message threads on the R-help mailing list had decreased since 2010, while the number of R-related questions asked on the Stack Exchange network had increased. Vasilescu also examined the *difference in activity* between contributions made by users active on both channels and users focused on a single medium. Similar to Vasilescu, Squire [147] studied a project's *transition to the Stack Overflow gamified channel*. She focused on examining whether four software projects that moved from mailing lists to Stack Overflow showed improvements in terms of developer participation and response time. She found that all four projects showed improvements on Stack Overflow compared to mailing lists. However, she also found that several projects moved back to using mailing lists despite achieving these improvements. The reasons for moving back included poor support for discussion in Stack Overflow and also closing of questions that were thought to be relevant but were not suited to Stack Overflow.

In our study, we examined Stack Overflow's R tag and the R-help mailing list to better understand the *knowledge types* used. This allowed us to characterize the different approaches for seeking and sharing knowledge on each channel. We found that both channels have knowledge support for question and answers, however, there are important differences between the two channels. For example, Stack Overflow's competitive environment gives more reputation points[4]—a sought after reward—to those who have an answer accepted than to those participating in other activities (e.g., editing). Additionally, upvoted questions and answers can receive many points over time if the question is frequently voted up. Other forms of participation (e.g., commenting) receive badges[5], which are finite and have less visibility than the number of reputation points achieved. For these reasons, we believe that participants have an incentive to be the first to provide the correct answer rather than improve other answers and participate in discussions. Moreover, we also found users of the R community that were active on both channels. As a result, we sought to understand *why users post to a particular channel*, and then understand *user participation activity over time.* This raises important questions about the role of newcomers, prolific members, contributors, and curators in communities of practice.

[4]https://stackoverflow.com/help/whats-reputation
[5]https://stackoverflow.com/help/badges

5.1.4 Community Participation

A community of practice is the embodiment of its members, their shared knowledge, relations and social interactions, and the activities that foster learning and participation. Wenger *et al.* [183] wrote about communities of practice from the perspective of learning, focusing on the role of technology in the formation of such 'digital habitats': *"While there is no question that digital habitats can give rise to new communities—by connecting people across time and space, by creating new spaces for engagement, by revealing affinities for shared domains, and by providing information about people—we need to make a clear distinction between the technology and the social conditions and processes that bring a community together. Just because the technological container remains, it does not mean the community is still functioning and alive."* In this study, we examine not only the channels and knowledge artifacts, but also the community participants and their activities.

We explore participation patterns in the R community and focus on two specific types of participants: *newcomers* and *prolific members*. Both groups of participants are important for a community's growth and well-being. Through a process of formal or informal mentorship, newcomers begin with simple peripheral tasks and participate in limited activities (physically or socially), gradually doing more until they become experienced members. *"The social structure of this practice, its power relations and its conditions for legitimacy define possibilities for learning"* [86]. Through these peripheral activities, novices gain 'legitimacy' and become acquainted with the community's rules, norms, principles, and vocabulary.

Experienced and prolific members tend to be very active within the community; they are often experts who want to keep the system clean with valuable content. Many of these members often serve as moderators or *knowledge curators* (referred to as "caretakers" by Srba and Bielikova [148]), bridging between different groups of people and nurturing the community. In the R community, we found that a small group of prolific members are responsible for a large proportion of answers.

5.2 Methodology

The main goal of this work was to empirically compare how knowledge, specifically knowledge in question-and-answer (Q&A) form, is sought, shared, and curated on both the R-help mailing list and the R tag on Stack Overflow. We also aimed to

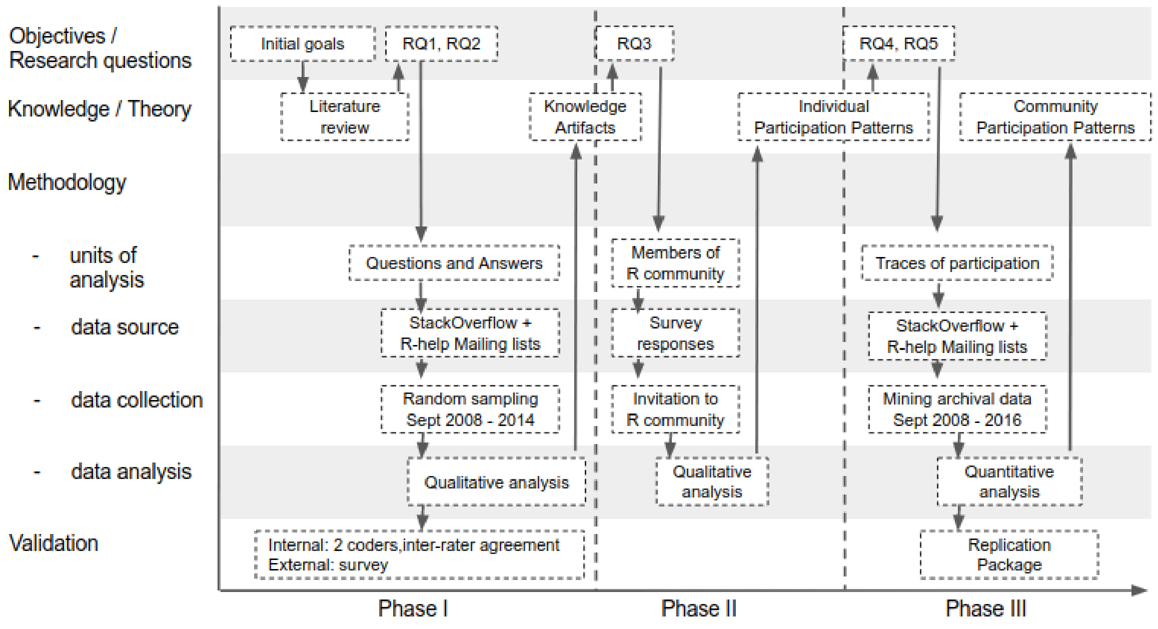


Figure 5.1: A timeline of our research process.

gain insights on the *knowledge curation process* used within the R community (e.g., participation patterns and behaviors). We used a mixed methods *exploratory case study* methodology [25, 131] to answer our research questions and we describe our research process timeline in Fig. 5.1.

This study employed two research methods over three phases: we *mined archival data* and conducted a *qualitative survey*. In the first phase, we sampled and qualitatively analyzed the mined data to characterize the types of discussions that occur. In the second phase, we surveyed members of the R community to validate our interpretation of the results from the previous phase. And in the third phase, we conducted a quantitative analysis of the archival data and focused on investigating participation patterns on both channels.

5.2.1 Phase I: Characterizing Types of Knowledge Artifacts

We mined data from the public archives of both the R-help mailing list and Stack Overflow. The R-help mailing list archive started in 1997, while the archives for Stack Overflow started in 2008 (when it was created). To make the datasets comparable, we analyzed both datasets from September 2008 until September 2014, a period of time that both channels were available[6].

[6]In the third phase of our study, we extended the mined datasets up to September 2016

Data collection process

For the Stack Overflow data, Stack Exchange releases a new data dump of all their Websites every three months[7]. We used only the questions, comments, and answers that included the tag `r` or its two synonyms[8] (`rstats` and `r-language`), and information about the users who contributed to them.

For the R-help mailing list data, we retrieved MBOX files of the mailing list archives from the R-help website. We downloaded all the threads and extracted the separate emails and corresponding metadata from each thread. As opposed to Stack Overflow, the R-help mailing list data doesn't include metadata to indicate whether a message is a question or an answer. Thus we used the following heuristic to classify the R-help messages: if a message contained an `In-Reply-To` or a `References` header, it was classified as an answer (225,254 responses); otherwise, a message was classified as a question (119,145 questions).

To study participation and determine which users were active in both channels, we assigned them a unique `person` identifier and performed a unification of identities. We extracted email addresses from the R-help mailing list data using the `From` field, and using a conservative approach, identified messages sent by the same person but from different email addresses. As a result, the number of unique individuals identified on R-help was reduced from 36,600 to 31,729 (a reduction of 15%)—a similar number was reported by Vasilescu *et al.* [170]. Next, by using MD5 hashes of the email addresses extracted from the R-help mailing list and Stack Overflow, we identified 1,449 persons who used both media channels.

To prepare and process the data, we wrote our own scripts (these are included in the replication package[9]). To ensure accurate results when processing the R-help mailing list, we followed a series of recommendations proposed by Bettenburg *et al.* [13]: extracted messages, removed duplicates, removed signatures, and reconstructed discussion threads. We unified identities of contributors in R-help using our own email unification tool[10]. To identify people common to both channels, we compared the hash of the email in the Stack Overflow data against the hash of every email used by a person. If there was a match, we considered the person in R-help to be the same as the person in Stack Overflow.

[7]http://stackexchange.com/sites

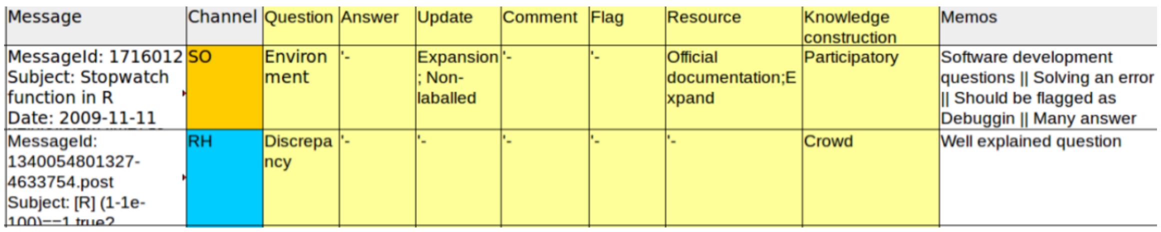
[8]http://stackoverflow.com/tags/r/synonyms

[9]Our scripts, sample data, and coded data are openly available at https://zenodo.org/record/831805

[10]https://github.com/dmgerman/unify-perl

Data analysis process

We followed an inductive approach [131] to analyze the data from Stack Overflow and the R-help mailing list. To reduce the risk of bias [131], the analysis was conducted by two computer scientists with a background in qualitative data analysis. To answer RQ1 and RQ2, we randomly sampled and iteratively coded questions in both channels to characterize the types of discussions that occur. This process was continued until we reached thematic saturation [17], i.e., no new themes were identified and no issues were raised regarding existing categories, which amounted to 400 threads in each channel. To answer RQ3, we focused on questions with identical subjects that were posted to both channels by the same author: we found and analyzed 79 such threads.



Message	Channel	Question	Answer	Update	Comment	Flag	Resource	Knowledge construction	Memos
MessageId: 1716012 Subject: Stopwatch function in R Date: 2009-11-11	SO	Environ ment	'-	Expansion ; Non-laballed	'-	'-	Official documentation;E xpand	Participatory	Software development questions || Solving an error || Should be flagged as Debuggin || Many answer
MessageId: 1340054801327-4633754.post Subject: [R] (1-1e-100)==1 true?	RH	Discrepa ncy	'-	'-	'-	'-	'-	Crowd	Well explained question

Figure 5.2: Example of data coding. Each row is a threaded message. Questions, comments, and answers are identified with the number in the first column. Columns in yellow (columns 4-10) contain the code for each message type. The last two columns contain the memos and URLs.

We used memoing, affinity diagrams, and a code book to support the data analysis process. We wrote reflective memos in a spreadsheet next to the applicable codes (see example in Fig. 5.2). These memos were used to create the codes and hypotheses about the relationships between concepts. We coded in multiple sessions, which allowed us to iteratively refine the definitions in the code book. Each entry is associated with a title, a formal definition, an example, and notes from the researchers. For inter-rater reliability, we used the Cohen Kappa inter-rater agreement coefficient [153]. Although it is suggested that one should aim for coefficient values above 0.6 to obtain substantial results [82], we aimed for 0.8 or above based on our previous experience with this method [53]. We used this coefficient after each coding session as a way to trigger discussion and to further refine the codes if necessary.

The analysis process required an *understanding of the context* surrounding each message. The process consisted of: (1) gathering the required information from each channel (i.e., the message analyzed, the relevant thread), and (2) mapping the messages from each channel to a specific knowledge type (see Section 5.3.1). The mapping was necessary as each channel contained a different data structure. We defined the following mappings between messages in both channels:

Question: The message is the first in the thread and contains the main question.

Answer: The message provides a solution to the main question in the thread.

Update: The message provides a modification to a question or an answer made by the author of said question or answer.

Comment: The message offers clarification to a specific part of the question or answer.

Flag: The message requests attention from the moderator or other community members (e.g., repeated questions, spam, or rude behavior).

As opposed to Stack Overflow, where the metadata indicates the corresponding mapping, for R-help we needed to manually examine the context of each message and classify them based on the knowledge artifact definitions. We elaborate on these definitions in Section 5.3.1.

5.2.2 Phase II: Exploring Why Users Post to a Particular Channel

The analysis from Phase I revealed that some developers are active on both channels, and in some cases, even post the same questions. To further understand this phenomena and explore the perceived benefits of using one channel over the other, we conducted a survey with users of the R community[11]. To test and refine the questions, format, and tone, we piloted the survey twice. We promoted our survey on Twitter, Reddit, the R-help mailing list, and Meta Stack Exchange to reach users of both channels and minimize selection bias. However, our survey invitation on Stack Exchange was deemed off topic and deleted a few minutes later. In total, we received 37 responses, 26 of which were valid (invalid responses occurred if the session ended or the participant did not complete the survey).

5.2.3 Phase III: An Extended Investigation of Participation Patterns

In this phase, we focused our analysis on the behavior and participation patterns of community users on the R-help mailing list and Stack Overflow. Since this phase

[11]A copy of the survey is available at http://cagomezt.com/lime/index.php/857211?lang=en

was conducted two years after our initial phase, we extended our analysis to include archival data from the beginning of each channel up to September 2016 (for both channels). Table 5.1 depicts a summary of the data used for this phase of the study.

Table 5.1: Raw data collected for each channel, up to September 2016.

Type	R-help	Stack Overflow
Questions	124,791	150,707
Answers	150,919	204,468
Comments	88,685	617,460
Different individuals	31,699	63,372

As before, we needed to compare identities between the two communication channels. However, due to privacy concerns, Stack Overflow has now removed all email information from their current dumps (the field is present but empty). Previously, Stack Overflow included the hash of email addresses. For this reason, we used two different data dumps of Stack Overflow: the first one was dated September 2014 (which contains the SHA of the email addresses); the second was dated in March 2017 (which does not contain any participant email information). We used the first dataset only when comparing identities between the two communication channels, while for the rest of the analysis, we used data from both channels (up to September 2016).

5.3 Findings

To understand how knowledge in the form of questions and answers is created, shared, and curated, we first identified and categorized the main types of knowledge artifacts contained within messages on the R-help mailing list and the Stack Overflow 'R tag' (RQ1). The emerging categories formed a typology and allowed us to identify and describe two approaches for constructing the knowledge supported by these channels (RQ2). Interestingly, we found that some developers are active on both channels, and in some cases, even post the same questions. As a result, we investigated the benefits they gain by doing so (RQ3). We also present our findings about participation on the two channels over time (RQ4) and look closely at the different levels of participation activity between community users (RQ5).

5.3.1 What Types of Knowledge Artifacts Are Shared on Stack Overflow and the R-help Mailing List

To answer RQ1, we randomly sampled message threads from both Stack Overflow and the R-help mailing list, where each thread included a question and the associated responses. We identified five types of artifacts that capture knowledge: (1) Questions; (2) Answers; (3) Updates; (4) Flags; and (5) Comments. Through our analysis, we further divided these types into subtypes.

Table 5.2 presents our typology of knowledge artifacts, their descriptions, and their frequency in the data sample. Even though we did not aim for a statistically significant sample size, the size of this sample (400 threads in each channel) guarantees a confidence level of approximately 95% $\pm$ 5% for both channels. Using the Chi-square test of independence, we tested whether the frequency distribution of the types and subtypes of questions was different between the two channels. Specifically, we tested if the distribution frequency of each subtype (as shown in Table 5.2) was statistically different between R-help and Stack Overflow. We also compared the overall frequency for each artifact subtype. In all cases, the distributions were found to be statistically different (with $\rho \ll 0.001$ in all cases).

Table 5.2: Typology of knowledge artifacts found on both Stack Overflow (SO) and the R-help (RH) mailing list, their frequency, and relative proportion in the analyzed sample.

		SO	**RH**	**Prop SO**	**Prop RH**
Questions					
How-to	Asks how to do something specific.	166	103	41.50%	25.75%
Discrepancy	Asks about an unexpected result of a specific function, process, or package.	53	88	13.25%	22.00%
Conceptual/Guidance	Asks for conceptual clarification or guidance on topics related to R or statistics.	48	49	12.00%	12.25%
Bug/Error/Exception	Asks for a solution to or reasons for an error message.	27	48	6.75%	12.00%
Decision help	Asks for advice in making a decision.	36	35	9.00%	8.75%
Code reviewing	Asks for a code review, explicitly or implicitly.	34	21	8.50%	5.25%

Set-up	Asks for possible ways to set up the R environment before or after deployment.	15	31	3.75%	7.75%
Non-functional	Asks for help (or suggestions) with a non-functional requirement such as performance or memory usage.	14	11	3.50%	2.75%
Future reference	Asks a question (often self-answering it) that might not exist on the channel, but that is interesting enough to warrant a thread for future reference.	5	4	1.25%	1.00%
Other	Asks for assistance unrelated to the channel, or the message contains unrelated information (e.g., announcements, ideas for improvement).	2	10	0.50%	2.50%
	Total	400	400	100%	100%
Answers					
Explanation	Provides an explanation of an approach that answers the question and lists steps on how to implement it.	203	101	25.15%	17.44%
Source code	Provides a source code snippet as a solution without an extensive explanation about the answer.	198	102	24.54%	17.62%
Redirecting	Provides a link to an existing solution that is not in the thread (e.g., external application, tutorial, project).	163	87	20.20%	15.03%
Clue/Hint/Suggestion	Provides a possible way to fix the issue without actually solving it.	43	105	5.33%	18.13%
Alternative	Provides a different approach to a solution that is related to but not exactly what is being asked (e.g., mathematical approach, data structure modification).	33	98	4.09%	16.93%
Tutorial	Provides a set of steps to teach people how to solve the issue.	105	15	13.01%	2.59%

Announcement	Provides a notification about some artifact (e.g., packages, libraries).	8	33	0.99%	5.70%
Opinion	Provides an opinion or an expansion of another answer by including scenarios and examples.	49	35	6.07%	6.04%
Benchmark	Provides a benchmark of multiple solutions posted by others or compares different answers.	5	3	0.62%	0.52%
	Total	807	579	100%	100%
Updates					
Expansion	Expands the question or answer by providing scenarios or examples.	116	83	18.92%	33.60%
Correction	Corrects format, grammar, spelling, and semantic mistakes.	301	2	49.10%	0.81%
Explanation	Explains or clarifies a specific point in the question or answer, such as why the user chose a specific data structure, or the meaning of a variable.	83	95	13.54%	38.46%
Announcement	Announces specific events (e.g., bounties, future updates).	27	3	4.40%	1.21%
Background	Adds additional context to the question or answer.	74	57	12.07%	23.08%
Solution	The user answers their own question.	12	7	1.96%	2.83%
	Total	613	247	100%	100%
Flags					
Repeated question	Notifies a user that the question has been answered previously.	48	8	59.26%	14.81%
Off-topic/ Opinion	Identifies questions that are unrelated to the channel's interests, or requests answers based on opinion.	22	19	27.16%	35.19%

Not an answer	Indicates answers that are out of scope of the question or that do not answer the question.	0	27	0.00%	50.00%
Too localized	Indicates questions that are too specific and might not help future readers.	6	0	7.41%	0.00%
Unclear	Indicates questions that are difficult to understand.	5	0	6.17%	0.00%
	Total	81	54	100%	100%
Comments					
Correction/ Alternative	Suggests a change to a question or answer, offers an alternative solution or a correction.	102	89	18.15%	33.33%
Expansion	Provides additional information.	127	65	22.60%	24.34%
Compliment/ Critic	Posts something good, offers thanks, or provides an opinion or criticism.	157	52	27.94%	19.48%
Clarification	Provides (or requests) additional information about a question or answer.	98	28	17.44%	10.49%
External reference	References an external resource.	78	33	13.88%	12.36%
	Total	562	267	100%	100%

Questions and Answers: Questions express one or more problems or concerns faced by a user on the R-help mailing list or on Stack Overflow, whereas answers represent solutions to questions. We observed that the types of questions on Stack Overflow are more specific than those on the R-help mailing list and are more likely to consist of tutorials. Stack Overflow also contains more answers per question: 2 per question compared to 1.4 for R-help (see Table 5.2). However, R-help answers tend to offer more suggestions or alternatives than Stack Overflow answers.

Updates: An update is a modification of a question or an answer. In Stack Overflow, updates are presented in one of two ways:

Labeled updates are explicitly shown in the body of questions and answers next to a label that identifies the update (e.g., edit, update, and p.s.). When multiple update labels appear in a message, each label is accompanied by a number (e.g., *"[Edit 1:]"*), a date (e.g., *"Edit/Update (April 2011):"*), or a bulleted list (e.g., "EDIT: - anova... -drop1...").

Non-labeled updates are only visually recognizable through the message history system. The only indication of a change is a box at the end of a message that identifies the user who performed the change and the date when it occurred.

We found that non-labeled updates are often used to correct formatting, grammar, semantic mistakes, and spelling, or to incorporate explanations, examples, and suggestions without changing the meaning of the question or answer. Labeled updates are for everything else.

On the R-help mailing list, all communication occurs through email, and authors do not explicitly tag messages as updates. For this reason, we define an update on R-help as *a message sent to a thread where the author has already participated once.*

Regarding update frequency in our sample, the Stack Overflow R tag contained 2.5 times more updates than the R-help mailing list. Corrections are more common on Stack Overflow (almost 50%), while R-help updates are often related to the adding of information to a thread (providing background, expansion, and explanation).

Flags: Flags are used to alert users that a question or an answer does not match community expectations. Stack Overflow contains a flagging mechanism that's often used to get a moderator's attention. These flags can accomplish various objectives: mark a message as containing spam or rude/abusive behavior, or identify duplicate questions, off-topic messages, unclear questions, opinion-based questions, and low-quality answers. Depending on the type of flag, this can lead to a thread being closed or the loss of user reputation points.

The R-help mailing list doesn't have a built-in flagging mechanism. However, R-help users utilize the concept of flags, which we define as *messages used to call the attention of other community users*, similar to the way flags are used in Stack Overflow. For example, a community member indicated that a question is off topic by responding that *"the main questions here are not R-related, but statistical modeling questions, and much too broad for the R list"*.

In terms of their frequency, R tag posts on Stack Overflow contained 1.5 times more flags than posts on the R-help mailing list. Stack Overflow flags are primarily used to mark repeated questions. In contrast, flags on R-help are often used to indicate that a previous answer is incorrect.

Comments: In Stack Overflow, comments are considered “temporary ‘Post-It’ notes left on a question or an answer”[12]. Comments are located below each question or answer and can be used to clarify information or follow up with further details. On the R-help mailing list, we define comments as messages written to *improve an answer or as a follow-up to a discussion.* It should be noted that for an email to qualify as a comment, it must not be written by the person who asked or answered the original question. Otherwise, the message would be considered an update. Because both Stack Overflow and the R-help mailing list permit participants to ask multiple questions in the same thread, the subcategories of comments are not mutually exclusive.

Regarding the frequency of comments, the main difference between the two channels is that Stack Overflow comments are less likely to be considered corrections or alternatives (Correction/Alternative subcategory) than on the R-help mailing list. The Stack Overflow R tag sample also contained 2.1 times more comments than the R-help sample (see Table 5.2).

5.3.2 How Knowledge Is Constructed on Stack Overflow and the R-help Mailing List

Our analysis helped us identify two different approaches for constructing knowledge (RQ2) on Stack Overflow and the R-help mailing list: participatory knowledge construction and crowd knowledge construction.

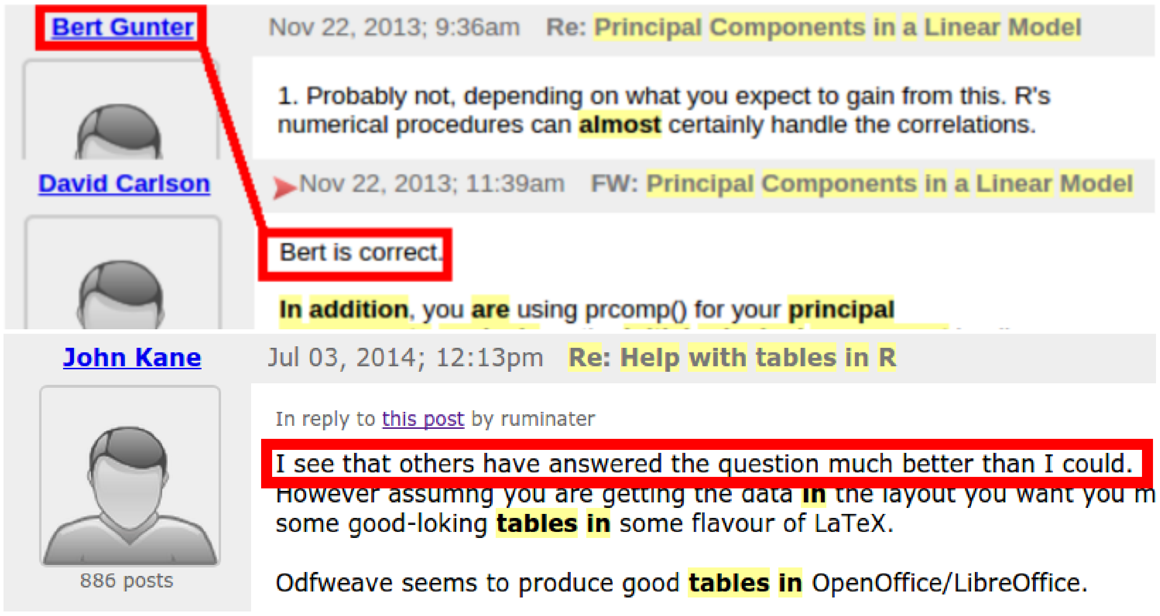
Participatory knowledge construction is an approach where answers are created through the cooperation of multiple users in the same thread. Participants complement each other’s solutions by discussing the pros and cons of each answer and by adding different viewpoints, additional information, and examples. This process is similar to a team working together towards a common objective.

Crowd knowledge construction leverages the experiences of many users who work in a relatively independent manner. Each user contributes to the thread, adding variety to the pool of solutions. However, the user’s priority is to provide a correct answer and not to discuss other solutions. This is comparable with the concept of a group in which people work towards the same objective but not necessarily together (e.g., Amazon’s Mechanical Turk). Participants can vote on others’ ideas, but the main idea is not constructed through discussion.

Figure 5.3 depicts two examples of the way participatory knowledge occurs on the

[12]http://stackoverflow.com/help/privileges/comment

R-help mailing list: direct citation of the author of a previous answer, and inferable links between answers.



Bert Gunter
Nov 22, 2013; 9:36am Re: Principal Components in a Linear Model

1. Probably not, depending on what you expect to gain from this. R's numerical procedures can **almost** certainly handle the correlations.

David Carlson
Nov 22, 2013; 11:39am FW: Principal Components in a Linear Model

Bert is correct.

In addition, you **are** using prcomp() for your **principal**

John Kane
Jul 03, 2014; 12:13pm **Re: Help with tables in R**

886 posts

In reply to this post by ruminater

I see that others have answered the question much better than I could.
However assumng you are getting the data **in** the layout you want you m
some good-loking **tables in** some flavour of LaTeX.

Odfweave seems to produce good **tables in** OpenOffice/LibreOffice.

Figure 5.3: Participatory knowledge construction on the R-help mailing list.

On the R-help mailing list, *participatory knowledge* construction occurs when:

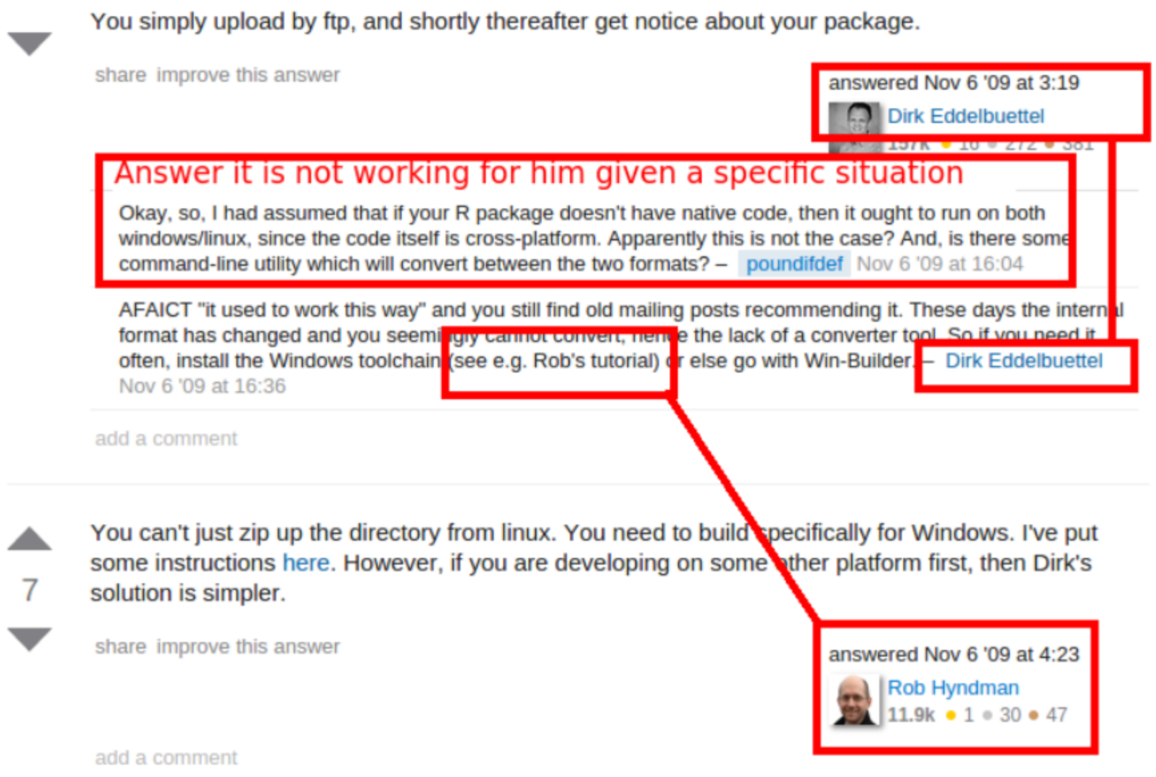
1. previous answers are included in the current answer, with clear links between them; or
2. a reply contains a direct reference to other answers or authors.

On Stack Overflow, *participatory knowledge* construction takes place when:

1. one can infer a link between answers, through either a direct or indirect reference; or
2. comments complement the answer or directly cite another author.

Participatory knowledge construction also occurs in different places on Stack Overflow, perhaps as a consequence of its rich interface. We observe this type of knowledge construction when a user answers a question and directly cites or links to someone else's answer in the thread, or when a user cites someone else's question or answer in a comment (a typical case is linking to a previously asked question). Figure 5.4 depicts an example of participatory knowledge construction on Stack Overflow: when an answer was deemed insufficient, a user helped out by adding a comment and referencing another author's answer.

On the R-help mailing list, *crowd knowledge* construction takes place when dif-


You simply upload by ftp, and shortly thereafter get notice about your package.

share improve this answer

answered Nov 6 '09 at 3:19
Dirk Eddelbuettel

Answer it is not working for him given a specific situation

Okay, so, I had assumed that if your R package doesn't have native code, then it ought to run on both windows/linux, since the code itself is cross-platform. Apparently this is not the case? And, is there some command-line utility which will convert between the two formats? – poundifdef Nov 6 '09 at 16:04

AFAICT "it used to work this way" and you still find old mailing posts recommending it. These days the internal format has changed and you seemingly cannot convert, hence the lack of a converter tool. So if you need it often, install the Windows toolchain (see e.g. Rob's tutorial) or else go with Win-Builder. – Dirk Eddelbuettel Nov 6 '09 at 16:36

add a comment

You can't just zip up the directory from linux. You need to build specifically for Windows. I've put some instructions here. However, if you are developing on some other platform first, then Dirk's solution is simpler.

share improve this answer

answered Nov 6 '09 at 4:23
Rob Hyndman
11.9k ● 1 ● 30 ● 47

add a comment

Figure 5.4: Example of participatory knowledge on Stack Overflow: users built on the comments and answers of other users.

ferent messages respond directly to the original question, rather than to another response.

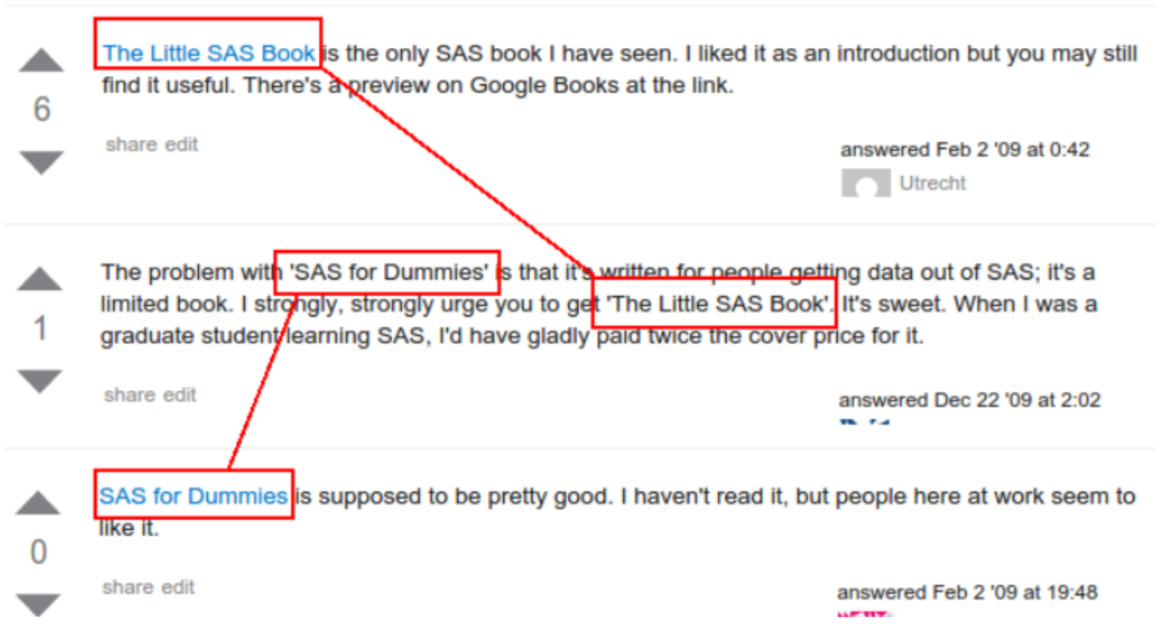
On Stack Overflow, *crowd knowledge* construction occurs when:

1. there is no direct or inferable reference between answers; or
2. an answer is a variation of one of the other answers in the thread.

Figure 5.5 depicts an example of crowd knowledge construction on Stack Overflow. As can be seen from the figure, two of the three answers provided the same solution.

5.3.3 Why Users Post to a Particular Channel or to Both Channels

From our survey, we were able to learn why some R community members preferred one channel over the other—we summarize their responses below. Using results from the analysis of the archived data, we discuss why some members post some questions to both channels.



6

The Little SAS Book is the only SAS book I have seen. I liked it as an introduction but you may still find it useful. There's a preview on Google Books at the link.

share edit

answered Feb 2 '09 at 0:42

Utrecht

1

The problem with 'SAS for Dummies' is that it's written for people getting data out of SAS; it's a limited book. I strongly, strongly urge you to get 'The Little SAS Book'. It's sweet. When I was a graduate student learning SAS, I'd have gladly paid twice the cover price for it.

share edit

answered Dec 22 '09 at 2:02

0

SAS for Dummies is supposed to be pretty good. I haven't read it, but people here at work seem to like it.

share edit

answered Feb 2 '09 at 19:48

Figure 5.5: Example of how crowd knowledge construction occurs on Stack Overflow. The three authors provided similar answers, but did it independently of each other.

Why participants post on Stack Overflow

Survey participants preferred using Stack Overflow for several reasons: (a) the ability to gain peer recognition (the advantage of gaining points—and visibility—is a major draw of Stack Overflow); (b) its rich and user-friendly interface; (c) answers are straight to the point; (d) questions are usually answered faster on Stack Overflow than on the R-help mailing list; and (e) it is easy to search for previous questions and answers.

However, the respondents reported a few main drawbacks of using Stack Overflow: (a) there is an overabundance of related questions; (b) one requires a certain level of experience to understand some of the answers; and (c) Stack Overflow's strict rules only allow questions and their answers, they do not allow discussions nor questions about opinions.

Why participants post on the R-help mailing list

Survey participants reported a few benefits of using the R-help mailing list: (a) the email format is convenient; (b) following the mailing list provides awareness and increases learning in new topics; (c) there is more flexibility regarding the topics that one can discuss; and (d) there is a high level of participation from experienced users.

The respondents did note a couple of disadvantages of R-help: (a) some discussions lead to aggressive behavior; and (b) searching the archives is not easy.

Why participants post to both channels

Our analysis of the archived data revealed that some users (79 in our sample) posted the same question on both channels. Based on the responses from the survey, we identified that being active on both channels brings benefits to those asking and answering questions (RQ3):

Find a better answer: One channel might result in a better answer than the other.

Support follow-up questions: We found that the R-help mailing list is often used to conduct follow-up discussions on specific answers provided to Stack Overflow questions. Stack Overflow's focus is on finding an answer to a question and provide a rudimentary method to discuss the specifics of an answer, either by adding comments to the answer—which cannot be threaded—or by asking another question (related to the answer). In contrast, a discussion on R-help can continue long after an answer has been found through follow-up questions involving a variety of people, not just the person who asked the original question.

Speed up answers: Users ask the same question on both channels in order to get an answer faster. However, in many cases this behavior is not encouraged by the community as it is deemed impolite[13]. This seems to be a matter of opinion, as other community members argue in favor of cross-posting questions[14].

5.3.4 How Participation Differs Between the Two Channels Over Time

Vasilescu *et al.* studied participation in the R-help and Stack Overflow communities over time [170]. Their research showed strong evidence that knowledge seeking activities are moving from R-help to Stack Overflow (as of the end of 2013). They also noticed a trend indicating that Stack Overflow continues to grow and R-help continues to decrease. These findings prompted us to also consider how participation differs between the two channels over time (RQ4). However, we take a different approach by

[13]https://stackoverflow.com/questions/3892033/r-why-this-doesnt-work-matrix-rounding-error#comment4151921_3892033

[14]http://meta.stackoverflow.com/questions/266053/is-it-ok-to-cross-post-a-question-between-non-stack-exchange-and-stack-exchange

slicing the data based on question score and participation activities over time. In the following, all findings come from our extended dataset that covers September 2008 to September 2016 for both channels (unless specified otherwise).

We first explore the evolution of the number of questions in both channels as the main proxy of activity (the results are shown in Figure 5.6). One aspect that is changing in Stack Overflow is the growth of questions that receive an overall negative score. A score in Stack Overflow is the sum of positive votes minus negative votes, which can be considered an indication of the question's quality. As can be seen in Fig. 5.6, the number of questions with an overall positive score has flattened and is starting to decrease. However, if we inspect the trends over the last eight years, this might be misleading. By looking at the most recent period (Jan. 2015 - Sept. 2016), Figure 5.7 shows that both channels are relatively flat in terms of the overall number of questions, but the number of questions in Stack Overflow is between 10 and 20 times the number of questions found in R-help. Additionally, the proportion of questions with a positive score in Stack Overflow has been decreasing steadily (shown in Fig. 5.8).

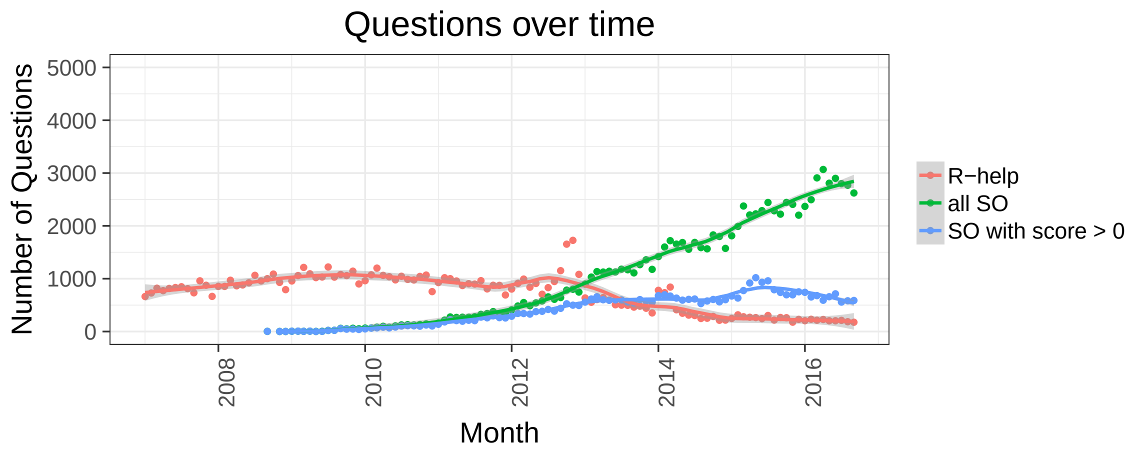


Figure 5.6: The number of questions asked over time: Stack Overflow activity has been much greater than R-help activity, however, the number of questions with a positive score has flattened.

As the Stack Overflow community grows and the number of questions increases, the community faces several challenges that may explain the decrease in questions with positive score. One challenge is handling duplicate questions[15] (cut-and-paste duplicates, accidental duplicates, and borderline duplicates). Another challenge is

[15]https://meta.stackoverflow.com/questions/315293/answering-borderline-duplicate-questions

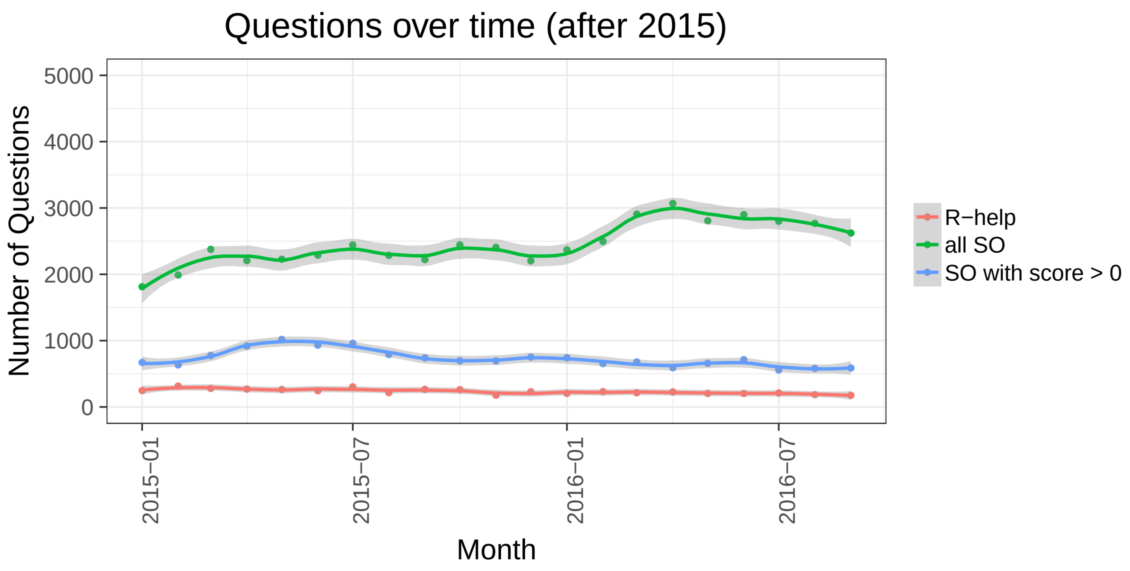


Figure 5.7: The number of questions asked after January 2015: both channels have flattened, but the number of Stack Overflow questions with a positive score continues to decrease.

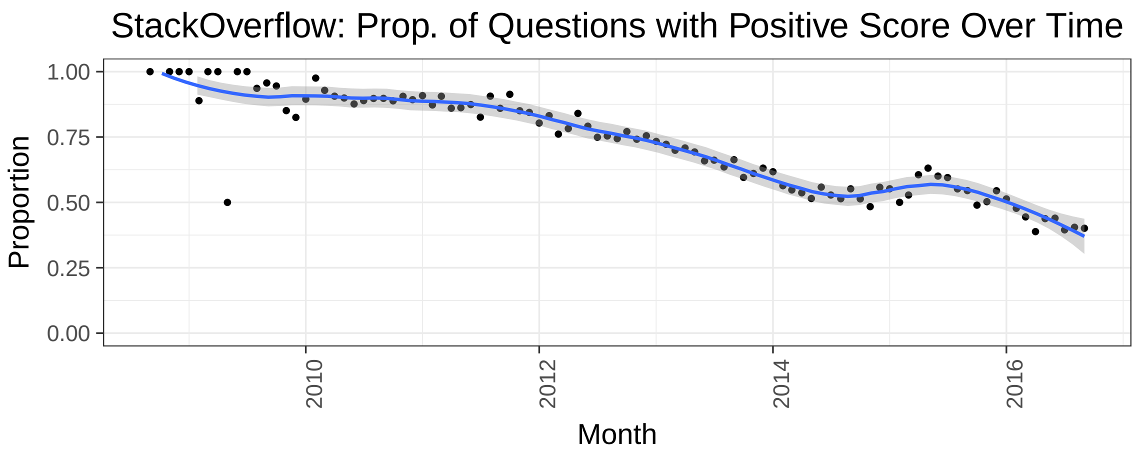


Figure 5.8: The proportion of Stack Overflow questions with a positive score has been decreasing steadily.

dealing with low quality questions: these questions are often poorly described, not directly related to R, or posted by users who didn't put much effort into searching for the answer themselves (referred to as "help vampires" by Srba and Bielikova [148]). At the same time, many questions are left unanswered and with no indication from the community (i.e., zero score). Srba and Bielikova found that there was a constant increase of poor quality questions, unanswered questions, and deleted questions on Stack Overflow [148].

We explored three potential reasons for the decrease in questions with a positive score:

1. We looked at the number of questions marked as *duplicates*. We found this proportion is increasing, but the overall number of questions marked as duplicates is only 3% of all questions.
2. Then we counted the number of *questions with a negative score*, but this only accounts for 2.9% of all questions.
3. Finally, we measured the number of *posts with a zero score*. We found that 29.2% of all posts have a score equal to zero. A small proportion of these questions (3%) had a zero score after being voted up and down.

We posit that the increase in the number of questions with zero score (and the corresponding decrease in questions with a positive score) is because newer questions tend to be too focused or obscure to be of interest to others, but further research is required to verify this assertion.

How long users participate in the channels

An important measure of a community and its ability to curate and maintain knowledge is whether their users continuously participate over time, even if their participation is small. To help us measure the continuity of participation, we divided the history of both channels into month-long segments. In R-help, we considered that a user participated in a given month if they posted at least one email to the list during that period. For Stack Overflow, we considered that a user participated in a given month if they posted, responded to, or commented on at least one question.

The results show that users do not participate in either channel for a long period of time—their participation tends to be brief. Figure 5.9 shows the accumulated proportion of users that participated during a given number of months (not necessarily consecutively). The curves are very similar and skewed: 62% of R-help users and 65%

of Stack Overflow users are active for a single month only; 90% of R-help users are active for 5 months or less; 90% of Stack Overflow users participate 4 months or less.

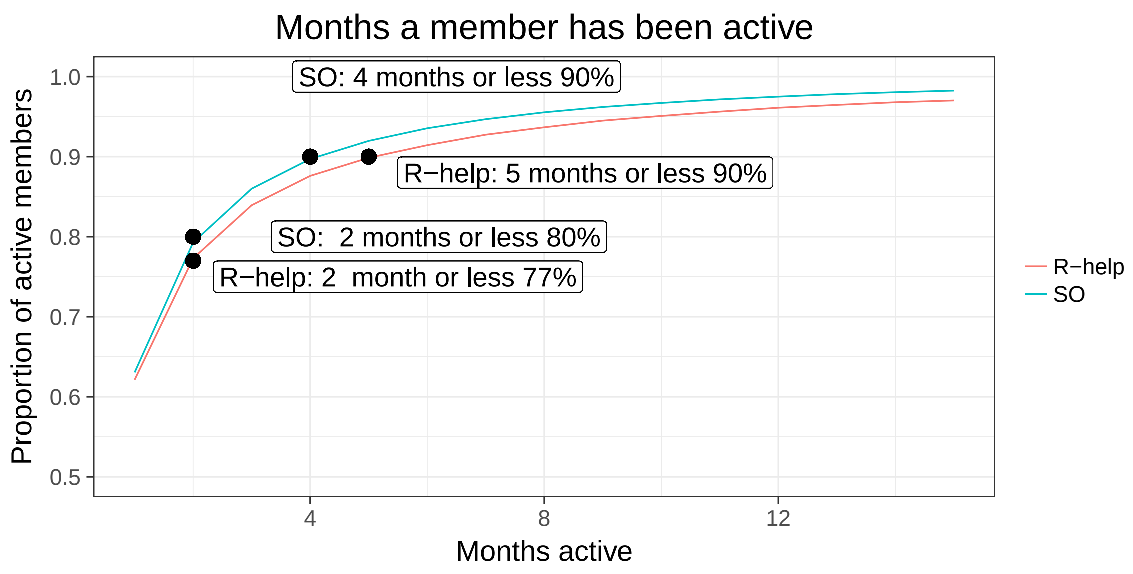


Figure 5.9: The number of months a user has been active. This plot shows the accumulated proportion of users who have been active for a given number of months: 62% of R-help users and 65% of Stack Overflow users are active for 1 month only; 90% of R-help users are active 5 or months or less; 90% of Stack Overflow users participate 4 or months or less. Months do not have to be consecutive.

We noticed that a large proportion of users ask questions but never contribute answers. The proportion of those who never post an answer in R-help is 57.3%, while it is 63.1% in Stack Overflow[16]. Figures 5.10 and 5.11 show the cumulative proportion of users that participate in either channel for a given number of months. As can be seen, people who answer questions tend to stay around longer in R-help than those contributing questions only. However, for Stack Overflow, the difference is very small: both people posting questions and those that answer them do not stay around very long.

[16]There is a threat to validity for this result in the R-help data: Stack Overflow separates responses into comments and answers, however, R-help does not have this distinction. For R-help, we consider that any direct reply to an email is an answer; and we consider a reply to an answer to be a comment.

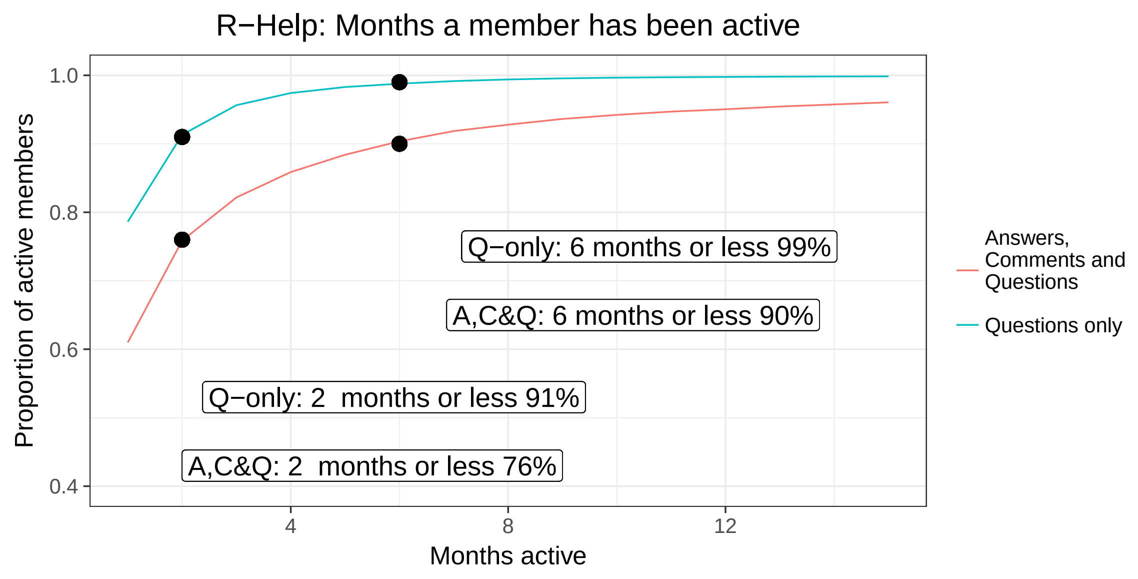


Figure 5.10: The months an R-help user has been active according to whether they only ask questions or only answer questions (and potentially ask questions, too). People who answer questions tend to stay around much longer than those who only ask questions.

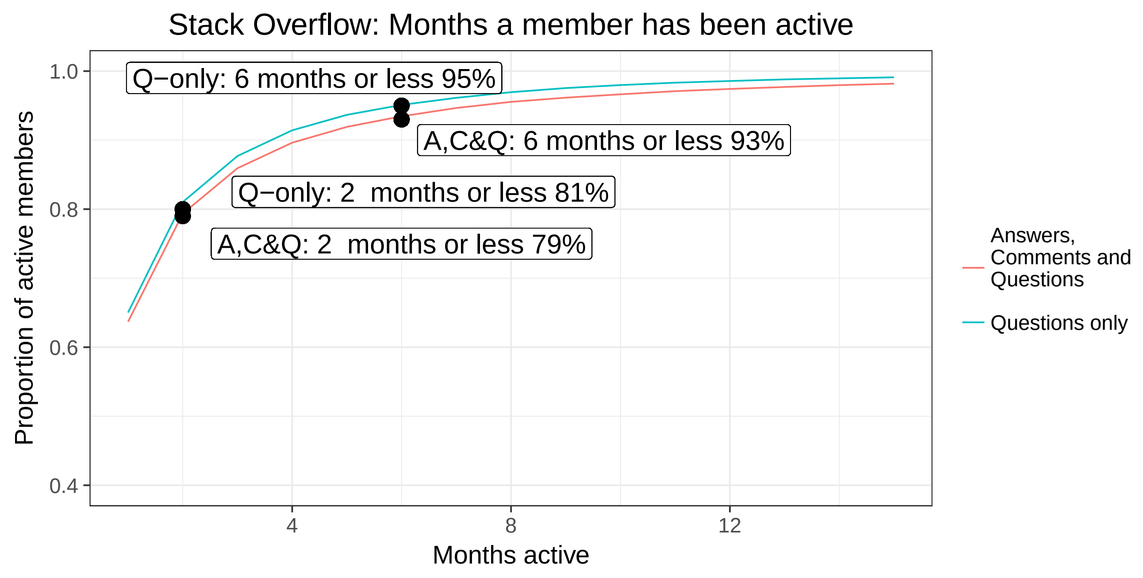


Figure 5.11: The months a Stack Overflow user has been active according to whether they only ask questions or only answer questions (and potentially ask questions, too). Both types of users do not stay around very long.

5.3.5 Are There Significant Differences in Participation Activity Between Community Members?

To consider participation activity within the two channels, we first consider new users by counting how many newcomers there are to both channels and checking if they are likely to post again. Secondly, we look at contributors that are responsible for the vast majority of posts on both channels.

Activity of new users

New users are important for the survival and continuity of a community. Unfortunately, since we can only track users when they actively participate in a channel, we do not know who is a passive participant. Hence, we use the date of a first contribution as the proxy of when a user joins the community. Figure 5.12 shows the proportion of users who have participated for the first time in a particular month. Both sub-communities show a similar pattern: in R-help, between 25% and 35% of all users are first-time participants; in Stack Overflow, between 40% and 50% are first-time participants. In recent years, both channels have seen a slight decline and many of these first-time participants only contribute once (see Fig. 5.13): except for the first 1.5 years of Stack Overflow, both channels see that the proportion of one-time participants is between 10% and 20%. In fact, the number of users who participate only once (at any time) is relatively large over the lifetime of the channels: 43% in R-help and 30% in Stack Overflow.

Common users of both channels

We found 1,449 unique users that participated in both channels. For this, we compared and matched the identities of contributors between both channels between September 2008 and September 2014 (September 2014 is the last Stack Overflow data dump with the email hashes). The number of members who participated in both sub-communities is relatively small (2.5% of all users), yet a handful of these contributors (also referred to as "caretakers" by Srba and Bielikova [148]) are responsible for a very large proportion of the answers in both channels. Figure 5.14 shows the accumulated proportion of answers that have been contributed by these authors. As we can see, the top contributor has contributed 3.9% and 3.7% of answers in R-help and Stack Overflow, respectively. In fact, the top 6 contributors to both

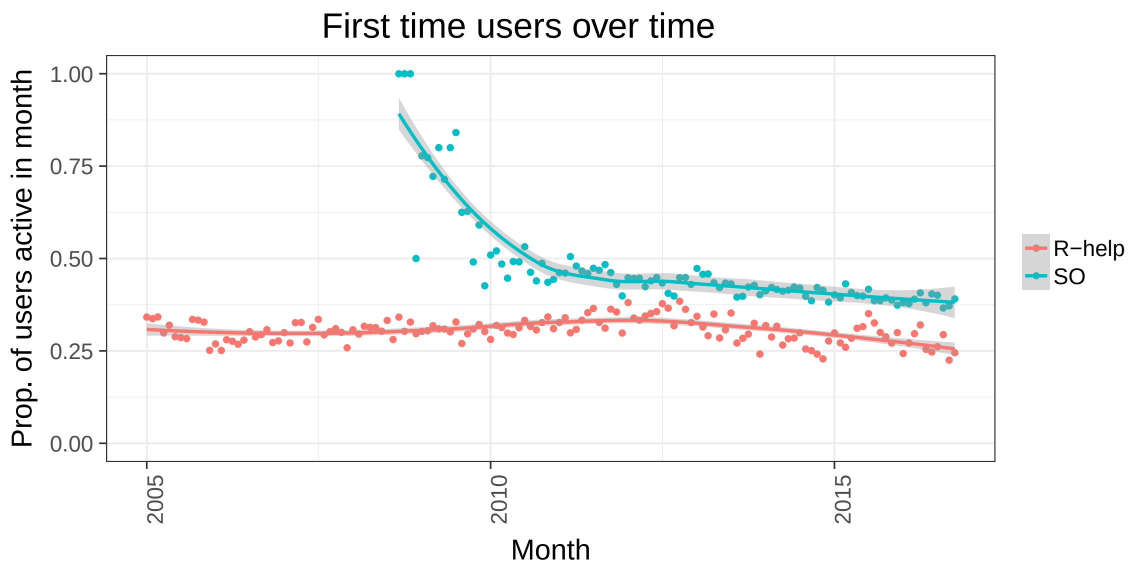


Figure 5.12: Proportion of new users over time. The top lines (thicker) correspond to the number of users who post their first question in that month. The bottom (thinner) lines correspond to the subset of new users who only ask one question and then never participate again.

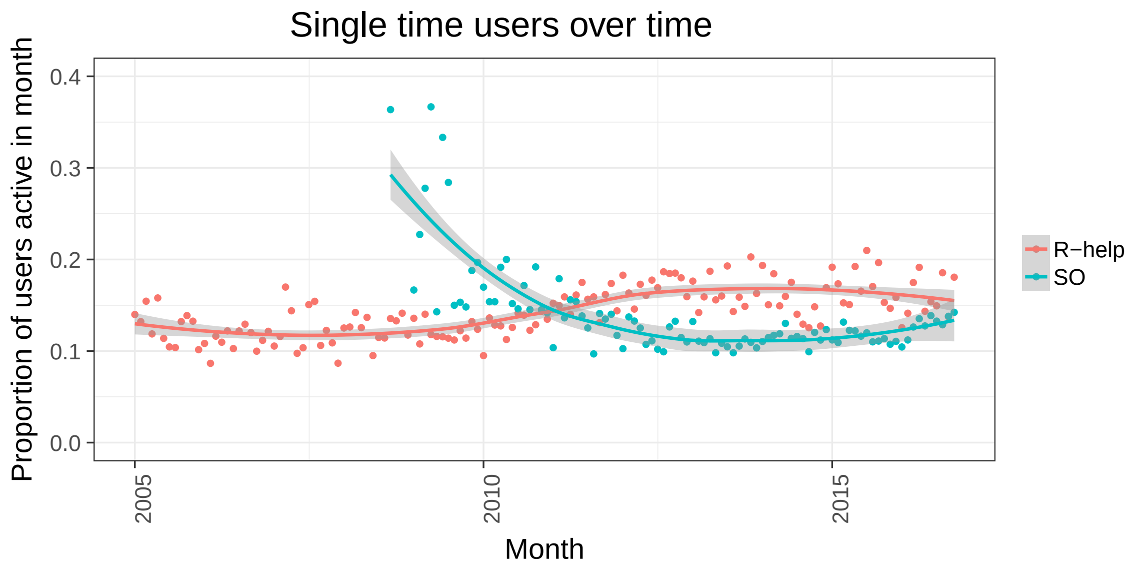


Figure 5.13: Proportion of one-time users in any given month.

channels are responsible for 10% of the answers (and only one of those belongs to the *R-core* group). Here answers on Stack Overflow include both posts and comments to questions.

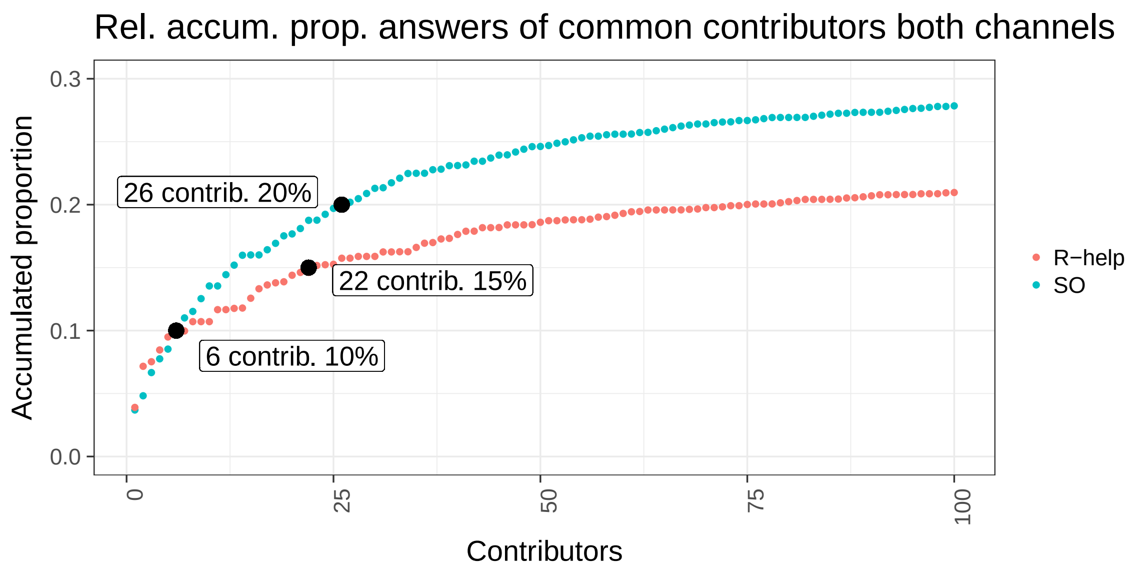


Figure 5.14: Accumulative proportion of answers by the most prolific users who post to both channels. As we can see, the top 6 most prolific users contribute approximately 10% of the answers. This plot only reflects posts between Sept. 2008 and Sept. 2014, the period when we could compare and match identities between both channels.

Prolific contributors

Both channels benefit from a handful of prolific members who answer most questions. Thus we refer to these users as *prolific contributors*. Figure 5.15 shows the most prolific contributors in each channel, ordered by the number of answers they have contributed. The vertical axis shows the accumulated proportion of contributions to all answers in each channel. In R-help, 8 users have contributed 25% of all answers and 40 have contributed 50% of all answers. In Stack Overflow, the distribution is not as steep, yet 27 users have contributed 25% of all answers and 146 have contributed 50% of all answers (including comments). This means that 0.13% of the R-help mailing list users contribute 50% of all the answers, while 0.23% of the Stack Overflow users contribute 50% of all the answers. We identified the professions of the top 10 contributors to both channels: five professors, three package maintainers, a core developer, and one retired person. For contributors who participated in only one of the channels:

- We identified the professions of 9 of the top 10 contributors in Stack Overflow: one professor, two package maintainers, three data scientists, one consultant, one retired person, and one anonymous contributor (there is no personal information in their profile).
- In the case of R-help, there are four package maintainers, one book author/professor, two professors, one core developer/professor, one consultant, and one retired person.

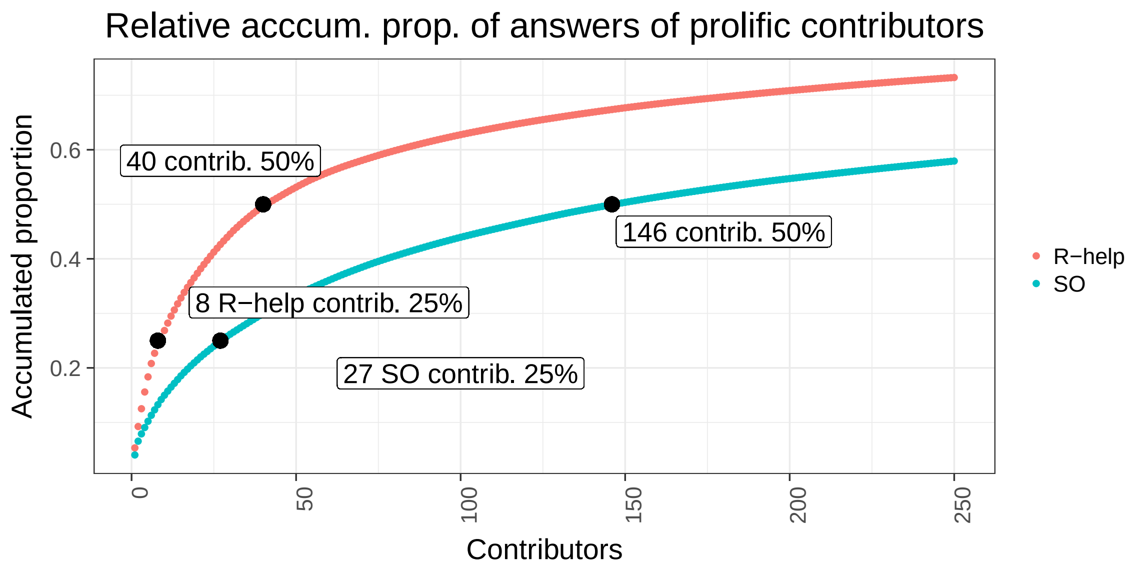


Figure 5.15: Accumulative proportion of answers by the most prolific users of each channel. The top 8 and top 27 contributors to R-help and Stack Overflow, respectively, are responsible for 25% of all the answers.

5.4 Discussion

In this section, we reflect on the results presented in the previous section, distill some themes from this work and relate them to relevant related research, and point out possibilities for future work. As we present the themes, we provide representative quotes extracted from the survey we conducted in the second phase of our study, using P# to indicate the participant ID.

5.4.1 Knowledge Creation and Curation

Our results show that both channels we studied provide similar knowledge support for asking questions and providing answers. However, there are some important differences between the channels, which we discuss in detail below and summarize in Table 5.3.

Table 5.3: Comparison of the ways knowledge is shared on Stack Overflow and the R-help mailing list.

	Stack Overflow	**R-help**
Knowledge construction	Mainly crowd	Mainly participatory
Topic restriction	Yes	No
Emphasis	Curating knowledge	Developing knowledge

Knowledge construction

Stack Overflow's gamification mechanism encourages users to be the first to answer questions [141]. In contrast, the R-help mailing list is a less competitive environment where users tend to build on other responses: users work as a team rather than acting as individuals searching for points. This perhaps explains why knowledge on Stack Overflow is built in a more crowdsourced manner, while knowledge on the R-help mailing list is usually built in a participatory manner.

Since the competitive Stack Overflow environment creates an incentive to be the first to answer rather than improve and build on other answers, it is common to find questions with several answers that provide the same information. For example, three of the six answers in the Stack Overflow question titled *"Resources for learning SAS if you are already familiar with R"*[17] reference the same books. And while Stack Overflow provides a powerful curation mechanism to ensure the best answers make it to the top, it does not explain why an answer is better than another.

In contrast, the R-help mailing list fosters a participatory environment where users discuss proposed answers—users tend to provide more background to answers and explain the rationale behind them. For example, the question *"Arrange elements on a matrix according to rowSums + short 'apply' Q"* that was posted to both Stack

[17]http://stackoverflow.com/questions/501917/resources-for-learning-sas-if-you-already-familiar-with-r

Overflow[18] and R-help[19] and illustrates how the two communities build knowledge. On Stack Overflow, each participant contributed a solution without any evidence of having collaborated with others. In fact, the person who asked the question actually stated that both answers worked. However, they selected one as the chosen answer and commented on the other with *"many thanks for your help, all! these did the trick"*, and there was no reflection on why they did so. On the R-help mailing list, users complemented each other's answers by providing further information and insights to existing answers.

Vasilescu *et al.* [170] showed that users who are active on both channels tend to provide answers faster on Stack Overflow than on R-help, suggesting that they are motivated by the gamification aspects of Stack Overflow, and thus, tend to gravitate towards crowd knowledge construction. While the crowd-based approach is prevalent, the construction of knowledge on Stack Overflow is not limited to the crowd-based approach. Participatory knowledge construction can also be observed, such as the up/down voting of questions and through the provision of comments. However, in most cases, participatory knowledge construction on Stack Overflow is used for editing answers (e.g., correcting grammar) or linking to previously asked questions. Similarly, some knowledge on the R-help mailing list is constructed in a crowd-based manner, but this is much less prevalent than participatory construction.

Tausczik *et al.* [161] examined how users of Math Overflow, a Q&A platform for mathematicians, collaborate and construct knowledge. They found that collaboration was diverse and fell on the spectrum between *independent* (crowd-based) and *interdependent* (participatory). Similar to our findings with Stack Overflow, the most common collaborative act was of an independent nature (i.e., providing information), while other contributions that built on existing work were less common (e.g., clarifying questions, critiquing answers, revising answers, and extending answers).

Our results seem to imply that Stack Overflow's gamification features, which have been effective at creating a large corpus of questions and answers, have the side effect of reduced collaborative knowledge creation between users. In their study on building Stack Overflow reputation, Bosu *et al.* [15] proposed six strategies for increasing reputation score, including *be the first to answer* and *answer at off-peak hours*, which indicates crowd knowledge creation. Furthermore, while Stack Overflow

[18] http://stackoverflow.com/questions/4333171/arrange-elements-on-a-matrix-according-to-rowsums-short-apply-q

[19] http://r.789695.n4.nabble.com/Arrange-elements-on-a-matrix-according-to-rowSums-short-apply-Q-td3068744.html

gives people the ability to vote on comments, it does not reward points to users that post comments. For example, some users search Stack Overflow for answers within comments and convert them to proper answers to gain reputation points[20].

An important research question that arises from these findings is whether Stack Overflow's model can be improved to provide better participatory knowledge construction support without hindering its ability to curate information for future use.

Topic restriction

Stack Overflow's participation rules only permit questions related to programming that have a clear answer, restricting the topics that can be discussed. In contrast, the R-help mailing list is suitable for discussing any topic related to the R language. For example, questions related to R but not focused on software development are not rejected by the R-help mailing list community—topics that trigger discussion are welcomed.

Stack Overflow questions that trigger a discussion are flagged as opinion-based or off-topic and typically closed. Correa and Sureka [23] found that 18% of deleted questions on Stack Overflow are subjective (i.e., ask for opinion). For example, a question titled *"What's a good example of really clean and clear [R] code, for pedagogical purposes?"*[21] was flagged as *off-topic* because the question was not related to software development. An R-help user wrote a fine explanation of the purpose of each channel in a message on the mailing list:[22]

> *"Got an R programming question that you think has a definite answer? Post to [Stack Overflow]. Want to ask something for discussion, like what options there are for doing XYZ in R, or why lm() is faster than glm(), or why are these two numbers not equal? Post to R-help. Questions like that do get posted to [Stack Overflow], but we [vote] them down for being off topic and they disappear pretty quickly."*

Squire reported that, despite the gains in participation and the response time provided by Stack Overflow, many development communities keep using mailing lists, either as a primary communication channel or as part of a hybrid solution where

[20]http://duncanlock.net/blog/2013/06/14/the-smart-guide-to-stack-overflow-zero-to-hero

[21]http://stackoverflow.com/questions/1739442/whats-a-good-example-of-really-clean-and-clear-r-code-for-pedagogical-purpos

[22]http://r.789695.n4.nabble.com/creating-an-equivalent-of-r-help-on-r-stackexchange-com-was-Re-Should-there-be-an-R-beginners-list-td4684587.html#a4684954

multiple channels are used, thus allowing for non-restrictive topics and fostering discussion [147]. Mailing lists are also favored for their simplicity and guaranteed delivery (i.e., knowing who will receive the email) [194].

Curated knowledge and knowledge development

One of the main benefits of Stack Overflow's crowd-based knowledge construction is the creation and curation of a pool of questions and answers. In contrast, R-help provides an environment in which users develop knowledge through participation, but this knowledge is not curated for future use. This makes it difficult for those that didn't participate in the creation of the knowledge (either actively or passively) to reuse the information.

While Stack Overflow has been successful, some users feel that by not fostering discussion, it restricts thinking that might lead to better answers, as P26 explained:

> *"Many developers share my view that [Stack Overflow] is a very bad model, ... [it] removes the value added by reading list traffic that doesn't seem directly relevant to a currently conceptualized question, but which may lead to a new conceptualization (out-of-the-frame thinking). [Stack Overflow] cannot do that."*

Similarly, P35 stated that they use the R-help mailing list if the questions are not 100% *"help-me-to-code-this"*.

However, Stack Overflow excels at creating and organizing a corpus of questions and answers. Its curation mechanisms provide tools for keeping the channel clean of what seems to be unnecessary information (e.g., flagging questions, deleting comments, editing messages, and demoting irrelevant answers), as P14 explained:

> *"[Stack Overflow] is an excellent model for providing a rich resource for users of R, which the R-help mailing list was not. Ability to include light markup, render code blocks nicely, no nested email threads all helps the experience of searching for and finding the help that a user needs, and I want to contribute to that."*

Another interesting aspect emerging from our findings is that the activity on the R-help mailing list is only marginally smaller than on Stack Overflow (the proportion of responses in each category fluctuated between 1.4 and 2 times). Further research is required to assess and verify the quality and effectiveness of answers.

Maturing as a community: from knowledge creation to knowledge curation

One of the major discoveries from Phase III of our study is the reduction in growth of active participation on Stack Overflow (asking or answering questions). In particular, the number of new questions with an overall positive score has started to decrease over time, but this is to be expected: it's likely that most sought after questions have already been asked. Given the large number of questions asked through Stack Overflow, many questions today are expected to be either duplicates of previous questions, be of low intrinsic value (too specific to be useful to others), or will end up flagged as not being a question. This decline of Stack Overflow participation has been discussed by community members[23] and the Stack Overflow development team[24], and both agree that Stack Overflow *"is not declining, it is serving its purpose quite well"*.

When we consider that the popularity of R continues to increase, this implies that when new members to the R community seek answers, they might not need to post questions—a testament to the value of the knowledge currently gathered by Stack Overflow. This also seems to imply that the activities of Stack Overflow contributors are shifting to tagging, flagging (e.g., as duplicates, unclear, or off-topic), or editing existing questions as opposed to answering new questions.

The frequency of postings on the R-help mailing list has now stabilized at approximately 200 questions per month. It is possible that by curating answers to the most frequently asked questions, Stack Overflow has contributed to a reduction of questions posted on the R-help mailing list, and the R-help community may now be able to concentrate on higher level questions than those asked in the past. It is also possible that R-help is now discussing questions that would be flagged as invalid on Stack Overflow. Further research should study the types of discussions that are now occurring on R-help and compare them to those from before the rise of Stack Overflow.

Active vs. passive participation

We found that more than 30% of users in both channels contributed only once, usually to ask a question. It is likely that these people continue to passively participate (e.g., consuming content), however, it is not clear why this is the case. It is possible that

[23]https://www.javacodegeeks.com/2016/09/stopped-contributing-stackoverflow-not-declining.html

[24]https://stackoverflow.blog/2016/10/podcast-89-the-decline-of-stack-overflow-has-been-greatly-exaggerated/

participants do not feel welcome, or that they feel that actively participating in each channel is not worth their time and effort. Another potential explanation is that these users contribute only when they have a very hard question for which they cannot find an answer anywhere else. Future work is needed in this area.

Similarly, approximately two thirds of users in both channels participate for a single month and then never actively participate again. This set of users includes the previous set (those that contribute only once) and might have related reasons for the halt in participation. It is also possible that these users are evaluating the channel, and after a short period of time, decide that they do not want to contribute further.

Nonetheless, new users continue to join both channels. In recent months, between 25% and 35% of users in any given month are new. This continuous growth seems to guarantee a steady flow of questions and answers in both channels, however, the experience and knowledge of the departed contributors is no longer present. It is possible that they become passive participants who are ready to continue contributing in the future. More research is needed to fully understand these behaviours.

Prolific contributors

There is a handful of users that are responsible for a large proportion of answers (less than 0.25% of users in both channels contribute 50% of the answers). It is not clear if the success of both channels can be attributed to them or not: e.g., it is possible that without them, other users might have answered those questions. Further research should look into the motivations of these individuals, consider the time they commit to help others, and also compare these patterns of activity with similar channels to see if this is a common phenomenon.

Several of these prolific contributors actively participate in both channels. The top contributor is responsible for almost 4% of the answers in either channel, and combined, the top 6 have authored approximately 10% of the answers. These prolific contributors are likely serving as a bridge of knowledge, moving information from one channel to the other.

When we looked at the professions of these contributors, we found that many of them are professors, book authors, or both. This implies that their goals include to learn and impart knowledge; for these reasons they probably see their participation as a major source of knowledge and an opportunity to diffuse it. Another group of prolific contributors are package maintainers who are likely experts in R and their corresponding packages, and perhaps see these channels as mechanisms to provide support to their users. These assertions should be confirmed in future work.

5.4.2 Threats to Validity

Here we examine and discuss threats to the validity of our approach [131].

Construct validity: To reach the emerging themes, we relied on subjective human judgment during the data coding phase. Researchers had to decide if a message fell within a specific coding category. To alleviate this issue, two researchers coded the qualitative data as part of the analysis process. We applied the Cohen Kappa coefficient on categories that were not mutually exclusive, but whose purpose was to trigger discussion between coders. We set a threshold of 0.8 as the minimum to obtain agreeable results, which is higher than the 0.6 suggested in the literature [82]. Regarding the quantitative analysis, there are several threats to validity associated with the mining of the R-help mailing list. First, our method to identify unique individuals might be inaccurate, however, we achieve results similar to those in [170]. Second, in our statistical analysis we considered the first email to the list as a question. And direct reply is considered an answer and further replies are considered comments; to do this we relied on the presence of *In-Reply-To* and *References*, which not all messages might include. On the other hand, Stack Overflow clearly identifies individuals and classifies posts into questions, answers and comments. It is also likely that a small proportion of emails are announcements or non-question emails. Regarding the mapping of individuals between the two datasets, as Stack Overflow stopped publishing the md5 of users, it is likely that the number of common contributors between the two channels is underestimated.

Internal validity: Stack Overflow's data is structured while the R-help mailing list consists of unstructured data. As a result, some of the mapping between the two channels was straightforward (e.g., a follow-up to a reply is a comment to that question), while in other cases it wasn't as obvious (e.g., identifying some emails as questions). To reduce the risk of bias when mapping the messages between channels (in the qualitative analysis), two researchers performed the mapping.

External validity: Our case study was exploratory in nature and we purposefully aimed to study the R community. Many R users are likely to be *casual devel-*

opers with limited or non-existent programming experience, with backgrounds that vary from biology to statistics, and thus, our findings may not apply to other developer communities. However, since Stack Overflow and mailing lists are widely used by other communities, we believe that our findings may be extended to these groups as well [147]. We do not claim the generalizability of our findings to other communication channels (e.g., Slack, GitHub), and further research is required to examine how knowledge is shared on other channels used by developers.

5.5 Conclusions

The purpose of this study was to understand how the R community collaborates when using different communication channels in the creation and curation of knowledge. In particular, we concentrated on studying how this community has used Stack Overflow (using the R tag) and the R-help mailing list to both ask and answer questions. Our analysis of a random sample of 400 threads from each channel shows that both channels are active communication channels where participants are willing to help others.

We found that knowledge contributed in response to a question can be classified into four main categories: answers, updates, flags, and comments. The number of responses in each of these categories was between 1.4 and 2.5 times greater on Stack Overflow than on the R-help mailing list. While all four types of contributions exist in both channels, they exhibit differences. For example, on Stack Overflow, answers are more focused towards step-by-step tutorials, while R-help answers are more likely to be suggestions or alternatives. Similarly, on Stack Overflow, updates are focused on language (grammar and spelling), while on R-help, updates are expansions of previous responses.

The analysis of these questions and answers shows that knowledge is constructed in each channel in a different manner. On Stack Overflow, there is a tendency to use a crowd approach: participants contribute knowledge independently of each other rather than improve other answers. This is likely a result of the gamification used on Stack Overflow where the person who provides the best answer is the one that gains the most points. In contrast, the R-help mailing list uses a participatory approach where participants are more likely to build on other answers, collaborating to find the best solution.

Another important difference between both channels is that Stack Overflow focuses on making knowledge available for future retrieval; knowledge on the R-help mailing list focuses on the discussion of knowledge, but not in its long-term storage or retrieval. Respondents to our survey also commented that while it is easy to find answers on Stack Overflow and make sense of them, on R-help it is not only hard to find the relevant answers to a question, but it is also hard to see how the many responses to a question relate to each other, and ultimately, what the best answer may be.

Another result of our research is that participants appear to prefer Stack Overflow for asking questions that are expected to have a direct answer. They prefer to use the R-help mailing list when the question requests opinions (Stack Overflow forbids them) or when they want to reach core developers of the R project. Some participants ask the same question in both channels in the hopes of gaining the advantages both channels offer. Additionally, R-help has the ability to complement Stack Overflow by providing a medium where the rationale of answers can be discussed.

A major result of our quantitative study is the impact of a small fraction of the members of both channels who answer a large proportion of the questions in both channels. There is also a dedicated core of members who contribute to both channels, connecting the two communities. At the same time, a large proportion of members participate very few times over a short period of time.

It was also interesting to observe that the frequency of new questions in Stack Overflow is starting to slow down, especially those with a positive score. This effect is probably due to the fact that the most frequently asked questions in R have already been asked and answered; the new questions are either inherently more difficult or more specialized and therefore have a smaller pool of members who would benefit from them. However, research is needed to understand this effect.

Overall, this research shows that the R community is committed to using both channels to help others. Each channel has advantages and disadvantages, and the community appears to be using both effectively to create and curate knowledge regarding the R language. Furthermore, our typology of knowledge artifacts summarized in Table 5.2 can be used by other researchers that wish to study and understand how knowledge is constructed and curated in other channels or across other communities. As new channels (such as Slack) become more widely adopted, studying these newer channels and comparing them to existing channels is an imperative aspect of understanding knowledge formation in software development.

Part III

A Theoretical Knowledge Building Framework

Chapter 6

Modeling Knowledge Transfer and Knowledge Activities in Software Development

"Knowledge flows along existing pathways in organizations. If we want to understand how to improve the flow of knowledge, we need to understand those pathways."

– Don Cohen and Laurence Prusak, 2001

Conceptualization plays an important role in scientific research [154]. It is a necessary step towards theory formulation [89], and often distinguishes between scientific reporting and conducting research [39]. Without conceptualization and theory generation, the scientific outcome often results in little value [129]. It is important to not only apply within-study conceptualization, but also across-study synthesis and conceptualization, which enables researchers to reflect and theorize beyond the scope of a single study, and to construct a bigger picture or theory.

In this chapter, I describe a *qualitative meta-synthesis*, i.e., an interpretive integration of qualitative findings in the form of an interpretive syntheses of data: either conceptual/thematic descriptions or interpretive explanations [133, p. 199]. The studies used as ground for the meta-analysis are: (1) the study on how social and communication media affect and disrupt software development (described in Chapter 4), and (2) the study about knowledge curation within the R community (described in Chapter 5). The specific meta-synthesis technique I used is *Taxonomic Analysis*, an inductive form of domain analysis useful for theory development. It has much in common with the axial and selective coding processes associated with grounded theory [158, 133]. Sandelowski and Barroso [133] explain that *"in contrast to effect sizes, which show the quantitative range of findings, taxonomies show the conceptual range of findings and provide a foundation for the development of conceptual descriptions and models, theories, or working hypotheses"*. The goal is not to determine the prevalence of findings, but rather to identify the underlying conceptual relations signified in, albeit not necessarily explicitly expressed, in the findings [133]. This process is characterized by a degree of innovation as it brings separate 'parts' together to form a 'whole' greater than the sum of its parts [178].

As described earlier in this thesis, we have studied communication and social media use in software development at length. Our research effort focused on understanding *how communication and social media affect and disrupt software development*, and consisted of a series of studies [157, 155, 191]. The findings from these studies provided valuable *descriptive* insights on the phenomena and revealed how developers were using social and communication media. However, we lacked a deeper *interpretive* understanding of **'why'** these observations and patterns were happening; e.g., Why were certain channels adopted over others? Why were the observed challenges happening? Why were the community participation patterns the way we observed? Kaplan [76] emphasized that *"data describe* ***which*** *empirical patterns were observed, and theory explains* ***why*** *empirical patterns were observed or are expected to be observed"*. Indeed, we realized that we needed to conceptualize our findings. As our understanding expanded over time, a mental model of *how software is built* was formed, and later a **theoretical framework of Knowledge Building in Software Development** emerged.

6.1 Background

First, we begin with an overview of knowledge work, as a basis for our conceptualization of knowledge transfer and knowledge activities in the context of software engineering.

6.1.1 Understanding The Importance of Knowledge Work

In 1957, Drucker was perhaps the first to recognize knowledge as the world's most important resource (i.e., as opposed to labor or capital) [36]. There were few indicators of upcoming social or technological transformations during that time, yet Drucker's remarkable foresight proved insightful and sparked an interest in understanding knowledge work. Not long after, Galbraith [48] and Toffler [162] predicted an emergence of a new class of *technical-scientific experts* and highly educated individuals—both are early mentions of knowledge workers. In Drucker's [37] view, what made *knowledge workers* different was not only the high level of education they obtained, but more importantly, that they owned the organization's means of production. Inspired by F.W. Taylor's contributions on the systematic analysis and study of work, Drucker wished to focus on the productivity of knowledge workers: *"from now on what matters is the productivity of non-manual workers."* [35]

With an increasing awareness for knowledge and the rise of "knowledge-intensive firms" [2, 152], where highly qualified members trade knowledge, scholars have started defining knowledge work. In Drucker's view [38], knowledge workers are capable of self-management, involved in defining the scope of their work, continuously innovating, focused on both the quality and quantity of the output, and continuously learning and teaching others. Reich [122] described these workers as "symbolic analytic" workers capable of identifying and solving problems, and connecting sets of information. Sveiby and Lloyd [159], noticing some of Drucker and Reich's points in their observations, suggested that knowledge-intensive firms need to be high on skilled know-how workers and have a high level of knowledge managerial skills. Winslow and Bramer [186] suggested that knowledge work happens when individuals use their cognitive abilities, technical know-how, interactions with others, and individual creativity to achieve work outcomes.

The term "knowledge worker" may give the impression of an expert individual engaged in solving a problem alone, however, this would be an incomplete understanding of knowledge work. In the literature, knowledge work is often seen as *a collective endeavor* to solve complex and novel problems. *"Perhaps more than any other form of work, knowledge work has pointed to the need for individuals to collaborate together, rather than work alone"* [73]. Trauth [163] states that in today's work environments, it's necessary for individuals to collaborate with others of similar or complementary areas of competence.

Some of the suggested definitions of knowledge work use the *level of familiarity* with the work situation or with the problem as an important distinction. For instance, Blackler [14] described a conceptual framework that categorized organizations and knowledge types according to two main dimensions: (1) whether the emphasis is on an *individual* or *collaborative* endeavor, and (2) whether the focus is on *familiar* or *novel* problems (as summarized in Table 6.1). In this framework, Blackler illustrated a shift towards organizations that focus on collaborative and novel types of activities, a quadrant representing knowledge work. Note that while these distinctions provide conceptual value, they are less than practical. In reality, the problems and work situations employees face can rarely be considered as completely novel or familiar. Even in situations of non-routine problems, knowledge workers can rely on their domain knowledge and past experience.

Table 6.1: Blackler's classification of organizations and knowledge types (source: Blackler, 1995 [14]).

	Focus on familiar problems	Focus on novel problems
Emphasis on collaborative endeavor	**Knowledge-Routinized** organizations	**Communication-Intensive** organizations
Emphasis on contributions of key individuals	**Expert-Dependent** organizations	**Symbolic-Analyst-Dependent** organizations

Knowledge work has invariably influenced work environments and relationships [73], thus leading to new modes of work that rely on group cooperation (e.g., collaborative work systems and computer-supported cooperative work).

6.1.2 Knowledge Work and Technology

Drucker's early predictions about knowledge work were not based on technological advances or social transformations, but rather on his perception of the increased cognitive demands on employees [36]. Since then, we have seen significant social and technological transformations; thus, an important question to consider is: *what is the role of technology in regard to knowledge work?* As modern knowledge work and technology have become inextricably linked together, some scholars argue [121] that contemporary definitions of knowledge work would be incomplete without involving technology [73]. Other scholars caution attributing so much dependence on technology when defining knowledge work, as technology seems more of a resource for accomplishing knowledge work more efficiently, and less of an attribute of the knowledge work itself.

For the scope of this thesis, instead of discussing technology in general, we'll focus on socially enabled tools and communication channels. We see three major roles of technology in regard to knowledge work [73]:

1. Social and communication media *provide the means for managing knowledge*
2. Software can help *automate many routine activities* in the workplace
3. Social and communication media can *serve as the content of the work itself*

Moreover, while there is a consensus in the literature that technology does not necessarily define knowledge work, technology has in fact increased the cognitive demands of doing knowledge work.

6.1.3 What is Knowledge?

Now that we have gained an understanding of the importance of knowledge work, we can address the question of *what is knowledge?* (and what is *not* knowledge?), and begin discussing the different knowledge types and their transfer in a software development context.

Unfortunately, knowledge is an elusive concept that is difficult to define. Although there is a large existing body of literature on knowledge management, knowledge work, and organizational knowledge [69, 12], there is no consensus among scholars on the definition of knowledge. In fact, the many different views of knowledge identified in the literature and the various types presented in this paper emphasizes the complexity of the topic [14]. Moreover, since knowledge is multi-faceted and complex, considering the different facets of knowledge independent of each other would be a mistake.

Knowledge management literature [67, 28, 106, 174, 107] commonly distinguishes between data, information, and knowledge (also referred to as the DIKW hierarchy). *"There is a consensus that data are discrete facts, but after that, consensus is lacking. The lack of consistent definitions for data, information, and knowledge make rigorous discussions of knowledge management difficult"* [69]. Nissen [107] operationalizes the triangular hierarchy between these concepts using two dimensions, *abundance* and *actionability*, to emphasize the difference between them (as shown in Fig. 6.1). Nissen explains that in a given domain, there is an abundance of data, with exponentially less information and even fewer chunks of knowledge. However, data is not particularly

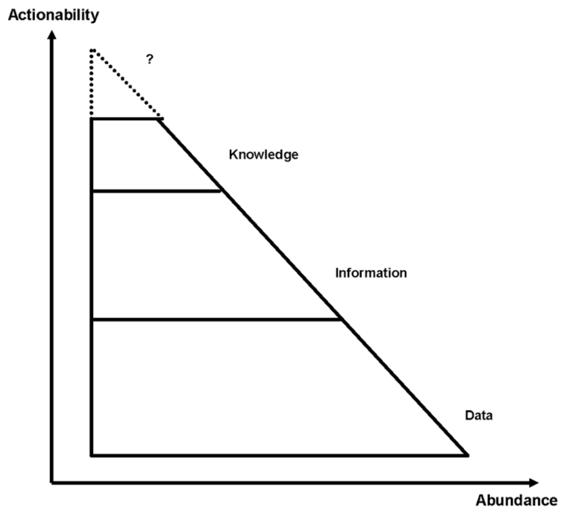


Figure 6.1: Knowledge hierarchy according to Nissen [107, p. 253].

powerful for supporting action, while knowledge allows individuals and organizations to enact and convert data into action. To further clarify these concepts, *information* is not a physical thing (e.g., data), but rather a physically-embodied arrangement of data. While *knowledge* is a mixture of information, experience, contextual information about specific situations, values, and intuition. We've chosen to rely on Davenport and Prusak's definition of knowledge [28]:

Definition: "Knowledge is a fluid mix of framed experiences, values, contextual information, and expert insight that provides a framework for evaluating and incorporating new experiences and information. It originates and is applied in the minds of knowledge workers. In organizations, it often becomes embedded, not only in documents or repositories, but also in organizational routines, processes, practices, and norms."
(source: Davenport and Prusak, 1998 [28, p. 5])

Next, we describe our research approach and process. Then in Section 6.3, we present a theoretical framework that emerged from our work, which describes different knowledge types and their transfer in a software development context.

6.2 Methodology

The proposed theoretical framework was created by applying a bottom-up approach consisting of a gradual and iterative analysis, and interpretive synthesis of empirical data from two previous studies (Fig. 6.2 shows an overview of our research process). I describe the study goal and method used in each of these studies below. I do not describe the full details of the studies here and refer readers to see additional information about these studies in Chapters 4 and 5 of this thesis. Both of our primary studies were iterative, and earlier iterations were also published [156, 192].

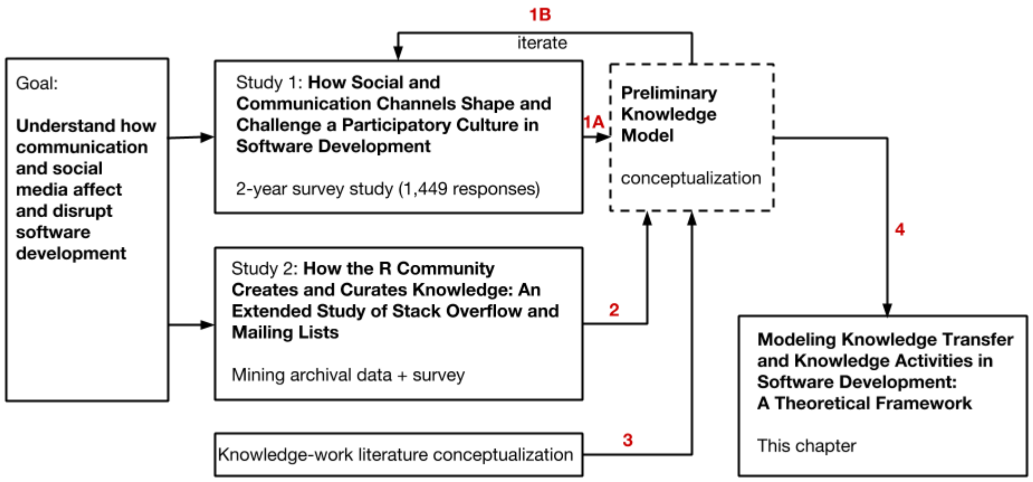

1B
iterate
Goal:
Understand how communication and social media affect and disrupt software development
Study 1: How Social and Communication Channels Shape and Challenge a Participatory Culture in Software Development
2-year survey study (1,449 responses)
1A
Preliminary Knowledge Model
conceptualization
Study 2: How the R Community Creates and Curates Knowledge: An Extended Study of Stack Overflow and Mailing Lists
Mining archival data + survey
2
4
Knowledge-work literature conceptualization
3
Modeling Knowledge Transfer and Knowledge Activities in Software Development: A Theoretical Framework
This chapter

Figure 6.2: An overview of the research process (and illustrating the relationship to our previous work).

6.2.1 The Studies Used as Ground for Meta-Analysis

Study 1: How Social and Communication Channels Shape and Challenge a Participatory Culture in Software Development [155]

We surveyed developers that are active on GitHub in two rounds of surveys: once in 2013 (1,492 responses), and once in 2014 (332 responses). Ignoring incomplete submissions, we had a total of 1,449 survey responses. We inquired about communication channel use for a set of 11 development activities, and investigated which channels were most important to developers, and why those channels were important. Additionally, we inquired about challenges developers face using communication media.

Study 2: How the R Community Creates and Curates Knowledge: An Extended Study of Stack Overflow and Mailing Lists [191]

The main goal of this work was to empirically compare how knowledge, specifically knowledge in question-and-answer (Q&A) form, is sought, shared, and curated on both the R-help mailing list and the R tag on Stack Overflow. We also aimed to gain insights on the knowledge curation process used within the R community (e.g., participation patterns and behaviors). For this purpose, we used a mixed methods exploratory case study methodology. We mined data from the public archives of both the R-help mailing list and Stack Overflow, initially from September 2008 until September 2014, and later extended up to September 2016.

For the sake of clarity, first we present the theoretical framework in Section 6.3, and then we provide its empirical grounding in Section 6.4.

6.3 Knowledge in Software Development: A Theoretical Framework

In this section, we introduce a theoretical framework of knowledge transfer in software development. The knowledge framework builds on existing concepts and models of knowledge work and CSCW, and on empirical findings from two of our recent studies. We combine these theoretical concepts and apply them to our model of software development, resulting in a novel knowledge framework. Later in Section 6.4, we demonstrate how our studies have led to this framework.

6.3.1 Individual Knowledge Types

In 1985, Naur [105] described programming as a *"(knowledge) theory building"* activity, suggesting that programming (in the loose sense) is not just about producing a program, but also includes the developers' insights, theory, and knowledge built in the process. By describing two real cases, Naur shows that programming involves not only the source code and documentation (i.e., **encoded knowledge**), but also the knowledge in developers' heads (i.e., **tacit knowledge**). He further explains that tacit knowledge is essential to software support and modification—if a developer was to leave the project, the tacit knowledge allowing these activities would be lost (e.g., design reasoning). Tacit knowledge can be further decomposed into procedural knowledge and declarative knowledge [128]. Procedural knowledge is dynamic and involves all the information related to the skills developed to interact with our environments; this knowledge is acquired from practice and experience. While declarative knowledge is static, based on facts, and concerned with the properties of objects, persons, and events and their relationships. Compared to procedural knowledge, declarative knowledge is easy to describe and communicate.

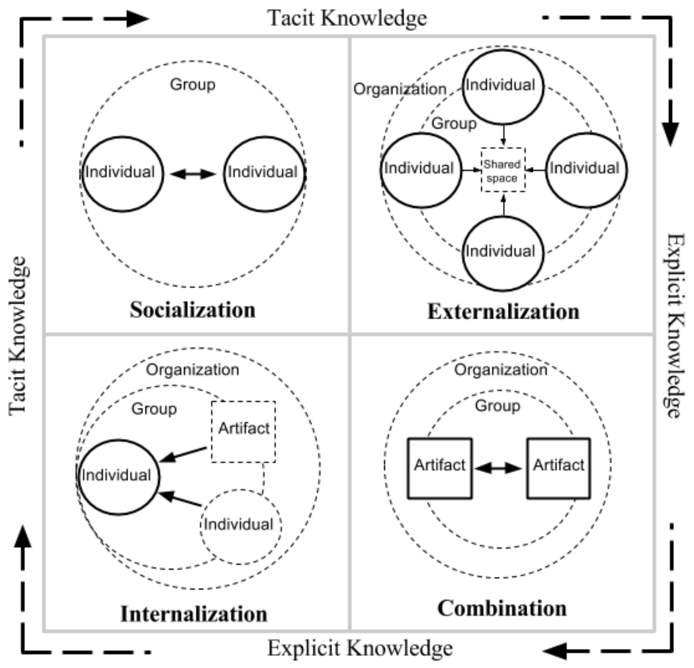


Figure 6.3: Our depiction of Nonaka's knowledge creation patterns. [108]. We visually illustrated his proposed knowledge transfer patterns.

The literature on knowledge management further distinguishes between different subtypes of knowledge embodied within people's heads: **tacit**, **implicit**, and **explicit** knowledge. Polanyi [118] talked about tacit knowledge, a form of knowledge that can't be articulated. Spender [146] suggested that some tacit knowledge has only yet to be articulated, while other tacit knowledge is incapable of it entirely. Nonaka [108, 109, 188] suggested that knowledge is created out of a dialog between tacit and explicit knowledge, forming four patterns: (1) from Tacit to Tacit (e.g., apprenticeship), (2) from Explicit to Explicit (e.g., reconfiguring existing pieces of data and information), (3) from Tacit to Explicit (e.g., learning craft skills, a dialog among organizational members and the use of metaphors to articulate hidden tacit knowledge), and (4) from Explicit to Tacit (e.g., internalization of new knowledge, learning by doing). These four patterns are often referred to as the SECI model—an acronym for socialization, externalization, combination, and internalization [109, 110]. We illustrate these patterns in Figure 6.3. A metaphor of an *upward spiral* is often used to illustrate how the process depicted by this model enables organizations to continuously create new knowledge. Later, Grant [54] re-framed Nonaka's model to not only consider tacit and explicit knowledge, but also to include two levels of knowledge: knowledge at the *individual* level and at the *organization* level (as shown in Fig. 6.4).

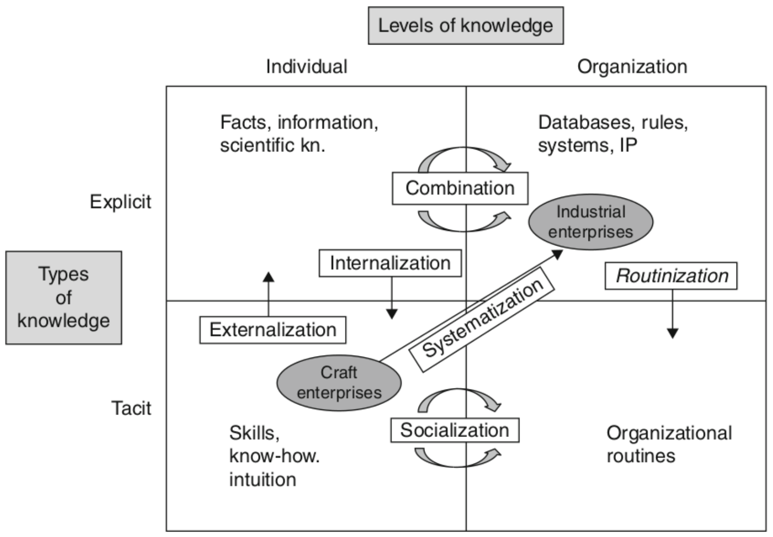
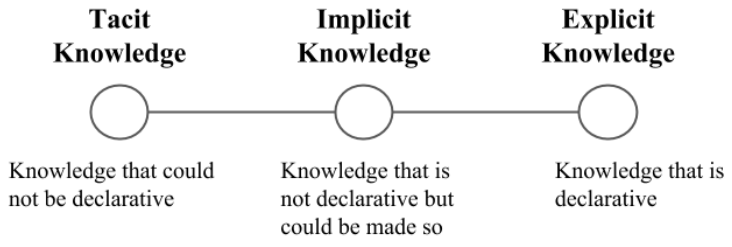


Figure 6.4: Grant's version of the SECI model, which re-frames the model to also consider the individual vs. organization levels of knowledge [54].

Leonard and Sensiper [90] suggest that instead of separate constructs, tacit and explicit knowledge can be conceptualized as a continuum. Building on this interpretation, Griffith *et al.* [56] position the knowledge forms at both ends and in the middle of the continuum (see Fig. 6.5): knowledge that is declarative (i.e., **explicit knowledge**), knowledge that is not declarative but could be made so (i.e., **implicit knowledge**), and knowledge that has never been and could not likely be declarative (i.e., **tacit knowledge**). Yet they acknowledge that while some knowledge fits nicely into one of these forms, other knowledge may share elements of all of them.

Other scholars, such as Collins [21] and Blackler [14], make use of a different categorization for knowledge types. Although defined differently, two of the knowledge types they describe, *embrained knowledge* (knowledge that is dependent on conceptual skills and cognitive abilities) and *embodied knowledge* (action-oriented and partially

Figure 6.5: Tacit, implicit, and explicit knowledge types conceptualized as a continuum. For our purposes of this chapter, I visualized below Griffith *et al.* [56] suggestion to portray them this way.


Tacit Knowledge
Implicit Knowledge
Explicit Knowledge
Knowledge that could not be declarative
Knowledge that is not declarative but could be made so
Knowledge that is declarative

explicit knowledge), align with the individual knowledge types we defined (tacit, implicit, and explicit). Additional knowledge types described by Blackler—*encultured*, *embedded*, and *encoded*—relate to social knowledge and are covered next.

6.3.2 Social Knowledge Types

The use of communication media in software development extends the knowledge building process by enabling the transfer of knowledge between members. This facilitates individual learning and expression, as well as coordination and collaboration between members of a community. Wasko *et al.* [179] distinguish different theories of knowledge based on the kind of knowledge they help capture or communicate: (1) 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 (i.e., tacit, implicit, or explicit knowledge); (2) knowledge that exists in artifacts and can be accessed independently from any human being (i.e., encoded knowledge); and (3) knowledge that is *"socially generated, maintained, and exchanged within emergent communities of practice"* (i.e., **social knowledge**).

Developers seek to form and to work in *communities of practice*, where knowledge can be generated and maintained by the community, preserving the knowledge even when individual members leave. This type of collective knowledge, refers to publicly available knowledge or knowledge embedded within the routines, culture, and norms of the team [146]. Nonaka [108] classified social knowledge as either: **objectified** (explicit knowledge known to all members), **collective** (explicit knowledge that has been externalized), or knowledge that has been formed as a **shared understanding** among the team members (tacit knowledge that has been formed through collective action). Similar to how they treated individual knowledge, Griffith *et al.* [56] position these social forms of knowledge on a continuum to represent different levels of interdependence (e.g., greater shared knowledge is more likely to have a high level of interdependence).

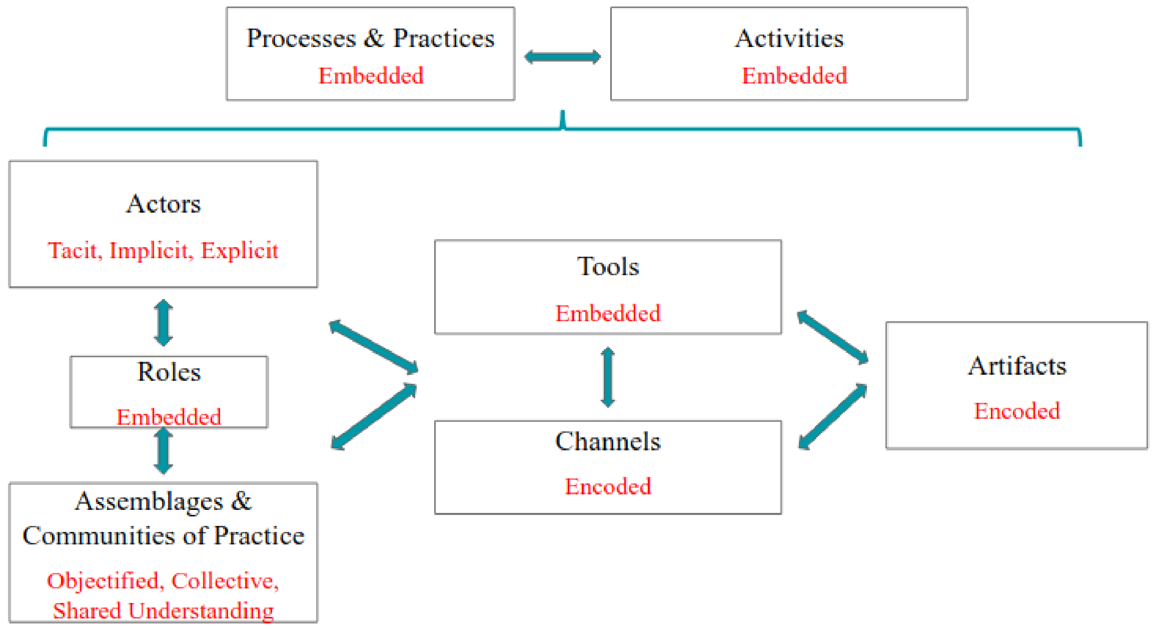


Figure 6.6: An illustration of the different knowledge types and where they reside in the context of software engineering.

Blackler's [14] *encultured knowledge* (the process of achieving a shared understanding) and *embedded knowledge* (which resides in roles, formal procedures, and systematic routines), align with the social knowledge types described above (objectified, collective, and shared understanding). However, we believe that there is added value in pointing out Blackler's **embedded knowledge** and using it on top of Nonaka's [108] social knowledge definitions, as it makes it clearer and more explicit when dealing with this type of knowledge. The fifth and final knowledge type described by Blackler, **encoded knowledge**, describes information that has been externalized, such as books, documentation, and information encoded and transmitted through social and communication channels.

Figure 6.6 depicts the different knowledge types and where they reside in the context of software engineering.

6.3.3 Team, Community, and Organizational Knowledge

Linking the individual knowledge types (Section 6.3.1) and social knowledge types (Section 6.3.2) is the team's *potential* and *usable* knowledge. Griffith *et al.* [56] describe **potential knowledge** as the combination of an individual member's tacit, implicit, and explicit knowledge, as well as the team's objectified, collective, and shared understanding knowledge. The amount of **usable knowledge**, i.e., potential knowledge that has been realized, depends on the individual member's *absorptive capacity* (i.e., an individual's ability to utilize available knowledge), the availability of relevant *communities of practice*, and the team's *transactive memory* and *synergy* [56].

The ability to transform individual's tacit and explicit knowledge into team-level social knowledge is expressed through **transactive memory** [181, 182, 56] (as illustrated in Fig. 6.7). *"The transactive memory system in a group involves the operation of the individuals' memory systems and the processes of communication that occur within the group"* [181]. It is a shared system for encoding, storing, and retrieving knowledge available to the group, and a collective awareness of who knows what. Wegner [182] describes three core factors necessary for a group's transactive memory to be effective: (1) directory updating ("who knows what?"), (2) information allocation (distributing knowledge based on whose expertise is best suited), and (3) retrieval coordination (retrieving knowledge given the expertise in the group). The known experts or assigned individuals in the group are usually responsible for the encoding, storage, and retrieval of any new information. In the absence of assigned experts, more subtle rules are used to direct continuing responsibility [181].

We can further distinguish between knowledge that is *brought into the team* (through transactive memory) and additional knowledge that is *generated within the team* (referred to as synergistic knowledge). **Synergistic knowledge** defines knowledge generated within the team (as illustrated in Fig. 6.7), beyond the potential knowledge initially held by the team's individuals [8]. Both transactive memory [181, 182, 56, 93, 103] and synergistic knowledge [180] moderate the transformation of potential team knowledge into usable knowledge. Additionally, access to communities of practice enables the transfer of both explicit and tacit knowledge, and these communities provide the learning context needed to enact potential team knowledge [90, 56]. In addition to long term knowledge held by the team, a team also relies on fleeting knowledge in the form of the **team's situational awareness** [22, 42] (also referred to as the *'team situation model'* or *'dynamic understanding'*). This

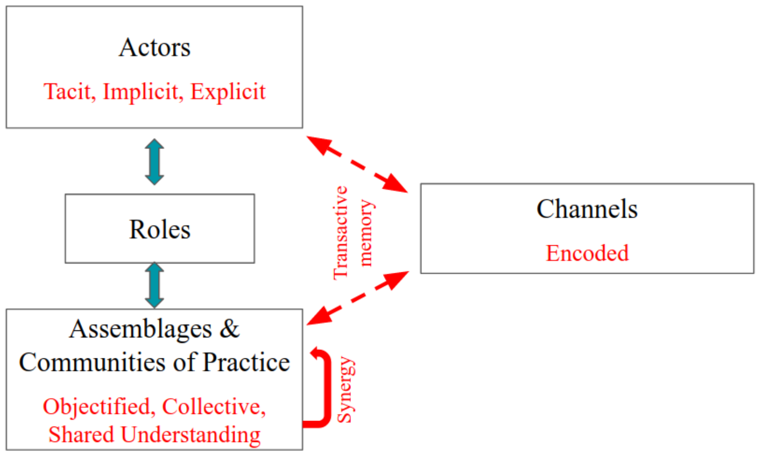

Actors
Tacit, Implicit, Explicit
Roles
Assemblages & Communities of Practice
Objectified, Collective, Shared Understanding
Channels
Encoded
Transactive memory
Synergy

Figure 6.7: An illustration of how transactive memory transfers individual knowledge to the team-level , and how synergy generates additional knowledge within the team. For illustration purposes, this figure focuses on the core constructs involved in the process, however, other constructs may also be affected by the process (e.g., encoded artifacts).

knowledge is the team's collective understanding of the specific situation acquired during the execution of a task, which changes depending on the specific task [22].

Broadening our perspective, we can further use the *framework constructs* defined in this section to describe community and organizational knowledge (as illustrated in Fig. 6.6). Thus organizational knowledge can consist of: (1) knowledge that resides in the individual members and teams, (2) knowledge that is captured in the community records and channels, and (3) knowledge that is embodied in the community's procedures, routines, norms, and layout [71]. Nonaka [108] believed that *"organizational [and community] knowledge creation hinges on a dynamic interaction between the different modes of knowledge conversion. That is to say, knowledge creation centers on the building of both tacit and explicit knowledge and, more importantly, on the interchange between these two aspects of knowledge through internalization and externalization."* The systems that moderate these knowledge exchanges are **knowledge activities**, which we describe next.

6.3.4 Knowledge Activities in Software Development

Knowledge management scholars have explored a variety of practices, actions, and activities carried out in personal and organizational knowledge work [28, 98, 144, 29, 40, 61, 172, 173, 124, 123]. As a result, multiple taxonomies for knowledge activities exist (a selection of recent taxonomies are shown in Table 6.2), each mapping the types of activities knowledge workers perform. While these sources describe the knowledge activities individuals partake in, an explicit definition of a knowledge activity is lacking in the literature. Knowledge management scholars further emphasize that *"there is no real consensus on definitions that describe the activities that are required to transform information to knowledge…This lack of consensus on the terms used to describe components of knowledge management makes rigorous debate difficult"* [12].

Table 6.2: A selection of existing taxonomies for knowledge activities.

Davenport, 1999 [28]	**Davis, 2003 [29]**	**Sellen and Harper, 2003 [137]**	**Efimova, 2004 [40]**	**Holsapple and Jones, 2004 [70]**	**Hädrich, 2008 [61]**	**Reinhardt *et al.* 2011 [124]**
Acquisition	Authoring	Acquiring	Awareness	Acquisition	Acquisition	Acquisition
Application	Review	Annotating	Collaboration	Assimilation	Authoring	Analyze
Creation	Planning	Composing	Conversations	Control	Co-authoring	Authoring
Dissemination	Collaboration	Organizing	Creativity	Coordination	Expert Search	Co-authoring
Documentation	Communication	Processing	Establishing and maintaining relations	Emission	Feedback	Dissemination
Packaging			Exposure	Generation	Invitation	Expert search
			Lurking	Leadership	Training	Feedback
			Making sense of information	Measurement	Update	Information organization
			Organizing ideas	Selection		Information search
						Learning
						Monitoring
						Networking
						Service search

For the purpose of defining the term *'knowledge activity'*, we rely on Kuutti's formulation of an activity [79]. However, we rephrased his definition to focus on knowledge activities, replacing the term 'object' with the term 'knowledge target'.

Definition: A knowledge activity is a 'form of doing' directed at a knowledge target, and knowledge activities are distinguished from each other according to their knowledge targets. Transforming the knowledge into an outcome motivates the existence of an activity. Knowledge can be a material thing, but it can also be less tangible (such as a plan) or completely intangible (such as a common idea) as long as it can be shared for manipulation and transformation by the participants of the activity.

(source: Kuutti, 1996 [79])

The most recent description of knowledge activities that is applicable to software development is the knowledge action typology for knowledge workers offered by

Table 6.3: Our taxonomy of knowledge activities and their description in a software development context.

Authoring:	Independently creating code and related artifacts. For example, documentation, tutorials, architecture documents, technical presentations, etc. This activity enables the externalization of implicit and explicit knowledge into encoded knowledge.
Co-Authoring:	Collaboratively creating code and related artifacts (e.g., shared documents, presentations). Through the use of transactive memory and team synergy, this activity enables the externalization of individual implicit and explicit knowledge into encoded knowledge, and the generation of added synergistic knowledge within the group.
Searching:	Looking up information, services, or experts. This activity refers to the search and retrieval of internalized or encoded knowledge, and may make use of the team's transactive memory (e.g., finding the expert on a specific API).
Learning:	Acquiring new knowledge, skills, or understanding for the purpose of supporting work. This activity refers to how individuals internalize knowledge. It can support tacit and implicit knowledge acquisition (i.e., procedural knowledge), and explicit knowledge acquisition (i.e., declarative knowledge)—the supported types are determined based on the actions executed. For example, this includes knowledge gained from formal sources such as books, courses, or training, and informal sources such as social interactions, feedback, learning by doing, and observing others. The acquired knowledge gains value when it comes from the context in which the knowledge is to be used.
Information Organization:	Managing personal or organizational information. This involves the management (e.g., transfer, filtering, moderation) of encoded knowledge stored within artifacts or within channels. This can also refer to the transfer of internally stored knowledge (i.e., implicit or explicit) into externally encoded knowledge (e.g., making a task list), and internally storing knowledge about *'where things are'* [181].
Feedback:	Includes both gaining and providing feedback on technical artifacts such as code (e.g., via code reviews or architecture reviews). This activity allows team members to contribute to and take advantage of the tacit and implicit knowledge within the team (knowledge which is typically difficult to transfer).
Dissemination:	Sharing information about work results, which can be either content (e.g., code, documentation, etc.) or developer activity (e.g., status reports). This activity enacts group knowledge processes (encoding and storage), and supports the core factors necessary for effective use of transactive memory: updating the directory of 'who knows what?', and information allocation among group members. This activity also contributes to the team's situational awareness by signaling and providing helpful cues and information [132].
Monitoring:	Staying up to date about new technologies, practices, trends, and tools for software development. It can also refer to keeping track of one's development activities or a collaborator's activities, thereby providing support to the team's situational awareness. Monitoring can also lead to unplanned or serendipitous discovery of new information.
Networking:	Creating and maintaining a work-related social network, or interacting with others to exchange information and develop contacts (e.g., at meet-ups or conferences).

Reinhardt *et al.* [124], which the authors formed based on previous work, existing taxonomies, and an empirical study (involving a survey and task execution study). However, Reinhardt *el al.*'s typology is aimed at knowledge work, not software development, and it needs to be adapted and refined in order to be used for capturing the activities found in software development. A preliminary use of Reinhardt's theoretical framework in software engineering by Ford *et al.* [46] helped identify software engineering personas.

More importantly, neither Reinhardt *el al.*'s knowledge framework nor any of the existing knowledge-activity taxonomies **make use of or link to the knowledge constructs and knowledge types described above**. Thus in this chapter, we propose a coherent taxonomy of knowledge activities in the context of software development. Each knowledge activity is defined and characterized with the knowledge terminology and constructs established earlier in this chapter. Table 6.3 shows our knowledge activity taxonomy.

6.4 Empirical Grounding: Reflecting Back on Our Original Studies

In this section, we show how our primary studies led to the proposed theoretical framework (as described in Section 6.3). In order to facilitate traceability between the empirical findings and the framework, we reflect back on our primary studies and include references to the relevant constructs of the theoretical framework. For conciseness, not all aspects of the framework are presented in this reflection, but only the most pertinent ones. We believe it demonstrates how this is grounded in empirical work. It is important to note that, the framework is not only based on our empirical findings (Chapters 4 and 5), but also on our implicit insights and extensive experience of studying communication media use in software development.

6.4.1 How Communication Media Supports and Impedes Knowledge Transfer in Software Development

From previous studies [166, 27, 141, 142], we learned that developers use social and communication media to support their development activities, however, we didn't know what those activities were and which social and communication channels supported these activities. Our study (which I described in Chapter 4) has provided insights into both. We found that software developers use a large assortment of communication media: face-to-face, books and magazines, web search, news aggregators, feeds and blogs, content recommenders, social bookmarking, podcasts, discussion groups, public and private chat tools, professional networking sites, developer profile sites, social networking sites, code hosting sites, microblogs, project coordination tools, and Q&A sites. Moreover, we formed and confirmed a preliminary list of 11 development activities (see Fig. 6.8), and linked these activities to specific channel types. For example, our findings showed that developers used Web Search and Q&A sites such as Stack Overflow to search for answers. Figure 6.8 summarizes our findings and shows which social and communication channels developers indicated to use when performing their professional activities. Furthermore, these findings helped us form a conceptual model of a software development knowledge ecosystem, linking developers, activities, and channels together.

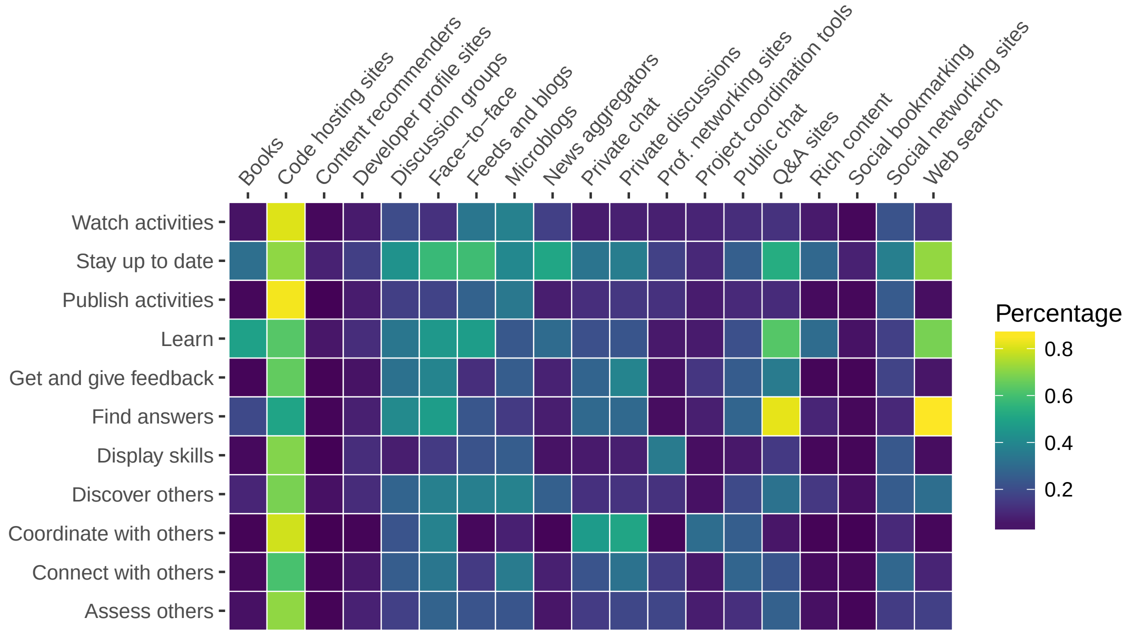


Figure 6.8: Channels used by software developers and the activities they support. Results are based on a survey with 1,449 developers (Chapter 4).

By building on our preliminary list of developer knowledge activities [155] and related work [124, 123], we formed a taxonomy of knowledge activities for software development. The revised taxonomy of knowledge activities, which is part of our proposed theoretical framework (presented in Section 6.3.4), has been refined and extended multiple times through successive studies. Table 6.4 illustrates how our conceptualization of the knowledge activities in software development has evolved from our original study [155]. Some activities were merged, some were generalized or renamed, and other new activities were added. The revised list of knowledge activities we propose in this chapter (shown in Table 6.3) has emerged from our findings. However, these activities have been also reinforced by Reinhardt *et al.*'s [124] findings from the knowledge work domain, and later these activities were validated by Ford *et al.* [46] in a software development context.

Table 6.4: Linking our preliminary formulation of knowledge activities for software development to the emerging theoretical framework knowledge activities.

Preliminary Activities		**Knowledge Framework Activities**
Stay up to date	⟶	Monitoring
Find answers	⟶	Searching
Learn	⟶	Learning
Discover others	⟶	Monitoring
Connect with others	⟶	Networking
Get and give feedback	⟶	Feedback
Publish activities	⟶	Dissemination
Watch activities	⟶	Monitoring
Display skills / accomplishments	⟶	Dissemination
Assess others		—
Coordinate with others	⟶	Co-authoring
—		Authoring
—		Information organization

Next, we needed to define each activity and articulate it in terms of the knowledge constructs—something that hasn't been done in any of the earlier existing studies nor in the other knowledge activity taxonomies. In our original study [155], we began to contemplate on the type of knowledge a channel can support, and considered four types of knowledge: (1) knowledge captured in developers' heads; (2) knowledge externalized in tools; (3) knowledge stored in community knowledge resources; and (4) knowledge captured in developer networks. As part of an interpretive conceptualization approach, we used well-established knowledge definitions (provided in Sections 6.1 and 6.3) to assign and link the different knowledge types to our knowledge building conceptual model of software development (shown in Figures 6.6 and 6.7), and to define and characterize the knowledge activities themselves (shown in Table 6.3), resulting in a theoretical framework. In doing so, we realized that software is the combined knowledge within a team, community, or organization, where social and communication media mediate knowledge transfer in the following ways:

- Between different individual knowledge types (e.g., implicit → explicit)
- Transactive memory: from the individual to a group level (individual → social knowledge types)
- Synergistic: create new shared knowledge (beyond the knowledge brought by group members)
- Encodifying knowledge and retrieving knowledge
- Facilitating or supporting awareness (e.g., who knows what)

Social and communication channels not only support but also impede developers in forming and sharing knowledge in a highly collaborative manner.

6.4.2 How Transactive Memory Manifests on Stack Overflow

In the second study [191], we empirically compared how knowledge, specifically knowledge in question-and-answer (Q&A) form, is sought, shared, and curated on both the R-help mailing list and the R tag on Stack Overflow. Additionally, we gained insights on knowledge activities members of the R community partake in (co-authoring, learning, dissemination).

Our findings indicated that there were two different approaches for constructing knowledge: *participatory knowledge construction* where members cooperate and complement each other's contributions, and *crowd knowledge construction* where members work towards the same objective but not necessarily together. A similar pattern was observed by Tausczik *et al.* [161] who examined how users on Math Overflow collaborate and construct knowledge. Our findings indicated knowledge transfer through Stack Overflow was done in a more crowdsourced manner, while knowledge transfer through the R-help mailing list was usually in a participatory manner.

By considering the definition of team knowledge from our framework (Section 6.3.3), we now understand that these two different knowledge transfer approaches relate to **transactive memory** support and **synergistic knowledge**. Stack Overflow's design and mechanisms enable transactive memory and facilitate the shared system needed for encoding, storing, and retrieving knowledge within a group. The system is designed to aid in information allocation (using tags and encouraging high quality answers), information retrieval (via search and answers from community experts), and managing of 'who knows what'. The mailing list channel does not support transactive memory by design, and members of the R community need to manage the knowledge exchanges themselves, assigning roles and setting up community rules. However, our results [191] also imply that Stack Overflow's gamification mechanisms seem to discourage collaborative knowledge creation between users, reducing synergistic knowledge creation (additional knowledge that is generated within the team beyond the potential knowledge initially held by the team's individuals).

6.5 Discussion

The theoretical framework we propose is a *descriptive framework*. It helps to describe and speculate on why a phenomena occurs and to explain its consequences. However, with additional investigations, this framework can be further extended to account for extra relations between the constructs (e.g., Actors → Roles → Assemblages) and to broaden our understanding of the existing relations (e.g., Actor → Channel). We emphasize, that our proposed theoretical framework is not yet a theory [154] , however, we believe that in the future it can be developed into a theory of knowledge in software engineering. We also recognize that our empirical grounding is based on two studies, which may imply that the generality power of the framework might not be high. Thus, further work is needed to test and extend our framework for a wider range of software development contexts and development activities. At the same time, we believe the usefulness of the framework was shown in part by applying it on our studies and revealing new insights.

Furthermore, while we chose to focus on software development, we believe that our knowledge framework and its contributions can be applicable to other knowledge work domains as well. Kelly [77] saw developers as the prototype of knowledge workers, pushing the boundaries of technology and shaping knowledge work. *"Modern knowledge work is enabled by and dependent on information technology-technologies that are created by software developers and used by legions of knowledge workers worldwide. The key difference between software knowledge workers and the others is that other knowledge workers can only use the tools that exist. If a tool doesn't exist, they can't use it. Conversely, software developers have the means to create any tool they can imagine."* By seeking to understand how knowledge is transfered and shared within software teams and communities, we can help future knowledge workers build knowledge, manage knowledge flow, and improve productivity. The relevance of our framework increases as more companies become *'knowledge-creating companies'* that create new knowledge, disseminate it effectively throughout the organization, and quickly embody it in new technologies and products [109].

6.6 Conclusions

The developed product of modern software companies and communities of practice is not only the produced code, but it is also the combined knowledge within the organization. This knowledge is indispensable and is distributed among the organizational systems, its members, routines, processes, and practices. For instance, Netflix's system *"is complex enough that there's not one person inside the organization that really understands all of it"*[1]. In today's software dominated world, industry professionals and researchers care deeply about supporting development teams to be more productive, improving existing work processes, and coping with increasing demand and scale. However, both practitioners and researchers lack the tools to *understand* the knowledge-building process of software development, or its consequences on productivity. To help with this effort, we sought to lay the theoretical ground for understanding knowledge activities and knowledge transfer in software development. In this chapter, we presented a theoretical framework grounded in empirical work, that allows to better articulate previously discovered patterns, e.g., revealing how transactive memory shapes knowledge construction on Stack Overflow. This articulation, informed by both empirical evidence and literature, helps us understand and describe the knowledge-building process better. Lastly, we believe that our conceptualization of related work and compilation of literature is also a valuable contribution.

[1]How Netflix thinks of DevOps (accessed on April 19th, 2018): `https://youtu.be/UTKIT6STSVM?t=6m16s`

Part IV

General Discussion

Chapter 7

Discussion and Insights

"He who loves practice without theory is like the sailor who boards ship without a rudder and compass and never knows where he may cast."

– Leonardo Da Vinci

So far in the thesis, I have described a theoretical framework for modeling and conceptualizing knowledge building in software development, and the theories and studies that led to the development of this framework. The previous chapters focused on describing *the framework* and *the process of building the framework*, however, there are a few additional topics that should be discussed and I cover them here.

I believe that newly proposed theories and frameworks should discuss two important aspects: *the need for a new theory*, and *ways it can be made actionable*. Thus, considering that my long-term goal (beyond my PhD) is to build upon the theoretical framework and form a theory of knowledge in software engineering, I begin this chapter by reflecting on why we need another theory. To support my arguments, I summarize existing theories used in software engineering that are relevant to the subject mater of the thesis. Then, I reflect on the insights that came from the studies described earlier in the thesis, and discuss how the proposed knowledge framework can be operationalized. Subsequently, I evaluate the framework along two main criteria: credibility and applicability. Lastly, I reflect on additional factors that have shaped my researcher bias as a result of the *additional studies I conducted or have been part of* (that are **not** part of this thesis).

7.1 Why Do We Need Another Theory?

As a maturing multidisciplinary field, software engineering relies on and uses a multitude of theories, theoretical frameworks, and models [65]. They can be used to: (1) **analyze** relationships (e.g., via taxonomies, classifications, and ontologies), (2) **influence designs**, (3) **generate post-hoc explanations** of observed phenomena, or (4) **predict** future cause-effect relationship(s) [143]. Some of these theories originate from within software engineering, while others are borrowed from other fields such as cognitive science, social psychology, and knowledge work. Earlier examination by Glass *et al.* [52] suggested that software engineering research relies little on other disciplines for its thinking, however, Hannay *et al.* [65, Table 4] showed a much higher degree of interdisciplinarity through their systematic review of theory use in software engineering. Nonetheless, there is no doubt that software engineering needs theories [75, 45].

Then the question is, within the subject matter of the thesis, do we need another theory? Indeed, there are existing theories that relate to the topic of this thesis:

- Naur's **Programming as Knowledge Building** theory [105] emphasized the importance of tacit knowledge within software teams. Polanyi's **Personal Knowledge** theory [118, 119] has also contributed to our understanding of tacit knowledge. These theories emphasize that certain knowledge, while difficult or even impossible to transfer, is essential to knowledge work. The implications of this transcend to the digital and social era and highlight that *"the larger, primarily tacit unit of context cannot be adequately represented in a computer system. Accordingly, the role of computer software should be to support human interaction and collaboration, rather than to replace or fully model human cognition"* [150].
- Lave and Wenger's theory of **Communities of Practice** [85, 86] highlighted that collective learning involves a deepening process of participation in communities of practice. Rather than considering learning as the acquisition of certain types of knowledge, Lave and Wenger place it in social engagements that provide the proper context for learning—situations of co-participation. They believe that humans are generally members of a number of such communities (either as core or peripheral members).
- Wasko's theory of **Public Knowledge** [179] (i.e., social knowledge) proposed that knowledge can be considered a public good, owned and maintained by

a community (as opposed to being owned either by the organization or by individual members). He suggested that organizations should use social media to develop communities of practice and manage knowledge as a public good.

- Reinhardt's [124, 123] **Knowledge Worker Theory** outlined a typology of knowledge workers and their respective knowledge actions (i.e., a framework of knowledge worker *roles* and *activities*). My work on developer knowledge activities was inspired and shaped by this framework.
- **Activity Theory** [43] views learning at both the individual and the community level. Vygotsky, who created activity theory, believed that the smallest possible unit for analysis of social processes should be an 'activity'. Activity theory highlights mediated cognition and *"situates the individual firmly in the activity system, which includes not only other individuals, but the mediating artifacts and the community or societal context as a contradictory whole"* [149]. Engeström updated the theory to include aspects of division of labor, the role of a community in the social process, and the existence of implicit and explicit rules (thus making it very relevant to CSCW and HCI). Activity theory aims to examine small-group interactions, but typically only reveals the larger societal issues. Rather than analyzing how a group interactionally builds knowledge, it is concerned with the group's situation in the larger organizational context and how the group deals with management issues [150].
- Stahl's **Group Cognition** theory [149, 150] highlights the goal of *"developing a post-cognitive view of cognition as the possible achievement of a small group collaborating so tightly that the process of building knowledge in the group discourse cannot be attributed to any individual or even reduced to a sequence of contributions from individual minds"* [150]. This theory emphasizes the idea of an *interactional space* for interactions within a small group, and directly relates to the concept of synergistic knowledge (Section 6.3.3).

The theories above have inspired and shaped my understanding of developer activities and of the software development process as a whole. Most of them were used as building blocks or supported the way we modeled knowledge building within software development. On top of these theories, there is also related work in the form of theses, which I consulted with throughout my research. Treude [164] examined the role of social media artifacts in collaborative software development; Aranda [4] proposed a theory of shared understanding for software organizations; Wagstrom [175] described vertical interaction in open software engineering communities; Arciniegas-Mendez [5] proposed a model of regulation (as support of collaboration) for software engineering; Mitchell [102] examined software process improvement through the identification and removal of project-level knowledge flow obstacles; Walenstein [176] proposed a distributed cognition framework to provide cognitive support in software engineering tools; and, Stam [151] examined knowledge productivity and proposed a way to design and test management methods within knowledge-intensive organizations to improve their knowledge productivity.

All these theories contribute to our understanding of knowledge transfer within software development, but none of them are capable of nor do they aim to model knowledge building within software development. Moreover, it would make little sense to compare these theories as they have different goals and scope; instead I use them to explain and support the links within our model of the software development process. Above, I described the key contribution of each theory and highlighted its limitation(s) in the context of this thesis. My thesis does not attempt to replace these theories; instead I use, expand, and build on their work to propose a knowledge building theoretical framework for software engineering.

7.2 How Can This Framework be Operationalized?

The role of a theory or a theoretical framework is to provide *"a logically interrelated set of propositions about empirical reality"* [135] (i.e., help explain or predict how the world works). However, when considering usefulness, I align with the philosophical school of pragmatism which asserts that in order to be useful, a theory should also be linked to practice. That link between theory and practice is often nontrivial. There can be several ways to operationalize (make actionable) theories and frameworks. Bednar *et al.* [11] suggest that *"the primary strategy for providing this 'link' between theory and practice has been to collect concepts and strategies suggested by the theories and make them available to the practitioners. The concepts and strategies are abstracted out of their theoretical framework, placed within a practitioner's framework, and grouped based on their relevance to a particular...task"*.

Next (in Section 7.3), I demonstrate one way in which the insights from the proposed knowledge framework can be made available to practitioners. By synthesizing the challenges developers face when using social media (that we uncovered in Chapter 4), and by using the knowledge framework and its knowledge activities (Chapter 6), I provide practitioners with a practical heuristic questionnaire to expose knowledge sharing challenges when choosing or designing communication media.

7.3 Exposing Knowledge Sharing Challenges in Communication Media: A Practical Heuristic

As indicated by the results in Section 4.3, on average modern developers find themselves using at least 12 different communication channels across different activities. While the need for multiple channels makes sense, as specific channels support individual activities differently, the ecosystem of socially-enabled channels and tools developers use is rife with challenges [155, 193]: Developers struggle to handle interruptions, and to effectively collaborate with others. Organizations and communities experience coordination, communication, participation, and social challenges. Furthermore, the use of many communication channels leads to knowledge fragmentation, overwhelmingly large quantities of information, and difficulties in evaluating the quality of information (see Fig. 4.9). In the meantime, new channels emerge and are adopted by developer teams (often adding to the number of channels used rather than replacing existing ones). It is important to note that while sage advice on how to use a single channel or improve its use is increasingly more prevalent, there is little advice on how to use or contemplate on the use of a constellation of communication media.

This leads to important questions. How does one *choose* the channels to use? How does one *evaluate* a channel or *compare* between multiple channels? How can a team or a community outline and *reflect* on the challenges involved with their choice of communication media? How does one *know when to retire a channel or replace* an existing channel with a different one?

To help organizations and development teams answer these questions, we describe an inspection method in the form of a *heuristic analysis* for revealing and mapping knowledge sharing challenges when using social media and communication channels. This section describes an actionable instrument for practitioners, that employs a practical method not guaranteed to be optimal or perfect, but sufficient for the specific goals. We kindly refer the interested reader in the research aspects to Chapters 4-5 and related work [193, 1, 142] that provide a descriptive analysis of the challenges.

7.3.1 Mapping The Challenges

Based on our previous research of communication channels and social media use in software development (described in Chapters 4-5) and related work [193, 183, 1, 142, 3], we synthesized insights on the knowledge sharing challenges developers face, and formed a practical heuristic analysis instrument for developer teams and communities. This heuristic is given in the form of a set of guiding questions (presented in Table 7.1), meant to be used by individuals or groups aiming to expose and map potential knowledge sharing challenges (similarly to the way a cognitive walkthrough method is used to expose usability issues). These questions should be applied per communication channel, and can be used to reflect on a combination of media. If possible, we recommend to use them as guiding questions in a focus group setting, discussing and prioritizing the challenges within a team or across the organization. Note, these questions are not designed to be used as a scorecard, but rather as a reflective and guiding tool.

We list 25 questions, under eight categories. (1) **Knowledge management** questions help reveal challenges about the cost of knowledge sharing, challenges in managing the quantity and quality of knowledge, and challenges of finding who knows what. (2) **Knowledge representation** questions deal with the channel's ability to properly represent knowledge. (3) **Team and community organizational structure** questions help reflect on challenges that may shape or be shaped by the organizational structure. (4) **Team cognition** questions prompt to consider the channel's support of knowledge transfer within the team. (5) **Collaboration and awareness** questions help evaluate the channel's ability to foster collaboration and awareness. (6) **Social attributes** is a category of questions to assess the channel's ability to support a healthy, reliable, and sustainable network. (7) **Security** questions prompt to consider the channel's mechanisms for protecting stored knowledge. Lastly, (8) the **literacy** category helps reflect about the expected tool literacy and proficiency for using a channel.

We believe this can be useful for mapping the knowledge sharing challenges involved when adopting new social and communication channels, for assessing the organizational communication flow, and by tool builders when designing new communication tools (such as Slack).

Table 7.1: Heuristic analysis technique to help practitioners reflect on the challenges in communication media use.

Knowledge Management

y / n Does the channel **introduce a cost to sharing or searching** knowledge? (e.g., as a result of distances, priority, cost of codifying knowledge, or lack of motivation to share knowledge).

y / n Does the channel **assist in dealing with knowledge loss**? (e.g., employee turn-over, fragmentation)

y / n Does the channel help in assessing the degree of **information accuracy**, **information relevance** (e.g., obsolete knowledge), and **information credibility** (e.g., dealing with spam)?

y / n Does the channel provide mechanisms to **manage the amount of incoming knowledge**? (e.g., dealing with information overload, finding the signal in the noise)

y / n Does the channel provide mechanisms to easily **locate knowledge**?

y / n Is it difficult to locate the **source of knowledge** on the channel? Is it difficult to **find domain experts** over the channel?

y / n Will this channel be used as a **central knowledge repository**?

y / n Does the channel provide mechanisms for **moderating knowledge** (e.g., up-voting, down-voting, editing, flagging)?

Knowledge Representation

y / n Does the channel **present knowledge in a consistent way**?

y / n Does the channel **support the required type(s) of knowledge** (i.e., capable of presenting it, allowing to edit and collaborate on it)?

Team / Community Organizational Structure

y / n Does the channel encourage or **dictate a specific communication structure** (centralized vs decentralized, synchronous vs asynchronous, private vs public, ephemeral vs archival, anonymous vs identified)? Does this communication structure match the organizational structure? (i.e., does it create a shadow organizational structure?)

y / n Does the channel encourage **sub-optimal knowledge flows** over the channel? (e.g., forcing specific communication structures)

y / n Is there a clear **ownership** over the generated knowledge? (e.g., a shared document may indicate who contributed which lines, etc.)

Team Cognition

y / n Does the channel help in **articulating tacit knowledge**?

y / n Does the channel allow users to **apply and mobilize knowledge in another context**?

y / n Does the channel help in **bridging gaps in education and technical knowledge** between the communicating members?

y / n Does the channel **promote or foster mentorship** (formal or informal) between the members?

Collaboration and Awareness

y / n Does the channel help in **maintaining group awareness** of newly shared knowledge and group members' activities?

y / n Does the channel introduce **distractions and interruptions**? Does the channel allow to **signal when you're too busy** for interruptions?

y / n Does the channel assist in **overcoming communication barriers** (e.g., due time zones, geographical distance, language barriers)

Social Attributes

y / n Does the channel **help to establish trust and rapport** among the members?

y / n Does the channel **provide mechanisms to deal with poor attitude and intimidation**? (e.g., fear of losing job, fear of criticism)

Security

y / n Does the channel **support security mechanisms for stored knowledge**?

y / n Does the channel allow to **distinguish between secure and non-secure knowledge transfers**?

Literacy

y / n Does the channel **require a high level of proficiency or technical expertise** to fully use the channel?

7.3.2 A Heuristic Analysis of Slack: Understanding How Slack Supports Knowledge Sharing

To illustrate how the heuristic analysis technique can be used, we briefly reflect on our experience with Slack, however, we emphasize that this method is intended to be applied to all the team's communication media.

Slack (stands for 'Searchable Log of All Conversation and Knowledge') is an IRC-like workplace messaging tool that has seen a rapid adoption by developer teams and other knowledge workers. Similar to other research labs [116], we use Slack as our main communication medium, after it had replaced Email, to accommodate a team of local and remote researchers, developers, and collaborators (16-20 weekly active users). Our conversation history includes 110k private and public messages to date, across 14 internal 'channels' (e.g., *weekly-updates*, *water-cooler*).

Slack allows our research group to overcome **collaboration and awareness** challenges, as it allows members to subscribe and unsubscribe to specific channels, and includes a notification mechanism for when one's attention is required. Additionally, Slack provides mechanisms to deal with interruptions, giving users the ability to signal when they are busy, and control the notification sensitivity. Slack also confirms with users before they mass-broadcast or contact members in different time zones—reducing unnecessary interruptions.

We also found Slack to be supportive of **team cognition** and effective for sharing tacit knowledge in the team (e.g., it allows group members to watch how others use it for devops).

At the same time, we identified significant challenges relating to **knowledge management**. Slack doesn't provide effective mechanisms to moderate and curate knowledge: there is no way to indicate or locate high quality content, to edit or build upon stored knowledge, or to locate domain experts. In our group, we mitigate these challenges by complementing Slack with additional channels such as GitHub. However, in larger communities (e.g., our city's tech community of professionals, designers, developers, and students has 1,796 active members on Slack) these challenges are amplified. Slack's lack of support for managing large amounts of incoming information, and its realtime-messaging design exacerbate and add to the knowledge management challenges.

Slack is capable of displaying many types of documents, but we found it lacking in terms of **knowledge representation**. Specifically, it does not support co-authoring of documents (e.g., collaboratively editing code snippets). Our group mitigates this with the use of integrations (Google Docs, GitHub).

Lastly, we found Slack to be accessible with a low barrier of entry, however **tool literacy** can still be an issue. Surprisingly, we saw people that found it harder to use than expected.

Notably, our intention in this reflection is to focus less on Slack itself, and more on how the heuristic technique helped us reflect on what a channel such as Slack brings to or removes from our group's communication. It helped us surface and spark a discussion on challenges we didn't realize we had or have implicitly solved.

7.3.3 "A Problem Well Put is Half Solved"

The heuristic analysis instrument described in this section isn't meant to prescribe which communication tools to adopt nor to rank them. Instead, this method should force communities and organizations to reflect on the challenges inherent in their use of social and communication media, and inform future or surface current challenges. By itself this instrument will not solve the challenges, but knowing the problem is often half the solution and awareness of the challenges can ensure they are addressed sooner. Organizations will be able to take mitigating steps, e.g., provide training to address literacy challenges, or raise awareness on ownership issues when the channel is first adopted or the workflow is determined. It is also important to emphasize that the severity of the challenges teams and organizations may face will depend on the organizational routines, processes, practices, and norms.

7.4 Evaluating the Theoretical Framework

While quantitative research uses **validity** and **reliability** to describe the quality of research (accuracy, consistency and equivalence), these criteria are not recommended for qualitative research [189]. It is because quantitative and qualitative types of research have different historic origins, and consequently, different interpretations of the words *meaning* and *truth*. Where researchers of one approach are *removed from the context*, researchers of the other are *embedded in the context and social situations*.

Instead, for evaluating the quality of my framework, I will use the terms **credibility** and **applicability**. Credibility refers to the trustworthiness of the findings and whether they reliably reflect a person's (e.g., researcher, participant, reader) experiences with the phenomenon and its interpretation [94, 24]. Credibility is increased when researchers (1) report on the data collection and analysis processes, (2) use triangulation, (3) acknowledge and reveal their researcher bias (reflexivity), (4) report and link to evidence, (5) understand and spend time in the domain, (6) consult with experts and stakeholders, and (7) reveal and consider variations and discrepant information. Applicability refers to the depth of the phenomenon, usefulness, fit, level of conceptualization, and confirmability of the findings.

Please note that while credibility is a commonly accepted term, I was not able to find an agreed upon term to represent usefulness [136], thus I chose to use applicability[1].

Credibility

The proposed theoretical framework was created by applying a bottom-up approach consisting of a gradual and iterative analysis, and an interpretive synthesis of two

[1]Different schools of thought and research paradigms have their own view on the criteria terminology for assessing quality of qualitative research.

empirical studies. The framework emerged from these studies and is grounded in them. In these studies, we extensively reported on the data collection and data analysis processes, and we linked to the generated evidence (e.g., codes, memos, and other traces of the analysis)[2]. Our observations were contextualized, taking into account the studied social situations and domain. In Chapter 6, I further describe the links between the empirical studies and the framework, and how these studies shaped the framework.

An important part of credibility stems from the researchers themselves and the trustworthiness of their interpretations. I include descriptions of these aspects in the dissertation as well. Section 3.5 describes how I positioned myself as a researcher in my studies, both an *insider* and an *outsider*, a combination that helps understand the impact of socio-technical systems on existing processes and behaviors. Creswell [24] stressed that understanding of the domain requires spending extensive time in the field. Over the period of my PhD, I have familiarized myself with the domain and gained a deep understanding of the domain under study, with specific focus on knowledge building, social media, and collaborative work within software engineering. I am well familiar with the landscape of tools and the processes developers use. At the same time, I also acknowledge, identify, and describe my researcher bias and how it affected my interpretations.

However, researchers are not expected to employ all the procedures that maximize credibility within a single study or body of work. For instance, Creswell [24] recommended employing at least two of the eight procedures he identified. In the case of the work described in this thesis, the following were not used: triangulation (from different sources of data, methods, or observers), consulting with domain experts and stakeholders, and negative case analysis.

Applicability

Beyond credibility, it is important to also consider the usefulness of the framework. Sections 7.2 and 7.3 described and demonstrated how the framework can be made actionable. Additionally, it should be noted that the proposed framework is not a theory yet [154]. However, with additional investigations, this framework can be further extended to broaden the existing relations and account for extra relations

[2]We shared the collected data used in the studies, excluding any identifiable information about the participants.

between the constructs (e.g., Actors $\rightarrow$ Roles $\rightarrow$ Assemblages), becoming a theory in the future. The usefulness is also determined by the conceptualization level of the framework [143]: little conceptualization (*"minor working relationships that are concrete and based directly on observations"*), medium conceptualization (*"theories of the middle range that involve some abstraction but are still closely linked to observations"*), and high conceptualization (*"all-embracing theories that seek to explain social behavior"*). I would categorize the theoretical framework as a medium-level conceptualization.

So far, I have discussed the position of the theoretical framework in relation to existing theories in software engineering, demonstrated how it can be made actionable, and evaluated it based on credibility and applicability. Next, I reflect on factors that may have shaped my interpretation and understanding of the knowledge building process.

7.5 My Work on Other Studies and How It Shaped My Researcher Bias and Interpretations

When Creswell [25, p. 8] described social constructivists, he said they *"recognize that their own backgrounds shape their interpretation, and they position themselves in the research to acknowledge how their interpretation flows from their personal, cultural, and historical experiences"*. This also includes the interpretations made and the insights gained by researchers from **other studies they previously conducted**, even if these studies may not have been directly related. For this reason, I reflect on how the additional studies I've conducted may have shaped my interpretation.

Throughout my PhD, I've been involved in additional studies which are **not** part of this dissertation. These studies are broadly (though not directly) related to the subject matter of this thesis. Hence, it is important to note that my experience with and insights from these studies have also shaped my bias in the work described in this thesis.

How GitHub Shapes the Workflow and Fosters Collaboration

In my studies of GitHub [190, 44], I examined how GitHub is used as a collaborative platform in an educational context. My goal was to understand how platforms such as GitHub—that provide social and collaborative features in conjunction with distributed version control—may improve (or possibly hinder) the educational experience for students and teachers. We conducted a qualitative study focusing on how GitHub is being used in university classes, first focusing on the educator's perspective [190] and then on the student's perspective [44].

We found that GitHub provided significant benefits compared to traditional learning management systems. It provided a shared space, i.e., a shared repository for group memory, for students and educators (collaborating over course notes, references, and other resources and material) and helped manage course versioning (giving access to previous iterations of a course). Moreover, GitHub fostered the reuse and sharing of course materials and knowledge (across semesters, among different instructors, and between universities). It enabled transparency of student activities (with mechanisms such as the *graph view* and the *news feed*), that promoted awareness and supported regulation and collaboration—helping transform passive observation into active participation. These benefits have **shaped the workflow** of students and educators as a whole. For example, educators were actively helping students that got stuck or were about to go 'off track'.

Interestingly, there are many parallels between these findings and our observations in the social media studies. In retrospect, I believe that these studies of GitHub in an educational context have shown me how social media platforms can shape workflows and knowledge flow, and introduced me to the role and impact of communities of practice.

Using Regulation to Understand Collaboration Practices and Tool Support Within Software Development

I worked with Maryi Arciniegas-Mendez [5] to seek an understanding of and explore how individuals and groups regulate each other within software teams and communities. *Regulation* is the combination of mindful processes developers engage in to determine what tasks they need to complete and who should be involved, what their goals are relative to those tasks, how they should meet their goals, what domain knowledge needs to be manipulated, and why they use a particular approach or tool.

As a result of this research work [6, 7], we formed a **Model of Regulation** to capture how individuals self-regulate their tasks, knowledge and motivation, how they regulate one another, and how they achieve a shared understanding of project goals and tasks within software teams and organizations. To do this, we borrowed

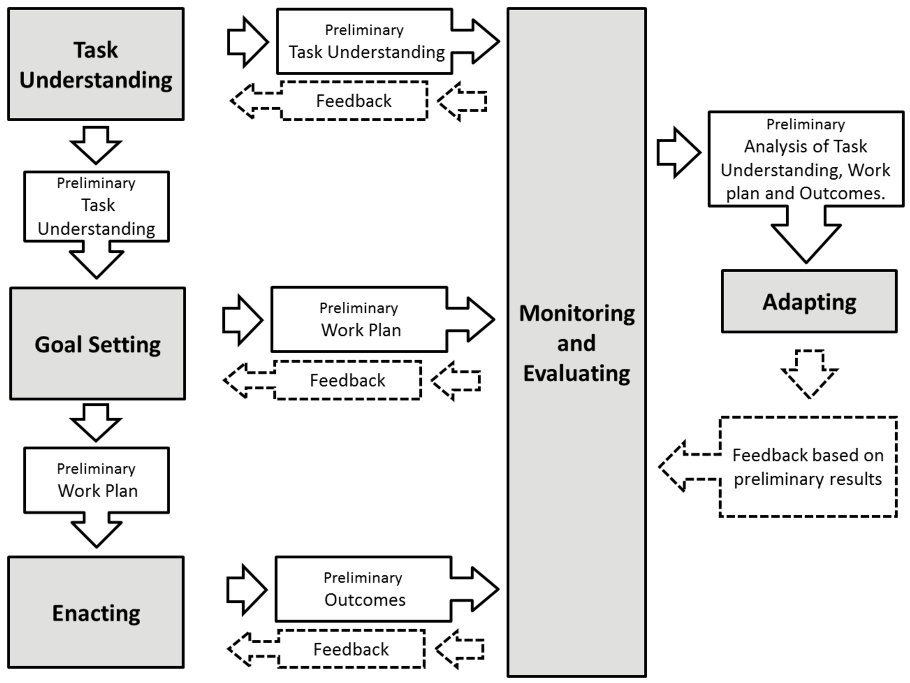

Task Understanding
Preliminary Task Understanding
Feedback
Preliminary Task Understanding
Goal Setting
Preliminary Work Plan
Feedback
Preliminary Work Plan
Enacting
Preliminary Outcomes
Feedback
Monitoring and Evaluating
Preliminary Analysis of Task Understanding, Work plan and Outcomes.
Adapting
Feedback based on preliminary results

Figure 7.1: Activation of the regulation processes. Solid arrows represent outcomes of the process, while dashed arrows indicate feedback going into the process. (Source: the model of regulation by Arciniegas-Mendez *et al.* [7])

and adapted regulation constructs from the learning sciences and computer supported collaborative learning literature (CSCL) [63, 62, 101] and applied them to the software engineering domain.

The model of regulation [7] included three modes of regulation: self-regulation, co-regulation, and shared regulation. **Self-regulation** refers to an individual's processes with respect to a task. **Co-regulation** refers to the recognition of each other's perspectives and the alignment of ideas regarding the tasks to be completed. However, co-regulation is afforded and constrained by the complete social context, including people, tools, technologies, and practices. **Shared regulation** involves joint control of a task through shared (negotiated), iterative fine-tuning of cognitive, behavioral, motivational, and emotional conditions/states as needed. The model also includes processes related to the strategic decisions required to perform a task. In particular, regulation in software engineering consists of five interconnected processes (see Fig 7.1).

This study exposed me to important dimensions of collaboration (behavior, cognition, and motivation) and granted me insights about sub-processes that govern activities (e.g., Task Understanding, Enacting) at different levels (self-, co-, shared). It further helped me familiarize myself with other existing collaboration and CSCW models (e.g., 3C model, MoCA). In hindsight, I believe this study encouraged me to not only focus on understanding the tools and channels developers use, but to explore the theoretical underpinnings of why certain tools and practices are used, and to consider collaboration at a metacognitive level.

How Bots are Disrupting and Augmenting Software Development

Together with Carlene Lebeuf and Margaret-Anne Storey, we examined the role of Bots in software development [157, 87, 88]. In general, Bots are seen as applications that automate repetitive or predefined tasks. In software development, they are used to help developers make smarter decisions and to support developers that need to communicate and coordinate with others[3]. The micro-services that Bots provide are not new, but the way they are presented to developers, through a conversational UI embedded in developer chat channels, is changing how tools are integrated in the developer's tool suite.

In their basic form, Bots serve as a conduit or an interface between *users* and *services*, typically through a conversational user interface (UI)[4], and are further enhanced by adding personalization and a memory. They can be designed to operate in *pull* mode, where the user initiates the interaction, *push* mode, where the Bot initiates the interaction, or a combination of both. Bots are most commonly used for automating tasks (e.g., running tests when certain conditions are met) or for gluing tools together. Bots may leverage AI or machine learning techniques, or they may capture or analyze data generated by other tools and Bots.

First, we outlined the modern Bot landscape and proposed a preliminary cognitive support framework that can be used to understand these roles and to reflect on the impact of Bots on productivity in software development [157]. Then, we explored how chatbots can help reduce the friction points software developers face when working collaboratively [87] (e.g., understanding and working towards team goals, adhering to team procedures and agreements, and coordinating team activities). We believe that chatbots have immense potential for supporting developers collaboration needs.

"The real potential of bots isn't going to be realized with one person using

[3]https://svenpet.com/talks/rise-of-the-machines-automate-your-development/
[4]http://www.wired.com/2015/06/future-ui-design-old-school-text-messages/

> *one bot (that's the old app model), but with multiple people having a normal conversation while the bots augment the stream with relevant context and functionality. This is a pretty far-reaching evolution of how humans interact with technology. It's cognitively ergonomic."*
>
> (Source: Phil Libin[5])

These studies allowed me to reflect on the role of Bots in the knowledge sharing process and how they would fit into my knowledge framework. The design and functionality of commonly used chatbots (and the way we defined Bots so far) would put them under the *'channel'* or *'tool'* category of the knowledge framework (as shown in Fig. 6.6). However, Bots are extremely versatile in terms of the types of knowledge they can transfer, e.g., Bots can elicit individual knowledge and help share and transform it into group knowledge, or they can accumulate tacit knowledge over time. In fact, I believe that future Bots could perhaps even fall into the *'actor'* category of the knowledge framework (i.e., be considered as knowledge workers), but I haven't seen a convincing example of that yet[6]. I believe that these insights have implicitly helped me refine the proposed knowledge framework.

[5]https://medium.com/@plibin/two-new-bot-investments-from-general-catalyst-dc5f1d61cab6
[6]http://fortune.com/2016/06/24/silicon-valley-last-job/

Chapter 8

Conclusions and Future Work

"I may not have gone where I intended to go, but I think I have ended up where I needed to be."

– Douglas Adams, 1988

This thesis described a theoretical framework for modeling the knowledge building process and understanding how knowledge transfer is mediated by the social and communication media used in software development. In this chapter, I conclude the thesis by reflecting on future work, discussing avenues for further research, and pointing out the research contributions I made along the way.

8.1 Contributions

The research work described in this dissertation makes the following overarching contributions:

Empirical studies of social and communication media use in software development communities.

In this thesis, I described the empirical studies I've conducted on the use of social and communication media in software developer communities. These studies have revealed valuable insights about the impact of communication channel use on developers and the software development process, and how different channels support different knowledge sharing activities. The findings of these studies helped us form the basis for a knowledge building framework in software engineering.

A theoretical framework of knowledge building in software development.

A key contribution of this thesis is the emerging theoretical framework. The knowledge framework aims to provide researchers with the 'theoretical mechanisms' needed to understand and articulate knowledge transfer within software development processes and organizations—thus, leading to a better understanding of software development itself. This framework is grounded in our empirical work and has been demonstrated to extend our findings.

A heuristic analysis instrument for practitioners.

To help organizations and development teams choose and design their communication infrastructure, I described an inspection method in the form of a heuristic analysis for revealing and mapping knowledge sharing challenges when using social media and communication channels. This heuristic is given in the form of 25 questions that are designed to be used as a reflective and guiding tool.

8.2 Future Work

The resulting theoretical framework is supported by multidisciplinary theoretical and empirical background and by our empirical findings. The framework is novel and useful, however, it is not extensive enough as currently stated to predict outcomes or prescribe tool and process designs. This leaves many opportunities for future research: I believe the following research directions are of the highest **urgency**.

Reinforcing the credibility of the framework

The findings described in this thesis come from studies of two specific communities, developers active on GitHub, and the R community. However, the software industry as a whole is both large, and diverse. I plan to evaluate and extend the framework further by investigating additional developer communities. In particular, it would be helpful to use the proposed framework within an industrial context (i.e., through a study within a company), because most of our findings are based on non-industrial communities (even though many of the participants are professional developers). Additionally, as mentioned in Chapter 7, replicating our findings or triangulating the findings with other data sources, methods, or research will be beneficial.

Extending the framework

While I believe the framework includes all the high-level constructs (actors, roles, assemblages & communities, channels, tools, artifacts, processes & practices, and activities), further research is required to confirm the existing relations and find new relations between these constructs. The next logical choice is to investigate the roles that developers take and how these roles are linked to their activities. This can be similar to the work by Reinhardt [124, 123], who examined the roles and activities among knowledge workers, or build on the work by Ford *et al.* [46], who identified developer personas based on tasks, collaboration styles, and perspectives of autonomy.

8.3 Conclusions

In this dissertation, I presented two empirical studies of social and communication media use in software development communities: (1) the study on how social and communication media affect and disrupt software development, and (2) the study about knowledge curation within the R community. These studies have revealed valuable insights about the impact of communication channel use on developers and the software development process, and how different channels support different knowledge sharing activities. Through an interpretive synthesis of empirical data from the two studies, I have formed a knowledge framework. This framework builds on existing concepts and models of knowledge work and CSCW, and is grounded in the empirical work presented in the thesis. The framework allows us better articulate previously discovered patterns, e.g., revealing how transactive memory shapes knowledge construction on Stack Overflow, and helps us better understand and describe the knowledge-building process in software engineering. Overall, the framework allowed us to move a step closer to understanding how modern software is built. As part of this thesis, I also provided a rich and contextualized background and conceptualization of social media, CSCW, cognition, and knowledge work. Furthermore, I provided an extensive discussion on the added value of the proposed framework and how it may be made actionable in the future.

When I started my PhD, I did not aim or plan to build a theoretical framework of knowledge (nor a theory of knowledge), but I think this is what I needed to do.

Bibliography

[1] Nitin Agarwal and Yusuf Yiliyasi. Information quality challenges in social media. In *International Conference on Information Quality (ICIQ)*, pages 234–248, 2010.

[2] Mats Alvesson. Organizations as rhetoric: Knowledge-intensive firms and the struggle with ambiguity. *Journal of Management studies*, 30(6):997–1015, 1993.

[3] Maurício Aniche, Christoph Treude, Igor Steinmacher, Igor Wiese, Gustavo Pinto, Margaret-Anne Storey, and Marco Aurélio Gerosa. How modern news aggregators help development communities shape and share knowledge. In *Proceedings of the 40th International Conference on Software Engineering*, 2018.

[4] Jorge Aranda. *A theory of shared understanding for software organizations.* University of Toronto, 2010.

[5] Maryi Arciniegas-Mendez. *Regulation in Software Engineering.* PhD thesis, 2016.

[6] Maryi Arciniegas-Mendez, Alexey Zagalsky, Margaret-Anne Storey, and Allyson F Hadwin. Regulation as an enabler for collaborative software development. In *Proceedings of the Eighth International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 97–100. IEEE Press, 2015.

[7] Maryi Arciniegas-Mendez, Alexey Zagalsky, Margaret-Anne Storey, and Allyson F Hadwin. Using the model of regulation to understand software development collaboration practices and tool support. In *Proceedings of the 20th ACM Conference on Computer Supported Cooperative Work & Social Computing.* ACM, 2017.

[8] Linda Argote. *Organizational learning: Creating, retaining and transferring knowledge.* Springer Science & Business Media, 2012.

[9] Lynda Baker. Observation: A complex research method. *Library trends*, 55(1):171–189, 2006.

[10] Ohad Barzilay, Christoph Treude, and Alexey Zagalsky. *Facilitating Crowd Sourced Software Engineering via Stack Overflow*, pages 289–308. Springer New York, New York, NY, 2013.

[11] Anne K Bednar, Donald Cunningham, Thomas M Duffy, and J David Perry. Theory into practice: How do we link. *Constructivism and the technology of instruction: A conversation*, pages 17–34, 1992.

[12] Lisa GA Beesley and Chris Cooper. Defining knowledge management (km) activities: towards consensus. *Journal of knowledge management*, 12(3):48–62, 2008.

[13] Nicolas Bettenburg, Emad Shihab, and Ahmed E. Hassan. An empirical study on the risks of using off-the-shelf techniques for processing mailing list data. In *ICSM'09: Proc. of the 25th Intl. Conf. on Software Maintenance*, pages 539–542, 2009.

[14] Frank Blackler. Knowledge, knowledge work and organizations. *The strategic management of intellectual capital and organizational knowledge*, pages 47–62, 2002.

[15] Amiangshu Bosu, Christopher S. Corley, Dustin Heaton, Debarshi Chatterji, Jeffrey C. Carver, and Nicholas A. Kraft. Building reputation in stackoverflow: An empirical investigation. In *Proc. of the 10th Intl. Conf. on Mining Software Repositories*, MSR '13, pages 89–92, 2013.

[16] Gargi Bougie, Jamie Starke, Margaret-Anne Storey, and Daniel 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.

[17] Glenn A. Bowen. Naturalistic inquiry and the saturation concept: a research note. *Qualitative Research*, 8(1):137–152, 2008.

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

[19] Michael Chui, James Manyika, Jacques Bughin, Richard Dobbs, Charles Roxburgh, Hugo Sarrazin, Geoffrey Sands, and Magdalena 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.

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

[21] Harry M Collins. The structure of knowledge. *Social research*, pages 95–116, 1993.

[22] Nancy J Cooke, Eduardo Salas, Janis A Cannon-Bowers, and Renee J Stout. Measuring team knowledge. *Human Factors*, 42(1):151–173, 2000.

[23] Denzil Correa and Ashish Sureka. Chaff from the wheat: Characterization and modeling of deleted questions on stack overflow. In *Proc. of the 23rd Intl. Conf. on World Wide Web*, WWW '14, pages 631–642, 2014.

[24] J. W. Creswell. *Qualitative Inquiry and Research Design: Choosing Among Five Approaches*. SAGE, 2012.

[25] J.W. Creswell. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. SAGE Publications, 2009.

[26] Michael Crotty. *The foundations of social research: Meaning and perspective in the research process*. Sage, 1998.

[27] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in github: Transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 1277–1286, New York, NY, USA, 2012. ACM.

[28] Thomas H Davenport and Laurence Prusak. *Working knowledge: How organizations manage what they know*. Harvard Business Press, 1998.

[29] Gordon B Davis. Affordances of ubiquitous computing and productivity in knowledge work. In *Workshop on Ubiquitous Computing Environments, Cleveland, OH*, 2003.

[30] Tom DeMarco and Tim Lister. *Peopleware: Productive Projects and Teams.* Addison-Wesley, 1987.

[31] Alan R Dennis and Susan T Kinney. Testing media richness theory in the new media: The effects of cues, feedback, and task equivocality. *Information systems research*, 9(3):256–274, 1998.

[32] Sebastian Deterding, Miguel Sicart, Lennart Nacke, Kenton O'Hara, and Dan Dixon. Gamification. using game-design elements in non-gaming contexts. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '11, pages 2425–2428, 2011.

[33] Paul Dourish and Victoria Bellotti. Awareness and coordination in shared workspaces. In *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, pages 107–114. ACM, 1992.

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

[35] Peter F Drucker. The rise of the knowledge society. *Wilson Quarterly*, 17(2):52–71, 1993.

[36] Peter F Drucker. *Landmarks of tomorrow: A report on the new.* Transaction Publishers, 2011.

[37] Peter Ferdinand Drucker. *Post-capitalist society.* Routledge, 1994.

[38] Peter Ferdinand Drucker. *Management challenges for the 21st century.* Routledge, 2007.

[39] Robert Dubin. Theory building (rev. ed.), 1978.

[40] Lilia Efimova. Discovering the iceberg of knowledge work: a weblog case. In *Proceedings of the Fifth European Conference on Organisational Knowledge, Learning and Capabilities (OKLC 2004*, pages 2–3, 2004.

[41] Clarence A. Ellis, Simon J. Gibbs, and Gail Rein. Groupware: Some issues and experiences. *Commun. ACM*, 34(1):39–58, January 1991.

[42] Mica R Endsley. Toward a theory of situation awareness in dynamic systems. *Human factors*, 37(1):32–64, 1995.

[43] Yrjö Engeström. Expansive learning at work: Toward an activity theoretical reconceptualization. *Journal of Education and Work*, 14(1):133–156, 2001.

[44] Joseph Feliciano, Margaret-Anne Storey, and Alexey Zagalsky. Student experiences using github in software engineering courses: A case study. In *Proceedings of the 38th International Conference on Software Engineering Companion*, ICSE '16, pages 422–431, New York, NY, USA, 2016. ACM.

[45] Daniel Méndez Fernández and Jan-Hendrik Passoth. Empirical software engineering: From discipline to interdiscipline. *arXiv preprint arXiv:1805.08302*, 2018.

[46] Denae Ford, Thomas Zimmermann, Christian Bird, and Nachiappan Nagappan. Characterizing software engineering work with personas based on knowledge worker actions. 2017.

[47] Jason Fried and David Heinemeier Hansson. *Remote: Office Not Required*. Ebury Digital, 2013.

[48] John Kenneth Galbraith. The new industrial state. *Antitrust L. & Econ. Rev.*, 1:11, 1967.

[49] Clifford Geertz. *The interpretation of cultures*, volume 5043. Basic books, 1973.

[50] D.M. German, B. Adams, and A.E. Hassan. The evolution of the r software ecosystem. In *Software Maintenance and Reengineering (CSMR), 2013 17th European Conf. on*, pages 243–252, March 2013.

[51] Marco Aurélio Gerosa. Analysis and design of awareness elements in collaborative digital environments: A case study in the aulanet learning environment. *Journal of Interactive Learning Research*, 14(3):315–332, 2003.

[52] Robert L Glass, Venkataraman Ramesh, and Iris Vessey. An analysis of research in computing disciplines. *Communications of the ACM*, 47(6):89–94, 2004.

[53] C. Gomez, B. Cleary, and L. Singer. A study of innovation diffusion through link sharing on stack overflow. In *Proc. of the 10th Intl. Conf. on Mining Software Repositories*, pages 81–84, May 2013.

[54] Robert M Grant. Nonaka's 'dynamic theory of knowledge creation'(1994): Reflections and an exploration of the 'ontological dimension'. In *Towards Organizational Knowledge*, pages 77–95. Springer, 2013.

[55] Eve Gregory and Mahera Ruby. The insider/outsiderdilemma of ethnography: Working with young children and their families in cross-cultural contexts. *Journal of Early Childhood Research*, 9(2):162–174, 2011.

[56] Terri L Griffith, John E Sawyer, and Margaret A Neale. Virtualness and knowledge in teams: Managing the love triangle of organizations, individuals, and information technology. *MIS quarterly*, pages 265–287, 2003.

[57] Egon G Guba. *The paradigm dialog.* Sage publications, 1990.

[58] Egon G Guba, Yvonna S Lincoln, et al. Competing paradigms in qualitative research. *Handbook of qualitative research*, 2(163-194):105, 1994.

[59] Carl Gutwin, Reagan Penner, and Kevin 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.

[60] Anja Guzzi, Alberto Bacchelli, Michele Lanza, Martin Pinzger, and Arie van Deursen. Communication in open source software development mailing lists. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, pages 277–286, Piscataway, NJ, USA, 2013. IEEE Press.

[61] Thomas Hädrich. Situation-oriented provision of knowledge services. *Doctoral dissertation*, 2008.

[62] Allyson Fiona Hadwin, Sanna Järvelä, and Mariel Miller. Self-regulated, co-regulated, and socially shared regulation of learning. *Handbook of self-regulation of learning and performance*, 30:65–84, 2011.

[63] Allyson Fiona Hadwin, Mika Oshige, Carmen LZ Gress, and Philip H Winne. Innovative ways for using gstudy to orchestrate and research social aspects of self-regulated learning. *Computers in Human Behavior*, 26(5):794–805, 2010.

[64] Mark Handel and James 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.

[65] Jo E Hannay, Dag IK Sjoberg, and Tore Dyba. A systematic review of theory use in software engineering experiments. *IEEE transactions on Software Engineering*, 33(2):87–107, 2007.

[66] Thomas N Headland, Kenneth L Pike, and Marvin Ed Harris. Emics and etics: The insider/outsider debate. In *This book had its genesis at a symposium of the 87th Annual Meeting of the American Anthropological Association in Phoenix, Arizona, on Nov 19, 1988.* Sage Publications, Inc, 1990.

[67] Nicholas L. Henry. Knowledge management: A new concern for public administration. *Public Administration Review*, 34(3):189–196, 1974.

[68] James D. Herbsleb, David L. Atkins, David G. Boyer, Mark Handel, and Thomas 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.

[69] Richard C Hicks, Ronald Dattero, and Stuart D Galup. The five-tier knowledge management hierarchy. *Journal of Knowledge management*, 10(1):19–31, 2006.

[70] Clyde W Holsapple and Kiku Jones. Exploring primary activities of the knowledge chain. *Knowledge and Process Management*, 11(3):155–174, 2004.

[71] George P Huber. Organizational learning: The contributing processes and the literatures. *Organization science*, 2(1):88–115, 1991.

[72] Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.

[73] Ronald L Jacobs. Knowledge work and human resource development. *Human Resource Development Review*, 16(2):176–202, 2017.

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

[75] Pontus Johnson, Mathias Ekstedt, and Ivar Jacobson. Where's the theory for software engineering? *IEEE software*, 29(5):96–96, 2012.

[76] Abraham Kaplan. *The conduct of inquiry: Methodology for behavioural science.* Routledge, 1998.

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

[78] Michael Koch. Cscw and enterprise 2.0-towards an integrated perspective. *BLED 2008 Proceedings*, page 15, 2008.

[79] Kari Kuutti. Activity theory as a potential framework for human-computer interaction research. *Context and consciousness: Activity theory and human-computer interaction*, 1744, 1996.

[80] Sawsen Lakhal, Hager Khechine, and Daniel 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.

[81] Cliff Lampe and Paul 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.

[82] J. Richard Landis and Gary G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977.

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

[84] Filippo Lanubile, Christof Ebert, Rafael Prikladnicki, and Aurora Vizcaino. Collaboration tools for global software engineering. *IEEE Software*, 27(2):52–55, 2010.

[85] Jean Lave and Etienne Wenger. *Situated learning: Legitimate peripheral participation.* Cambridge University Press, 1991.

[86] Jean Lave and Etienne Wenger. Legitimate peripheral participation in communities of practice. *Supporting lifelong learning*, 1:111–126, 2002.

[87] Carlene Lebeuf, Margaret-Anne Storey, and Alexey Zagalsky. How software developers mitigate collaboration friction with chatbots. *arXiv preprint arXiv:1702.07011*, 2017.

[88] Carlene Lebeuf, Margaret-Anne Storey, and Alexey Zagalsky. Software bots. *IEEE Software*, 35(1):18–23, January 2018.

[89] Meir M Lehman and JF Ramil. An approach to a theory of software evolution. In *Proceedings of the 4th international workshop on Principles of software evolution*, pages 70–74. ACM, 2001.

[90] Dorothy A Leonard and Sylvia Sensiper. The role of tacit knowledge in group innovation. In *Managing Knowledge Assets, Creativity and Innovation*, pages 301–323. World Scientific, 2011.

[91] Bo Leuf and Ward Cunningham. *The Wiki Way: Quick Collaboration on the Web*. Addison-Wesley Professional, 2001.

[92] Hongwei Li, Zhenchang Xing, Xin Peng, and Wenyun Zhao. What help do developers seek, when and how? In *Reverse Engineering (WCRE), 2013 20th Working Conference on Reverse Engineering*, pages 142–151. IEEE, 2013.

[93] Diane Wei Liang, Richard Moreland, and Linda Argote. Group versus individual training and group performance: The mediating role of transactive memory. *Personality and Social Psychology Bulletin*, 21(4):384–393, 1995.

[94] YS Lincoln and EG Guba. Naturalistic inquiry. newburry park, 1985.

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

[96] Wendy E Mackay and Anne-Laure Fayard. Hci, natural science and design: a framework for triangulation across disciplines. In *Proceedings of the 2nd conference on Designing interactive systems: processes, practices, methods, and techniques*, pages 223–234. ACM, 1997.

[97] Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. Design lessons from the fastest Q&A site in the west. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, CHI '11, pages 2857–2866, 2011.

[98] Lynne M Markus. Toward a theory of knowledge reuse: Types of knowledge reuse situations and factors in reuse success. *Journal of management information systems*, 18(1):57–93, 2001.

[99] Joseph E McGrath. Methodology matters: Doing research in the behavioral and social sciences. In *Readings in Human–Computer Interaction*, pages 152–169. Elsevier, 1995.

[100] Margaret Mead. Visual anthropology in a discipline of words. *Principles of visual anthropology*, 3:3–12, 1975.

[101] Mariel Miller and Allyson Fiona Hadwin. Scripting and awareness tools for regulating collaborative learning: Changing the landscape of support in cscl. *Computers in Human Behavior*, 52:573–588, 2015.

[102] Susan Marie Mitchell. *Software process improvement through the removal of project-level knowledge flow obstacles: the perceptions of software engineers.* University of Maryland, Baltimore County, 2012.

[103] Richard L Moreland, Linda Argote, and Ranjani Krishnan. Training people to work in groups. In *Theory and research on small groups*, pages 37–60. Springer, 2002.

[104] Anthony Naaeke, Anastacia Kurylo, Michael Grabowski, David Linton, and Marie L Radford. Insider and outsider perspective in ethnographic research. *Proceedings of the New York State Communication Association*, 2010(1):9, 2011.

[105] Peter Naur. Programming as theory building. *Microprocessing and microprogramming*, 15(5):253–261, 1985.

[106] Mark Nissen, Magdi Kamel, and Kishore Sengupta. Integrated analysis and design of knowledge systems and processes. *Knowledge Management and Virtual Organizations*, 13(1):24–43, 2000.

[107] Mark E Nissen. An extended model of knowledge-flow dynamics. *Communications of the Association for Information Systems*, 8(1):18, 2002.

[108] Ikujiro Nonaka. A dynamic theory of organizational knowledge creation. *Organization science*, 5(1):14–37, 1994.

[109] Ikujiro Nonaka. *The knowledge-creating company.* Harvard Business Review Press, 2008.

[110] Ikujiro Nonaka, Ryoko Toyama, and Noboru Konno. Seci, ba and leadership: a unified model of dynamic knowledge creation. *Long range planning*, 33(1):5–34, 2000.

[111] Donald A Norman. Things that make us smart, 1993.

[112] Tim Oreilly. What is web 2.0, 2005.

[113] Shelly Park and Frank 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.

[114] Chris Parnin and Christoph Treude. Measuring api documentation on the web. In *Proc. of the 2nd Intl. Workshop on Web 2.0 for Software Engineering*, Web2SE '11, pages 25–30, 2011.

[115] Chris Parnin, Christoph Treude, Lars Grammel, and Margaret-Anne Storey. Crowd documentation: Exploring the coverage and the dynamics of api discussions on stack overflow. Technical Report GIT-CS-12-05, Georgia Tech, 2012.

[116] Jeffrey M Perkel. How scientists use slack. *Nature News*, 541(7635):123, 2017.

[117] Raphael Pham, Leif Singer, Olga Liskin, Fernando Figueira Filho, and Kurt Schneider. Creating a shared understanding of testing culture on a social coding site. In *Proceedings of the 35th International Conference on Software Engineering*, ICSE '13, pages 112–121, 2013.

[118] Michael Polanyi. *The tacit dimension.* University of Chicago press, 2009.

[119] Michael Polanyi. *Personal knowledge: Towards a post-critical philosophy.* University of Chicago Press, 2015.

[120] Hortense Powdermaker. *Stranger and friend: The way of an anthropologist.* Number 410. WW Norton & Company, 1966.

[121] Pasi Pyöriä. The concept of knowledge work revisited. *Journal of Knowledge Management*, 9(3):116–127, 2005.

[122] Robert B Reich. *The Work of Nations: Preparing Ourselves for 21st Century Capitalis.* Vintage, 2010.

[123] Wolfgang Reinhardt. *Awareness support for knowledge workers in research networks.* PhD thesis, Open University in the Netherlands (CELSTEC), 2012.

[124] Wolfgang Reinhardt, Benedikt Schmidt, Peter Sloep, and Hendrik Drachsler. Knowledge worker roles and actions—results of two empirical studies. *Knowledge and Process Management*, 18(3):150–174, 2011.

[125] Peter C. Rigby, Brendan Cleary, Frederic Painchaud, Margaret-Anne Storey, and Daniel M. German. Contemporary peer review in action: Lessons from open source development. *IEEE Software*, 29(6):56–61, 2012.

[126] Peter C. Rigby and Margaret-Anne Storey. Understanding broadcast based peer review on open source software projects. In *Proc. of the 33rd Intl. Conf. on Software Engineering*, pages 541–550, 2011.

[127] Lionel P Robert and Alan R Dennis. Paradox of richness: A cognitive model of media choice. *IEEE transactions on professional communication*, 48(1):10–21, 2005.

[128] Pierre N. Robillard. The role of knowledge in software development. *Commun. ACM*, 42(1):87–92, January 1999.

[129] Colin Robson. Real world research: A resource for social scientists and practitioner-researchers, 2002.

[130] Everett M Rogers. *Diffusion of innovations.* The Free Press, 2003.

[131] P. Runeson, M. Host, A. Rainer, and B. Regnell. *Case Study Research in Software Engineering: Guidelines and Examples.* Wiley, 2012.

[132] Eduardo Salas, C Shawn Burke, and Janis A Cannon-Bowers. Teamwork: emerging principles. *International Journal of Management Reviews*, 2(4):339–356, 2000.

[133] Margarete Sandelowski and Julie Barroso. *Handbook for synthesizing qualitative research*. Springer Publishing Company, 2006.

[134] Christian Sauter, Thomas Mühlherr, and Stephanie Teufel. Sozio-kulturelle auswirkungen von groupware: Ein ansatz zur adaption und operationalisierung eines sozialpsychologischen modells für die gestaltung und den einsatz von groupware. In *ISI*, pages 517–526, 1994.

[135] Russell K Schutt. *Investigating the social world: The process and practice of research*. Pine Forge Press, 2011.

[136] Clive Seale. Quality in qualitative research. *Qualitative inquiry*, 5(4):465–478, 1999.

[137] Abigail J Sellen and Richard HR Harper. *The myth of the paperless office*. MIT press, 2003.

[138] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, January 2001.

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

[140] Leif Singer, Fernando Figueira Filho, Brendan Cleary, Christoph Treude, Margaret-Anne Storey, and Kurt Schneider. Mutual assessment in the social programmer ecosystem: An empirical investigation of developer profile aggregators. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 103–116, New York, NY, USA, 2013. ACM.

[141] Leif Singer, Fernando Figueira Filho, Brendan Cleary, Christoph Treude, Margaret-Anne Storey, and Kurt Schneider. Mutual assessment in the social programmer ecosystem: An empirical investigation of developer profile aggregators. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 103–116, New York, NY, USA, 2013. ACM.

[142] Leif Singer, Fernando Figueira Filho, and Margaret-Anne Storey. Software engineering at the speed of light: How developers stay current using twitter. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 211–221, New York, NY, USA, 2014. ACM.

[143] Dag IK Sjøberg, Tore Dybå, Bente CD Anda, and Jo E Hannay. Building theories in software engineering. In *Guide to advanced empirical software engineering*, pages 312–336. Springer, 2008.

[144] David Skyrme. Knowledge networking. *Creating the collaborative enterprise*, 1999.

[145] Brent D Slife, Richard N Williams, and Richard N Williams. *What's behind the research?: Discovering hidden assumptions in the behavioral sciences*. Sage, 1995.

[146] J-C Spender. Making knowledge the basis of a dynamic theory of the firm. *Strategic management journal*, 17(S2):45–62, 1996.

[147] Megan Squire. Should we move to Stack Overflow?: measuring the utility of social media for developer support. In *37th Intl. Conf. on Software Engineering*, pages 219–228, 2015.

[148] Ivan Srba and Maria Bielikova. Why is stack overflow failing? preserving sustainability in community question answering. *IEEE Software*, 33(4):80–89, 2016.

[149] Gerry Stahl. Group cognition: Computer support for building collaborative learning, 2006.

[150] Gerry Stahl. Theories of cognition in cscw. In *ECSCW 2011: Proceedings of the 12th European Conference on Computer Supported Cooperative Work, 24-28 September 2011, Aarhus Denmark*, pages 193–212. Springer, 2011.

[151] Christiaan D Stam. Knowledge productivity: designing and testing a method to diagnose knowledge productivity and plan for enhancement. 2007.

[152] William H Starbuck. Learning by knowledge-intensive firms. *Journal of management Studies*, 29(6):713–740, 1992.

[153] Steven E.v Stemler. A comparison of consensus, consistency, and measurement approaches to estimating interrater reliability. *Practical Assessment, Research & Evaluation, 9, 4*, 2004.

[154] Klaas-Jan Stol and Brian Fitzgerald. Uncovering theories in software engineering. In *Software Engineering (GTSE), 2013 2nd SEMAT Workshop on a General Theory of*, pages 5–14. IEEE, 2013.

[155] M.-A. Storey, Alexey Zagalsky, Fernando Figueira Filho, Leif Singer, and Daniel M. German. How social and communication channels shape and challenge a participatory culture in software development. In *IEEE Transactions on Software Engineering*. IEEE, 2016.

[156] Margaret-Anne Storey, Leif Singer, Brendan Cleary, Fernando Figueira Filho, and Alexey Zagalsky. The (r) evolution of social media in software engineering. In *Proceedings of the on Future of Software Engineering*, FOSE 2014, pages 100–116, New York, NY, USA, 2014. ACM.

[157] Margaret-Anne Storey and Alexey Zagalsky. Disrupting developer productivity one bot at a time. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2016, pages 928–931, New York, NY, USA, 2016. ACM.

[158] Anselm Strauss and Juliet Corbin. Basics of qualitative research: Procedures and techniques for developing grounded theory, 1998.

[159] Karl Erik Sveiby and Tom Lloyd. *Managing knowhow*. Bloomsbury London, 1987.

[160] Enock Takyi. The challenge of involvement and detachment in participant observation. *The Qualitative Report*, 20(6):864, 2015.

[161] Yla R. Tausczik, Aniket Kittur, and Robert E. Kraut. Collaborative problem solving: A study of mathoverflow. In *Proc. of the 17th ACM Conf. on Computer Supported Cooperative Work and Social Computing*, CSCW '14, pages 355–367, 2014.

[162] Alvin Toffler. *Future shock*. Bantam, 1971.

[163] Eileen Moore Trauth. *The culture of an information economy: Influences and impacts in the Republic of Ireland.* Springer Science & Business Media, 2012.

[164] Christoph Treude. *The role of social media artifacts in collaborative software development.* PhD thesis, 2012.

[165] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. How do programmers ask and answer questions on the web? (NIER track). In *Proc. of the 33rd Intl. Conf. on Software Engineering*, pages 804–807, 2011.

[166] Christoph Treude and Margaret-Anne Storey. Effective communication of software development knowledge through community portals. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ESEC/FSE '11, pages 91–101, New York, NY, USA, 2011. ACM.

[167] Linda Klebe Trevino, Robert H Lengel, and Richard L Daft. Understanding managers' media choices: A symbolic interactionist perspective. *Organizations and communication technology*, page 71, 1990.

[168] Jason Tsay, Laura Dabbish, and James D Herbsleb. Social media in transparent work environments. In *Cooperative and Human Aspects of Software Engineering (CHASE), 2013 6th International Workshop on*, pages 65–72. IEEE, 2013.

[169] Bogdan Vasilescu. *Social aspects of collaboration in online software communities.* PhD thesis, Eindhoven University of Technology, 2014.

[170] Bogdan Vasilescu, Alexander Serebrenik, Premkumar T. Devanbu, and Vladimir Filkov. How social Q&A sites are changing knowledge sharing in open source software communities. In *Proc. of the 17th ACM Conf. on Computer Supported Cooperative Work and Social Computing*, pages 342–354, 2014.

[171] Diego M. Virasoro, Pauline Leonard, and Mark Weal. An analysis of social news websites. In *Proc. ACM WebSci'11.* ACM, June 2011.

[172] Max Völkel. *Personal knowledge models with semantic technologies.* Doctoral dissertation, 2010.

[173] Max Völkel. *Personal knowledge models with semantic technologies.* BoD–Books on Demand, 2011.

[174] Georg Von Krogh, Kazuo Ichijo, and Ikujiro Nonaka. *Enabling knowledge creation: How to unlock the mystery of tacit knowledge and release the power of innovation*. Oxford University Press on Demand, 2000.

[175] Patrick A Wagstrom. *Vertical interaction in open software engineering communities*. PhD thesis, 2009.

[176] Andrew Walenstein. *Cognitive support in software engineering tools: A distributed cognition framework*. PhD thesis, Citeseer, 2002.

[177] Xiaofeng Wang, Ilona Kuzmickaja, Klaas-Jan Stol, Pekka Abrahamsson, and Brian Fitzgerald. Microblogging in open source software development: The case of drupal using twitter. *IEEE Software*, 99(PrePrints):1, 2013.

[178] Spencer A Ward and Linda J Reed. Knowledge structure and use: Implications for synthesis and interpretation. 1983.

[179] Molly McLure Wasko and Samer 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.

[180] Warren E Watson, Larry K Michaelsen, and Walt Sharp. Member competence, group interaction, and group decision making: A longitudinal study. *Journal of applied psychology*, 76(6):803, 1991.

[181] Daniel M Wegner. Transactive memory: A contemporary analysis of the group mind. In *Theories of group behavior*, pages 185–208. Springer, 1987.

[182] Daniel M Wegner. A computer network model of human transactive memory. *Social cognition*, 13(3):319–339, 1995.

[183] Etienne Wenger, Nancy White, and John D Smith. *Digital habitats: Stewarding technology for communities*. CPsquare, 2009.

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

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

[186] Charles D Winslow and William L Bramer. *FutureWork: Putting knowledge to work in the knowledge economy.* Free Press New York, 1994.

[187] Harry F Wolcott. *Ethnography: A way of seeing.* Rowman Altamira, 1999.

[188] Fangqi Xu. *The Formation and Development of Ikujiro Nonaka's Knowledge Creation Theory*, pages 60–73. Palgrave Macmillan UK, London, 2013.

[189] Olive Yonge and Len Stewin. Reliability and validity: Misnomers for qualitative research. *Canadian Journal of Nursing Research Archive*, 20(2), 1988.

[190] Alexey Zagalsky, Joseph Feliciano, Margaret-Anne Storey, Yiyun Zhao, and Weiliang Wang. The emergence of github as a collaborative platform for education. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '15, pages 1906–1917, New York, NY, USA, 2015. ACM.

[191] Alexey Zagalsky, Daniel M. German, Margaret-Anne Storey, Carlos Gómez Teshima, and Germán Poo-Caamaño. How the R community creates and curates knowledge: an extended study of stack overflow and mailing lists. *Empirical Software Engineering*, 23(2):953–986, Aug 2017.

[192] Alexey Zagalsky, Carlos Gómez Teshima, Daniel M German, Margaret-Anne Storey, and Germán Poo-Caamaño. How the R community creates and curates knowledge: a comparative study of stack overflow and mailing lists. In *Proceedings of the 13th International Conference on Mining Software Repositories*, pages 441–451. ACM, 2016.

[193] Mansooreh Zahedi, Mojtaba Shahin, and Muhammad Ali Babar. A systematic review of knowledge sharing challenges and practices in global software development. *International Journal of Information Management*, 36(6):995–1019, 2016.

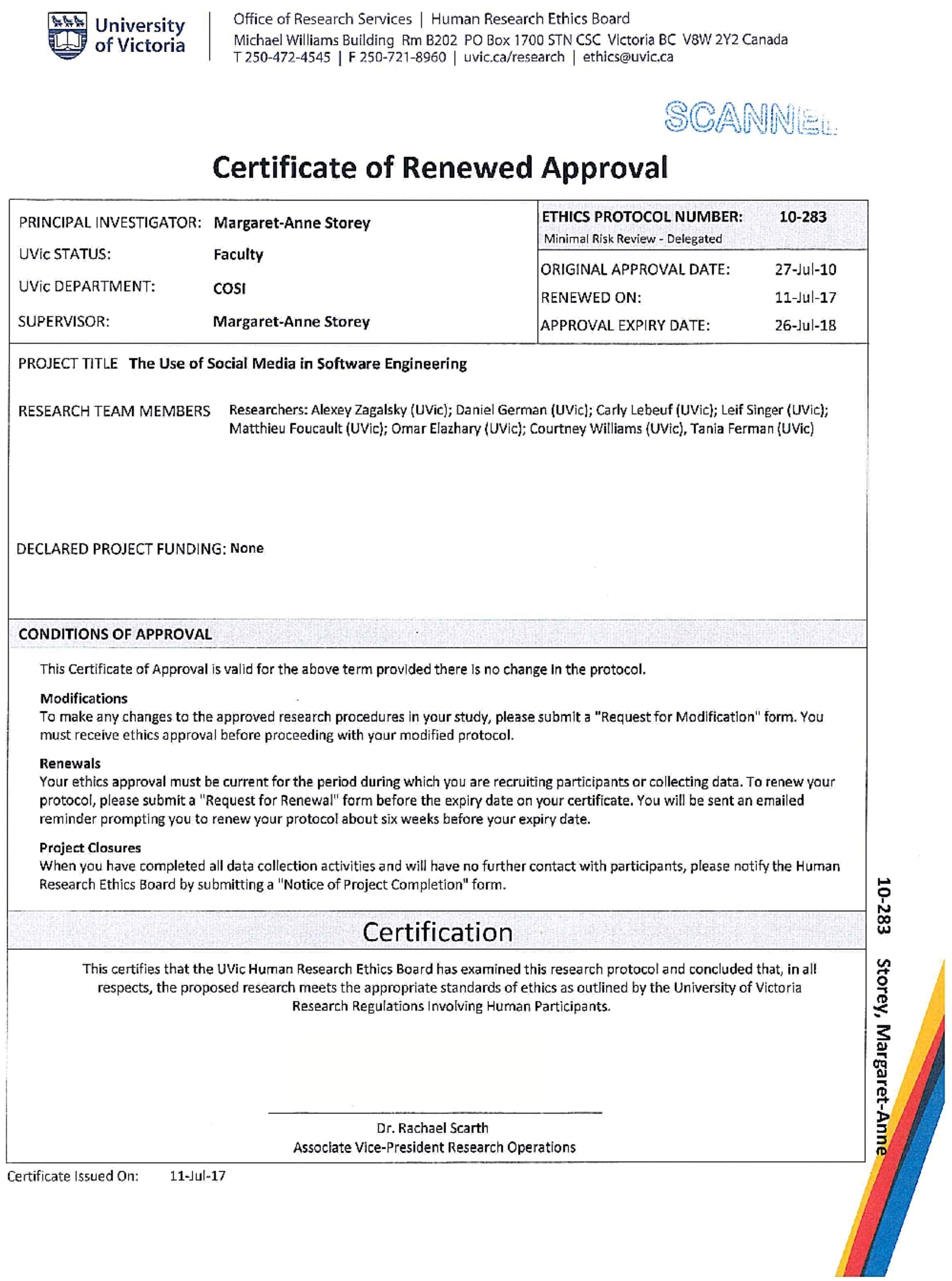
[194] Amy X. Zhang, Mark S. Ackerman, and David R. Karger. Mailing lists: Why are they still here, what is wrong with them, and how can we fix them? In *Proc. of the 33rd SIGCHI Conf. on Human Factors in Computing Systems*, 2015.

Appendices

Appendix A

Data Collection Instruments for Chapter 4

A.1 Ethics Approval



University of Victoria | Office of Research Services | Human Research Ethics Board
Michael Williams Building Rm B202 PO Box 1700 STN CSC Victoria BC V8W 2Y2 Canada
T 250-472-4545 | F 250-721-8960 | uvic.ca/research | ethics@uvic.ca

SCANNED

Certificate of Renewed Approval

PRINCIPAL INVESTIGATOR:	**Margaret-Anne Storey**	**ETHICS PROTOCOL NUMBER:** Minimal Risk Review - Delegated	**10-283**
UVic STATUS:	**Faculty**	ORIGINAL APPROVAL DATE:	27-Jul-10
UVic DEPARTMENT:	**COSI**	RENEWED ON:	11-Jul-17
SUPERVISOR:	**Margaret-Anne Storey**	APPROVAL EXPIRY DATE:	26-Jul-18

PROJECT TITLE **The Use of Social Media in Software Engineering**

RESEARCH TEAM MEMBERS Researchers: Alexey Zagalsky (UVic); Daniel German (UVic); Carly Lebeuf (UVic); Leif Singer (UVic); Matthieu Foucault (UVic); Omar Elazhary (UVic); Courtney Williams (UVic), Tania Ferman (UVic)

DECLARED PROJECT FUNDING: None

CONDITIONS OF APPROVAL

This Certificate of Approval is valid for the above term provided there is no change in the protocol.

Modifications
To make any changes to the approved research procedures in your study, please submit a "Request for Modification" form. You must receive ethics approval before proceeding with your modified protocol.

Renewals
Your ethics approval must be current for the period during which you are recruiting participants or collecting data. To renew your protocol, please submit a "Request for Renewal" form before the expiry date on your certificate. You will be sent an emailed reminder prompting you to renew your protocol about six weeks before your expiry date.

Project Closures
When you have completed all data collection activities and will have no further contact with participants, please notify the Human Research Ethics Board by submitting a "Notice of Project Completion" form.

Certification

This certifies that the UVic Human Research Ethics Board has examined this research protocol and concluded that, in all respects, the proposed research meets the appropriate standards of ethics as outlined by the University of Victoria Research Regulations Involving Human Participants.

Dr. Rachael Scarth
Associate Vice-President Research Operations

Certificate Issued On: 11-Jul-17

10-283 Storey, Margaret-Anne

A.2 Survey

University of Victoria Developer Survey:

How do you communicate and collaborate in 2015?

Hello!

We're researchers from the University of Victoria and we're interested in how software developers communicate and collaborate.

We'd be grateful if you could help us understand this better by filling the survey below. It should only take about 10 minutes of your time.

This is a purely academic research project with no commercial interests. We will ***OPENLY PUBLISH*** the results so everyone can benefit from them, but will ***ANONYMIZE*** everything before doing so. We will handle your response confidentially. If at some point during the survey you want to stop, you're free to do so without any negative consequences.

Please find our Letter of Information for Implied Consent here (http://leif.me/files/Consent_Social-Media.pdf), it includes the details on anonymity, confidentiality, and related issues.

Thanks so much for your help!

Margaret-Anne Storey (http://webhome.cs.uvic.ca/~mstorey), Alexey Zagalsky (http://alexeyza.com), Daniel German (http://turingmachine.org), and Leif Singer (http://leif.me) from the University of Victoria (https://www.uvic.ca) in Canada. Fernando Figueira Filho (http://fernandofilho.me/) from the Federal University of Rio Grande do Norte in Brazil.

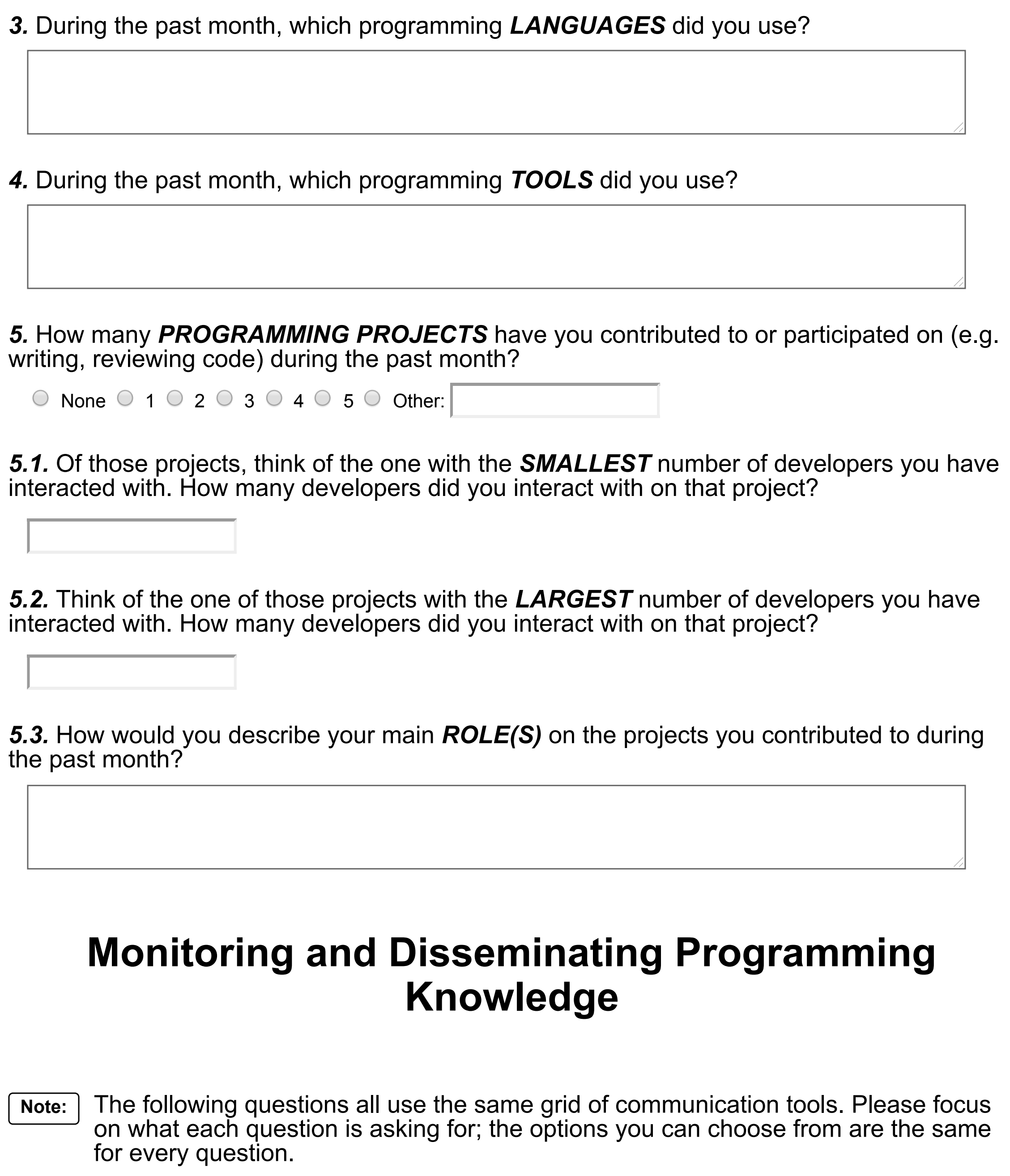
Basics

1. Do you develop software?

☐ Yes, professionally ☐ Yes, non-professionally -- e.g. pet projects, tinkering, ... ☐ Yes, I contribute to one or more open source projects (irrespective of size)

2. How many years have you been programming?

☐



3. During the past month, which programming ***LANGUAGES*** did you use?

4. During the past month, which programming ***TOOLS*** did you use?

5. How many ***PROGRAMMING PROJECTS*** have you contributed to or participated on (e.g. writing, reviewing code) during the past month?

○ None ○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ Other:

5.1. Of those projects, think of the one with the ***SMALLEST*** number of developers you have interacted with. How many developers did you interact with on that project?

5.2. Think of the one of those projects with the ***LARGEST*** number of developers you have interacted with. How many developers did you interact with on that project?

5.3. How would you describe your main ***ROLE(S)*** on the projects you contributed to during the past month?

Monitoring and Disseminating Programming Knowledge

Note: The following questions all use the same grid of communication tools. Please focus on what each question is asking for; the options you can choose from are the same for every question.

6. During the past month, the following were important at helping me stay ***UP TO DATE*** **about new technologies, practices, trends and tools for software development**.
Please check all that apply.

- ☐ **Face-to-face communication**
- ☐ **Books and Magazines**
- ☐ **Web Search** (Google, Bing, Duckduckgo, ...)
- ☐ **News Aggregators** (Hackernews, Reddit, Digg, Slashdot, ...)
- ☐ **Feeds and Blogs** (RSS, Feedly, Newsletters, blogs in general, ...)
- ☐ **Content Recommenders** (Stumble Upon, Prismatic, Flipboard, ...)
- ☐ **Social Bookmarking** (Pinterest, Pinboard, Delicious, ...)
- ☐ **Rich Content** (Podcasts, Screencasts, ...)
- ☐ **Discussion Groups** (Mailing Lists, Google Groups, Usenet, Forums, ...)
- ☐ **Private Discussions** (Email, ...)
- ☐ **Public Chat** (IRC, ...)
- ☐ **Private Chat** (IM, Skype Chat, Google Chat, ...)
- ☐ **Professional Networking Sites** (LinkedIn, Xing, ...)
- ☐ **Developer Profile Sites** (Coderwall, Geekli.st, Masterbranch, ...)
- ☐ **Social Network Sites** (Facebook, Google Plus, vk.com, Diaspora, ...)
- ☐ **Microblogs** (Twitter, Tumblr, App.net, Sina Weibo, Plurk, ...)
- ☐ **Code Hosting Sites** (GitHub, BitBucket, Launchpad, Google Code, Sourceforge, ...)
- ☐ **Project Coordination Tools** (Basecamp, Bugtrackers, ...)
- ☐ **Question & Answer Sites** (Stack Overflow, Quora, ...)
- ☐ **Other:** ________

7. During the past month, I have used the following to ***KEEP TRACK*** of **my development activities**.
Please check all that apply.

- ☐ **Face-to-face communication**
- ☐ **Books and Magazines**
- ☐ **Web Search** (Google, Bing, Duckduckgo, ...)
- ☐ **News Aggregators** (Hackernews, Reddit, Digg, Slashdot, ...)
- ☐ **Feeds and Blogs** (RSS, Feedly, Newsletters, blogs in general, ...)
- ☐ **Content Recommenders** (Stumble Upon, Prismatic, Flipboard, ...)

- ☐ **Social Bookmarking** (Pinterest, Pinboard, Delicious, ...)
- ☐ **Rich Content** (Podcasts, Screencasts, ...)
- ☐ **Discussion Groups** (Mailing Lists, Google Groups, Usenet, Forums, ...)
- ☐ **Private Discussions** (Email, ...)
- ☐ **Public Chat** (IRC, ...)
- ☐ **Private Chat** (IM, Skype Chat, Google Chat, ...)
- ☐ **Professional Networking Sites** (LinkedIn, Xing, ...)
- ☐ **Developer Profile Sites** (Coderwall, Geekli.st, Masterbranch, ...)
- ☐ **Social Network Sites** (Facebook, Google Plus, vk.com, Diaspora, ...)
- ☐ **Microblogs** (Twitter, Tumblr, App.net, Sina Weibo, Plurk, ...)
- ☐ **Code Hosting Sites** (GitHub, BitBucket, Launchpad, Google Code, Sourceforge, ...)
- ☐ **Project Coordination Tools** (Basecamp, Bugtrackers, ...)
- ☐ **Question & Answer Sites** (Stack Overflow, Quora, ...)
- ☐ **Other:** ______

8. During the past month, I have used the following to ***MONITOR*** **changes to projects**that I care about.
Please check all that apply.

- ☐ **Face-to-face communication**
- ☐ **Books and Magazines**
- ☐ **Web Search** (Google, Bing, Duckduckgo, ...)
- ☐ **News Aggregators** (Hackernews, Reddit, Digg, Slashdot, ...)
- ☐ **Feeds and Blogs** (RSS, Feedly, Newsletters, blogs in general, ...)
- ☐ **Content Recommenders** (Stumble Upon, Prismatic, Flipboard, ...)
- ☐ **Social Bookmarking** (Pinterest, Pinboard, Delicious, ...)
- ☐ **Rich Content** (Podcasts, Screencasts, ...)
- ☐ **Discussion Groups** (Mailing Lists, Google Groups, Usenet, Forums, ...)
- ☐ **Private Discussions** (Email, ...)
- ☐ **Public Chat** (IRC, ...)
- ☐ **Private Chat** (IM, Skype Chat, Google Chat, ...)
- ☐ **Professional Networking Sites** (LinkedIn, Xing, ...)
- ☐ **Developer Profile Sites** (Coderwall, Geekli.st, Masterbranch, ...)

- ☐ **Social Network Sites** (Facebook, Google Plus, vk.com, Diaspora, ...)
- ☐ **Microblogs** (Twitter, Tumblr, App.net, Sina Weibo, Plurk, ...)
- ☐ **Code Hosting Sites** (GitHub, BitBucket, Launchpad, Google Code, Sourceforge, ...)
- ☐ **Project Coordination Tools** (Basecamp, Bugtrackers, ...)
- ☐ **Question & Answer Sites** (Stack Overflow, Quora, ...)
- ☐ **Other:**

9. During the past month, I have used the following to ***DISSEMINATE*** software engineering **content**.
Please check all that apply.

- ☐ **Face-to-face communication**
- ☐ **Books and Magazines**
- ☐ **Web Search** (Google, Bing, Duckduckgo, ...)
- ☐ **News Aggregators** (Hackernews, Reddit, Digg, Slashdot, ...)
- ☐ **Feeds and Blogs** (RSS, Feedly, Newsletters, blogs in general, ...)
- ☐ **Content Recommenders** (Stumble Upon, Prismatic, Flipboard, ...)
- ☐ **Social Bookmarking** (Pinterest, Pinboard, Delicious, ...)
- ☐ **Rich Content** (Podcasts, Screencasts, ...)
- ☐ **Discussion Groups** (Mailing Lists, Google Groups, Usenet, Forums, ...)
- ☐ **Private Discussions** (Email, ...)
- ☐ **Public Chat** (IRC, ...)
- ☐ **Private Chat** (IM, Skype Chat, Google Chat, ...)
- ☐ **Professional Networking Sites** (LinkedIn, Xing, ...)
- ☐ **Developer Profile Sites** (Coderwall, Geekli.st, Masterbranch, ...)
- ☐ **Social Network Sites** (Facebook, Google Plus, vk.com, Diaspora, ...)
- ☐ **Microblogs** (Twitter, Tumblr, App.net, Sina Weibo, Plurk, ...)
- ☐ **Code Hosting Sites** (GitHub, BitBucket, Launchpad, Google Code, Sourceforge, ...)
- ☐ **Project Coordination Tools** (Basecamp, Bugtrackers, ...)
- ☐ **Question & Answer Sites** (Stack Overflow, Quora, ...)
- ☐ **Other:**

Learning and Feedback

10. During the past month, the following were important at helping me ***LEARN*** and improve my **skills**.
Please check all that apply.

- ☐ **Face-to-face communication**
- ☐ **Books and Magazines**
- ☐ **Web Search** (Google, Bing, Duckduckgo, ...)
- ☐ **News Aggregators** (Hackernews, Reddit, Digg, Slashdot, ...)
- ☐ **Feeds and Blogs** (RSS, Feedly, Newsletters, blogs in general, ...)
- ☐ **Content Recommenders** (Stumble Upon, Prismatic, Flipboard, ...)
- ☐ **Social Bookmarking** (Pinterest, Pinboard, Delicious, ...)
- ☐ **Rich Content** (Podcasts, Screencasts, ...)
- ☐ **Discussion Groups** (Mailing Lists, Google Groups, Usenet, Forums, ...)
- ☐ **Private Discussions** (Email, ...)
- ☐ **Public Chat** (IRC, ...)
- ☐ **Private Chat** (IM, Skype Chat, Google Chat, ...)
- ☐ **Professional Networking Sites** (LinkedIn, Xing, ...)
- ☐ **Developer Profile Sites** (Coderwall, Geekli.st, Masterbranch, ...)
- ☐ **Social Network Sites** (Facebook, Google Plus, vk.com, Diaspora, ...)
- ☐ **Microblogs** (Twitter, Tumblr, App.net, Sina Weibo, Plurk, ...)
- ☐ **Code Hosting Sites** (GitHub, BitBucket, Launchpad, Google Code, Sourceforge, ...)
- ☐ **Project Coordination Tools** (Basecamp, Bugtrackers, ...)
- ☐ **Question & Answer Sites** (Stack Overflow, Quora, ...)
- ☐ **Other:** ________

11. During the past month, the following were useful for getting and giving ***FEEDBACK*** to **developers**.
Please check all that apply.

- ☐ **Face-to-face communication**
- ☐ **Books and Magazines**

- ☐ **Web Search** (Google, Bing, Duckduckgo, ...)
- ☐ **News Aggregators** (Hackernews, Reddit, Digg, Slashdot, ...)
- ☐ **Feeds and Blogs** (RSS, Feedly, Newsletters, blogs in general, ...)
- ☐ **Content Recommenders** (Stumble Upon, Prismatic, Flipboard, ...)
- ☐ **Social Bookmarking** (Pinterest, Pinboard, Delicious, ...)
- ☐ **Rich Content** (Podcasts, Screencasts, ...)
- ☐ **Discussion Groups** (Mailing Lists, Google Groups, Usenet, Forums, ...)
- ☐ **Private Discussions** (Email, ...)
- ☐ **Public Chat** (IRC, ...)
- ☐ **Private Chat** (IM, Skype Chat, Google Chat, ...)
- ☐ **Professional Networking Sites** (LinkedIn, Xing, ...)
- ☐ **Developer Profile Sites** (Coderwall, Geekli.st, Masterbranch, ...)
- ☐ **Social Network Sites** (Facebook, Google Plus, vk.com, Diaspora, ...)
- ☐ **Microblogs** (Twitter, Tumblr, App.net, Sina Weibo, Plurk, ...)
- ☐ **Code Hosting Sites** (GitHub, BitBucket, Launchpad, Google Code, Sourceforge, ...)
- ☐ **Project Coordination Tools** (Basecamp, Bugtrackers, ...)
- ☐ **Question & Answer Sites** (Stack Overflow, Quora, ...)
- ☐ **Other:** ____

12. During the past month, the following helped me ***FIND ANSWERS*** to specific **technical questions**.
Please check all that apply.

- ☐ **Face-to-face communication**
- ☐ **Books and Magazines**
- ☐ **Web Search** (Google, Bing, Duckduckgo, ...)
- ☐ **News Aggregators** (Hackernews, Reddit, Digg, Slashdot, ...)
- ☐ **Feeds and Blogs** (RSS, Feedly, Newsletters, blogs in general, ...)
- ☐ **Content Recommenders** (Stumble Upon, Prismatic, Flipboard, ...)
- ☐ **Social Bookmarking** (Pinterest, Pinboard, Delicious, ...)
- ☐ **Rich Content** (Podcasts, Screencasts, ...)
- ☐ **Discussion Groups** (Mailing Lists, Google Groups, Usenet, Forums, ...)
- ☐ **Private Discussions** (Email, ...)
- ☐ **Public Chat** (IRC, ...)

- ☐ **Private Chat** (IM, Skype Chat, Google Chat, ...)
- ☐ **Professional Networking Sites** (LinkedIn, Xing, ...)
- ☐ **Developer Profile Sites** (Coderwall, Geekli.st, Masterbranch, ...)

- ☐ **Social Network Sites** (Facebook, Google Plus, vk.com, Diaspora, ...)
- ☐ **Microblogs** (Twitter, Tumblr, App.net, Sina Weibo, Plurk, ...)
- ☐ **Code Hosting Sites** (GitHub, BitBucket, Launchpad, Google Code, Sourceforge, ...)
- ☐ **Project Coordination Tools** (Basecamp, Bugtrackers, ...)
- ☐ **Question & Answer Sites** (Stack Overflow, Quora, ...)
- ☐ **Other:** ______

13. During the past month, I used the following to ***PROVIDE ANSWERS*** to specific **technical questions** that other developers or users asked.
Please check all that apply.

- ☐ **Face-to-face communication**
- ☐ **Books and Magazines**
- ☐ **Web Search** (Google, Bing, Duckduckgo, ...)
- ☐ **News Aggregators** (Hackernews, Reddit, Digg, Slashdot, ...)
- ☐ **Feeds and Blogs** (RSS, Feedly, Newsletters, blogs in general, ...)
- ☐ **Content Recommenders** (Stumble Upon, Prismatic, Flipboard, ...)
- ☐ **Social Bookmarking** (Pinterest, Pinboard, Delicious, ...)

- ☐ **Rich Content** (Podcasts, Screencasts, ...)
- ☐ **Discussion Groups** (Mailing Lists, Google Groups, Usenet, Forums, ...)
- ☐ **Private Discussions** (Email, ...)
- ☐ **Public Chat** (IRC, ...)
- ☐ **Private Chat** (IM, Skype Chat, Google Chat, ...)
- ☐ **Professional Networking Sites** (LinkedIn, Xing, ...)
- ☐ **Developer Profile Sites** (Coderwall, Geekli.st, Masterbranch, ...)

- ☐ **Social Network Sites** (Facebook, Google Plus, vk.com, Diaspora, ...)
- ☐ **Microblogs** (Twitter, Tumblr, App.net, Sina Weibo, Plurk, ...)
- ☐ **Code Hosting Sites** (GitHub, BitBucket, Launchpad, Google Code, Sourceforge, ...)
- ☐ **Project Coordination Tools** (Basecamp, Bugtrackers, ...)

- ☐ **Question & Answer Sites** (Stack Overflow, Quora, ...)
- ☐ **Other:** ______

Networking

14. During the past month, the following helped me ***DISCOVER OR CONNECT*** with interesting **developers** that work on technologies or projects of interest to me.
Please check all that apply.

- ☐ **Face-to-face communication**
- ☐ **Books and Magazines**
- ☐ **Web Search** (Google, Bing, Duckduckgo, ...)
- ☐ **News Aggregators** (Hackernews, Reddit, Digg, Slashdot, ...)
- ☐ **Feeds and Blogs** (RSS, Feedly, Newsletters, blogs in general, ...)
- ☐ **Content Recommenders** (Stumble Upon, Prismatic, Flipboard, ...)
- ☐ **Social Bookmarking** (Pinterest, Pinboard, Delicious, ...)
- ☐ **Rich Content** (Podcasts, Screencasts, ...)
- ☐ **Discussion Groups** (Mailing Lists, Google Groups, Usenet, Forums, ...)
- ☐ **Private Discussions** (Email, ...)
- ☐ **Public Chat** (IRC, ...)
- ☐ **Private Chat** (IM, Skype Chat, Google Chat, ...)
- ☐ **Professional Networking Sites** (LinkedIn, Xing, ...)
- ☐ **Developer Profile Sites** (Coderwall, Geekli.st, Masterbranch, ...)
- ☐ **Social Network Sites** (Facebook, Google Plus, vk.com, Diaspora, ...)
- ☐ **Microblogs** (Twitter, Tumblr, App.net, Sina Weibo, Plurk, ...)
- ☐ **Code Hosting Sites** (GitHub, BitBucket, Launchpad, Google Code, Sourceforge, ...)
- ☐ **Project Coordination Tools** (Basecamp, Bugtrackers, ...)
- ☐ **Question & Answer Sites** (Stack Overflow, Quora, ...)
- ☐ **Other:** ______

15. During the past month, the following helped me ***COMMUNICATE*** with other ***DEVELOPERS***.

Please check all that apply.

- ☐ **Face-to-face communication**
- ☐ **Books and Magazines**
- ☐ **Web Search** (Google, Bing, Duckduckgo, ...)
- ☐ **News Aggregators** (Hackernews, Reddit, Digg, Slashdot, ...)
- ☐ **Feeds and Blogs** (RSS, Feedly, Newsletters, blogs in general, ...)
- ☐ **Content Recommenders** (Stumble Upon, Prismatic, Flipboard, ...)
- ☐ **Social Bookmarking** (Pinterest, Pinboard, Delicious, ...)
- ☐ **Rich Content** (Podcasts, Screencasts, ...)
- ☐ **Discussion Groups** (Mailing Lists, Google Groups, Usenet, Forums, ...)
- ☐ **Private Discussions** (Email, ...)
- ☐ **Public Chat** (IRC, ...)
- ☐ **Private Chat** (IM, Skype Chat, Google Chat, ...)
- ☐ **Professional Networking Sites** (LinkedIn, Xing, ...)
- ☐ **Developer Profile Sites** (Coderwall, Geekli.st, Masterbranch, ...)
- ☐ **Social Network Sites** (Facebook, Google Plus, vk.com, Diaspora, ...)
- ☐ **Microblogs** (Twitter, Tumblr, App.net, Sina Weibo, Plurk, ...)
- ☐ **Code Hosting Sites** (GitHub, BitBucket, Launchpad, Google Code, Sourceforge, ...)
- ☐ **Project Coordination Tools** (Basecamp, Bugtrackers, ...)
- ☐ **Question & Answer Sites** (Stack Overflow, Quora, ...)
- ☐ **Other:**

16. During the past month, the following helped me ***COMMUNICATE*** with ***USERS***.
Please check all that apply.

- ☐ **Face-to-face communication**
- ☐ **Books and Magazines**
- ☐ **Web Search** (Google, Bing, Duckduckgo, ...)
- ☐ **News Aggregators** (Hackernews, Reddit, Digg, Slashdot, ...)
- ☐ **Feeds and Blogs** (RSS, Feedly, Newsletters, blogs in general, ...)
- ☐ **Content Recommenders** (Stumble Upon, Prismatic, Flipboard, ...)
- ☐ **Social Bookmarking** (Pinterest, Pinboard, Delicious, ...)
- ☐ **Rich Content** (Podcasts, Screencasts, ...)

- ☐ **Discussion Groups** (Mailing Lists, Google Groups, Usenet, Forums, ...)
- ☐ **Private Discussions** (Email, ...)
- ☐ **Public Chat** (IRC, ...)
- ☐ **Private Chat** (IM, Skype Chat, Google Chat, ...)
- ☐ **Professional Networking Sites** (LinkedIn, Xing, ...)
- ☐ **Developer Profile Sites** (Coderwall, Geekli.st, Masterbranch, ...)
- ☐ **Social Network Sites** (Facebook, Google Plus, vk.com, Diaspora, ...)
- ☐ **Microblogs** (Twitter, Tumblr, App.net, Sina Weibo, Plurk, ...)
- ☐ **Code Hosting Sites** (GitHub, BitBucket, Launchpad, Google Code, Sourceforge, ...)
- ☐ **Project Coordination Tools** (Basecamp, Bugtrackers, ...)
- ☐ **Question & Answer Sites** (Stack Overflow, Quora, ...)
- ☐ **Other:** ______

17. During the past month, I have used the following to ***DISPLAY*** my technical **skills or accomplishments**.
Please check all that apply.

- ☐ **Face-to-face communication**
- ☐ **Books and Magazines**
- ☐ **Web Search** (Google, Bing, Duckduckgo, ...)
- ☐ **News Aggregators** (Hackernews, Reddit, Digg, Slashdot, ...)
- ☐ **Feeds and Blogs** (RSS, Feedly, Newsletters, blogs in general, ...)
- ☐ **Content Recommenders** (Stumble Upon, Prismatic, Flipboard, ...)
- ☐ **Social Bookmarking** (Pinterest, Pinboard, Delicious, ...)
- ☐ **Rich Content** (Podcasts, Screencasts, ...)
- ☐ **Discussion Groups** (Mailing Lists, Google Groups, Usenet, Forums, ...)
- ☐ **Private Discussions** (Email, ...)
- ☐ **Public Chat** (IRC, ...)
- ☐ **Private Chat** (IM, Skype Chat, Google Chat, ...)
- ☐ **Professional Networking Sites** (LinkedIn, Xing, ...)
- ☐ **Developer Profile Sites** (Coderwall, Geekli.st, Masterbranch, ...)
- ☐ **Social Network Sites** (Facebook, Google Plus, vk.com, Diaspora, ...)

- ☐ **Microblogs** (Twitter, Tumblr, App.net, Sina Weibo, Plurk, ...)
- ☐ **Code Hosting Sites** (GitHub, BitBucket, Launchpad, Google Code, Sourceforge, ...)
- ☐ **Project Coordination Tools** (Basecamp, Bugtrackers, ...)
- ☐ **Question & Answer Sites** (Stack Overflow, Quora, ...)
- ☐ **Other:** ____

Collaborating with Others

18. During the past month, I have used the following to ***COORDINATE TASKS*** with other developers.
Please check all that apply.

- ☐ **Face-to-face communication**
- ☐ **Books and Magazines**
- ☐ **Web Search** (Google, Bing, Duckduckgo, ...)
- ☐ **News Aggregators** (Hackernews, Reddit, Digg, Slashdot, ...)
- ☐ **Feeds and Blogs** (RSS, Feedly, Newsletters, blogs in general, ...)
- ☐ **Content Recommenders** (Stumble Upon, Prismatic, Flipboard, ...)
- ☐ **Social Bookmarking** (Pinterest, Pinboard, Delicious, ...)
- ☐ **Rich Content** (Podcasts, Screencasts, ...)
- ☐ **Discussion Groups** (Mailing Lists, Google Groups, Usenet, Forums, ...)
- ☐ **Private Discussions** (Email, ...)
- ☐ **Public Chat** (IRC, ...)
- ☐ **Private Chat** (IM, Skype Chat, Google Chat, ...)
- ☐ **Professional Networking Sites** (LinkedIn, Xing, ...)
- ☐ **Developer Profile Sites** (Coderwall, Geekli.st, Masterbranch, ...)
- ☐ **Social Network Sites** (Facebook, Google Plus, vk.com, Diaspora, ...)
- ☐ **Microblogs** (Twitter, Tumblr, App.net, Sina Weibo, Plurk, ...)
- ☐ **Code Hosting Sites** (GitHub, BitBucket, Launchpad, Google Code, Sourceforge, ...)
- ☐ **Project Coordination Tools** (Basecamp, Bugtrackers, ...)
- ☐ **Question & Answer Sites** (Stack Overflow, Quora, ...)
- ☐ **Other:** ____

19. During the past month, I have used the following to ***COLLABORATIVELY AUTHOR*** development **artifacts** (e.g., code, documentation, blogs, or user manuals, etc).
Please check all that apply.

- ☐ **Face-to-face communication**
- ☐ **Books and Magazines**
- ☐ **Web Search** (Google, Bing, Duckduckgo, ...)
- ☐ **News Aggregators** (Hackernews, Reddit, Digg, Slashdot, ...)
- ☐ **Feeds and Blogs** (RSS, Feedly, Newsletters, blogs in general, ...)
- ☐ **Content Recommenders** (Stumble Upon, Prismatic, Flipboard, ...)
- ☐ **Social Bookmarking** (Pinterest, Pinboard, Delicious, ...)
- ☐ **Rich Content** (Podcasts, Screencasts, ...)
- ☐ **Discussion Groups** (Mailing Lists, Google Groups, Usenet, Forums, ...)
- ☐ **Private Discussions** (Email, ...)
- ☐ **Public Chat** (IRC, ...)
- ☐ **Private Chat** (IM, Skype Chat, Google Chat, ...)
- ☐ **Professional Networking Sites** (LinkedIn, Xing, ...)
- ☐ **Developer Profile Sites** (Coderwall, Geekli.st, Masterbranch, ...)
- ☐ **Social Network Sites** (Facebook, Google Plus, vk.com, Diaspora, ...)
- ☐ **Microblogs** (Twitter, Tumblr, App.net, Sina Weibo, Plurk, ...)
- ☐ **Code Hosting Sites** (GitHub, BitBucket, Launchpad, Google Code, Sourceforge, ...)
- ☐ **Project Coordination Tools** (Basecamp, Bugtrackers, ...)
- ☐ **Question & Answer Sites** (Stack Overflow, Quora, ...)
- ☐ **Other:** ______

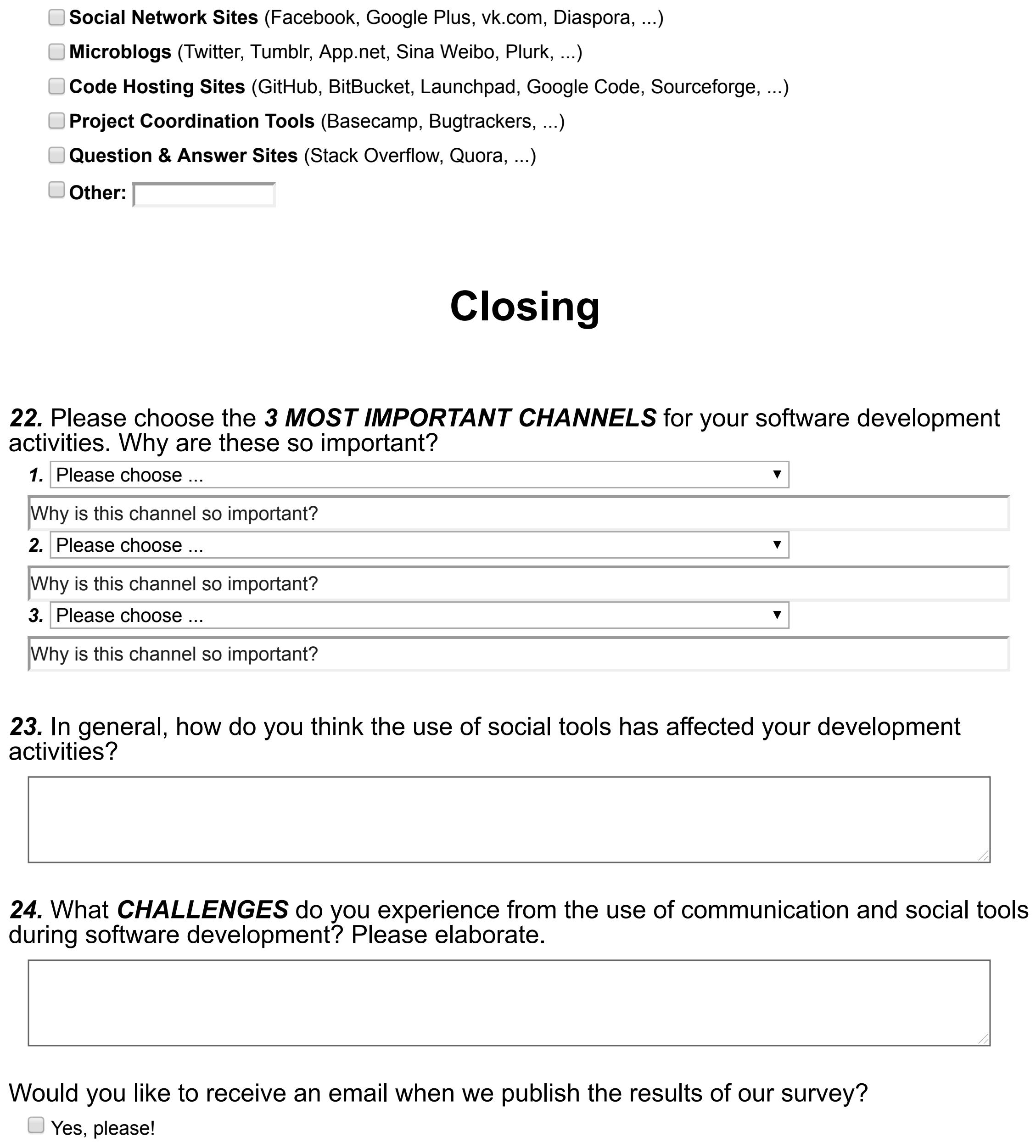
20. During the past month, I have used the following to ***MONITOR*** my collaborators' activities.
Please check all that apply.

- ☐ **Face-to-face communication**
- ☐ **Books and Magazines**
- ☐ **Web Search** (Google, Bing, Duckduckgo, ...)
- ☐ **News Aggregators** (Hackernews, Reddit, Digg, Slashdot, ...)
- ☐ **Feeds and Blogs** (RSS, Feedly, Newsletters, blogs in general, ...)

- ☐ **Content Recommenders** (Stumble Upon, Prismatic, Flipboard, ...)
- ☐ **Social Bookmarking** (Pinterest, Pinboard, Delicious, ...)
- ☐ **Rich Content** (Podcasts, Screencasts, ...)
- ☐ **Discussion Groups** (Mailing Lists, Google Groups, Usenet, Forums, ...)
- ☐ **Private Discussions** (Email, ...)
- ☐ **Public Chat** (IRC, ...)
- ☐ **Private Chat** (IM, Skype Chat, Google Chat, ...)
- ☐ **Professional Networking Sites** (LinkedIn, Xing, ...)
- ☐ **Developer Profile Sites** (Coderwall, Geekli.st, Masterbranch, ...)
- ☐ **Social Network Sites** (Facebook, Google Plus, vk.com, Diaspora, ...)
- ☐ **Microblogs** (Twitter, Tumblr, App.net, Sina Weibo, Plurk, ...)
- ☐ **Code Hosting Sites** (GitHub, BitBucket, Launchpad, Google Code, Sourceforge, ...)
- ☐ **Project Coordination Tools** (Basecamp, Bugtrackers, ...)
- ☐ **Question & Answer Sites** (Stack Overflow, Quora, ...)
- ☐ **Other:** ____

21. During the past month, I have used the following for ***FINDING EXPERTS*** to address specific technical problems.
Please check all that apply.

- ☐ **Face-to-face communication**
- ☐ **Books and Magazines**
- ☐ **Web Search** (Google, Bing, Duckduckgo, ...)
- ☐ **News Aggregators** (Hackernews, Reddit, Digg, Slashdot, ...)
- ☐ **Feeds and Blogs** (RSS, Feedly, Newsletters, blogs in general, ...)
- ☐ **Content Recommenders** (Stumble Upon, Prismatic, Flipboard, ...)
- ☐ **Social Bookmarking** (Pinterest, Pinboard, Delicious, ...)
- ☐ **Rich Content** (Podcasts, Screencasts, ...)
- ☐ **Discussion Groups** (Mailing Lists, Google Groups, Usenet, Forums, ...)
- ☐ **Private Discussions** (Email, ...)
- ☐ **Public Chat** (IRC, ...)
- ☐ **Private Chat** (IM, Skype Chat, Google Chat, ...)
- ☐ **Professional Networking Sites** (LinkedIn, Xing, ...)
- ☐ **Developer Profile Sites** (Coderwall, Geekli.st, Masterbranch, ...)



☐ **Social Network Sites** (Facebook, Google Plus, vk.com, Diaspora, ...)

☐ **Microblogs** (Twitter, Tumblr, App.net, Sina Weibo, Plurk, ...)

☐ **Code Hosting Sites** (GitHub, BitBucket, Launchpad, Google Code, Sourceforge, ...)

☐ **Project Coordination Tools** (Basecamp, Bugtrackers, ...)

☐ **Question & Answer Sites** (Stack Overflow, Quora, ...)

☐ **Other:**

Closing

22. Please choose the ***3 MOST IMPORTANT CHANNELS*** for your software development activities. Why are these so important?

1. Please choose ... ▾

Why is this channel so important?

2. Please choose ... ▾

Why is this channel so important?

3. Please choose ... ▾

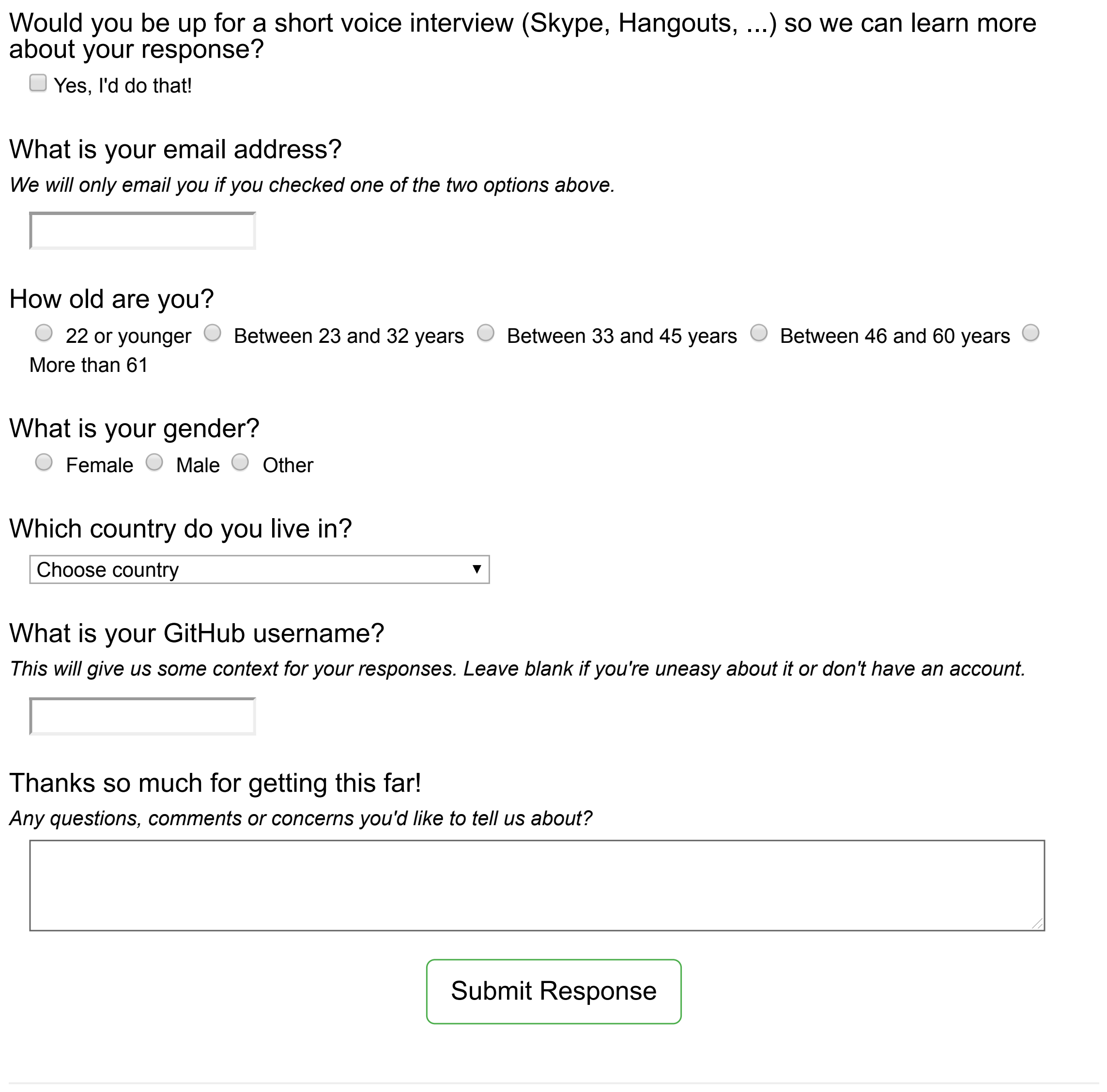
Why is this channel so important?

23. In general, how do you think the use of social tools has affected your development activities?

24. What ***CHALLENGES*** do you experience from the use of communication and social tools during software development? Please elaborate.

Would you like to receive an email when we publish the results of our survey?

☐ Yes, please!



Would you be up for a short voice interview (Skype, Hangouts, ...) so we can learn more about your response?

Yes, I'd do that!

What is your email address?

We will only email you if you checked one of the two options above.

How old are you?

22 or younger · Between 23 and 32 years · Between 33 and 45 years · Between 46 and 60 years · More than 61

What is your gender?

Female · Male · Other

Which country do you live in?

Choose country

What is your GitHub username?

This will give us some context for your responses. Leave blank if you're uneasy about it or don't have an account.

Thanks so much for getting this far!

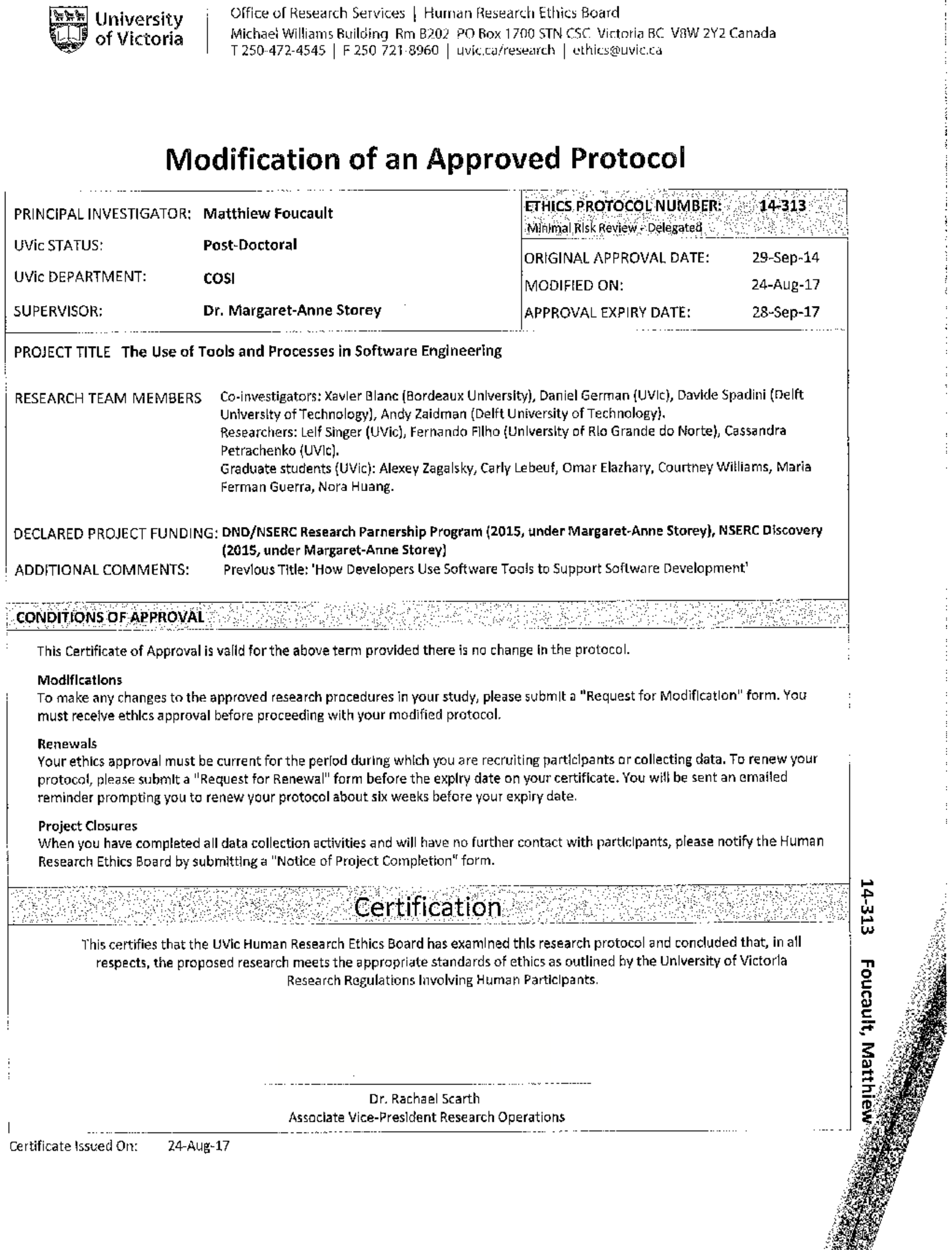
Any questions, comments or concerns you'd like to tell us about?

Submit Response

Appendix B

Data Collection Instruments for Chapter 5

B.1 Ethics Approval



University of Victoria | Office of Research Services | Human Research Ethics Board
Michael Williams Building Rm B202 PO Box 1700 STN CSC Victoria BC V8W 2Y2 Canada
T 250-472-4545 | F 250 721 8960 | uvic.ca/research | ethics@uvic.ca

Modification of an Approved Protocol

PRINCIPAL INVESTIGATOR:	**Matthiew Foucault**	**ETHICS PROTOCOL NUMBER:**	**14-313**
		Minimal Risk Review - Delegated	
UVic STATUS:	**Post-Doctoral**	ORIGINAL APPROVAL DATE:	29-Sep-14
UVic DEPARTMENT:	**COSI**	MODIFIED ON:	24-Aug-17
SUPERVISOR:	**Dr. Margaret-Anne Storey**	APPROVAL EXPIRY DATE:	28-Sep-17

PROJECT TITLE **The Use of Tools and Processes in Software Engineering**

RESEARCH TEAM MEMBERS Co-investigators: Xavier Blanc (Bordeaux University), Daniel German (UVic), Davide Spadini (Delft University of Technology), Andy Zaidman (Delft University of Technology).
Researchers: Leif Singer (UVic), Fernando Filho (University of Rio Grande do Norte), Cassandra Petrachenko (UVic).
Graduate students (UVic): Alexey Zagalsky, Carly Lebeuf, Omar Elazhary, Courtney Williams, Maria Ferman Guerra, Nora Huang.

DECLARED PROJECT FUNDING: **DND/NSERC Research Parnership Program (2015, under Margaret-Anne Storey), NSERC Discovery (2015, under Margaret-Anne Storey)**

ADDITIONAL COMMENTS: Previous Title: 'How Developers Use Software Tools to Support Software Development'

CONDITIONS OF APPROVAL

This Certificate of Approval is valid for the above term provided there is no change in the protocol.

Modifications
To make any changes to the approved research procedures in your study, please submit a "Request for Modification" form. You must receive ethics approval before proceeding with your modified protocol.

Renewals
Your ethics approval must be current for the period during which you are recruiting participants or collecting data. To renew your protocol, please submit a "Request for Renewal" form before the expiry date on your certificate. You will be sent an emailed reminder prompting you to renew your protocol about six weeks before your expiry date.

Project Closures
When you have completed all data collection activities and will have no further contact with participants, please notify the Human Research Ethics Board by submitting a "Notice of Project Completion" form.

Certification

This certifies that the UVic Human Research Ethics Board has examined this research protocol and concluded that, in all respects, the proposed research meets the appropriate standards of ethics as outlined by the University of Victoria Research Regulations Involving Human Participants.

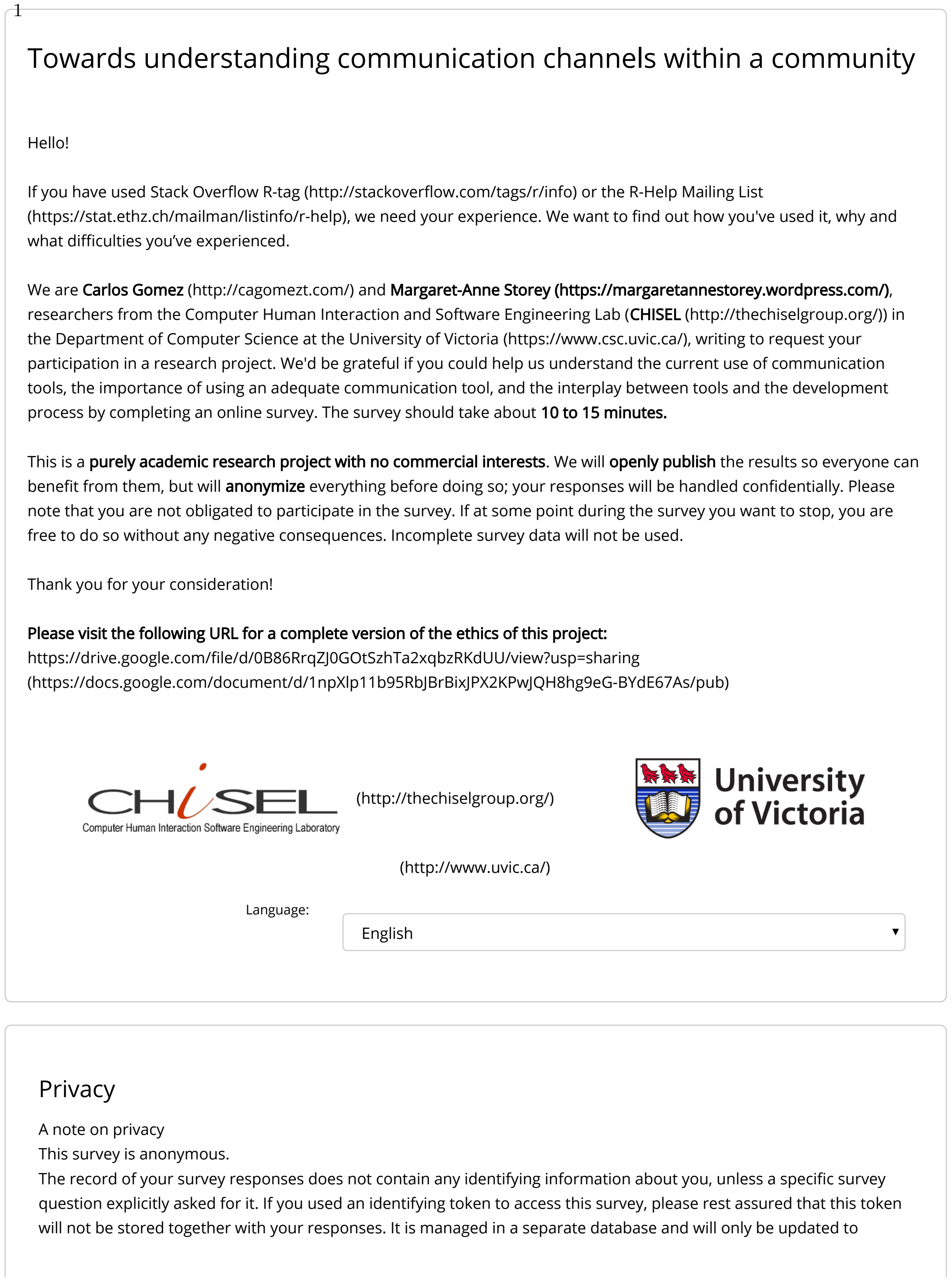
Dr. Rachael Scarth
Associate Vice-President Research Operations

Certificate Issued On: 24-Aug-17

14-313 Foucault, Matthiew

B.2 Survey

[1]



Towards understanding communication channels within a community

Hello!

If you have used Stack Overflow R-tag (http://stackoverflow.com/tags/r/info) or the R-Help Mailing List (https://stat.ethz.ch/mailman/listinfo/r-help), we need your experience. We want to find out how you've used it, why and what difficulties you've experienced.

We are **Carlos Gomez** (http://cagomezt.com/) and **Margaret-Anne Storey (https://margaretannestorey.wordpress.com/)**, researchers from the Computer Human Interaction and Software Engineering Lab (**CHISEL** (http://thechiselgroup.org/)) in the Department of Computer Science at the University of Victoria (https://www.csc.uvic.ca/), writing to request your participation in a research project. We'd be grateful if you could help us understand the current use of communication tools, the importance of using an adequate communication tool, and the interplay between tools and the development process by completing an online survey. The survey should take about **10 to 15 minutes.**

This is a **purely academic research project with no commercial interests**. We will **openly publish** the results so everyone can benefit from them, but will **anonymize** everything before doing so; your responses will be handled confidentially. Please note that you are not obligated to participate in the survey. If at some point during the survey you want to stop, you are free to do so without any negative consequences. Incomplete survey data will not be used.

Thank you for your consideration!

Please visit the following URL for a complete version of the ethics of this project:
https://drive.google.com/file/d/0B86RrqZJ0GOtSzhTa2xqbzRKdUU/view?usp=sharing
(https://docs.google.com/document/d/1npXlp11b95RbJBrBixJPX2KPwJQH8hg9eG-BYdE67As/pub)

CHiSEL Computer Human Interaction Software Engineering Laboratory (http://thechiselgroup.org/) University of Victoria

(http://www.uvic.ca/)

Language: English ▾

Privacy

A note on privacy
This survey is anonymous.
The record of your survey responses does not contain any identifying information about you, unless a specific survey question explicitly asked for it. If you used an identifying token to access this survey, please rest assured that this token will not be stored together with your responses. It is managed in a separate database and will only be updated to

[1]The survey included above was part of the study described in Chapter 5. It was designed and conducted by Carlos Gómez Teshima.

Towards understanding communication channels within a community

Language: English ▾

Demographics

1 What is your **area of expertise**?

ⓘ Check all that apply

☐ Computer Science

☐ Mathematics

☐ Life Science

☐ Healthcare

☐ Other:

2 What is your **occupation**?

ⓘ Check all that apply

☐ Academia (e.g., professor, or researcher)

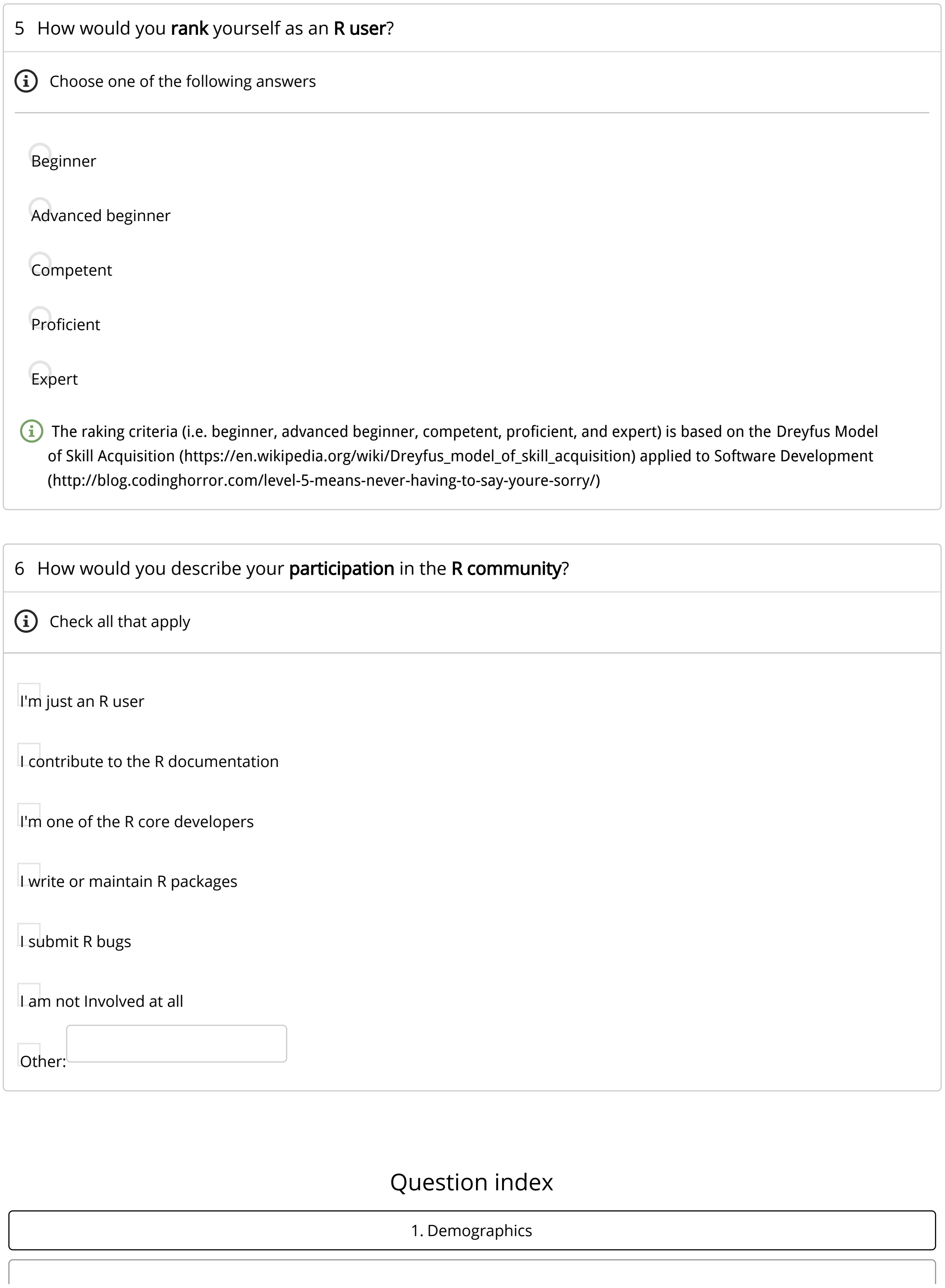
☐ Student (e.g., graduate or undergraduate student)

☐ Industry

☐ Other:

3 Do or did you have **experience** as a software developer/programmer prior to learning or using R?

Yes No



5 How would you **rank** yourself as an **R user**?

Choose one of the following answers

- Beginner
- Advanced beginner
- Competent
- Proficient
- Expert

The raking criteria (i.e. beginner, advanced beginner, competent, proficient, and expert) is based on the Dreyfus Model of Skill Acquisition (https://en.wikipedia.org/wiki/Dreyfus_model_of_skill_acquisition) applied to Software Development (http://blog.codinghorror.com/level-5-means-never-having-to-say-youre-sorry/)

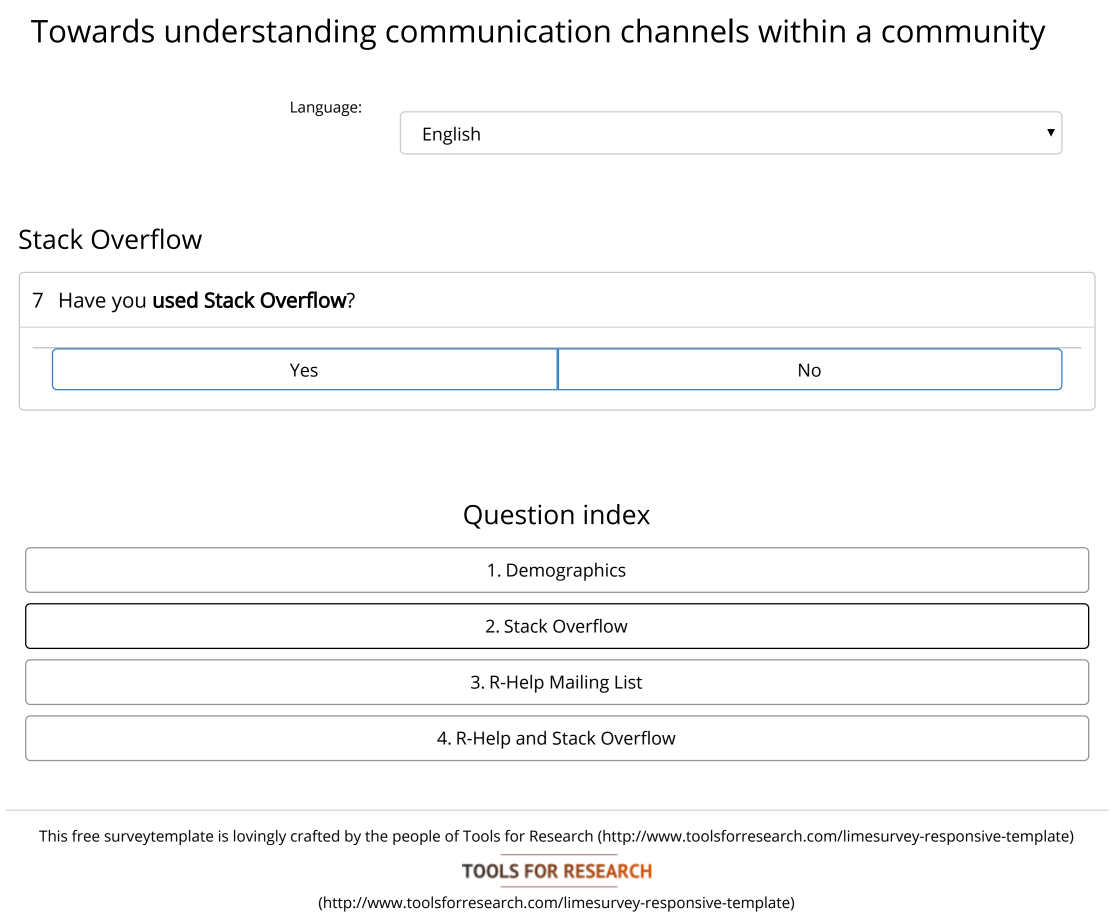
6 How would you describe your **participation** in the **R community**?

Check all that apply

- I'm just an R user
- I contribute to the R documentation
- I'm one of the R core developers
- I write or maintain R packages
- I submit R bugs
- I am not Involved at all
- Other:

Question index

1. Demographics



Towards understanding communication channels within a community

Language: English

Stack Overflow

7 Have you **used Stack Overflow**?

Yes No

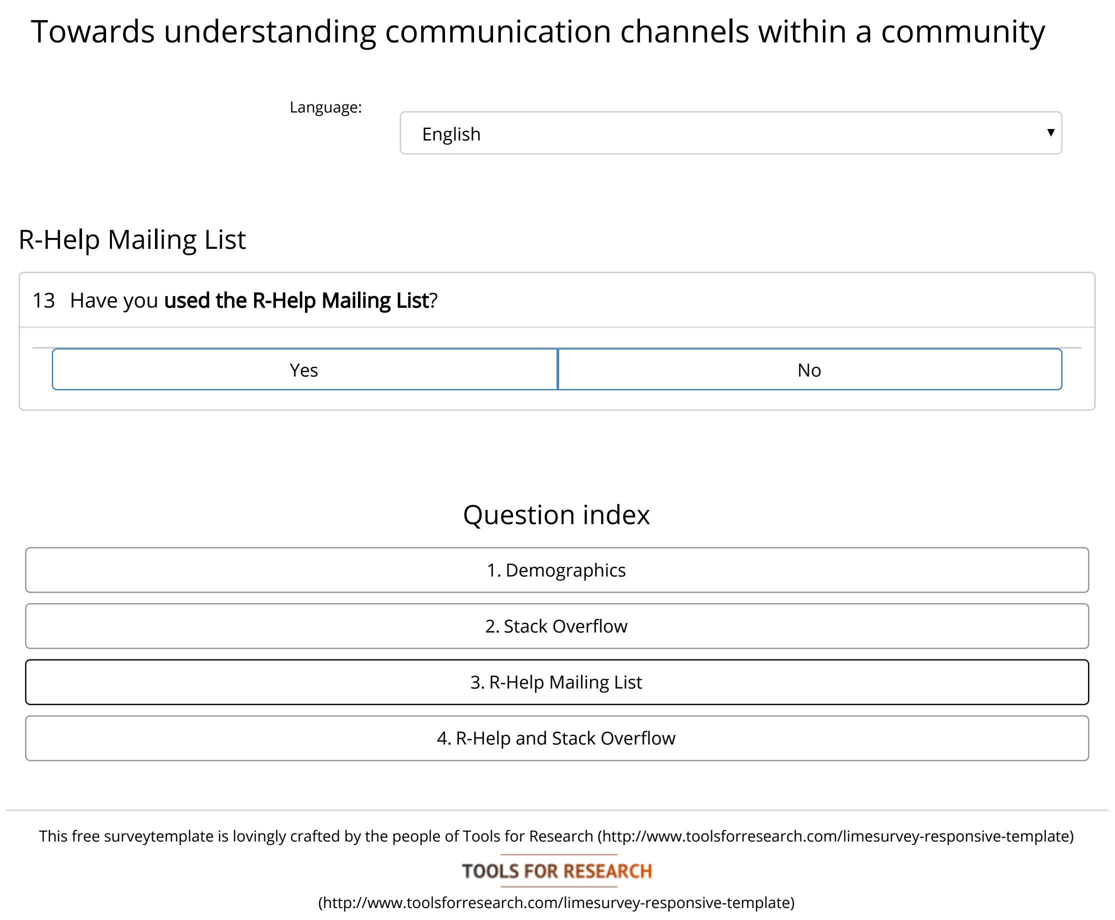
Question index

1. Demographics
2. Stack Overflow
3. R-Help Mailing List
4. R-Help and Stack Overflow

This free surveytemplate is lovingly crafted by the people of Tools for Research (http://www.toolsforresearch.com/limesurvey-responsive-template)

TOOLS FOR RESEARCH

(http://www.toolsforresearch.com/limesurvey-responsive-template)



Towards understanding communication channels within a community

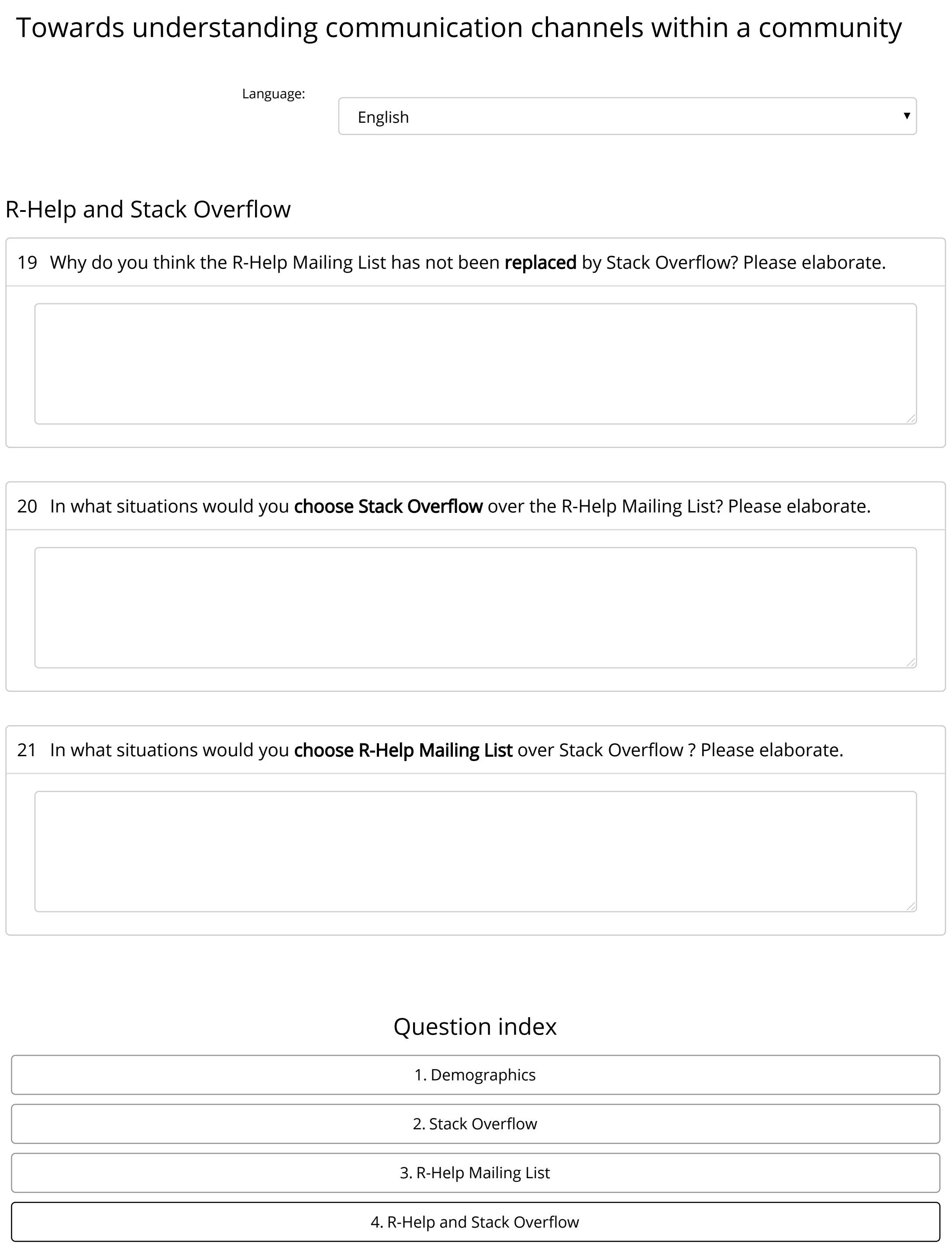
Language: English ▾

R-Help Mailing List

13 Have you **used the R-Help Mailing List**?

Yes No

Question index



Towards understanding communication channels within a community

Language: English ▾

R-Help and Stack Overflow

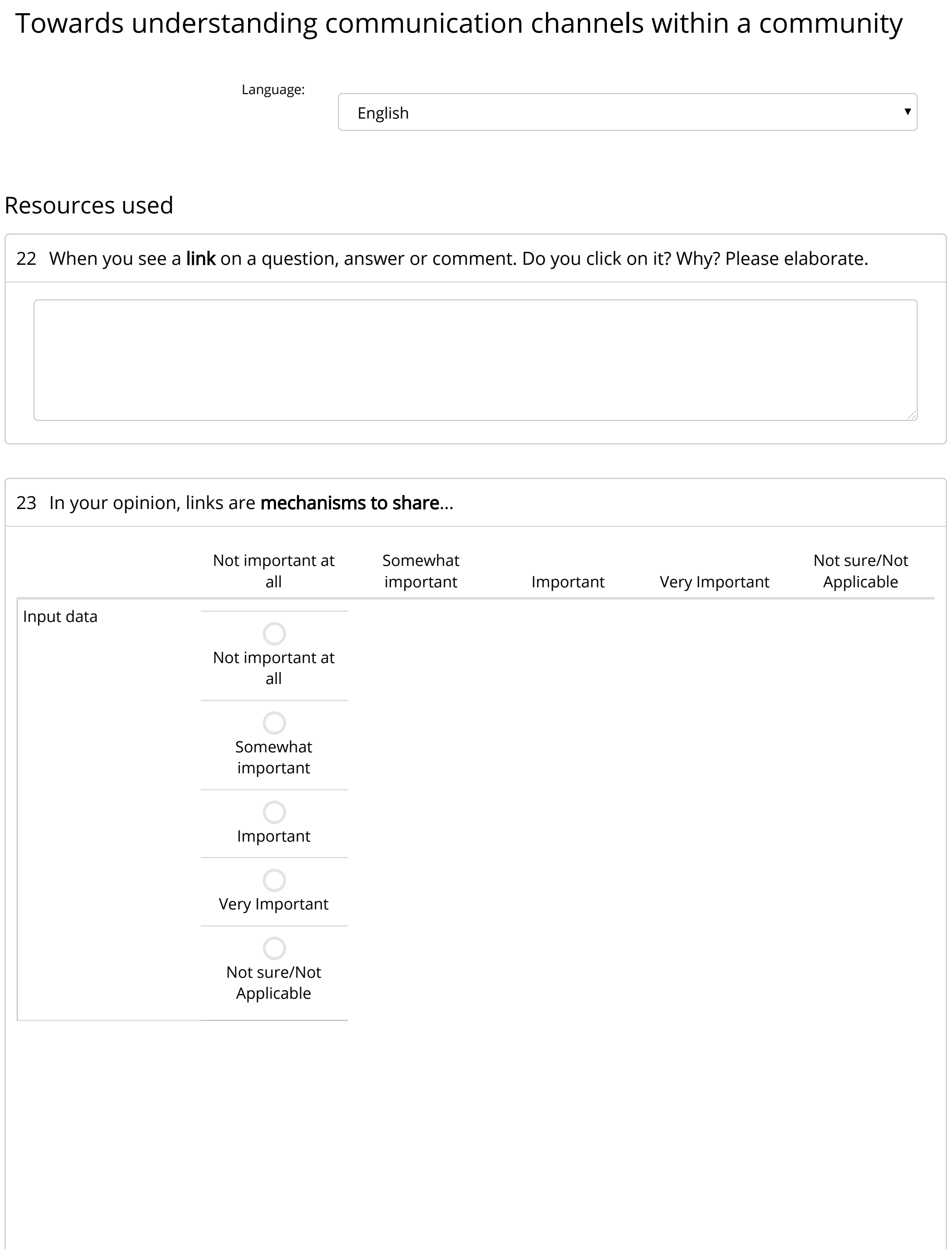
19 Why do you think the R-Help Mailing List has not been **replaced** by Stack Overflow? Please elaborate.

20 In what situations would you **choose Stack Overflow** over the R-Help Mailing List? Please elaborate.

21 In what situations would you **choose R-Help Mailing List** over Stack Overflow ? Please elaborate.

Question index

1. Demographics
2. Stack Overflow
3. R-Help Mailing List
4. R-Help and Stack Overflow



Towards understanding communication channels within a community

Language: English ▾

Resources used

22 When you see a **link** on a question, answer or comment. Do you click on it? Why? Please elaborate.

23 In your opinion, links are **mechanisms to share**…

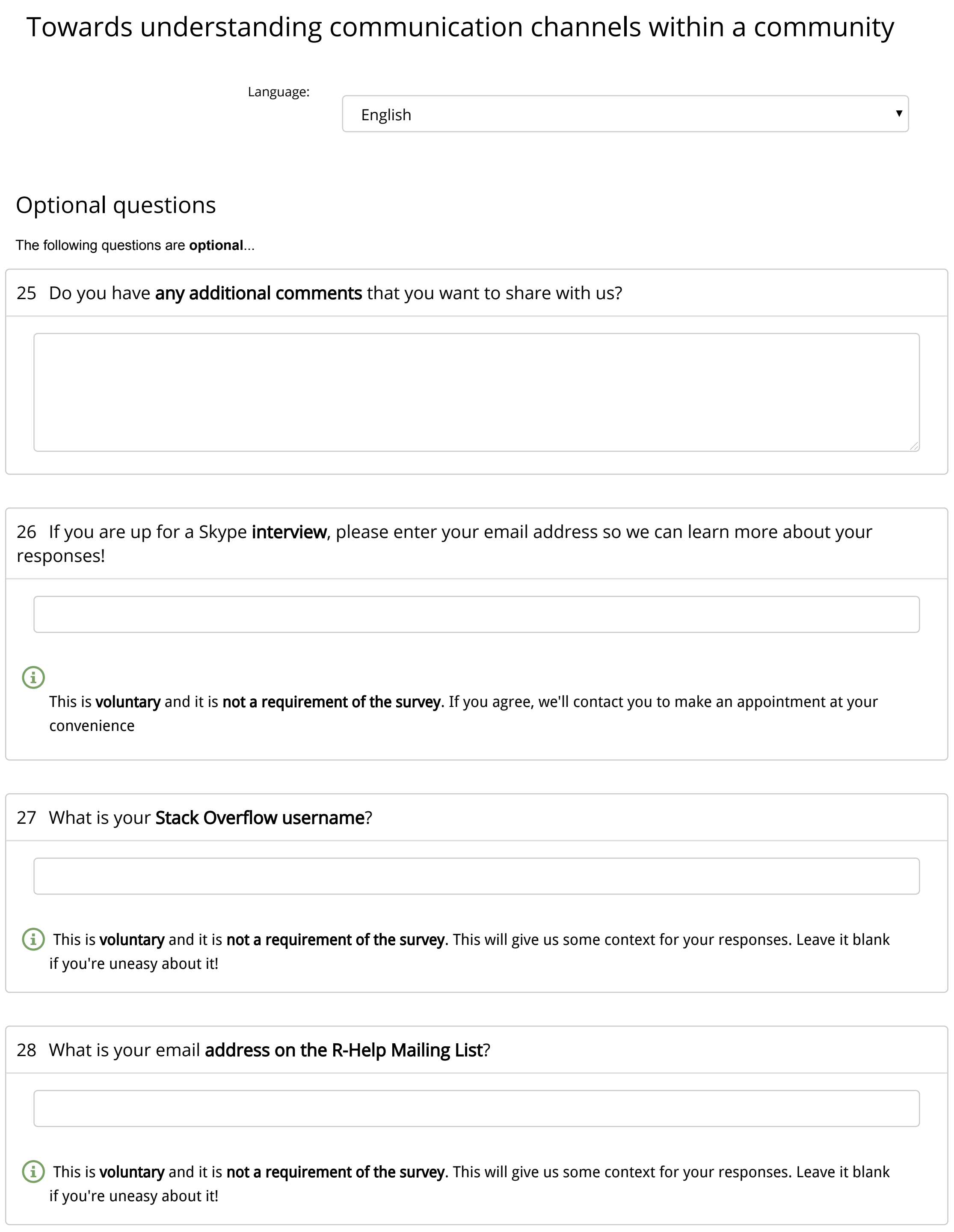
	Not important at all	Somewhat important	Important	Very Important	Not sure/Not Applicable
Input data	○ Not important at all ○ Somewhat important ○ Important ○ Very Important ○ Not sure/Not Applicable				

	Not important at all	Somewhat important	Important	Very Important	Not sure/Not Applicable
Source code	○ Not important at all				
	○ Somewhat important				
	○ Important				
	○ Very Important				
	○ Not sure/Not Applicable				
Documentation	○ Not important at all				
	○ Somewhat important				
	○ Important				
	○ Very Important				
	○ Not sure/Not Applicable				

	Not important at all	Somewhat important	Important	Very Important	Not sure/Not Applicable
External libraries	○ Not important at all ○ Somewhat important ○ Important ○ Very Important ○ Not sure/Not Applicable				
Authors/users	○ Not important at all ○ Somewhat important ○ Important ○ Very Important ○ Not sure/Not Applicable				

	Not important at all	Somewhat important	Important	Very Important	Not sure/Not Applicable
Publicity	○ Not important at all ○ Somewhat important ○ Important ○ Very Important ○ Not sure/Not Applicable				
Apps	○ Not important at all ○ Somewhat important ○ Important ○ Very Important ○ Not sure/Not Applicable				

24 Within the context of Stack Overflow and the R-Help Mailing List, can you think of any **other benefits** of using links? Please elaborate.



Towards understanding communication channels within a community

Language: English ▾

Optional questions

The following questions are **optional**...

25 Do you have **any additional comments** that you want to share with us?

26 If you are up for a Skype **interview**, please enter your email address so we can learn more about your responses!

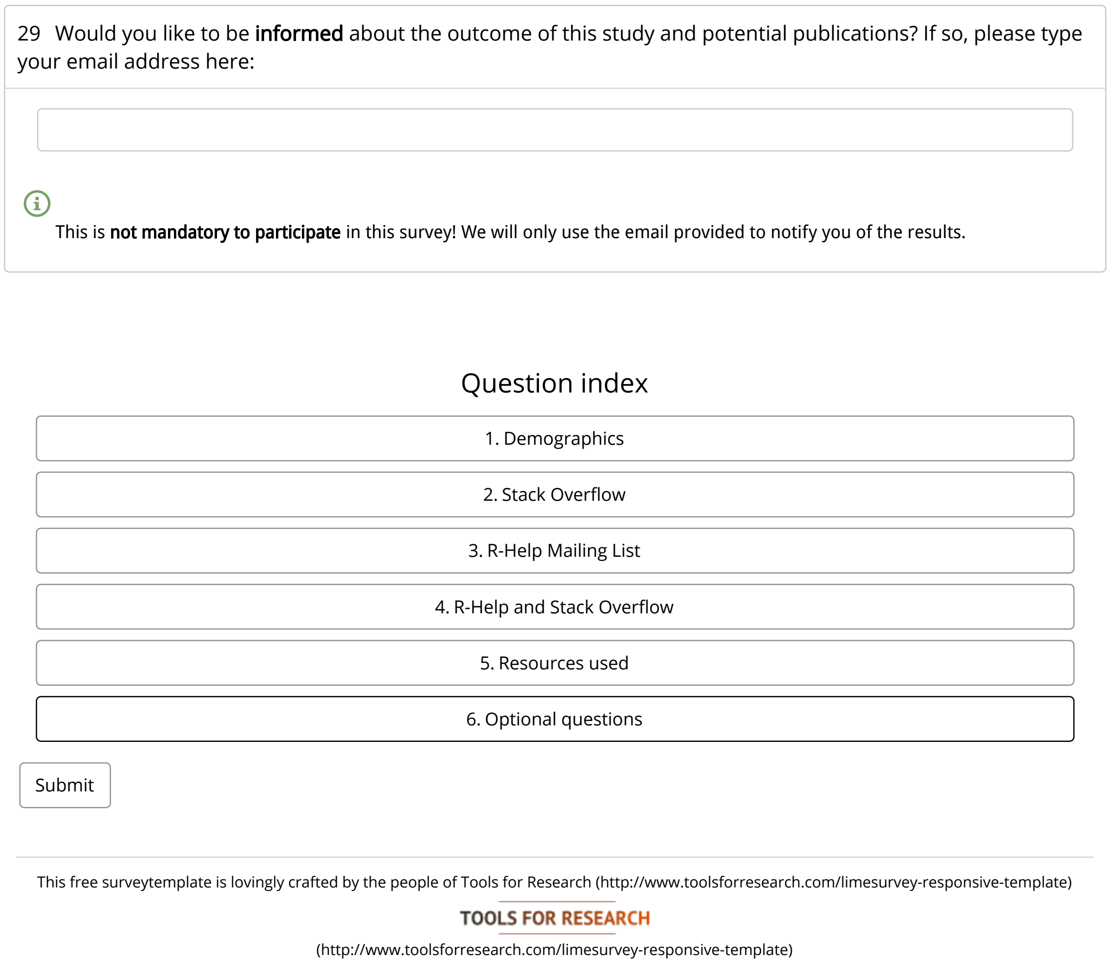
This is **voluntary** and it is **not a requirement of the survey**. If you agree, we'll contact you to make an appointment at your convenience

27 What is your **Stack Overflow username**?

This is **voluntary** and it is **not a requirement of the survey**. This will give us some context for your responses. Leave it blank if you're uneasy about it!

28 What is your email **address on the R-Help Mailing List**?

This is **voluntary** and it is **not a requirement of the survey**. This will give us some context for your responses. Leave it blank if you're uneasy about it!



29 Would you like to be **informed** about the outcome of this study and potential publications? If so, please type your email address here:

This is **not mandatory to participate** in this survey! We will only use the email provided to notify you of the results.

Question index

1. Demographics
2. Stack Overflow
3. R-Help Mailing List
4. R-Help and Stack Overflow
5. Resources used
6. Optional questions

Submit

This free surveytemplate is lovingly crafted by the people of Tools for Research (http://www.toolsforresearch.com/limesurvey-responsive-template)

TOOLS FOR RESEARCH

(http://www.toolsforresearch.com/limesurvey-responsive-template)

Appendix C

A Model of Knowledge in Software Development: Evolving Over Time

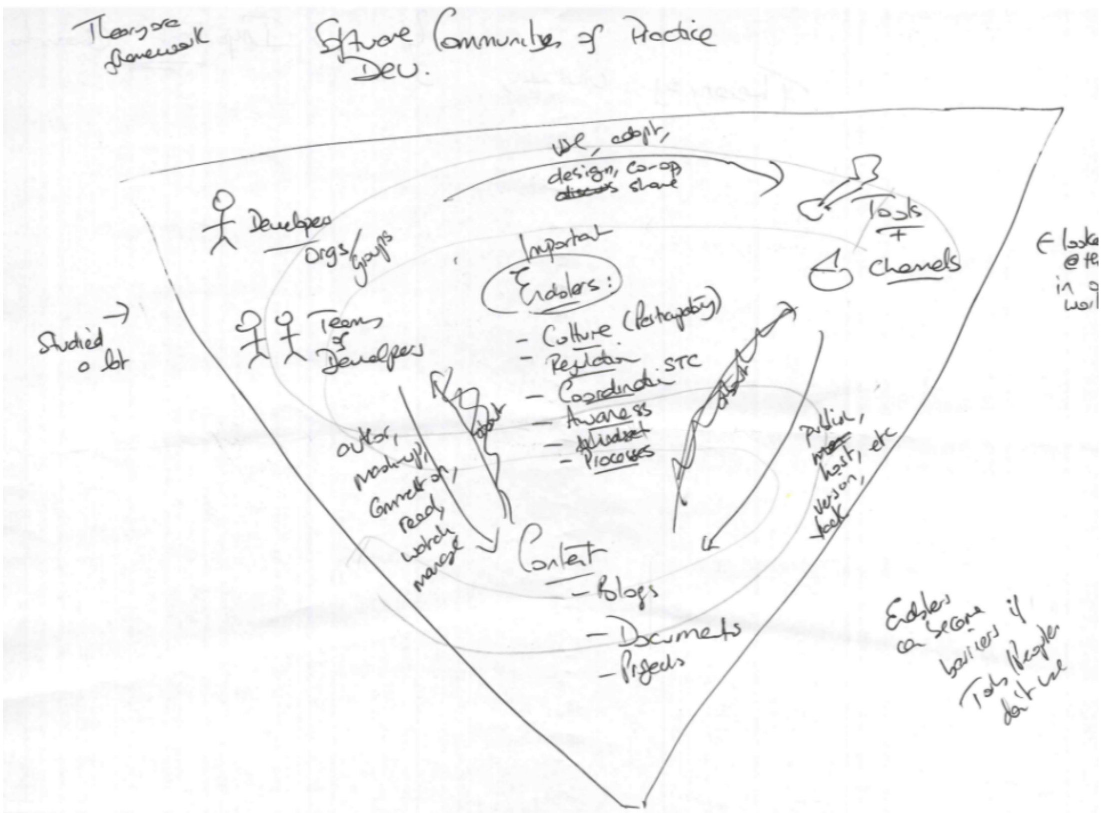


Figure C.1: An early mental model of how we envisioned the software development ecosystem and the role of communities of practice.

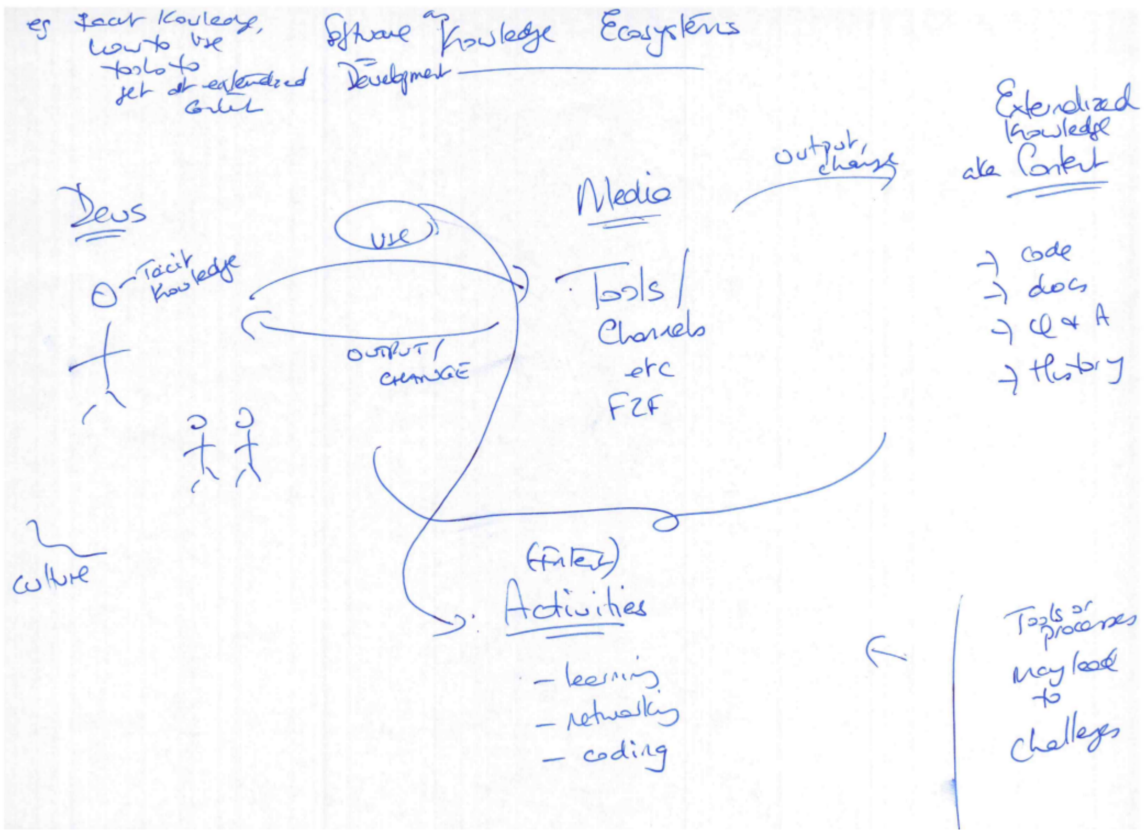

eg tacit knowledge, how to use tools to get at externalized content
Software as Knowledge Ecosystems
Development
Externalized knowledge aka Content
output, change
Media
Devs
use
Tacit knowledge
Tools / Channels etc F2F
OUTPUT / CHANGE
code
docs
Q & A
history
culture
(filter)
Activities
– learning
– networking
– coding
Tools or processes may lead to challenges

Figure C.2: A more mature mental model of the developer knowledge ecosystem.

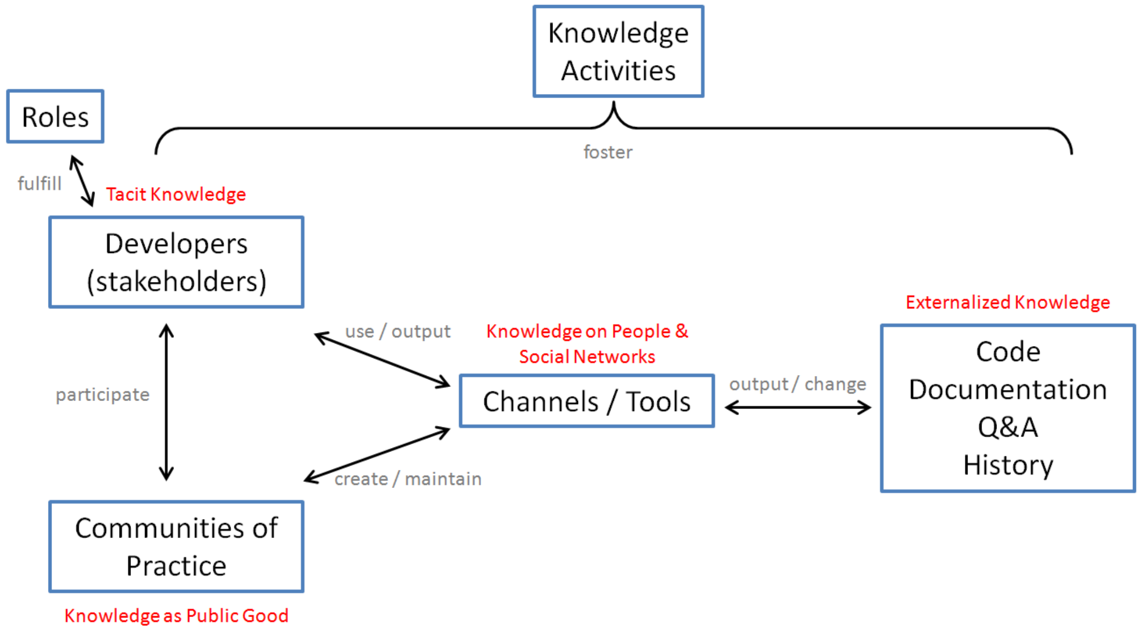


Figure C.3: The knowledge model I used for my candidacy exam (December 2, 2015).