UNDERSTANDING ONLINE INNOVATION COMMUNITY: AN INVESTIGATION OF GROUP DIVERSITY IN OPEN COLLABORATION

WANG ZHIYI
(*B. Eng., Renmin University of China*)

A THESIS SUBMITTED
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
DEPARTMENT OF INFORMATION SYSTEMS AND ANALYTICS
NATIONAL UNIVERSITY OF SINGAPORE

2018

Supervisor:
Associate Professor Hahn Jungpil

Examiners:
Associate Professor Goh Khim Yong
Associate Professor Tan Chuan Hoo
Associate Professor Steven L. Johnson, University of Virginia

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Wang Zhiyi

30 July 2018

ACKNOWLEDGEMENTS

This thesis would not be finished without the help and support from many individuals. Here I would like to mention some of them particularly for their advice and influences during my journey of PhD.

First, I would like to express my gratitude to my advisor, Professor Jungpil Hahn, who guided me to complete this thesis and develop research capabilities. In the past five years, he devoted a significant amount of time to and provided insightful suggestions for my research. He is a great advisor and friend in my academic life. I learned from him not only the rigor of conducting high quality research but also the importance of keeping my belief in the academic journey.

I would also like to appreciate the help from my thesis committee members, Professor Khim Yong Goh, Professor Chuan Hoo Tan and Professor Steven Johnson. They provided useful guidelines and suggestions during the development of my dissertation, as well as valuable feedbacks on my academic career.

I am also grateful to my colleague, Dr. Lusi Yang, who always has inspirational thoughts and determined commitment. I would like to thank her for her continuous encouragement and unconditional support in my research development and academic life. She is also an excellent collaborator to me with great dedication to research and deep thinking on problems, which helped me become a better scholar.

I also benefited a lot from my friends, colleagues, and faculty members in the Department of Information Systems and Analytics. In particular, I would like to thank Professor Atreyi Kankanhalli, Professor Cheng Suang Heng and Professor Yuanyuan Chen for their guidance on research methodologies and my thesis development. I also want to thank my colleagues and faculty members in my department to give me suggestions on academic career and provide feedbacks in research seminars.

Last but not least, I want to thank my family members for their unconditional loves, especially my parents Jianyu Wang and Mei Xie, for their everlasting encouragement, patience and understanding. They guided me for being a nice person and cultivated my habit of reading and thinking. I would not achieve my current status without them.

SUMMARY

The emergence of online innovation communities has provided a new business model to break the boundaries of innovation in organizations. Although these communities have created significant values for organizations and society, effective management of such communities is still challenging for companies and platform operators. Specifically, it is important to explore how to facilitate value co-creation as well as group efficiency in the open form collaboration. This dissertation seeks to examine open collaboration management and group formation in innovation communities to extend the literature on the effectiveness and efficiency in open innovation communities. Drawing on the group diversity perspective, two essays are incorporated to understand the organization of diverse individuals in online innovation collectives.

The first essay titled "Organizing the Online Crowds: Diversified Experience and Collective Performance in Crowdsourced New Product Development" investigates the role of knowledge variety in crowdsourced new product development. From the knowledge diversity and creativity perspectives, I develop a research model to understand: 1) different types of crowd members and 2) the value contributions to collective crowd performance from different member types. Using data from 425 crowdsourced product development campaigns, I empirically find that both diverse knowledge and specialized knowledge are important for the collective performance of the crowd. In addition, generalists may not be valued in the online new product development context.

The second essay titled "Can I Touch Your Code? The Effects of Programming Style on Open Source Collaboration" focuses on the management of individuals with diverse work styles in open source software development. Drawing on the literature in software engineering and group diversity, I develop hypotheses on the effects of

programming style on open source collaboration and development, as well as the factors that can shape the effects of programming style. I develop comprehensive measures to quantify programming style inconsistency at multiple levels and test these hypotheses using empirical data and source code from a prominent open source community. With large scaled static code analysis and econometric analysis, I find that style inconsistency exhibits negative effects through within file inconsistency on contribution activities rather than other collaboration outcomes. The negative effects are mitigated by team familiarity but unexpectedly intensified by developer experience. In addition, the enactment of coding standards to control programming style can only reduce style inconsistency within files but style inconsistency across files.

Overall, my dissertation takes a group diversity perspective to examine the effective management of open innovation communities. The open collaboration process, although can increase the reach to innovators, engenders uncertainties on creating innovations and organizing the groups. By focusing on the diversity of knowledge and work style in these groups, this dissertation seeks to explore the ways to better form online groups in the open collaboration process for value co-creation and collaboration efficiency. It provides implications on the management of open innovation, online group dynamics and group diversity.

Keywords: innovation communities, open innovation, crowdsourcing, new product development, open source software, open collaboration, group diversity, knowledge variety, work style, programming style, software engineering, econometric analysis, cluster analysis, static code analysis, propensity score matching, difference-in-difference

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 Research Background

Innovation has become one of the most important competencies for organizations in recent years. The creation of new products not only provides significant economic value for organizations but also cultivates their capability for sustainable development (Ahlstrom 2010). Organizations recognize that continued investment in innovation is important for long-term firm survival. It is both a capitalization process of existing knowledge as well as an exploration process for new business opportunities (March 1991). Recent trends in innovation and entrepreneurship also highlight the plentiful opportunities and rewards for creating new solutions for the market.

However, organizations still face numerous formidable challenges with respect to innovation and not all organizations ultimately benefit from their costly innovation efforts (Lichtenthaler 2011). Traditional R&D activities are typically limited to within the boundaries of an organization such that the number of experts and the breadth of their perspectives may be constrained for substantial innovation. In addition, firms may not fully understand what (external) consumers/users really want if they only focus on (internal) R&D. New products may turn out not to be successful in the market despite large amounts of investments. These challenges make pursuing an innovation strategy uncertain and risky for organizations.

In recent years, the emergence of new web technologies and platform-based ecosystems have enabled novel ways for creating innovations – via open innovation communities. Open innovation communities break the traditional organizational boundary of innovation and democratize innovation activities by empowering regular individuals to participate in the innovation process. It affords the opportunity of creating innovations available for anyone in the world, which benefits organizations (West and Lakhani 2008). For example, the Linux Operating System community consists of developers from all around the world and the system itself has been

continuously developed for over 20 years. It has even been adopted and is being used by many firms for their mission critical business activities. Meanwhile, organizations have also built such communities (i.e., firm-sponsored open innovation communities) to extract value from users outside of the organizational boundary (Lettl et al. 2006). For instance, Dell created IdeaStorm in 2007 to collect useful solutions from regular users to better satisfy its customers and the market. The "wisdom of crowds" has been significantly exploited by these Internet-based innovation communities (Surowiecki 2004).

Given the great potential for economic value creation from open innovation communities, researchers and practitioners alike have exerted significant efforts to understand how these communities operate and have experimented with new community models (Chesbrough and Appleyard 2007; Di Gangi et al. 2010; Nambisan and Baron 2010). Nevertheless, the effective management of these communities and the overall ecosystem is still challenging for platform stakeholders. Different incentives may induce unintended strategic behaviors, uneven quality of crowd contributions, which ultimately may lead to inefficient management of the innovation process. This is because innovation communities usually operate in an open form collaboration model, where participation barriers are low and control mechanisms are weak (Ren et al. 2015). Under such conditions, individual contributors could freely choose what to contribute and how they contribute, which may lead to undesired outcomes for both innovation effectiveness and coordination efficiency. Such individuals with different knowledge, although enrich the available ideas and perspectives in the group, can lead to diverse group composition, making the creation of innovations uncertain. Thus, it is important to understand how to effectively manage the innovation activities and diverse individuals in these communities.

My dissertation seeks to better understand open collaboration in innovation communities. It consists of two essays focusing on crowdsourced new product

development communities and open source software communities, respectively. In these communities, individuals with diverse backgrounds can voluntarily participate in innovation activities. To explore effective open collaboration management, I draw on the group diversity perspective to understand how diverse individuals can be formed into open collaboration groups to create value for innovation and advance collaboration efficiency (Harrison et al. 1998; van Knippenberg and Schippers 2007; Williams and O'Reilly 1998). I also attempt to examine group diversity by differentiating the role of subgroup members and exploring work style preference diversity based on the nature of the product. Specifically, the first essay examines *knowledge variety* in large online groups in crowdsourced new product development. It aims to understand the open collaboration process among individuals with diverse knowledge and how different member groups in the crowd provide value contributions for product development. The second essay focuses on open collaboration in open source software development communities, where strict control on an individual's work style preference is the exception rather than the norm. It draws upon the notion of "*diversity as separation*" and studies the role of programming style in open source collaboration. It intends to explore a specific type of diversity – diversity in work style preferences, which is typically not observed in traditional contexts or examined from studies following the behavioral perspective, to extend the current literature on group diversity.

1.2 Research Questions

Online innovation communities are characterized by open collaboration, diverse individuals and innovation-based tasks (Bogers et al. 2010). In crowdsourcing-based innovation communities, participants can accumulate their experiences in different tasks and develop their own knowledge portfolios (Huang et al. 2012). Thus, individuals in a large online group for new product development can be characterized by their experiences. How individuals with different experiences across knowledge

domains are brought together may influence the overall innovation performance of the crowd. Therefore, the first research question of this dissertation is:

How do knowledge distributions of crowd members impact the collective innovation performance of crowds?

Having diverse individuals not only provides a variety of perspectives, but also characterize online groups with different member compositions. A group with all members having similar knowledge portfolios may not perform well in open collaboration (Harrison and Klein 2007). Thus, differences in group composition may also impact performance. Meanwhile, one group of members may strengthen or weaken the effect of another group of members. This highlights the importance of the composition of crowd members in open collaboration. Hence, the second research question is:

How does the composition of crowd members affect collective crowd performance?

In online innovation communities, the open collaboration process typically lacks strict control mechanisms compared with in-house closed-form collaboration. Thus, diversity of work style preferences will be amplified in open collaboration communities, especially when the work style is embedded in the product itself. In open source innovation communities, programming style is such an example. It is usually strictly controlled in proprietary software development using coding standards but such controls are not common in open source development and individuals resort to using their own work (programming) styles. This kind of individual preference may lower effectiveness and efficiency in collaboration and development. Therefore, the third research question in my dissertation asks:

How does work style preference in open source development affect open source collaboration?

In addition to the direct impact of programming style on open source collaboration, it is important to be aware of the possible solutions to alleviate the

issues arising from different programming styles. Given the potential negative effects of separation on group performance and the extra efforts to comprehend source code with inconsistent programming styles, the factors (i.e., group formation mechanisms) that can moderate the negative consequences and mitigate the issues arising from coding style differences are important for managing different individual styles in open innovation communities. Thus, the last research question is:

What factors may resolve the issues arising from programming style diversity in open source collaboration?

In summary, these research questions examine open collaboration in innovation communities from a group diversity perspective. To resolve the challenges on the uncertainty of *what* are contributed and *how* contributions are made in open innovation community due to diverse individual participation, I focus on two aspects of diversity – knowledge diversity for the "what" aspect (quality) and work style diversity for the "how" aspect (coordination). Therefore, the first two research questions pay attention to knowledge variety in large scale online collaboration groups. The knowledge of individual participants could affect what kind of contributions are made in innovation communities, which help to resolve the challenge of contribution quality. The other two research questions examine the work style diversity in open source collaboration. Individual style of participation influences how the contributions look like and has important implications for the challenge of coordination. Overall, answering these questions help us to further understand the broader phenomena in open innovation communities and ecosystems.

1.3 Research Implications

My dissertation offers several implications for research and practice. First, the two essays extend the literature on online innovation communities by providing insights into the nature of open collaboration in crowd-based innovation activities. They examine the research context with open collaboration elements for creating

innovative products but with different organizing mechanisms. The research models have the potential to enrich our knowledge of the management of innovation communities. Second, the research design and empirical method in my dissertation capture the dynamics of group formation in innovation communities. Given the low barriers of entry and exit and the nature of voluntary contribution in open collaboration, the fluidity and dynamics of online groups are important characteristics to be considered in designing research models and conducting empirical analysis (Ransbotham and Kane 2011; Ren et al. 2015). This dissertation, therefore, extends the literature by generating research implications for understanding the dynamic nature of open collaboration collectives. Third, this dissertation extends the boundaries of the group diversity literature by examining knowledge variety in large scale online groups and work style diversity in open collaboration. It seeks to understand the role of diversity in more complicated contexts and in the nature of the product.

For practical implications, this dissertation offers insights into group formation in online innovation communities. Firms and teams can learn from the research findings to build effective online work groups for product innovations and efficiently organize diverse individuals for innovation. Knowledge variety, work style, team familiarity and member experience can be used as important characteristics for group formation and governance in innovation communities.

1.4 Dissertation Structure

My dissertation examines open collaboration in innovation communities from group diversity perspective. It includes two independent essays that study knowledge diversity in a crowdsourced new product development community and work style diversity in an open source software community, respectively. The current chapter provides the research background, presents the research questions and outlines the high-level contributions. Chapter 2 introduces the research context – online

innovation communities. I review and summarize and discuss the key characteristics, common types of organizing, and the existing research streams related to innovation communities. Chapter 3 presents the theoretical background on group diversity, including the definition and mechanisms of groups diversity, a theoretical typology of diversity, and a discussion of diversity in online groups. Chapters 4 and 5 are the two independent essays. As standalone essays, each chapter includes a focused literature review, theory and hypotheses development, a discussion of the specific study context, the empirical method and results, followed by discussions and a conclusion. Specifically, chapter 4 is the first essay about knowledge variety in crowdsourced new product development. It includes the literature review on crowdsourcing and generalist vs. specialist framework, a typology of crowd members, hypotheses development on value contributions of crowd members, study context, empirical method, results and discussions, and conclusions. Chapter 5 is the second essay about work style diversity in open source software community. It presents the literature review on open source community and programming style in software engineering, research hypotheses on the main effects of programming styles and factors that shape the effects, research context, empirical methods including the measure of programming style and econometric models, results and conclusions. Finally, I conclude my dissertation in Chapter 6 with a summary of the essays and discussion of contributions.

CHAPTER 2 RESEARCH CONTEXT: ONLINE INNOVATION COMMUNITIES

An online innovation community (or simply, innovation community) is a form of online community whose purpose is to create innovations. An innovation community can be defined as "a voluntary association of actors who lack a common organizational affiliation but share the common instrumental goal to create innovations" (Gläser 2001; West and Lakhani 2008, pp. 224). In such communities, innovative users from anywhere in the world are able to contribute their talents to various innovations, such as new products for organizations (Bayus 2013; Di Gangi and Wasko 2009), extensions or modifications to existing products (Arakji and Lang 2007; Zhang et al. 2013), and even original products that can be consumed by others (Feller et al. 2008). These communities have created great economic value for companies and the public since its emergence and researchers in Information Systems have devoted significant efforts to better understand how these communities function and create value. In this chapter, I summarize the key characteristics of innovation communities and review the important extant research streams. Finally, I discuss the focus of this dissertation in terms of the research context and how the two essays in this dissertation attempt to add to the literature.

2.1 Characteristics of Innovation Communities

The definition of online innovation community suggests that such communities follow the nature of online communities and the purpose of these communities are for creating innovations (usually open innovation) (West and Lakhani 2008). Up to now, several salient characteristics of online innovation communities have been observed, practiced and documented by both researchers and practitioners. These characteristics not only exhibit the uniqueness of innovation communities, but also suggest important research directions in examining this phenomenon.

IT-mediated and Geographically Distributed. Online innovation communities typically operate on the Internet such that users participating in the communities may come from anywhere in the world (Gläser 2001). On the one hand, this makes innovations more accessible by gathering innovators from distant places. Innovative users have more opportunities to contribute or collaborate through the Internet and which can consequently increase the likelihood that a greater number of innovations may be created. On the other hand, geographical separation and cultural difference can also present challenges to effective communication, collaboration and evaluation of innovation (Daniel et al. 2013). Time zone differences, diverse culture backgrounds (e.g., different spoken languages) and a lack of face-to-face interaction may undermine the effectiveness of such communities (Kankanhalli et al. 2006).

The Wisdom of Crowds. It has been realized that the promise of innovation communities is largely characterized by the "wisdom of the crowd" (Boudreau and Lakhani 2013). Users who may not be experts are able to view problems from novel perspectives so that the crowd can create tremendous value for organizations or society (Mannes 2009). Many practices have tried to utilize the wisdom from the crowd to solve a variety of problems and innovation communities provide ideal Internet-based platforms to sustain such practices. In summary, the diversity of knowledge from a heterogeneous crowd can help to generate novel ideas for creating innovations.

Voluntary Participation. Similar to other online communities, innovation communities are also characterized by voluntary participation of members (Bagozzi and Dholakia 2006). Users or members in these communities are not forced to work and they can freely choose what and when to contribute. Although this is common in other online communities (e.g., knowledge sharing communities such as Quora), it is different with innovation tasks in online labor markets (which is contract-based) or in offline innovation contexts. The combination of innovation activity and voluntary participation gives users the freedom to explore and devote their knowledge, but also

raises concerns related to task regulation and member commitment (Crowston et al. 2007).

Knowledge Intensive Work. Different from general online communities (e.g., discussion forums, Q&A communities and healthcare communities), innovation communities are typically characterized by knowledge-intensive tasks that are for creating innovations (Chesbrough and Appleyard 2007). Users in these communities not only share their knowledge, but also use their knowledge to generate new ideas. The context of innovation usually requires community members to create new things that may have value for organizations or the public.

2.2 Types of Innovation Communities

In addition to the aforementioned common characteristics, online innovation communities can be heterogeneous in terms of their mechanisms of operation. In general, how the community members are organized for innovation creation can be categorized into three ways: firmed-oriented, two-sided and self-organized.

Firm-oriented Communities. This type of innovation community is usually built and led by a firm for its own innovation initiatives. These communities are commonly called "user innovation communities" (Bogers et al. 2010; Füller et al. 2006). Firms create communities to organize for innovations beyond their boundaries. The outcomes of community innovation (e.g., ideas, solutions and products) are usually utilized by the firm for their own business (Di Gangi and Wasko 2009). In such communities, participation is not limited to community members and firms also communicate with users in the community and organize them for innovations (Greer and Lei 2012). For example, in Dell's IdeaStorm community, users can contribute their new product ideas and Dell will select ideas that are proposed by the community and also in line with Dell's own business strategy (Di Gangi et al. 2010; Huang et al. 2014). By operating such communities, firms are able to gather a group of individuals outside of their boundaries but with diverse perspectives to facilitate their new

product and service innovations. These community members not only serve as innovators to contribute ideas but also as product users who understand market requirements (Bogers et al. 2010; Lettl et al. 2006). The major challenge in these communities is how to select ideas from community members. Contributors in these communities may have strong expectations of being selected or rewarded, and sometimes the crowd evaluation of ideas is quite different from expert panel evaluation in the firm (Liu et al. 2018). It is important for the firm to balance the participants' motivation and the selection of ideas for implementation.

Two-sided Communities. This type of online innovation community is operated by third-party platform owners with two group of users – innovation seekers and innovation contributors. Seekers in the community post innovation tasks and contributors are tasked to provide solutions (Yang et al. 2009). Some crowdsourcing communities, such as TopCoder, adopt this type of community model where programming challenges are posted by innovation seekers and solved by contributors. Different from the first type (i.e., firm-oriented communities), innovation seekers in two-sided communities do not have to be firms but can be individuals who seek new solutions for their personal use. Moreover, the innovation task is the glue that connects seekers with contributors (Estelles-Arolas and Gonzalez-Ladron-de-Guevara 2012). Innovation activities in these communities are mostly task-based and innovative users only submit their ideas or solutions to specific tasks. The major challenge in this type of community is about the choice of task and strategic behaviors under such a competitive environment. Users face the trade-off between performing diverse tasks to acquire new experiences and the probability of winning the reward. Strategic users may choose the timing of participation and difficulty of the task to increase the chance of winning, which in turn may harm the community in the long run (Huang et al. 2014; Yang et al. 2010). Therefore, it is necessary to understand how to motivate users to perform more tasks and reduce the strategic behaviors.

Self-organized Communities. In the previous two types, innovative users in the communities are organized by specific parties, i.e., the firms or the seekers. However, innovation communities can also self-organize to create innovations – these constitute the third type – self-organized communities. Users in such communities start innovation activities by themselves. The most exemplary case of self-organized communities is the open source software development community, where developers work together to make innovative software. In the Linux OS community, the development of the system is not for any firm or individuals but is to create a novel and useful operating systems (Bagozzi and Dholakia 2006; Lee and Cole 2003). In such communities, users can freely create their innovations and the innovation outcomes can be consumed by a broader group, such as the public or those with interests (von Hippel 2005). The key challenge in the self-organized community is how to effectively form the community and organize the members, as there is no explicit party to control the innovation activities. Community members are expected to build their own logics on the collaboration, communication and governance (Bagozzi and Dholakia 2006).

2.3 Research on Innovation Communities

Given the key characteristics and various types of online innovation communities, researchers have put in significant efforts to understand the phenomenon of online innovation communities. Besides the research challenges in each type of innovation communities, there are several streams of research examining different behavior and community mechanisms across these community types. Specifically, existing research on innovation communities mainly focus on the following aspects: individual's motivation to contribute, antecedents of contribution behaviors, effective team management and collaboration, and the effective design of innovation communities.

Individual Motivation. An important stream of research in online communities and innovation communities is to understand why people contribute to these communities

when oftentimes there are no direct monetary rewards. The existing literature in innovation communities has suggested different types of motivations – intrinsic motivation, extrinsic motivation and internalized extrinsic motivation (Roberts et al. 2006; von Krogh et al. 2012). *Intrinsic motivation* is usually about the enjoyment of participating in the communities, which has been shown as an important factor on developer's participation in open source communities (Ke and Zhang 2009; Zhang et al. 2013) and crowdsourcing communities (Zheng et al. 2011). *Extrinsic motivation* is about the monetary rewards and career benefits from contributions, and is usually regarded as the dominating motivation when economic incentive is provided (Hann et al. 2013; Roberts et al. 2006; Yang et al. 2010). In addition, related studies have shown that *internalized extrinsic motivations*, which benefit individuals in non-monetary ways, have strong implications on user's participation. It has been examined that garnering reputation or identity (Fang and Neufeld 2009; Roberts et al. 2006; Zheng et al. 2011) and learning new knowledge (Fang and Neufeld 2009; Hars and Ou 2002; Lakhani and Von Hippel 2003) are important motivations for participating in different types of innovation communities.

Contribution Behavior. Another stream of research in innovation communities examines the factors beyond motivation that affect a user's contribution and the quality of their contributions. Two major aspects have been widely discussed in related works: the social network and experience. A contributor's social network has been documented as an important antecedent of participation and contribution (Hahn et al. 2008; Moqri et al. 2015; Oh and Jeon 2007). Through social influences and social connections, individuals will make more contributions and improve the quality of their works. Moreover, an individual's experience can have salient impacts on his/her contributions. Learning-by-doing has been broadly investigated in research on crowdsourcing communities (Archak and Ghose 2010; Huang et al. 2012) and open source software development (Singh et al. 2010). However, some unexpected effects of experience in the innovation context are also present, such as cognitive fixation

(i.e., successful experiences constrain the creativity of subsequent contributions) (Bayus 2013) and strategic behaviors (i.e., contributors tend to maximize their probability of winning contests so that they strategically participate in crowdsourcing to get higher chances of being rewarded) (Huang et al. 2014; Yang et al. 2008).

Team Collaboration. In addition to individual behaviors in innovation communities, researchers in this area also focus on group level phenomena, especially team collaboration in such communities. The majority of research focuses on this aspect in self-organized communities. Existing studies on open source communities have examined various dimensions of open source team collaboration including social capital (Singh et al. 2011), network embeddedness (Grewal et al. 2006), governance and coordination (Blincoe and Damian 2015; Shah 2006), team ideology (Stewart and Gosain 2006), and team diversity (Daniel et al. 2013). In addition, studies in crowdsourcing and open innovation communities (i.e., the first two types of innovation communities discussed above) have started to explore the factors and mechanisms of collaboration process in these communities (Boudreau et al. 2014; Dissanayake et al. 2014; Levine and Prietula 2013). The new business models that focus on collaborative design in crowd-based communities have created opportunities for researchers to investigate some important aspects such as the collective design (Paulini et al. 2013), collaboration engineering (Nguyen et al. 2013) and co-development success (Oh et al. 2015).

Community Mechanism. In addition to user behaviors and collaboration, there is a stream of literature that focuses on how the design of innovation communities create better innovations at the macro level. Existing studies usually adopt theoretical development and case-based approach to develop frameworks or practices for effective innovation communities and conceptualize the mechanisms of crowd-based innovation communities (Chesbrough and Appleyard 2007; Piller and Walcher 2006). Important topics in this stream include strategies to incorporate users for innovation in a community (Fichter 2009; Füller et al. 2008; Lakhani and Von Hippel 2003; von

Hippel and von Krogh 2003), theoretical frameworks to effectively utilize the crowd in new product development (Di Gangi et al. 2010; Füller et al. 2006), and community mechanisms for innovators (Di Gangi and Wasko 2009; Franke and Shah 2003). This stream of research has provided important implications for both the creation of innovation community ecosystems and the practices to incorporate the crowd into the innovation process (Boudreau and Lakhani 2013).

2.4 Research Focus of this Dissertation

This dissertation aims to understand the open collaboration process in the online innovation process. Although existing research in innovation communities have widely examined the team collaboration perspective, there are still several gaps in related literature. First, most studies focus on collaboration in self-organized communities such as the open source software development community. In the context of firm-led open innovation communities, there is limited understandings on how firms organize the users into an open collaboration process. Although previous examinations widely discuss crowdsourcing-based communities, their focus is usually on the competition-based mode or idea generation process, instead of the collaboration mode. Second, existing studies on collaboration in innovation communities usually examine the behavioral factors such as social networks or team attributes, while ignoring the importance of the nature of the product itself in the innovation process. The focus of behavioral perspective helps to understand the overall management of open collaboration, but fails to explain more nuanced phenomena rooted in the product development process (e.g., software engineering in open source software development) and deeper level team collaboration. Therefore, my dissertation intends to fill these gaps by conducting two empirical studies to extend the current literature on innovation communities as follows.

The first essay studies a firm-oriented open innovation community to fill the first gap. Specifically, it focuses on how firms use the crowdsourcing approach to

incorporate the community into the new product development process. It emphasizes the collaboration perspective in crowdsourcing to complement our current understanding that centers on competition and idea collection. Using data on crowdsourced new product development campaigns that involve diverse crowd members, this study attempts to explore how to effectively organize online crowds in the innovation process. The second essay examines self-organized communities, i.e., open source communities, to fill the second research gap. It seeks to identify the key metrics in the software source code and examine how different programming styles in an open source project affect the open collaboration process. By studying the role of programming style, I intend to explore deep level collaboration mechanisms on the product itself and discover nuanced insights in the open source ecosystem. Using data and source code from open source projects, the second study explores the challenges in open collaboration and how to resolve these issues. Overall, the two essays help to enrich our understanding of online innovation communities by highlighting effective mechanisms for open collaboration.

CHAPTER 3 THEORETICAL BACKGROUND: GROUP DIVERSITY

In this dissertation, the theoretical perspective to understand open collaboration process in online innovation communities is group diversity. There has been a long history of research trying to understand the effects and dynamics of group diversity in the organization and strategy literature (van Knippenberg and Schippers 2007). The extant literature has discussed the effects of group diversity on group performance and the theoretical mechanisms underlying these effects. In this chapter, I summarize the classical views on group diversity in the literature and discuss its relevance to online innovation communities and the current dissertation.

3.1 The Definition of Diversity

Diversity in work groups is usually defined as the differences among the group members in terms of specific attributes that will lead to a perception that others are different from the self (van Knippenberg et al. 2004). The attributes that lead to diversity can range from demographic attributes such as age, gender and culture background (Bayazit and Mannix 2003) to informational attributes such as education, tenure and functional positions (van der Vegt and Bunderson 2005). In addition to surface-level attributes, deeper-level attributes such as attitudes, beliefs and values (Harrison et al. 1998) can also lead to diversity. It has been well recognized that group diversity can affect performance, but this is dependent on whether the difference in an attribute is visible to the group members and whether the difference can shape the perspectives needed to perform the group task (Pelled 1996). However, studies which examine the effects of diversity on performance have not drawn conclusive implications (van Knippenberg and Schippers 2007). In general, diversity in work groups may lead to conflicts but mixed findings were observed regarding the effects of diversity on group task performance.

3.2 Two Mechanisms of Diversity

Although there were mixed observations and findings on the effects of group diversity, two core mechanisms have been agreed upon in the literature – social categorization and information/decision-making (van Knippenberg and Schippers 2007; Williams and O'Reilly 1998). The interaction of these two mechanisms helps to understand and interpret the mixed effects of group diversity.

Social Categorization. It is commonly acknowledged that individual differences in work groups can lead to social categorization, where members are more likely to interact with similar others and have biases against dissimilar others (Williams and O'Reilly 1998). Such differences can undermine the group functioning process and lead to subgroup dynamics. From this perspective, work groups with homogeneous individuals will perform better than those with heterogeneous members since individuals with different social demographic attributes will try to categorize themselves into different subgroups in order to be more satisfied or comfortable during the group works (Pelled et al. 1999). Conflicts are more likely to be generated across different social groups when subgroup dynamics are constructed, leading to worsened group performance. In general, social or demographic differences such as age, gender, culture, tenure and position are regarded to engender this mechanism.

Information/Decision-Making. The core of this mechanism is that group diversity can be associated with differences in terms of expertise, knowledge and perspectives (van Knippenberg and Schippers 2007). From the informational or decision-making perspective, heterogeneous groups can access and bring to bear broader knowledge, opinions and expertise so that these groups are able to make decisions from more diverse perspectives and potentially produce better outcomes than groups with homogeneous members. Groups with diverse informational perspectives are more likely to exchange information, acquire various knowledge and make greater use of the information (Dahlin et al. 2005). In this process, task-related attributes such as

knowledge background, individual skills and personal network connections are the focal ones that engender this mechanism (Horwitz and Horwitz 2007).

3.3 Types of Diversity

In addition to the mechanisms on explaining the role of diversity, researchers have proposed typologies to understand what diversity is about and decompose different dimensions of diversity. According to the pattern, operationalization and consequences of diversity, it can be categorized into three types: separation, variety and disparity (Harrison and Klein 2007).

Diversity as Separation. Separation is about the differences of values, beliefs and attitudes in a group (Williams and O'Reilly 1998). It captures individual differences on a continuous attribute where individuals collocate at different positions across the attribute (Harrison and Klein 2007). Separation happens when group members have different opinions or attitudes towards the tasks (no separation when everyone has the same attitude, i.e., all at the same position with respect to attribute), and maximum separation emerges when one half of the group members have a completely opposite attitude from the other half (Harrison and Klein 2007). In terms of its consequences, following the attraction-selection-attribution mechanism (Schneider 1987; Schneider et al. 1995), separation usually leads to low levels of group cohesion, high likelihood of member withdrawal, greater conflicts and poorer performance (Ely 2004; Harrison et al. 1998; Jackson and Joshi 2004). Similar to the social categorization mechanism, individuals find it more pleasurable to work with others with similar values or characteristics in the same group (Harrison et al. 2002).

Diversity as Variety. Variety is about the knowledge, skills and information across a set of categories (McGrath et al. 1995). It captures the difference between individuals on a categorical attribute. Variety exists when members in the group have diverse information or knowledge in performing a task. It is minimal when all the individuals belong to the same category, while it is maximal when each individual occupies a

unique category (Harrison and Klein 2007). Generally, this type of diversity is positively associated with group performance. Variety helps the group to access different knowledge and resources, and facilitates the decision making and task completion within the group (Jackson and Joshi 2004; van Knippenberg et al. 2004). Consistent with the information processing mechanism, greater variety of information in the group can lead to better decisions and more creative solutions (Jackson et al. 1995).

Diversity as Disparity. Disparity relates to the resources, power and status distributed within the group (Blau 1977). It captures the inequality of individuals in terms of the resources they possess in the group. Disparity emerges when some individuals in the group hold more resources or power compared to others. Different from separation and variety, disparity is asymmetric because it only happens when some individuals have more power than others instead of the opposite. It is minimal when all members share the same amount of resources, while it is maximal when only one individual has a disproportionate share of the power or the resource (Harrison and Klein 2007). Following the tournament competition mechanisms, disparity can induce competition, differentiation and deviations within the group (Siegel and Hambrick 2005). But from the social stratification perspective that addresses power and status hierarchies, a moderate level of disparity (where individuals in the group exhibit some but limited differences in terms of power or resources) can lead to better conformance to group norms (Phillips and Zuckerman 2001).

3.4 Diversity in Online Groups

More pertinent to the focus of this dissertation, diversity in online groups has also been examined in the existing literature (Daniel et al. 2013; Ren et al. 2015). Different from traditional offline work groups, online groups have several unique characteristics that can lead to different implications with regards to the effects of diversity (Carte and Chidambaram 2004; Ren et al. 2015).

First, there is a lack of face-to-face interaction in online groups since members are geographically dispersed and interact primarily through technology-enabled channels. In such contexts, on the one hand, visibility of attributes is usually not salient so that diversity is less likely to be observed by group members. Individuals may not readily be aware of others' background and differences. On the other hand, these Internet platforms also enable users to access others' information such as demographics and participation history. This leads to unclear effects with respect to the social categorization process (Pelled 1996; Pelled et al. 1999).

Second, the structure of online groups is quite different from that of offline work groups. Power and status in online groups usually follow the logic of meritocracy rather than that of hierarchy (Stewart 2005). An individual gains power or social status in an online group through contribution, recognition and reputation instead of predetermined role from hierarchical position (Ren et al. 2015). Thus, inequality from some resources or privileges is not likely to emerge in online groups, which can curtail the negative effects of disparity on group functioning.

Third, in online groups, collaborative technologies are commonly used to coordinate members and generate rich information to be shared. These IT-enabled systems in online groups can help members track diverse information they have obtained and contributed (Carte and Chidambaram 2004; Chiravuri et al. 2011). Online groups are more likely to search and utilize various knowledge and expertise compared with offline work groups. Thus, it is expected that the information processing mechanism will play a more prominent role in online groups.

Lastly, online groups are characterized by fluid membership, low entry/exit barriers and voluntary participation (Faraj et al. 2011). Any individual with an intention to contribute can join a group, which further leads to much greater diversity of information in the group. However, the low entry/exit barrier and voluntary participation make online groups more sensitive to conflicts (Cramton 2001). Group

members may stop their participation when different opinions or conflicts arise in the collaboration process.

3.5 Theoretical Background of the Current Dissertation

My dissertation uses group diversity as the theoretical lens to understand the open collaboration process in online innovation communities. Specifically, it focuses on diversity as variety and as separation in online innovative collectives. The first essay aims to explore the effects of knowledge variety in large online collaboration groups. Although research on diversity has examined the effect of diversity or variety in different contexts, these groups are usually team-based and of small size. However, firms that collaborate with the crowd using innovation communities need to organize a large number of participants, implying the importance of understanding the overall effects of diversity (variety) in large online groups. These large online groups, characterized by diverse participants with diverse expertise, difficulties in quality control and information exchange, as well as multiple incentives of the crowd members, make the management of diverse individuals more challenging. In addition, I attempt to differentiate the role of different subgroup members in the collaboration process. Existing studies on group diversity usually examine the impact of overall group diversity (using some group level diversity indices) on group performance instead of exploring the composition of different types of members. The second essay focuses on individual work style separation as diversity. In general work groups, individuals can complete their tasks on their own such that the work style is merely a personal characteristic not visible to others (Pelled 1996). However, when this attribute can be observed by others, its effects on collaboration and performance are not well understood. Open source software communities, where work style can be observed from the source code, provides an opportunity to investigate this diversity attribute and allows us to explore the approaches to alleviate separation when individuals exhibit their work styles in the group. Overall, this dissertation seeks to

examine two important types of diversity in innovation communities and develops its theoretical framework based on the group diversity literature (Harrison and Klein 2007; van Knippenberg and Schippers 2007; Williams and O'Reilly 1998). It aims to extend the literature on the attributes and effects of group diversity, especially in the context of online groups.

CHAPTER 4 ESSAY I – ORGANIZING THE ONLINE CROWDS: DIVERSIFIED EXPERIENCE AND COLLECTIVE PERFORMANCE IN CROWDSOURCED NEW PRODUCT DEVELOPMENT

4.1 Abstract

Crowdsourcing has widely been used as a strategy for sourcing ideas and efforts to facilitate innovation. However, research into the value creation mechanism of crowdsourcing and the efficacy of the crowd in innovation creation is still limited. In this study, we investigate a crowdsourced new product development context to understand the collective intelligence and the crowd-creation process. Drawing on the theory of diversity and professionals' experience portfolios (i.e., generalists vs. specialists), we examine the role of crowd participants with different experience distributions in affecting crowd performance and innovation outcomes. Our empirical analysis shows that participants with both diverse and specialized experience are helpful in enhancing crowd performance in terms of efficient product development. The results also show that participants with T-shaped experience in non-focal tasks may be beneficial. Contrary to other group contexts, generalists do not seem to be helpful, at least in our study context. The findings provide insights for understanding this new form of organizing using crowdsourcing with collaboration and value co-creation in open innovation communities.

4.2 Introduction

Creating innovations is no longer the sole purview of domain experts but has recently become accessible to ordinary contributors in the crowd. The *wisdom of crowds*, or *collective intelligence*, has been appropriated by multiple stakeholders including firms, governments, scientists, technical experts as well as researchers (Howe 2006; Howe 2008; Malone et al. 2010; Savage 2012). Numerous crowd-based platforms have been established to facilitate innovations and the creation of economic value

(Avital et al. 2014). *Crowdsourcing*, a term coined by Jeff Howe in 2006 (Howe 2006), has been widely used for seeking various information and knowledge in a variety of domains. Platforms such as IdeaStorm, TopCoder and InnoCentive have attracted numerous users who are not formal domain experts but nonetheless contribute important resources (e.g., ideas, knowledge or solutions) to problem-solving and innovation creation. Leading companies including Dell, Starbucks, SAP, GE and Apple have also built crowdsourcing platforms and communities to attract value creation from the crowd (Krcmar et al. 2009; Ramaswamy and Gouillart 2010).

Along with the popularity of crowdsourcing and crowd-based platforms, a variety of business models concerning the way of organizing for crowdsourcing have been developed by companies and studied by scholars. Traditional crowdsourcing typically takes the form of a one-time collection of ideas, while Web 2.0 technologies and social media introduced social platforms and communities for crowdsourcing. To organize crowd participants, many crowdsourcing platforms adopt a competition model (e.g., crowdsourcing contest) and participants compete against one another to win the contests (Terwiesch and Xu 2008). Another form of organizing for crowdsourcing is firm-oriented idea generation, a model whereby a company hosts a platform and community members contribute new ideas to the company's business and services. IdeaStorm, the well-known platform hosted by Dell, is a representative example of this model (Bayus 2013).

Although competition and idea generation have been regarded as dominant forms of crowdsourcing, novel ways of organizing have recently been introduced. One of the new emergent organizing approaches in the crowdsourcing business model, which is the focus of this study, is collaboration. In contrast to the competition model where participants do not interact with one another, this new form of organizing emphasizes the notion of "crowd co-creation" and introduces interdependencies in crowdsourcing projects (Avital et al. 2014; Malone et al. 2010). A typical example of such a crowd co-creation process is the LEGO Ideas

Community, which enables user collaboration for value co-creation (Antorini et al. 2012). The penetration of social media, online communities and digital communication enables such collaboration-based crowdsourcing to be practically feasible. Compared to traditional crowdsourcing of work using contests and ideation forums, the concept of value *co*-creation is more salient in the collaboration process, which could lead to more predictable value propositions from the crowd and innovation creation from larger-scaled crowds (Nguyen et al. 2013; Nickerson et al. 2014). Such a new approach of organizing crowdsourcing and innovation has altered how crowds work and how value is created in open innovation communities. Instead of only seeking ideas, this new form of *crowd-creation* (Geiger et al. 2011) highlights the depth of crowd work and the importance of creating tangible outcomes from ideas. However, despite the growing trend of collaboration and value co-creation in crowdsourcing, research has yet to thoroughly investigate them, and challenges remain in this new form of organizing. Innovation outcomes may be subject to the uncertainty of contributions from diverse participants organized in the co-creation and from the voluntary nature of participation from the community (Ren et al. 2015). The existing literature has primarily focused on contest-based competitions and idea generation processes in crowdsourcing; only limited attention has been paid to new forms of innovation in crowdsourcing communities (Dissanayake et al. 2014), especially the organizing with collaborative intelligence and the challenges in the crowd co-creation process.

In this study, we fill the literature gap and resolve the challenges in crowd-creation by investigating collective intelligence and innovation outcomes in *crowdsourced new product development* (Malone et al. 2010). Concerning how the crowd works and creates value, an important aspect is to organize the knowledge contributed and created by the participants (Faraj et al. 2011). Participants possess different sets of knowledge in crowd-creation and make contributions based on their knowledge. To understand how to organize participants with different types of

knowledge, we adopt the theoretical lens from the organizational literature on diversity to explain differences in outcomes derived by the crowd in new product development. Specifically, by considering prior contributions or experiences of participants in the community as a proxy of knowledge (Menon et al. 2017), we examine how different participants are in terms of knowledge and how participants with different knowledge drive innovation performance. Although the role of experience has been examined in the crowdsourcing literature, little is known about how the characteristics of participants' experiences have an effect on their innovation outcomes. Inspired by the diversity literature on the distribution of knowledge, our study intends to answer the following research question: *how do crowd participants with different knowledge distributions affect their collective innovation performance in crowdsourced new product development?*

To examine the various types of the crowd members based on their knowledge (experience), we develop a typology of crowd participants based on the framework of generalist and specialist, and propose hypotheses about their value contributions to innovation outcomes by drawing on the diversity literature about teams and groups. Data on new product development from *Quirky.com* allows us to empirically identify six types of crowd participants. Furthermore, we find that a crowd with a greater proportion of members who possess experiences that are both high in diversity and in specialization is associated with better performance in terms of product development efficiency. In addition, a greater proportion of participants with a T-shaped experience distribution and with specialization in non-focal tasks in the crowd was also found to positively affect performance. Finally, we find that generalists are not an ideal type in our context. A series of supplemental analyses further confirm our findings and offer additional insights into experience accumulation, interaction effects and product effectiveness. By capturing the dynamics of crowd construction in a crowd-creation context and examining the characteristics of crowd member experiences, our study contributes to the literature

on crowdsourcing, open innovation, online communities, as well as to the theory of diversity and group composition.

4.3 Literature Review and Theoretical Background

4.3.1 Crowdsourcing

Crowdsourcing has been studied by many scholars in recent years. It denotes the outsourcing of internal tasks to outside individuals through an open call (Howe 2006; Howe 2008). Crowdsourcing typically calls for collective intelligence to facilitate the process of new product development, business analytics and problem-solving. However, different types of crowdsourcing exist and have been investigated by different research streams (Huang et al. 2012; Moqri et al. 2014). Here, we review major forms of organizing crowds for innovation (Geiger et al. 2011; Pedersen et al. 2013): competition (e.g., TopCoder), idea generation (e.g., IdeaStorm and Giffgaff) and collaboration (e.g., Lego). These organizing approaches are in line with the recent research agenda for crowdsourcing (Estelles-Arolas and Gonzalez-Ladron-de-Guevara 2012; Majchrzak and Malhotra 2013).

The first organizing approach is competition-based, where crowdsourcing participants provide their solutions to a specific task such as logo design, and only those who submit the best solutions are selected as winners and earn the rewards. Competition-based crowdsourcing is typically regarded as an auction or a contest and economic models are used to investigate how participants behave in the contest (Archak and Ghose 2010; Archak and Sundararajan 2009; Huang et al. 2012; Koh 2014; Yang et al. 2009; Zhang et al. 2017). Factors affecting crowdsourcing quality (Krcmar et al. 2009; Yang et al. 2009) and motivation (Hou et al. 2011; Jiang and Wagner 2014) have also been examined. In these studies, how to attract a greater number of problem solvers (project level) and how to win the contest (individual level) are the focal questions examined. Another way to organize crowd participants is through idea generation, which is typically adopted by firms to crowdsource ideas

for new products and services (Bayus 2013; Ramaswamy and Gouillart 2010) and typically does not require a specific task for the crowd. Instead, through an open call, firms benefit from new innovative ideas (Krcmar et al. 2009) and consumers also take the opportunity to make their ideas come to life. Studies on this element of crowdsourcing generally focus on how individual ideators' knowledge and experience affect their participation and performance. Bayus (2013) found that successful experiences have negative effects on subsequent idea adoption and Hwang et al. (2014) found that an individual's performance in generating ideas can be increased by her knowledge, especially knowledge depth. Huang et al. (2014), from a different perspective, suggested that by accumulating experiences, users updated their ability and understanding of the firm's implementation costs on their ideas to provide better ideas. The adoption and quality of ideas are the focal interest for this organizing approach in the literature.

The previous two ways of organizing for crowdsourcing, mainly focus on the collection of ideas or solutions from the crowd. Differently, the third recent way of organizing enable collaboration among participants in the crowd so that they collaboratively work toward a collective outcome. This approach highlights the depth of crowdsourcing for creating more tangible outcomes from ideas through the integration of the co-creation process (Paulini et al. 2013). Conceptual models including the peer-production model (Haythornthwaite 2009; Nguyen et al. 2013) and patterns of collaboration (Nguyen et al. 2013; Vreede et al. 2009) have been used to provide insights into the collaboration process. Paulini et al. (2013) investigated the collaborative design process in innovation communities through qualitative analysis of communication in online forums. However, discussion about this form of organizing in crowdsourcing is still limited in the literature. This form of organizing, in general, takes the opportunity to organize the crowds for co-creation and exhibits great potential with high crowd engagement. Therefore, more theoretical and empirical examinations towards the efficiency and effectiveness in this crowd-

creation process are required to better understand the collaborative form of crowdsourcing for value co-creation. Our study focuses on this form of organizing in crowdsourcing and tries to fill this gap.

4.3.2 The Role of Experience in Crowdsourcing

Our study is also related to the stream of literature about knowledge or experience in crowdsourcing. Knowledge plays an important role in innovation tasks and crowdsourcing, and a series of studies have shown the effects of experience in crowdsourcing. Most of them focused on the effects of learning from prior experiences on an individual's behavior in crowdsourcing contests. Archak and Ghose (2010) examined the learning-by-doing effects at TopCoder and they found that coders would both myopically learn from their experience through participating in the same programming language projects and try other new language projects in a forward-looking manner. Huang et al. (2012) also examined experiential learning at Threadless, showing that experience reduces the effort in submitting solutions. Menon et al. (2017) further compared downstream and upstream experiences to understand the effects of related experiences on individual performance. However, past experience does not always improve an individual's performance. Successful experience in idea generation was shown to cause cognitive fixation on ideators and negatively affect their subsequent performance (Bayus 2013). In addition, other outcomes such as subsequent participation (i.e., users with unsuccessful experiences may leave the community) and strategic behaviors (i.e., experienced users are more likely to strategically choose the time and difficulty of task participation) were also investigated (Huang et al. 2014; Yang et al. 2008).

The existing literature provides evidence that a participant's prior experience does matter in crowdsourcing, but competing effects are identified at the individual level. Our study, in this regard, extends previous research by examining the role of various dimensions of experience in past crowdsourcing tasks in crowdsourced

innovation at the group level, which is particularly important in the value co-creation process.

4.3.3 Generalist and Specialist

Our research focus on the distribution of participants' knowledge is in line with the framework of generalist and specialist in the diversity literature. Generalist and specialist are generally defined by the distribution of their knowledge (Rulke and Galaskiewicz 2000). Based on the portfolio of knowledge and experience, generalist and specialist can be determined by how diverse/specialized their experience and knowledge are (Boh et al. 2007; Kang et al. 2012; Narayanan et al. 2009). The diversity literature suggests that generalists usually possess higher knowledge diversity (breath), while specialists possess deep knowledge within restricted domains.

It has been widely shown that generalists (with greater knowledge diversity) can perform better and be more productive. Greater knowledge diversity is regarded as a source of creativity (Taylor and Greve 2006), and generalists have the ability to learn new knowledge more efficiently. They know how to learn new knowledge and skills with less error because they may better relate to their existing stock of knowledge which has more hooks for relating because of its great diversity (Narayanan et al. 2009). When facing a new task, they are more likely to find out related and helpful solutions from their known solutions because they have more diverse experiences in performing tasks. (Narayanan et al. 2009). In addition, Hwang et al. (2014) proposed that broad knowledge can induce novelty and creativity (i.e., think outside the box) because greater knowledge breadth facilitates the search of new ideas and the recombination of existing knowledge. Bayus (2013) found that the diverse experience in commenting activities could mitigate the cognitive fixation problem in idea generation. That is, the diversity of knowledge domains, or knowing other different areas is able to expand one's perspectives in creating ideas and reduce

the cognitive constraints of repeating successful experiences. Also, from a group perspective, Rulke and Galaskiewicz (2000) found that having more generalists in a group could enhance group performance by facilitating knowledge exchange and sharing as there will be a greater likelihood of having common ground across individuals, which is essential to knowledge exchange. These findings imply that diverse experience (i.e., generalists) should help to enhance performance.

Conversely, some studies caution that the diversity of experience may hinder performance. The variety of experience has an inverted U-shaped effect on software maintenance productivity, which means that too much diversity may harm productivity (Narayanan et al. 2009). The diverse experiences, although benefit performance with broader knowledge base and perspectives, may cause great loads of cognition or memory and increase the difficulty of searching for knowledge and ideas (Johnson and Hasher 1987). Also, the effect of experience variety was shown to be moderated by tasks relatedness (i.e., the overlap between the current task and past experiences) (Armstrong and Hardgrave 2007; Boh et al. 2007). Thus, it is not always the case that diversity of experience improves performance.

On the other hand, there is also evidence suggesting that specialists (with deep knowledge in limited domains instead of knowledge in diverse domains albeit of limited extent) can perform better and be more productive. The literature in software development has shown that specialized experience enhances team learning and productivity, generally through experience curve and learning-by-doing (Boh et al. 2007; Huckman et al. 2009; Kang et al. 2012; Narayanan et al. 2009). Hwang et al. (2014) also showed that in the innovation context, generalists with deep knowledge in at least one domain could generate better ideas than those with shallow but diverse knowledge, emphasizing the importance of knowledge depth in innovation. This is because the depth of knowledge help to improve the feasibility of ideas, which might be low when an individual only creates innovations with diverse knowledge (i.e., she may generate some novel but unrealistic ideas). However, other studies also showed

that specialized experience is not always beneficial for performance and productivity. Specialization in some knowledge types may cause learning myopia and undermine the ability to learn new knowledge (Archak and Ghose 2010). In addition, there is a learning plateau effect such that the effect of past experience is marginally decreasing and becomes less significant after a certain point (Argote 2012). Cognitive fixation is also more likely to occur as specialized experience increases (Bayus 2013). Thus, there are also conflicting results regarding the role of specialists and specialized experiences.

Given the conflicting results in the literature, instead of measuring group-level experience directly, we develop a typology of different types of crowd members based on generalist (diverse experience) and specialist (specialized experience) and use this to develop hypotheses using the mechanisms in the diversity literature. This approach allows us to differentiate the effects of member types (i.e., subgroups) in the collaboration process and examine the role of experience diversity through a composition perspective.

4.4 Theory and Hypotheses

In this section, we develop our theory concerning different member types and how they contribute to product development outcomes. To better clarify the member types in the crowd and elaborate the mechanisms in our hypotheses, we first introduce our study context, and then develop a typology of members and hypotheses related to their effectiveness in collective innovation.

4.4.1 Study Context

Our study context is new product development at *Quirky.com*, a crowdsourcing platform for online inventions. As a company focusing on community-driven new product development, Quirky uses a crowdsourcing approach for both its product portfolio and product development. Users can submit product ideas using a problem-solution paradigm. Then Quirky, along with its community members, will choose the

promising ideas and start to develop the product concepts. During the development process, Quirky also crowdsources important product development tasks and decisions to community members through different development projects. When a product is released into the real world, Quirky will manufacture and sell the product through various channels (e.g., via partnerships with major brick and mortar retailers as well as via Quirky's own e-commerce website). Anyone who has contributed to the development of the product, including the initial inventor (i.e., the individual who submitted the original product idea), community members (i.e., those who participated in the product development projects), and Quirky (who is in charge of manufacturing) will share the proceeds from the product sales.

The first stage of Quirky's business model, called *Invent*, comprises the idea generation process where new product ideas from the crowd are submitted. It includes both competition among ideators and collaboration from the community through comments, identification of similar products and social interactions among members. Promising ideas are then selected into the development process, called *Influence* stage, where Quirky sets up a series of collaborative projects based on the tasks crowdsourced to the community. In these collaborative projects, community members can directly create submissions of their own solutions or submit improvement to others' submitted solutions, or indirectly contribute to the project by commenting or voting on others' submissions. Finally, when a product successfully *completes* its development (on paper or as a prototype), it will be launched into *Concept Portfolio* and prepared for production. Specifically, our study focus is the second stage (i.e., the *Influence* stage), which comprises the crowdsourced product development process. This process is characterized by the co-creation from Quirky and the crowd, and the quality of this process is generally captured by the efforts (e.g., time of development) and the value of the product (e.g., whether the product moves further). In addition, although Quirky emphasizes the concept of team in the *Influence* stage, we find that it may not completely fit the term "team" (Dissanayake et al. 2014) because of the

mixture of modes[1] and the large number of participants in the co-creation process. Instead, we use "crowd" to characterize the participants in a whole product development process (Pedersen et al. 2013).

4.4.2 Experience Typology

To understand how different participants contribute to their collective innovation outcome, we first present a typology of crowd members (participants) in crowdsourced new product development following our research context. We define each type based on the distribution of prior experiences in crowdsourcing tasks. Although there are two generic types of professionals (i.e., generalists and specialists), hybrid forms may emerge depending on the concentration of experiences with respect to some focal task (or knowledge requirement). Specifically, in line with the concept of experience in our study context, we use members' prior experiences in other crowdsourced product development projects to quantify three metrics: diverse experience, specialized experience and concentration of experience.

Diverse experience is defined as the breadth of task areas the member has experienced in past crowdsourced product development projects. *Specialized experience* is defined as the depth of prior experience the member has obtained with respect to some focal task. Here, *focal task* refers to the tasks in which the member participates for the current product development project. However, using only diverse and specialized experience cannot portray the whole picture regarding generalists and specialists. If one has both highly diverse experience and highly specialized experience, the domain in which the experiences are concentrated will matter. Concentrated experience in limited domains renders one a T-shaped professional,

[1] The process Quirky adopts for harnessing collective intelligence is *Collection* and *Collaboration* (Malone et al. 2010). At the project level, Quirky crowdsources specific tasks to the community and community members independently work on the solutions, indicating the *Collection* mode. But at the product level, different tasks are interdependent and members should consider others' works in order to complete their own work, consistent with the *Collaboration* mode.

while equally distributed specialized experience on all domains leads to an omniscient expert (Hansen and Von Oetinger 2001). Thus, in our study, *concentration of experience* is the extent of concentration of all the experiences the member has obtained in past product development tasks, which implies the balance between diverse and specialized experiences (Kang et al. 2012; Narayanan et al. 2009). Based on these three metrics, we can specify the distribution of participants' experience and classify crowd members into different types. Table 4-1 shows a theoretical typology of crowd members according to combinations of values for these dimensions.

When the extent of diverse experience and experience concentration are all high, such a member can be classified as T-shaped with respect to the task requirements (Hansen and Von Oetinger 2001; Kang et al. 2012; Narayanan et al. 2009). If the specialized experiences are concentrated on a focal task, then the member is considered T-shaped with respect to the focal task. Alternatively, if the specialized experiences are concentrated on some other non-focal task, then the member would be considered as T-shaped but in tasks other than the focal task requirement. When diverse experience and specialized experience are high but experience concentration is low, which means that the experiences are sparsely distributed in different types of task, the member will have highly specialized experiences in many of the task types, indicating an omniscient type (i.e., omniscient polymath). The fourth type indicates a type of generalist, which has broad experiences across different tasks but limited experiences in each of the different types of tasks. The fifth type represents a specialist in the focal tasks, with specialized experiences concentrated in the focal areas. In contrast to this (i.e., the sixth type), when the specialized experience on focal tasks is low, this means the experiences are concentrated on other non-focal tasks, indicating a specialist in some other tasks. The remaining combinations are distinct from the others. When all of the three metrics are low, the member would only have few experiences in each domain, suggesting the

novice type. Therefore, the seventh type corresponds to a group of inexperienced members. However, it is not possible to identify any specific type when only specialized experience is high because such a combination is theoretically impossible. Thus, a total of seven types exist in the typology. In the empirical analysis, we classify the type of crowd members based on such a typology.

Table 4-1. A Typology of Crowd Members by Experience

No.	Diverse Experience	Specialized Experience	Experience Concentration	Type
1	High	High	High	T-Shape in focal tasks
2	High	Low	High	T-Shape in other tasks
3	High	High	Low	Omniscient
4	High	Low	Low	Generalist
5	Low	High	High	Specialist in focal tasks
6	Low	Low	High	Specialist in other tasks
7	Low	Low	Low	Novice

Note: The typology consists of seven types. The remaining possibility (i.e., low in diverse experience, high in specialized experience and low in experience concentration) is theoretically impossible and does not indicate any meaningful type.

4.4.3 Hypotheses Development

Consistent with the criteria of our typology, our hypotheses are based on the three metrics of experience and the types emerging from them. Specifically, we provide theoretical arguments about experience diversity, specialization and concentration in crowdsourced new product development to further predict how each type of crowd member would influence the product development process based on their value contributions to collective innovation performance. To develop the hypotheses about the crowd, we apply the mechanisms from the diversity literature on teams and groups, and discuss both individual level (each crowd member) effects from the creativity perspective and group level (the whole crowd) effects from the information processing perspective.

In crowdsourced new product development, a group of individuals form a crowd and collectively work on tasks that are designed to produce ideas or solutions.

As crowd members need to search their own knowledge bases, their prior experience in the relevant tasks become critically important. Therefore, diverse experiences would provide members with richer knowledge components and a greater number of available ideas or solutions (Weisberg 1999). The availability of ideas is regarded as the basis for generating novel outputs (Taylor and Greve 2006). Therefore, diversity of experience should help individuals to search and combine existing knowledge to generate creative and novel works (Hwang et al. 2014; Taylor and Greve 2006; Wulf and Schmidt 1997), resulting in better performance. In addition, participants in the crowd do not always work independently. Besides collective works (i.e., independent tasks), they also collaboratively commit to a joint outcome (i.e., the product) with interdependencies among their tasks, ideas and solutions (Malone et al. 2010). Members in the crowd will interact with one another like in a virtual group or community. From this perspective, members with high experience diversity will be able to facilitate knowledge transfer through more effective group communication (Rulke and Galaskiewicz 2000). Other members that may lack task-relevant information and experience would be able to receive assistance from these members with high experience diversity (Paulini et al. 2013). Also, the total level of diversity in terms of task experiences in the crowd can be further increased by experience diversity because of the interaction and broader exposition of knowledge (Dahlin et al. 2005; Nonaka and Takeuchi 1995), eventually leading to more creative ideas or solutions from the crowd. Thus, we argue that experience diversity is positively associated with crowd performance (Huckman et al. 2009).

In terms of the crowd member types defined in our typology, four are associated with high diverse experiences. However, we notice that only the type "generalist" independently conforms to our argument for diverse experience, while others such as "T-shaped specialist" and "Omniscient" are also connected with the other two metrics (i.e., specialization and concentration). For generalists, they serve as the source of ideas and the driver of knowledge transfer to increase the diversity

and creativity of the crowd (Hwang et al. 2014; Perry-Smith and Shalley 2003; Rulke and Galaskiewicz 2000). Thus, a crowd with more generalists is expected to perform better. Since our crowd composition is conceptualized as member types within the crowd, we define the baseline as the novice type who do not have much experience in any dimension and use the proportion of member types to explain the effects. Therefore, for a given size of the crowd, a higher proportion of generalists will benefit the crowd in terms of performance, which leads to higher efficiency (for the firm) in the development process. We propose:

Hypothesis 1*: A crowd with a greater proportion of generalists will have better performance.*

Besides diverse experiences, crowd members also rely on specialized experiences to generate ideas or solutions for the crowdsourced tasks. From the individual level perspective, an individual in the crowd has to figure out the problem in the task and search from her existing knowledge to produce some outputs. Although specialized experience does not increase diversity, it could reduce the cost for the crowd member to work on the final output (Huang et al. 2012; Narayanan et al. 2009). Therefore, even though task specialization does not necessarily lead to novel or creative works, the efficiency and quality of the task output could be higher. Furthermore, although diversity can help to generate a greater number of ideas, it may often lead to unexpected outcomes due to the lack of maturity or uncertainty of the generated ideas (Taylor and Greve 2006). Deep knowledge, from this view, can help individuals in the crowd effectively combine their knowledge and make their solutions more feasible (Boh et al. 2014; Hwang et al. 2014). Thus, the ideas or solutions generated by the individuals in the crowd may be better when they possess deep expertise in the task domain. In addition, the same expectation emerges from the collaborative perspective (i.e., crowd level). Members with high levels of specialized experiences can provide practical suggestions for improving the ideas of other members who do not possess deep knowledge (Paulini et al. 2013). The deep

knowledge shared by these members will help to make the ideas or solutions from the crowd more feasible and reliable, reducing the uncertainty resulting from diverse but shallow knowledge in the product development (West 2002). Therefore, we expect that specialized experience is positively associated with crowd performance.

The two types of crowd members in our typology that conform to our arguments about specialized experience are "specialists in the focal tasks" and "specialists in non-focal (other) tasks." Focal tasks, as defined earlier in our typology discussion, refer to the tasks in which the crowd member participated in the product development process. Thus, having more experience in focal tasks will not only guarantee an individual's own work quality but also increase the reliability of others' works more effectively. This effect, on the other hand, may not be salient for specialized experiences in non-focal tasks due to the absence of direct interaction and specialization. Following the same baseline in H1, we propose:

Hypothesis 2*: A crowd with a greater proportion of specialists in focal-tasks will have better performance.*

In addition to diverse and specialized experiences that affect crowd performance directly, the concentration of experience may also matter. The concentration of experience shifts the interaction between diversity and specialization, leading to more nuanced types of individuals in the crowd. In traditional task contexts, a balance of diversity and specialization (i.e., T-shaped) will be preferred (Narayanan et al. 2009). It may be costly to be specialized in many different areas and acquiring knowledge that is less relevant can be ineffective for task performance. Also, a concentrated focus on limited knowledge domains may enhance the learning outcome (Yang et al. 2008). However, given the innovative nature of tasks and voluntary participation in crowdsourced new product development, a balance between diversity and specialization may not be necessary. In a context of innovation, both diverse experience and specialized knowledge are important (Boh et al. 2014). Without deep knowledge in broad domains, individuals

cannot combine their knowledge effectively to reduce the uncertainty of creative works and generate reliable solutions (Hwang et al. 2014; Novick 1988). Furthermore, due to the interdependencies among tasks in crowd-based product development (Malone et al. 2010), only having (limited) specialization in the focal domains will be insufficient because it is necessary to refer to other tasks or crowd works to effectively search from individual's own knowledge, which requires some understanding of non-focal tasks. Moreover, due to the nature of voluntary participation and knowledge sharing in online communities (Ren et al. 2015), participants in the crowd could voluntarily choose whether or not to work on a task and easily access past contributions on the digital platform instead of costly searching from offline archives, so the effect of cognitive learning cost may not be salient in this context (Archak and Ghose 2010; Huang et al. 2012). Thus, we argue that the concentration of experience will not matter much in crowd performance.

The types of crowd members relevant to experience concentration are the "T-shaped" and "Omniscient" members. Combining our discussion about diverse experience and specialized experience, both of these types have the potential to enhance crowd performance. Thus, crowd member types with both a high level of diverse experience and specialized experience will lead to similar prediction. The two types – T-shaped specialist in focal and other tasks – possess both a high level of diversity and depth in prior experience. Similarly, the omniscient type members have a high level of specialization in most of the domains. Consistent with our discussion about specialists in non-focal tasks, the effect of specialization is not salient for T-shaped specialist in other tasks either, but diverse experience is expected to matter. Given our discussion about experience concentration, we do not expect significant differences due to concentration. Thus, in line with the same baseline condition in H1 and H2, we propose:

Hypothesis 3a*: A crowd with a greater proportion of T-shaped members in focal tasks will have better performance.*

Hypothesis 3b*: A crowd with a greater proportion of T-shaped members in other tasks will have better performance.*

Hypothesis 3c*: A crowd with a greater proportion of omniscient members will have better performance.*

4.5 Data and Method

4.5.1 Data Collection

We retrieved all product information and product development information from Quirky with the details of crowdsourced projects in each product development process, including the type, task, submissions and comments for each project. User profiles and information related to the idea of each product were also retrieved to build a holistic picture of the product development process. The time window of our data is from May 2009, the start of Quirky's operations, to June 2014. The full dataset includes 828 product development campaigns, with a total of 3,044 sub-projects for these products, as well as 33,789 unique users who participated in the crowdsourced product development process. In our analysis, we focus only on those products with community participation (574 products) so that a crowd was actually constructed in the product development process. They were used to measure members' prior experiences, which are then used to classify the members into experience-based types. Furthermore, some products were not completed (i.e., stopped at the early development stage) due to some unexpected development difficulties. Since we do not have complete information about these products, they were not included in the estimation sample. Nonetheless, crowd members' experiences in these products were included. We also only use data starting from 2010 since members on the platform generally do not have any prior experiences in the first year.[2] Finally, we excluded repeated products which are based on the same idea since they usually have duplicated development processes. Outliers and other records with incomplete

[2] That said, data from 2009 were used in capturing the experiences of crowd members that participated in products from 2010 and onwards.

information were also examined and excluded from the analysis dataset.[3] Our final sample includes 425 product development campaigns (with 2,097 sub-projects and 29,980 crowd members) that were developed at least on paper between 2010 and 2014.

4.5.2 Measures

To measure the prior experience, we first identified the crowd members in each product's development. Specifically, we use submissions as the criterion for member selection. A user is considered as a member in the crowd if he or she submits at least one contribution to the projects during product development.[4] Thus, we set this criterion for crowd members and measure experiences based on user contributions within each product development. This procedure generated 274,281 observations of product-user pairs.

We quantify members' past experiences based on the three metrics – diverse experience, specialized experience and concentration of experience – in terms of two aspects – process and domain. Similar to related studies (Pedersen et al. 2013), we note that the crowd in product development is dynamically constructed. Also, the experiences of members are dynamic and product-crowd specific. Thus, we computed each member's experience in each product using the join time, which is the time at which the member started to participate in the product development, as the cutoff time for experience. In addition, experience is accumulated at the product level, which means all relevant experiences for a product development process can be

[3] The rationale of this step is that our dependent variables are about Quirky's decision on the development process and product. Although we believe Quirky should make consistent decisions on the outcomes, there may be still some disturbances or unobserved factors that can lead to extreme or unexpected decisions. Therefore, the exclusion of these observations can reduce the potential errors in the regressions. Actually, our results are consistent with the inclusion of these outliers.

[4] As discussed in section 3.1 (study context), submissions are regarded as the core contributions from crowd members. Although peripheral contributions exist in the product development, they are usually not creative works and not influential. Future work may examine the significance of peripheral contributions in this context.

included in a member's experience portfolio only after the product development completes (or is stopped prematurely). This is because in the product development context, the product is a whole entity or unit for acquiring complete experiences (i.e., partial experiences are not counted). This procedure allows us to capture both the dynamic crowd formation process as well as the differences in experience accumulation for each user within a product development campaign.

In our context, there are five different types of projects corresponding to five specific tasks (i.e., a 1-to-1 mapping) and eight product categories corresponding to eight domains. Process tasks (projects) include *Research*, *Design*, *Styling*, *Naming* and *Tagline* setting, while domain tasks (categories) are *Electronics*, *Health*, *Home*, *Kitchen*, *Parenting*, *Play*, *Travel* and *Wildcard*. Projects are mainly about the experience in performing specific tasks in the *process* of product development, while categories are related to *domain* knowledge on product development in different categories. Thus, for the *process* task aspect, we operationalized diverse experience as the number of distinct tasks the user has participated in past product development projects. Specialized experience was operationalized as the amount of prior experience in the *focal tasks*, normalized by the number of tasks. Specifically, it could be written as $SpecializedExperience_{ij} = \sum_j^{T_i} Exp_{ij} \,/\, P_i$, where P_i is the number of task types crowd member i has participated in the focal product development, T_i is the total number of product development campaigns in which the user has participated, and Exp_{ij} is the number of focal tasks the user participated in the development of product j. Similarly, for *domain* tasks, diverse experience is measured as the number of distinct product categories the user has participated, and specialized experience is operationalized as the numbers of product development campaigns that are in the same domain (category) with the focal development campaign the user has participated.

To measure the concentration of experience for crowd members, we use the

Herfindahl-Hirschman Experience Index (*HHEI*), which is derived from the Herfindahl-Hirschman Index frequently used to measure market concentration in economic studies. This measurement has been used in software development studies to represent the concentration of experience (Kang et al. 2012; Narayanan et al. 2009). Specifically, the computation of experience concentration by *HHEI* is $\sum_k^N (Exp_{ik} / SumExp_i)^2$, where N is the total number of distinct tasks (N=5 for process tasks and N=8 for domain tasks) in all product development campaigns, Exp_{ik} is the number of task experiences of member i in task type k, and $SumExp_i$ is the total number of tasks crowd member i participated in the past across all the task areas.

4.5.3 Identifying Experience-based Crowd Member Types

With the operationalizations of three metrics, we computed the time-varying (i.e., product-varying) experience measures for each crowd participant within each product development process. Then a clustering approach was used to identify the types of members for each product-crowd pair. Cluster analysis has been widely used in the related literature as an exploratory approach to empirically discerning different types of users given patterns of observable behaviors (Hahn and Lee 2013; Lin et al. 2014). Clusters are partitioned to have high intra-cluster similarity and inter-cluster dissimilarity. We adopt the two-step procedure in Ketchen and Shook (1996) to conduct the cluster analysis. First, we performed hierarchical clustering (Maimon and Rokach 2005) and investigate the dendrogram to determine a suitable number for clusters. We also validate the solution using the *k-means* clustering approach with model fit indicators. Specifically, we use Caliński-Harabasz index (Caliński and Harabasz 1974), which captures the balance between within-cluster sum of squared errors and between-cluster sum of squared errors. After determining the suitable number of clusters, we use the *k-means* approach to assign the membership of observations to clusters on the 274,281 product-member pairs. Finally, we matched the clustering results (i.e., generated clusters) to our typology.

4.5.4 Empirical Model

To test the impact of each type of crowd members on product development performance, we build a product-crowd level econometric model. The variables used in our model are specified as follows.

Dependent Variable

Development Duration (*Duration*): We measure product development performance based on the duration of the crowdsourced product development projects.[5] Several reasons support the use of development project duration to measure crowd and development performance. First, in a specific project, Quirky (usually a team assigned for the product) will review the submissions and close the project when it is deemed that it can utilize the solutions for development decisions. Therefore, the duration captures how Quirky perceives the quality the crowdsourced work. Second, in the product development process, Quirky will typically control the quality of product and complete the development process only if reaches an acceptable level of quality. Thus, all else being equal, a shorter duration means that Quirky spends less time on examining the crowd contributions from multiple tasks, implying better performance from the crowd. In contrast, a longer development duration implies that the crowd did not generate satisfactory solutions for Quirky and as a result took Quirky more time for overall product development (i.e., close the projects and launch the product). However, since the crowd dynamically emerges, using the product development duration from its start to end may raise some confounding and causality issues (i.e., reverse causality whereby a longer time causes more participants and more experiences of participants). To address this issue, we adopt an aggregation approach for the operationalization. Since the crowdsourced product development

[5] Ideally, the dependent variable should be the overall performance of the crowd (e.g., the quality of their contributions). Unfortunately, it is not possible to observe each crowd member's actual quality to measure the internal crowd-level performance with only publicly observable data on product development outcomes.

(i.e., crowd work) is comprised of several development projects and solutions are typically submitted during the initial days in these projects, the durations of all development projects are used. Specifically, we computed the duration of each project and summed them to derive the duration at the product level. Such an operationalization helps to avoid the issues caused by the dynamic formation of the crowd and does not affect the validity of the performance measure.

Independent Variables

Type of crowd member: To examine the impact of experience-based member types on crowd performance, we used the proportion of each experience-based user type in the crowd as the independent variables (Hahn and Lee 2013; Inbar and Barzilay 2014). These types are determined by the empirical identification of crowd member types from the cluster analysis.

Control Variables

Given that our dependent variable for crowd performance is product development duration, we need to control for several factors that not only affect crowd performance but also product development time.

Number of Participants (*Members*): The number of participants who submitted solutions, ideas and other creative works to the product development campaign is controlled. In our context, because each member can only submit a limited number of solutions in each sub-project, the number of participants captures not only the size of crowd but also the number of contributions. In addition, it is necessary to control the size of crowd when the proportion of different member types are used as independent variables.

Number of Projects (*NumProjects*): Since we use the sum of project durations as the dependent variable, it is necessary to control for the number of projects. In addition, the number of projects indicates the amount of required crowdsourced work for product development, which is also useful for explaining the duration of product

development.

Brainstorming (*HasBrainstorm*): Brainstorming is an offline activity for product development using an expert panel. Typically, if a brainstorming session is conducted, a video section will be displayed in the product's timeline. It may help to attract different members from the crowd to participate in the tasks and facilitate the overall development process. Since brainstorming is done before the start of product development, we can use it as a control variable. We use an indicator variable to indicate whether the product used a brainstorming session.

Inventor Products (*InventorProducts*): We also control the characteristics of the product inventor (idea submitter). Specifically, we use the number of prior successful ideas (selected into development stage) created by the inventor as a control variable. The successful experiences of product inventors may help to attract more participants and make the product more likely to be successfully developed.

Number of Comments (*Comments*): We control for the total number of comments on the submissions by the crowd members during the product development process. This captures the interaction and collaboration among the crowd participants.

Average Crowd Ideation Influence (*AvgIdeaInfluence*): Since each crowd for product development could be different due to members' own intelligence levels or other experiences, it is necessary to account for the confounding effect caused by the crowd intelligence. Specifically, we utilize the performance of crowd members in the *Invent* stage. We compute the average amount of *influence* points (a measure of contribution for ideation stage defined by Quirky) across products earned by each crowd member before joining the crowd, and then take the average across the members within the crowd.

Product-Specific Factors: The product development process is usually affected by the complexity and uniqueness of the product. Since it is difficult to know a-priori the complexity of the final product, we use three proxy variables to capture product-level heterogeneity. We use the number of comments, the number of similar

products and the length of solution in idea description to control for product characteristics. First, we control for the number of comments in the ideation (invent) stage of the product (*IdeaComments*). We only compute the number of comments before the product moves into development (i.e., prior to selection for product development). More community feedbacks may indicate that the product has more elements to be discussed, reflecting greater complexity. Next, we control for the number of similar products submitted by community members about the product (*SimilarProducts*). The existence of more similar products may suggest lower uniqueness of the product idea but higher number of elements in the product. We use this to control for the uniqueness of product. Finally, the length of solution provided by the ideators is controlled for the complexity of product idea (*Solution*). Since idea submitters follow a problem-solution paradigm, the solution indicates the possible approaches for product design and signals the complexity of the problem.

Category and Time: We also control the category of each product using dummy variables. We classify the eight categories into three main categories: electronic-related, home-related and play-related.[6] Year dummies are also controlled for time trends, policy change and year-specific effects.

In the empirical analysis, we log-transform product development duration, number of participants, number of comments, average crowd intelligence, number of ideation comments, number of similar products and length of solution because of their scales and skewed distributions. Since our data is cross-sectional at the product-crowd level (i.e., product development campaign with a crowd as the unit), we used OLS to estimate the parameters in the model. To address the potential sample selection issue of using completed product development campaigns with crowd participation, we use Heckman selection model to test the robustness of our results

[6] This is to classify the categories into intuitive groups to identify their specific effects. The main results of key independent variables are actually not affected by including all the category dummies.

(see discussions in §4.6.3).

4.6 Results and Discussions

4.6.1 Clustering Results

We first performed hierarchical clustering to determine the suitable number of clusters. However, given that our sample includes 274,281 observations, it is not practical to employ hierarchical clustering (or other approaches that use distance matrix to cluster) due to computational limitations.[7] To overcome this, we employed a bootstrapping approach where we selected a random sample of 10,000 observations to perform hierarchical clustering and repeat the process with different random subsamples. The dendrograms indicate that the solution of six clusters is optimal and stable across bootstrapped samples. We also computed the Gap statistic (Tibshirani et al. 2001) to verify the reliability of clusters across subsamples. Table 4-2 presents the results in 4 subsamples from 2 to 10 clusters. The first peak was selected to indicate the optimal number of clusters and all the subsamples suggest the solution of six clusters. We then performed *k-means* cluster analysis on the full sample using six clusters solution. To further verify the results, we calculated Caliński-Harabasz indices from 2 to 10 clusters. The peak of indices suggests the reasonable number of clusters is six in the data (also shown in Table 2). Finally, as the *k-means* approach randomly selects the initial values of cluster centroid (starting points), we ran the analysis for the solution of six clusters using multiple random starting points (with 1,000 replications) and the clusters are quite stable in terms of cluster centroid and size (reliability > 0.8). In addition to the quantitative approaches for understanding clustering results, we compared the generated clusters and found that there aren't any

[7] The hierarchical clustering (or other distance matrix based approaches) requires the distance matrix between each observation, which will generate *N(N-1)/2* pairs in the matrix. Given that our sample size (274,281), it is impossible to address the exponentially increased number of pairs in the matrix (approximately 280GB memory). Thus, we follow the existing literature to use random subsamples. In addition, due to the dynamic nature in our sample, we use *k-means* approach on the whole sample to verify the clusters.

additional meaningful clusters when number of clusters is above six, lending greater support for our six-cluster solution.

Table 4-2. Robustness Checks of Clustering Solution

	Gap Statistic				Caliński - Harabasz Index
#Clusters	Sample 1	Sample 2	Sample 3	Sample 4	Full Sample
2	1.4462	1.4625	1.4491	1.4733	76,299
3	1.5262	1.5415	1.5407	1.5567	445,057
4	1.6691	1.6351	1.6413	1.6834	545,461
5	1.7358	1.6879	1.7390	1.7499	566,929
6	**1.7596**	**1.7680**	**1.7549**	**1.7964**	**573,522**
7	1.7312	1.7662	1.7354	1.7861	548,585
8	1.7808	1.7818	1.7534	1.8010	529,387
9	1.7571	1.7638	1.7654	1.7792	511,941
10	1.7930	1.7922	1.8007	1.7931	474,397

Table 4-3 summarizes the clusters and the corresponding crowd member types. We categorize each cluster by comparing the mean of each metric within cluster with their mean for the whole sample to match our typology. Interestingly, not all types in the theoretical typology were identified from our data. T-shaped in non-focal tasks (Cluster 1) and Generalist (Cluster 2) are matched to the types in our typology. Two types for novice can be distinguished (novice with zero experience and novice with very low experience): New comers (Cluster 4) / Triers (Cluster 6). However, we found two types of omniscient members (i.e., those who have specialized experience in most task areas) in our data – Cluster 2 and Cluster 5. Both clusters have high levels of experience and low levels of experience concentration. Cluster 2 shows a group of crowd members with a high level of both diverse experience and specialized experience, but a moderate level of experience compared to Cluster 5. Based on the statistics, we classify Cluster 2 as “Deep Generalist” (Hwang et al. 2014) and Cluster 5 as “Omniscient” members (Kang et al. 2012). We classify them based the pattern of process experience for better interpretation and matching but note that these types are mostly consistent for both process and domain

tasks (except slight difference for Cluster 1 and Cluster 2).

In addition, we note that Cluster 6 could be specialists in other tasks (type 6 in the typology) based on the three metrics. However, when we consider the total product level experiences of this cluster, we note that their experiences are quite limited (around 1.3). Thus, they are actually more representative for triers instead of specialist in other tasks. The reason for their high experience concentration is that the computation of *HHEI* would cause extremely high value when a member has only one or two experience in one or two task areas (i.e., one unit of experience in one area does not necessarily mean a high concentration, even though the *HHEI* computation results in an extremely high value). To further verify this, we computed a balanced measure of *HHEI* by adding 1 experience in each task area before computing *HHEI*. The clustering results can distinguish the cluster of triers with low experience concentration from specialists in other tasks. Thus, a total of six types of members were identified based on experience breadth, depth and concentration from the empirical data.

Table 4-3. Clustering Results and Corresponding Types

	Cluster 1	**Cluster 2**	**Cluster 3**	**Cluster 4**	**Cluster 5**	**Cluster 6**	**Total**
Type	*T-shaped in non-focal*	*Generalist*	*Deep generalist*	*New comers*	*Omniscient*	*Triers*	
N	28,706	38,853	74,492	96,133	20,121	20,446	274,281
ProcessDiverse Exp	3.3191	4.3665	4.757	0.0000	4.9632	1.6352	2.7401
ProcessSpecialized Exp	3.4502	7.8074	38.117	0.0000	147.2767	0.7012	22.6756
ProcessExpConcentration	0.4718	0.3031	0.2861	0.2000	0.2634	0.7389	0.3113
DomainDiverse Exp	3.0367	4.1215	7.2245	0.0000	7.9906	1.0662	3.5294
DomainSpecialized Exp	1.1055	2.801	9.7364	0.0000	45.2437	0.3108	6.499
DomainExpConcentration	0.4101	0.3349	0.2316	0.1250	0.225	0.9682	0.2837
# of Product Exp.	5.3481	12.5419	56.1804	0.0000	194.6382	1.3049	31.9702

Note: Six variables are used to cluster the crowd members. Based on the centroids of the clusters, we classify them by comparing the centers and sample means. The generated clusters are matched with the theoretical typology except *Deep Generalist*, which is newly conceptualized based on the characteristics of this cluster compared with other clusters.

4.6.2 Analysis Results

With the results of the cluster analysis, the independent variables were

operationalized as the proportion of five cluster types (out of the six identified). Thus, the proportion of T-shaped in other tasks members (*TinOther*), deep generalists (*DeepGeneralist*), generalist members (*Generalist*), Omniscient members (*Omniscient*) and Triers (*Triers*) in the crowd are the five independent variables and the largest type, the new comer type, is used as the benchmark type (i.e., as the novice type). Table 4-4 shows the descriptive statistics of key variables and the correlation matrix. Correlations between key variables were found not to be excessively high. Heteroskedasticity-robust standard errors are used in the estimation. Variance Inflation Factors (VIFs) for multicollinearity were checked and below the recommended thresholds (all below 5 with the exception of a year indicator slightly above 5) (Belsley et al. 2005; Cohen et al. 2013).

Table 4-4. Descriptive Statistics and Correlation Matrix

	Mean	SD	(1)	(2)	(3)	(4)	(5)	(6)
1. Duration	154.3	160.2	1					
2. TinOther	0.0999	0.0375	-0.24***	1				
3. Generalist	0.160	0.0644	-0.05	0.20***	1			
4. DeepGeneralist	0.277	0.106	-0.09	-0.13**	-0.34***	1		
5. Omniscient	0.0585	0.0560	-0.09	-0.06	-0.45***	0.32***	1	
6. Trier	0.0758	0.0290	0.04	0.15**	0.14**	-0.26***	-0.16**	1
7. Members	539.6	385.7	-0.06	0.21***	-0.37***	-0.01	0.41***	-0.06
8. NumProjects	4.793	1.404	0.05	0.07	-0.04	-0.35***	-0.17*	0.12*
9. HasBrainstorm	0.129	0.336	-0.10*	-0.01	-0.17***	0.17**	0.21***	-0.11*
10. InventorProducts	0.605	1.645	0.08	-0.04	-0.18***	0.06	0.10*	-0.04
11. Comments	348	296.3	-0.04	0.13**	0.04	-0.40***	-0.35***	0.17***
12. AcgIdeaInfluence	0.211	0.146	0.03	-0.08	0.13**	0.09	-0.19***	0.05
13. IdeaComments	40.84	39.46	0.03	0.01	0.03	-0.17***	-0.17***	0.10*
14. SimilarProducts	6.224	4.171	0.02	-0.08	-0.47***	0.36***	0.69***	-0.17***
15. Solution	191.8	82.71	0.05	-0.09	-0.12*	0.04	0.12*	-0.06

	(9)	(10)	(11)	(12)	(13)	(14)	(15)
7. Members							
8. NumProjects							
9. HasBrainstorm	1						
10. InventorProducts	0.06	1					
11. Comments	-0.10*	-0.06	1				
12. AcgIdeaInfluence	-0.15**	-0.05	0.06	1			
13. IdeaComments	-0.09	0.05	0.20***	0.04	1		
14. SimilarProducts	0.33***	0.15**	-0.38***	-0.32***	-0.19***	1	
15. Solution	0.09	-0.05	-0.16**	-0.15**	0.02	0.19***	1

Significance levels: *** $p<0.001$, ** $p<0.01$, * $p<0.05$

Table 4-5 presents the main analysis. We estimated our parameters progressively by first estimating a model with control variables only (Model 1) and then adding the independent variables of interest (i.e., crowd type variables) in Model

2. In Model 1, we only include the control variables. We observe that the effect of number of participants is negative and significant (*Members*: β=-0.264, p<0.01), which means larger crowd and more crowd works facilitate (i.e., shorten) product development. Attracting a greater number of crowd participants help to gather more solutions and indicate better crowd performance. As an important component of development duration, it is not surprising that the coefficient of *NumProjects* is positive and significant (β=0.125, p<0.01). The coefficient of *HasBrainstorm* is negative but not significant (β=-0.0601, ns). Although the brainstorm section could help the crowd understand the product and be an indicator of product feasibility, it does not seem to significantly affect crowd performance in product development. The coefficient of *ProductInventor* is not significant (β=-0.0137, ns), showing no effect of inventor's past success. Interestingly, the number of comments in the development projects is positively associated with the duration of development (*Comments*: β=0.147, p<0.01), suggesting that more discussions among the crowd members through comments does not lead to better performance. One possibility is that the number of comments reflects the level of consensus (or lack thereof) in the collaboration – more comments indicate that it is difficult for crowd participants to achieve consensus in the collaboration. The coefficient of *AvgIdeaInfluence* in Model 1 is negatively (marginally) significant (β=0.894, p<0.1), showing that on average a more intelligent crowd would perform better by reducing development time. In terms of the three proxy variables for product heterogeneity, they are for the most part positively associated with product development duration (*IdeaComments*: β=0.0986, p<0.05; *SimilarProducts*: β=0.366, p<0.01; *Solution*: β=0.0383, ns). A greater number of feedbacks through comments in the ideation stage indicates more discussions and suggestions, which could be related to greater product complexity, while more similar products may be an indicator for the lack of product uniqueness, which is also related to the complexity for designing a unique and novel product. The length of solution is not significant, which means the solutions from the ideator do

not affect crowd performance in product development.

In Model 2, we add the crowd member type variables. The control variables are generally stable except for *IdeaComments* and *AvgIdeaInfluence*. The lack of significance of *AvgIdeaInfluence* implies that when including the crowd members' experience and composition, the crowd intelligence level or other experiences do not matter much. In Model 2, we first observe that the type that saliently contributes to product development performance is the omniscient one. The coefficient of *Omniscient* is negatively significant (β=-2.320, p<0.05). This is not that surprising since members with high levels specialized experiences in many areas should be more knowledgeable than others. Thus, a crowd with a higher proportion of such members would perform better, which is consistent with the supportive evidence of diverse experiences and specialized experiences. Consistently, deep generalists also have positive effects on collective crowd performance (*DeepGeneralist*: β=-1.383, p<0.05). These findings suggest that H3c is supported. In addition, we note that in Model 2, the type of T-shaped experience in other tasks is helpful for crowd performance (β=-2.937, p<0.01). Members of this type have relatively broad process experience but do not have deep experience in the focal tasks. They also have very limited domain experiences. However, they do have some experience in other task areas. Such members may have stronger motivations to explore in the focal tasks and their experiences in other tasks could help them in the focal task areas (Amabile 1983; Taylor and Greve 2006). H3b is supported. By calculating the average number of tasks each cluster of members participates in product development campaign, we find that members with T-shaped experiences in other tasks perform relatively fewer tasks (i.e., more focused in the tasks) compared with generalists, suggesting a stronger tendency for exploration (as opposed to exploitation). It seems that in crowdsourced product development, crowdsourcers should not only attract highly experienced participants but also those who are T-shaped in other tasks (i.e., those with more room and stronger motivation to explore). This could be achieved by

increasing the relatedness of tasks across different task areas.

Interestingly, we do not find any other types to significantly affect collective crowd performance in our main analysis. More generalists in the crowd do not affect product development performance (*Generalist*: β=0.214, ns), which is not entirely consistent with the existing literature on generalists in the group context (e.g., Rulke and Galaskiewicz 2000; Taylor and Greve 2006). H1 is thus not supported. One possible reason may be the nature of limited communication among the crowd participants in virtual communities. Generalists are able to facilitate knowledge transfer and sharing within a group, but this benefit would only materialize if there is vivid communication among the members, which is not the case in these loosely organized crowds. On the other hand, new comers and triers may have more incentives in exploring the tasks than generalist who have already experienced many if not all of the task areas, while generalists need more time to develop deep expertise and evolve into other (more experienced) types over time (Cahalane et al. 2014). It is also worth noting that these generalists, according to our clustering results, do not have rich experience in term of domain tasks, which may also affect their value contributions to collective performance. This implies the importance of both deep knowledge and domain knowledge. The coefficient of *Trier* is also not significant (β=-1.030, ns). Compared with new comers, although triers have a little more experience in some tasks, their experiences are still limited and do not produce better submissions.

Table 4-5. Main Regression Results

	DV: ln*(Duration)*				
Variables	Model 1	Model 2	Model 3	Model 4	Model 5
TinOther		-2.937***	-2.598**	-2.832***	-2.516**
		(1.087)	(1.086)	(1.049)	(1.046)
Generalist		0.214	0.219	0.266	0.276
		(0.721)	(0.689)	(0.709)	(0.680)
DeepGeneralist		-1.383**	-1.312**	-1.414***	-1.333***
		(0.554)	(0.522)	(0.463)	(0.442)
Omniscient		-2.320**	-2.019*	-2.341**	-2.038**
		(1.062)	(1.049)	(1.046)	(1.039)
Trier		-1.030	-1.077	-1.027	-1.073
		(1.340)	(1.328)	(1.235)	(1.225)
ln(Members)	-0.264***	-0.263***		-0.266***	
	(0.0797)	(0.0874)		(0.0776)	
ln(Submissions)			-0.241***		-0.241***
			(0.0628)		(0.0559)
NumProjects	0.125***	0.103***	0.120***	0.103***	0.120***
	(0.0356)	(0.0352)	(0.0348)	(0.0319)	(0.0323)
HasBrainstorm	-0.0601	-0.0640	-0.0736	-0.0615	-0.0716
	(0.107)	(0.101)	(0.100)	(0.111)	(0.110)
InventorProducts	-0.0137	-0.00838	-0.00736	-0.0116	-0.0102
	(0.0290)	(0.0273)	(0.0268)	(0.0218)	(0.0215)
ln(Comments)	0.147***	0.134***	0.129***	0.132***	0.127***
	(0.0418)	(0.0381)	(0.0376)	(0.0371)	(0.0367)
ln(AvgIdeaInfluence)	-0.894*	-0.455	-0.534	-0.468	-0.542
	(0.481)	(0.461)	(0.458)	(0.383)	(0.380)
ln(IdeaComments)	0.0986***	0.0706**	0.0685**	0.120**	0.112**
	(0.0338)	(0.0335)	(0.0332)	(0.0560)	(0.0722)
ln(SimilarProducts)	0.366***	0.375***	0.373***	0.462***	1.006***
	(0.0728)	(0.0763)	(0.0740)	(0.113)	(0.111)
ln(Solution)	0.0383	0.0367	0.0310	0.0305	-0.0543
	(0.0254)	(0.0273)	(0.0267)	(0.0277)	(0.0274)
IMR				0.458	0.402
				(0.403)	(0.402)
Constant	3.589***	4.795***	4.854***	4.449***	4.531***
	(0.487)	(0.711)	(0.630)	(0.735)	(0.677)
Observations	425	425	425	425	425
R-squared	0.416	0.451	0.460	0.456	0.464

Significance levels: *** $p<0.01$, ** $p<0.05$, * $p<0.1$

Notes: *IMR* = Inverse Mills Ratio. Category and time dummies are included. Robust standard errors in parentheses.

4.6.3 Supplemental Analysis

Alternative Control Variable

We conduct additional analysis to verify the main results and explore further insights from these member types. We start with alternative measures in our empirical model. First, in our main model, we control the size of crowd but not the total number of contributions submitted. In Model 3 (Table 4-5), we use number of submissions instead of number of participants as a control variable to clearly account for the amount of crowd works (as the two variables cannot be included together given the high correlation of 0.988, which is attributed to the design of submission mechanism). The results in Model 3 are consistent. Second, although our operationalization of the dependent variables through an aggregation approach helps to alleviate the potential reverse effects of development duration on member experiences, such a measure of performance may not be intuitive to capture the overall efforts devoted by Quirky. Alternatively, we simply use the total duration from the first day of development to the product launch date, the results remain consistent despite the potential reverse causality issues. Third, instead of using the proportion of each cluster in the empirical model, we alternatively use number of participants in each cluster as the independent variables and the results are consistent.

Sample Selection

Another concern of our analysis is the sample selection issue. As we only focus on the product development campaigns with crowd participation and completed design, potential sample selection bias may exist in the analysis. Specifically, there are two potential sources of potential sample selection bias – the selection of crowd participation in the development (i.e., there are crowds in the product development) and the selection of complete product development (i.e., with a full duration of development). To address the selection issue, we specify a Heckman selection model (Heckman 1979) for Model 2 and 3 using 708 products with complete information in

the first stage estimation (i.e., 425 out of 708 selected for the second stage in the selection equation). To meet the identification of the selection model (i.e., exclusion restriction) (Wooldridge 2010), we utilize the characteristics of the multiple-stage crowdsourcing process and use variables from the ideation stage (i.e., the first stage of business model) in the selection equation. Specifically, we incorporate the length of overall product idea description in the ideation stage (*Description*) and the length of the problem proposed by the ideators (*Problem*) as additional first stage variables (which only affect selection). We also include inventor's successful ideas (*InventorProduct*), number of comments on the product idea (*IdeaComments*), number of similar products submitted for the product idea (*SimilarProducts*) and length of solution in the problem-solution paradigm (*Solution*). They are used in both stages since they may affect both selection and outcome. Category and time[8] dummies are also controlled in the first stage equation.

The results in Model 4 and 5 are shown to be robust to sample selection. The inverse mills ratios (*IMR*) are not significant in the estimations and other variables remain consistent after controlling the correction term. In addition, the selection equation has substantial explanatory power (*Pseudo* R^2 = 0.763), further confirming the validity of our Heckman selection model specification (Certo et al. 2016). These imply that the selection issue is not severe in our setting and the results are not affected by the correction due to selection.

Alternative Experience Portfolio

We also conduct several supplemental analyses on the measures of experience portfolios and member types. First, we consider a different specification for the clustering analysis to identify the crowd member types. Instead of separating process

[8] The time dummies controlled in the selection equation are the year of ideation (since the first stage utilizes the multiple-stage nature of crowdsourcing process) instead of the year of development used in the second stage equation.

and domain tasks for constructing the clustering metrics, we interact the process and domain tasks to construct crowd members' knowledge portfolio, which generates 40 fine-grained areas (5 process tasks × 8 domain tasks). Using the same set of metrics (i.e., diverse experience, specialized experience and concentration of experience) constructed by the new specification, we replicate the clustering analysis and regression analysis. Interestingly, the derived clusters are quite similar with those in Table 3 and the results are mostly consistent, as shown in Table 4-6. The results are also consistent with the alternative control variable (Model 7) as well as with the selection model (Model 8 and 9). We check the separate process and domain experience of members in this new set of clusters and find the distributions are very similar with the clusters in Table 4-3.

Second, in our main analysis, we construct each individual's experience for each product development campaign by the end of development (i.e., knowledge accumulate at the product level). Alternatively, we relax this assumption by measuring experiences at the project (or task) level; in other words, we assume that experience culminates into useful knowledge when a sub-project is completed rather than when the overall product development is completed. The results for both clustering and regressions are consistent with our main analysis.

Third, we note that H2 and H3a could not be directly tested from the results since the exact corresponding types did not emerge from the empirical data, even though multiple ways of constructing experience portfolios have been considered. To alleviate this concern, we conduct a theory-driven analysis and classify members into different types simply by the mean of each metric (domain, process, or domain×process-based) such that the resulting types fit the theoretical typology. The regression results with this new specification confirmed our previous findings (the effects of T-shaped in other task, deep generalist and omniscient members), but do not lend support for H2 (generalist in focal task) and H3a (T-shaped in focal task). Therefore, these two hypotheses do not receive support from our analysis.

Table 4-6. Regression Results using Interaction-based Clusters

	DV: ln*(Duration)*			
Variables	Model 6	Model 7	Model 8	Model 9
TinOther	-2.057**	-1.894**	-2.157**	-1.981**
	(0.908)	(0.898)	(0.856)	(0.848)
Generalist	0.0673	0.369	0.325	0.606
	(0.928)	(0.919)	(0.873)	(0.859)
DeepGeneralist	-1.987**	-1.985**	-2.121***	-2.096***
	(0.791)	(0.772)	(0.739)	(0.725)
Omniscient	-2.759***	-2.384***	-2.835***	-2.446***
	(0.869)	(0.842)	(0.832)	(0.820)
Trier	-0.0819	-0.0580	0.126	0.120
	(1.774)	(1.754)	(1.458)	(1.445)
ln(Members)	-0.370***		-0.375***	
	(0.0843)		(0.0721)	
ln(Submissions)		-0.310***		-0.311***
		(0.0593)		(0.0519)
NumProjects	0.111***	0.129***	0.111***	0.129***
	(0.0350)	(0.0345)	(0.0318)	(0.0321)
HasBrainstorm	-0.0517	-0.0606	-0.0463	-0.0560
	(0.102)	(0.100)	(0.113)	(0.112)
InventorProducts	-0.0100	-0.00789	-0.0142	-0.0116
	(0.0267)	(0.0260)	(0.0223)	(0.0219)
ln(Comments)	0.126***	0.121***	0.124***	0.118***
	(0.0386)	(0.0378)	(0.0372)	(0.0368)
ln(AvgIdeaInfluence)	-0.462	-0.514	-0.468	-0.516
	(0.463)	(0.459)	(0.380)	(0.377)
ln(IdeaComments)	0.0663**	0.0645*	0.137**	0.518***
	(0.0336)	(0.0331)	(0.0575)	(0.0722)
ln(SimilarProducts)	0.419***	0.406***	0.541***	0.515***
	(0.0778)	(0.0756)	(0.115)	(0.112)
ln(Solution)	0.0365	0.0314	0.0282	-0.0543
	(0.0288)	(0.0281)	(0.0292)	(0.0287)
IMR			0.643	0.576
			(0.400)	(0.400)
Constant	5.466***	5.326***	4.982***	4.870***
	(0.683)	(0.599)	(0.714)	(0.661)
Observations	425	425	425	425
R-squared	0.448	0.459	0.456	0.466

Significance levels: *** $p<0.01$, ** $p<0.05$, * $p<0.1$

Notes: *IMR* = Inverse Mills Ratio. Category and time dummies are included. Robust standard errors in parentheses.

Uncertainty of Clusters

Although we use cluster analysis to empirically explore member types in the data, there may be uncertainties from the clusters since the algorithm only maximizes the differences between clusters instead of perfectly capturing unique groups. To resolve this uncertainty, we first incorporate a control variable that captures the variances of clusters from the cluster analysis. The main results are consistent after controlling the variances. Furthermore, we consider a group average approach without clustering member types. This can also help to verify our arguments on the role of diverse experience, specialized experience and experience concentration. We calculate the experience metrics (diversity, specialization and concentration for domain, process or domain×process-based) at the group level by averaging individual experiences across group members. The results suggest that specialization plays an essential role but diversity and concentration do not. This is consistent with our insights from member clusters.

Moderation and Interaction Effects

Our research questions and main analyses mainly focus on the overall effects of different crowd members on product development outcome. However, there might be potential heterogeneity and interactions within these effects. Specifically, we consider two aspects – the role of crowd size (i.e., number of members) in moderating these effects, and the interaction effects between different member types. Empirically, we first use the size of crowd to interact with each crowd composition variable. The results suggest that only the proportion of generalist is significantly moderated by crowd size – when the size of crowd is larger, generalists tend to be more helpful to collective performance product development process (i.e., a negative interaction effect between *Generalist* and *Members*). The increased size of crowd requires more members with diverse experience to facilitate the flow of information and expertise, which is less likely to be necessary for smaller crowds. In larger crowds, generalists

are able to transfer knowledge and offer available knowledge base for better knowledge contributions and collective performance. In addition, we conduct interaction analysis between different types of participants in the crowd – the results imply that most interactions between member types are not significant except the *substitution* effect between generalists and deep generalists (i.e., a positive interaction effect between *Generalist* and *DeepGeneralist*). The effects of generalist and deep generalist tend to be weaker there are more of the other type, implying certain overlaps between them in terms of their roles in the crowd. When there are more generalist to transfer knowledge and afford knowledge diversity, the influence of deep generalist is weaker as their roles have been partially achieved by generalists.

Predicting Development Success

Although the focus of our analysis is on the performance of the crowd reflected in the duration of product development, this measure mainly captures the efficiency of product development for the firm instead of the actual value of the product. To explore how crowd participants affect the effectiveness of the developed product, in Table 4-7, we predict the success of development measured by whether a product was selected into production (*Production*) by Quirky (Bayus 2013).[9] As this is a binary variable, we use logistical regression for product effectiveness and include the same independent variables in our main analysis.

[9] Although the selection of a product into production may suggest the success of development, whether a product can go into the production stage can be decided by various factors such the actual manufacturer, materials and production cost, which is quite different from the "on paper" development process. In addition, the selection for production does not fully imply the success of the product. According to our observations and media articles about the platform, some of the products only achieved very limited sales even though production costs were high. Therefore, we only include the analysis with this dependent variable as an extension of our main analysis.

Table 4-7. Predicting Product Development Success using Clusters

	DV: *Production*			
Variables	Model 10	Model 11	Model 12	Model 13
TinOther	-1.505	-2.055	-1.686	-2.172
	(3.772)	(3.841)	(3.788)	(3.852)
Generalist	0.980	-0.393	0.982	-0.413
	(2.219)	(2.150)	(2.218)	(2.153)
DeepGeneralist	4.658***	3.544**	4.811***	3.629**
	(1.660)	(1.550)	(1.667)	(1.559)
Omniscient	5.442	4.050	5.668*	4.195
	(3.339)	(3.255)	(3.331)	(3.253)
Trier	4.720	4.737	4.843	4.816
	(4.025)	(4.065)	(4.025)	(4.067)
ln(Members)	1.571***		1.591***	
	(0.281)		(0.284)	
ln(Submissions)		1.145***		1.151***
		(0.204)		(0.205)
NumProjects	-0.00155	-0.0345	0.000736	-0.0325
	(0.104)	(0.105)	(0.105)	(0.105)
HasBrainstorm	0.301	0.358	0.303	0.360
	(0.409)	(0.406)	(0.412)	(0.408)
InventorProducts	0.0485	0.0463	0.0559	0.0512
	(0.0731)	(0.0721)	(0.0737)	(0.0728)
ln(Comments)	-0.231*	-0.202	-0.231*	-0.202
	(0.132)	(0.131)	(0.133)	(0.132)
ln(AvgIdeaInfluence)	-1.047	-1.031	-1.040	-1.034
	(1.320)	(1.338)	(1.325)	(1.341)
ln(IdeaComments)	-0.225**	-0.229**	-0.360	-0.320
	(0.104)	(0.102)	(0.229)	(0.232)
ln(SimilarProducts)	-0.143	-0.0579	-0.392	-0.222
	(0.254)	(0.252)	(0.457)	(0.459)
ln(Solution)	-0.000867	0.0275	0.0246	0.0441
	(0.0819)	(0.0826)	(0.0882)	(0.0880)
IMR			-1.284	-0.853
			(1.943)	(1.947)
Constant	-9.334***	-7.535***	-8.339***	-6.824**
	(2.299)	(2.021)	(2.838)	(2.688)
Observations	425	425	425	425
Pseudo R^2	0.168	0.169	0.169	0.169

Significance levels: *** $p<0.01$, ** $p<0.05$, * $p<0.1$

Notes: *IMR* = Inverse Mills Ratio. Category and time dummies are included. Robust standard errors in parentheses

The results suggest that only incorporating more deep generalists can increase the potential of developed products to go into production. Omniscient members (as well as T-shaped in other task), although they can facilitate the development process, cannot guarantee the final value of the product as perceived by the company. Members with very high level of knowledge depth may be bounded with their existing knowledge portfolio and be less likely to create novel solutions. Using the number of submissions as an alternative control variable (Model 11) and the Heckman selection model specification (Model 12 and 13) shows similar results. These findings suggest that crowd members may have differential effects on the development process (efficiency) and on the product value (effectiveness). Extremely deep experiences from omniscient members may help to improve the overall performance of crowd and development efficiency, but are not able to increase the possibility of creating substantially novel contributions in the crowd (i.e., creativity might be constrained).

4.7 Conclusions

In this study, we empirically investigate the role of member types in a crowdsourced new product development context. We develop a typology and corresponding hypotheses for the types of crowd members based on the diversity literature and the theoretical framework of generalists and specialists. With data on new product development at *Quirky.com*, we empirically identified six types of crowd members. Our empirical results showed that in addition to omniscient members and deep generalists, members with T-shaped experiences in other task areas may also benefit the crowd performance in terms of reducing the crowdsourced product development duration. Interestingly, generalists were found not to be very valuable, especially for smaller crowds.

4.7.1 Theoretical Contributions

Our study contributes to several aspects of theory and literature. First, this study

contributes to the literature on crowdsourcing and open innovation. As an important partner for open innovation to organizations, the value of the crowd has been recognized, but a large body of the crowdsourcing and innovation literatures focuses primarily on collection of ideas, such as crowdsourcing contest and idea generation, and examining the motivations, antecedents and consequences (Boudreau et al. 2011; Hou et al. 2011; Yang et al. 2010; Yang et al. 2009). However, as an emergent form of organizing crowdsourcing, the collaboration approach and crowd co-creation are still understudied (Pedersen et al. 2013). These new forms of crowdsourcing have led to new business models for open innovation and innovation community and have enabled new ways of organizing the crowd to create tangible new products. Our study examines such a new form of organizing crowdsourcing and empirically identifies the important drivers of success in crowdsourced new product development. The collaborative crowdsourcing process makes use of the interdependence among tasks and participants to harness collective intelligence. The concept of crowd-creation and collective intelligence have attracted much attention in crowd-based contexts (e.g., crowdsourcing and crowdfunding) (Avital et al. 2014; Nickerson et al. 2014). Our study serves as one of the initial empirical works examining the key components in crowd-creation and unpacks the value co-creation process in innovation communities.

Second, our study contributes to the literature on online innovation communities. We identified the important types of crowd members in a crowdsourcing community where community members work in both collective and collaborative manner. We find that generalists do not improve crowd co-creation works in a large virtual group (crowd) context possibly due to limited needs for (or natural occurrence of) information transfer and communication. Members with T-shaped experience in non-focal tasks were actually found to facilitate the product co-design / co-creation process in crowdsourced new product development. These findings provide further insights related to the impact of innovation community on value co-creation (Antorini et al. 2012). We suggest that when using open innovation

community for value co-creation, it is important to attract experienced members and facilitate communication to increase the values of other participants. In addition, a larger size of the crowd in the community would be preferred for value co-creation.

Third, we contribute to the theoretical perspective on generalists and specialists, and extend the diversity literature to large and dispersed online groups. In traditional group work such as software development, T-shaped experience distributions with specialized experiences in the focal knowledge domain is typically preferred (Kang et al. 2012; Narayanan et al. 2009). Furthermore, groups composed of generalists was also shown to exhibit better performance (Rulke and Galaskiewicz 2000). However, in our context, generalists were not found to be a particularly beneficial type (only partially effective in large crowds). When a generalist reaches the limit of knowledge areas, she may have a weaker motivation to improve performance and needs more time to further explore the depth of her knowledge portfolio to evolve into a more experienced type. Meanwhile, the lack of domain knowledge may make generalists less influential. Furthermore, limited communication may also lead to unexpected implications. In the diversity literature, communication and coordination cost are discussed for large groups but there is a lack of empirical examination (Taylor and Greve 2006). We extend the theory about diversity in large and loose online groups and find that diversity may only have limited influence for innovation and creativity in the absence of knowledge depth. In addition, a T-shaped member in other tasks may be an important type in online crowdsourced groups. Specialized experience may have the potential to transfer from other areas to the focal area when diverse experience is sufficient for such a transfer. Such a potential of transfer is in line with stronger motivation of exploration, which could induce better collective performance in innovation. Finally, our results show that only diverse experience may not work in our context. Only with specialized experiences in various task areas could members collectively produce better performance, which is consistent with Hwang et al. (2014) in the context of

individual idea innovation. However, highly specialized experience may only be helpful for the efficiency of innovation process but not for potential product effectiveness.

4.7.2 Practical Implications

In terms of practical implications, we empirically uncover the collaboration and crowd co-creation process in crowdsourcing and open innovation community. Firms which frequently crowdsource works to community members or seek crowd co-creation from a community should attract more experienced members in both diverse knowledge and specialized knowledge by increasing the variety and specificity of crowdsourcing tasks. Community members with only diverse experiences may not be very useful. Firms also need to pay attention to those who have T-shaped knowledge in other areas by designing tasks with high relatedness and interactions along with a more flexible crowd co-creation process. Second, by attracting more experienced members and the right type of members, firms could also spend less effort in assimilating the crowdsourced works because of better crowd performance. Third, it may be important for the designer of crowdsourcing tasks to understand the members' balance between task exploration and specialization in crowdsourcing to guarantee the innovativeness and engagement of members.

4.7.3 Limitations

Our study has several limitations that require further investigations. First, our empirical results suggest that participants tend to explore different types of tasks broadly to accumulate experience. It would be meaningful to further examine this exploration process for diversity to get deeper understanding of community participation. Second, the analysis of participation patterns of community members generated six clusters from the empirical data, which did not perfectly fit the theoretical typology. Future research may attempt to comprehensively investigate this typology in different contexts to enrich our knowledge of crowd participants. Third,

we mainly focus on the group level crowd performance and firm-level efficiency/effectiveness. Future research can also test individual level performance and compare individual level contribution quality with group level innovation outcomes. Lastly, although our study takes the initial step on how to organize community members in value co-creation, our results suggest the room of further examination on factors affecting crowd-creation process, such as member interaction and product complexity.

CHAPTER 5 ESSAY II – CAN I TOUCH YOUR CODE? THE EFFECTS OF PROGRAMMING STYLE ON OPEN SOURCE COLLABORATION

5.1 Abstract

Open source software (OSS) development has recently garnered much attention from both industry practitioners and academic researchers. However, existing research on OSS usually focus on the role of behavioral factors in affecting collaboration outcomes but has neglected to critically consider the nature of the artifact (i.e., the software) itself. In this study, we seek to integrate collaboration factors and software factors in extending our understanding of OSS collaboration. Specifically, we investigate the role of programming style in open source collaboration, where strict guidelines for coding are typically not enforced. We develop three implications of programming style on contributor participation, development efficiency and OSS diffusion from a diversity perspective. Additionally, two team level factors (i.e., team familiarity and developer experience) that moderate the negative effects of programming style are discussed. We also examine how project teams can effectively control coding styles for collaboration. With a list of metrics identified from the literature and industrial standards, we quantify programming style for both within file inconsistency and across file consistency. The empirical analysis suggests that style inconsistency can exhibit negative effects, but mainly through within file inconsistency, and on contribution activities. We also find that team familiarity can alleviate the negative effects, but developer experience unexpectedly intensifies them. In addition, the practice of project control through coding standards is found to only reduce within file inconsistency. Our study contributes to the literature on OSS development, software engineering and diversity in distributed work groups, and offers practical insights for open source software teams.

5.2 Introduction

Open source software (OSS) development has witnessed much popularity and growth in recent years (Hahn et al. 2008; von Krogh and von Hippel 2006). The open source production model transforms software development process from a proprietary development mechanism into an open collaboration model where developers from anywhere in the world are able to collaborate with each other and create new software products (Fitzgerald 2006).

In the OSS collaboration model, developers from diverse backgrounds and with different knowledge are able to contribute and collaborate in the development of software. However, it is commonly acknowledged that every developer exhibits unique characteristics in writing computer programs (Graham 2004; Reiss 2007). Such personal characteristics, if not reconciled, may be harmful for collaboration, which is similar to the situation where two authors with different writing styles have difficulties in effectively collaborating on co-authoring an article. This may not be an issue with proprietary software development in traditional organizations as organizations typically enforce strict and detailed coding guidelines (Shah 2006). But this may not be the case with OSS. Without monetary rewards, open source developers work for free (Hars and Ou 2002), and their primary motivations are enjoyment, learning, reputation and code sharing (Roberts et al. 2006; Sheoran et al. 2014). Such self-organized new organizational forms usually cannot impose specific rules for developers so that individual characteristics can easily get embedded in their code contributions (Crowston et al. 2007). These personal preferences in code writing may make it difficult for team members to understand others' contributions and also undermine subsequent collaboration and development efforts. However, the existing literature on OSS development generally focuses on the behavioral factors on collaboration without much attention paid to the characteristics of the source code. As the core of the software product, the source code is the conduit for developers to

understand the task, conduct maintenance and promote the software (Mohan and Gold 2004). Its importance is even further intensified by the transparency of source code on today's social coding platforms (Dabbish et al. 2012).

According to the literature on programming languages, one of the most intuitive and visible individual characteristics in code writing is programming (or coding) style (Arabyarmohamady et al. 2012; Caliskan-Islam et al. 2015). Programming style goes beyond the grammar of the programming language and also captures stylistic elements (Caliskan-Islam et al. 2015). A poor coding style usually impedes the effective comprehension of the source code, which is an important precursor to software team collaboration and software maintenance (Buse and Weimer 2010). In addition, developers receive different training in programming which cultivates their own preferences in writing code (Soloway and Ehrlich 1984). They usually develop a strong sense about what the code should look like and find it difficult to comprehend source code written with different styles (Spinellis 2011). Although style guides in different languages have been developed and organizations (including some large OSS projects) have enacted guidelines for development, it is not common practice for OSS projects to enforce programming style requirements in the collaboration process. Therefore, different programming styles may coexist within a project's source code. The objective of this study is to *explore the implications of different programming styles in the source code on open source collaboration*.

Several reasons highlight the importance of understanding the role of programming style in OSS development. First, in open source collaboration, which is characterized by free style collaboration, it is important for project teams to alleviate the potential consequences of programming style if poor coding style can hamper collaboration efficiency. Second, team members not only directly interact with each other, but also indirectly through the software product itself (which may actually be the preferred approach). Understanding the role of software itself in the collaboration process can advance our knowledge on deeper level collaboration mechanisms in

OSS development. Third, analyzing the styles (as well as other metrics) in source code can reveal how developers contribute to open source projects from a more nuanced perspective, and help to facilitate the sustained development of the open source ecosystem. Therefore, exploring the implications of programming style can be useful for understanding the deeper level collaboration process and the development of community in open source context.

To understand the implications of programming style both theoretically and empirically, we first conceptualize differences in programming styles in the source code as "separation" from the diversity literature. Anchoring on this theoretical perspective and the software engineering literature, we first develop the core hypotheses on the overall implications of programming style on collaboration, development and diffusion of OSS. Two mechanisms from the source code and group cognition are discussed to develop the hypotheses. Then we explore the possible team level factors and strategies that can help to alleviate the challenges of different programming styles – the role of network connections in the team, developers' experience and project control (i.e., coding standard).

We test these hypotheses using data and source code collected from GitHub. We quantify differences in programming styles by the inconsistency of programming style within and across code files using large-scale static code analysis. Through econometric models at the project month level, we find that style inconsistency exhibits negative effects mostly through within file inconsistency and on contribution activities (so no clear effects on other collaboration indicators). The negative effects can be alleviated by team familiarity but unexpectedly intensified by developer experiences. In addition, by utilizing a quasi-experiment setting of enacting coding standards, we find that project control is effective for decreasing within file inconsistency but not for across file inconsistency. Our study contributes to the literature on OSS development by highlighting the importance of the software artifact in open collaboration and the interaction between software factors and behavioral

factors. We also provide insights for software engineering research on the role of programming style, and group diversity literature in terms of work style diversity and different opinions (i.e., separation) towards the product. OSS teams can benefit from our study on managing programming style in the code repository and leverage team governance mechanisms together to achieve higher collaboration efficiency.

5.3 Literature Review

5.3.1 Open Source Collaboration

Researchers in OSS development area have investigated various aspects in this emergent collaboration model. Examined topics include the developer's social network (Hahn et al. 2008; Oh and Jeon 2007; Singh 2010; Singh and Phelps 2013), team structure (Daniel et al. 2013; Grewal et al. 2006; Singh and Tan 2010; Singh et al. 2011), individual motivation and learning (Ke and Zhang 2009; Roberts et al. 2006; Singh et al. 2010; Zhang et al. 2013) and the value of the OSS model (August et al. 2013; Boudreau 2010; Zhu and Zhou 2011). In summary, from the developer perspective, OSS developers are shown to form project teams and choose project licenses based on their network connections. They learn by contributing to projects and extending the source code. From the project perspective, social capital, network position and team structure have been shown to affect the technical and commercial success of OSS projects. From a broader perspective, researchers compared the OSS model with the proprietary development model, and investigated the OSS ecosystem (Haefliger et al. 2007; Levine and Prietula 2013).

However, the extant literature has yet to delve into the nature of the software product itself when examining project collaboration and performance. It has been documented in the software engineering literature that the examination of source code and software metrics is an important aspect in understanding software outcomes (Capiluppi et al. 2009). In this study, we seek to fill this gap by integrating behavioral factors and software factors in understanding open collaboration.

5.3.2 Programming Style

Programming style describes a developer's preferences in writing source code and efforts to make source code easy to understand (Kernighan and Plauger 1978). Existing studies have employed the stylistic traits of source code in various contexts including programming education (Moghadam et al. 2015; Ohno 2013), plagiarism detection (Arabyarmohamady et al. 2012), program authorship identification (Caliskan-Islam et al. 2015), and software engineering (Lee et al. 2013b; Smit et al. 2011b). This study draws upon the software engineering area relating to source code comprehension and software maintenance (Binkley et al. 2013; Mi et al. 2016; Miara et al. 1983).

Research in software engineering has highlighted the importance of programming style (Prause and Jarke 2015; Reed 2010). Specifically, a series of studies have documented the linkage between coding style and important software metrics (e.g., readability, maintainability and quality). Good programming style increases readability and understandability of the code, facilitating code comprehension (Lee et al. 2013b; Oman and Cook 1988). Source codes with better (consistent) style are also easier to maintain in the long run (Buse and Weimer 2010; Prause and Jarke 2015). In addition to comprehension and maintainability, programming style is also related to software quality (Capiluppi et al. 2009; Smit et al. 2011b). However, Boogerd and Moonen (2008) claimed that not all the programming style standards are helpful in reducing software faults. Smit et al. (2011a) also evaluated coding style conventions and found that not all the conventions are equally important according to software engineers. Moreover, some studies in software engineering attempt to quantify programming style by tracing the fingerprint of developers (Caliskan-Islam et al. 2015; Frantzeskou et al. 2006; Mi et al. 2016; Mohan and Gold 2004).

Although existing works have examined programming style, their focus has usually been on the impact on software itself instead of the collaboration process. However, due to the possible implications on collaborative software development processes, especially given the voluntary contribution nature of OSS, we focus our attention on the role of programming style in open collaboration.

5.4 Hypotheses Development

Given the concerns of programming style in the open source community, we develop hypotheses about the impact of different programming styles on open source collaboration. We first intend to examine how programming styles can affect open source collaboration. Drawing on the theory of diversity, more specifically *diversity as separation* (Harrison and Klein 2007), we discuss three implications of programming style on open source collaboration and development. Specifically, there are two mechanisms to explain the effects of programming style, from the software engineering perspective and from the diversity as separation perspective, which we term as the *material* mechanism and the *cognitive* mechanism, respectively. The material mechanism comes from the software development process where inconsistent programming styles increase the efforts in software comprehension and maintenance. The cognitive mechanism occurs with cognitive conflicts in terms of work styles across developers when different styles co-exist, which subsequently decreases the intention to collaborate and contribute. Then, we explore how to alleviate the challenges arising from inconsistent programming styles in OSS development by proposing hypotheses on the role of team familiarity, developer experience and project control in shaping the impact of programming style. In these hypotheses, project control serves as an antecedent of programing style, while the other two are proposed to moderate the effects of programming style by mitigating the cognitive mechanism and the material mechanism, respectively. Figure 5-1 shows the basic framework of this study.

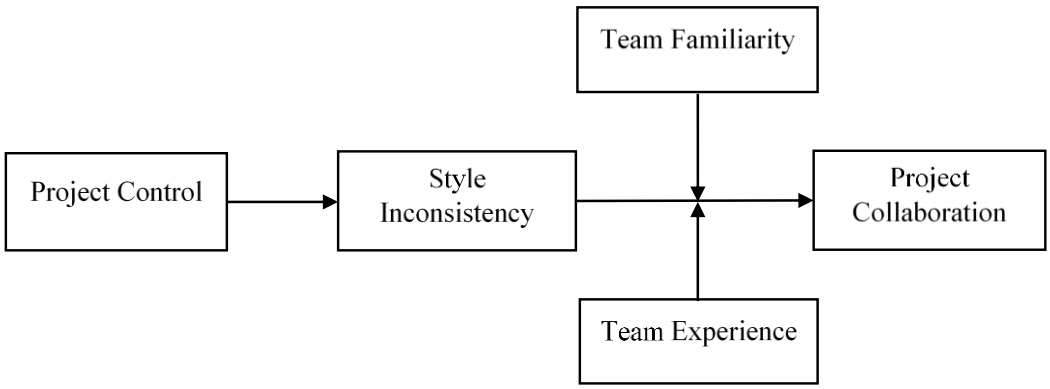

Figure 5-1. A Brief Research Framework

5.4.1 The Effects on Contributors

Contributors are important actors in the OSS community. It is necessary for open source projects to keep attracting developers to sustain development (Fang and Neufeld 2009). Existing studies have examined the roles of various behavioral factors in developer's participation and contribution (Grewal et al. 2006; Hahn et al. 2008; Singh et al. 2011). Here, we claim that the nature of the source code will also affect the participation and contribution process. Drawing on the OSS literature, we discuss the role of programming style in both potential contributors' and existing contributors' participation.

In open source communities, a developer who wants to contribute to a project not only considers the social or behavioral factors but also needs to evaluate the project based on its functionality, maturity or scale (Hahn et al. 2008). One of the most important tasks during the evaluation is to understand the source code so that the developer is able to modify the code for contribution (Prause and Jarke 2015). If the current project has inconsistent programming styles (due to the contributions of multiple developers with different styles), this implies divergence in developers' opinions about how the software should be coded (i.e., how to write the source code to implement functions) and there are no clear norms within the project team (Bechky 2003). Developers in the team keep their personal opinions about what the source

code should look like and there are no rules to regulate any conflicts. Thus, inconsistent programming style not only makes it difficult for potential contributors to understand the source code but also signals a potential lack of conflict resolution in the project team (Kankanhalli et al. 2006), which creates barriers to entry and may reduce intentions to participate and contribute. Conversely, a consistent programming style in the source code facilitates software comprehension and thereby lowers the barriers for new contributors to participate. Hence, we propose:

Hypothesis 1a: *Programming style inconsistency is negatively associated with the number of new contributors in open source projects.*

To collaborate on a project, existing members also need to comprehend the evolving source code. Source code with diverse coding styles will be more difficult to understand and impede subsequent contributions (Prause and Jarke 2015). In addition, team members may feel uncomfortable when there is strong individualism or cognitive separation within the team (Earley and Mosakowski 2000). If developers insist on keeping their own style preferences when writing code, there could be salient cognitive conflicts in the collaboration process. The literature on diversity suggests such conflicts can lead to social categorization and undermine group integration (Schneider et al. 1995). Developers not only face heavier load on code comprehension and maintenance, but also have less intentions to collaborate. Thus, developers will be less like to be engaged in the collaboration and become less active due to the inconsistent programming styles in the source code. We propose:

Hypothesis 1b: *Programming style inconsistency is negatively associated with the activeness of existing contributors in open source projects.*

5.4.2 The Effects on Development Process

Another important aspect of OSS collaboration is the software development process. Research in OSS has investigated various indicators for the success of software development, such as commits and downloads. However, these indicators mainly capture the engagement of community and end user interest rather than the process of

software development and enhancements. In the present study, we focus on the release of open source software, a repeated activity in the development life cycle (Lerner and Tirole 2002). To make a software release, the project team needs to make the software both technically and functionally acceptable to the community (Godfrey and Tu 2000; Lakhani and Von Hippel 2003). Thus, the *event of release* captures the success of the development process (Mallapragada et al. 2012).

It has been documented in the software engineering literature that programming style is an important element for software quality and maintainability (Capiluppi et al. 2009; Smit et al. 2011b). Therefore, poor programming style implies that software may encounter issues and difficulties in testing and maintenance. This will lead to more fixing works and more time to create new features, resulting in a delayed release. In addition, when there are multiple styles in the source code, it may take the project team a longer time to unify the individual 'fingerprint' in the software and integrate the resources for a release (van Knippenberg and van Ginkel 2010). It requires the project team to reconcile the dissimilarities in contribution style to ensure a maintainable version of the software. Conversely, a consistent programming style will make it easier for the team to fix bugs and incorporate new features (Mohan and Gold 2004). Different programming styles, as exhibited by diversified work styles, can lead to greater cognitive conflicts and lowered productivity (van Knippenberg and Schippers 2007), which also increases the time to achieve important milestones such as the software release. Taken together, we hypothesize:

Hypothesis 2: *Programming style inconsistency is negatively associated with the release of open source projects.*

5.4.3 The Effects on Project Diffusion

Open source project diffusion refers to the attention of the project in the broader developer community and the reuse of source code by other developers (Zhang et al. 2014). The diffusion of projects is important for further collaboration and sustainability of the open source community (Zhang et al. 2014) and thus it is a key

aspect of the OSS ecosystem. For example, on GitHub, developers can fork or watch other projects for knowledge improvement and/or further extension (Dabbish et al. 2012; Sheoran et al. 2014).

Prior research has suggested that open source developers pay attention to projects because of contribution and code sharing (Kalliamvakou et al. 2014a). Projects mainly receive attention from those developers who have the intention to contribute due to interest in the source code. These developers are potential contributors and modifiers as well as issue reporters for new releases (Sheoran et al. 2014; Zhang et al. 2013). Thus, garnering more attention from external developers is beneficial for project collaboration and sustained improvements. However, inconsistency in programming style will lead to more efforts in code comprehension and imply conflicts within the team. For external developers, it is difficult to clearly understand the source code in such project repositories (with different coding styles) and track the evolution of the software development (Blincoe and Damian 2015). Thus, external developers would perceive less potential for further usage and contribution, which in turn reduces the attention received by such projects. In a similar vein, the low maintainability and different code writing opinions as a result of inconsistent programming style will make the source code less readable and reusable. The diversity literature suggests that it is difficult to integrate and organize resources in groups with high separation (He et al. 2007), which impedes team functioning and further progress (Harrison et al. 2002). Compared with projects without style control, projects with clear and consistent programming style are more likely to be followed by developers in the community for further opportunities and be re-developed by external community members. Taken together, we propose the following hypotheses about project diffusion:

Hypothesis 3a: *Programming style inconsistency is negatively associated with external attention of open source projects.*

Hypothesis 3b: *Programming style inconsistency is negatively associated with the extent of code reuse of open source projects.*

5.4.4 The Moderating Role of Team Familiarity

We now focus on the characteristics of the project team that may help to resolve the issues arising from programming style inconsistency. These factors usually relate to within team collaboration and development processes instead of to new contributors and project diffusion which are more pertinent to developers in the whole open source ecosystem external to the focal OSS project. Thus, our hypotheses will be developed on the basis of collaboration and development aspects discussed above.

We first examine the role of team member familiarity in open source collaboration. Team members who have prior connections with each other are more likely to collaborate with each other (Hahn et al. 2008) and resolve the conflicts in the group (Cannella et al. 2008). Given the two mechanisms of inconsistent programming style (material- and cognition-based), the cognitive process may be curtailed when team members are more familiar with each other. Although the inconsistency of style in source code can lead to greater efforts in comprehension and maintenance, the intention to collaborate and contribute is less likely to be decreased when team members know each other very well. Thus, the negative effects of programming style inconsistency on activeness of developers should be weaker in projects where members are familiar with one another. Similarly, even though developers still need to exert great efforts for maintenance to release the project given different programming styles, the cognitive process will not be affected if team members are familiar with each other. They are more likely to trust other developers in the collaboration and tolerate the different style of others if they know each other well. This suggests that the negative effect of inconsistent style on project release can be weaker with greater team familiarity.

In open source communities or other innovation/collaboration communities, two types of connections are typically documented – collaboration ties and friendship

ties (Hahn et al. 2008; Moqri et al. 2015; Oh et al. 2015). Collaboration ties refer to the collaboration experiences among project members in other projects, while friendship ties capture the extent to which team members know each other. Both of them suggest the familiarity among team members (Huckman et al. 2009) so that they are able to resolve the concern of inconsistent coding style by mitigating the cognitive process of separation. Thus, we propose the following hypotheses:

Hypothesis 4a: *The negative relationship between programming style inconsistency and the activeness of developers is moderated by the familiarity of team members such that the negative effect is weaker when team familiarity is greater.*

Hypothesis 4b: *The negative relationship between programming style inconsistency and the speed of project release is moderated by the familiarity of team members such that the negative effect is weaker when team familiarity is greater*.

5.4.5 The Moderating Role of Developer Experience

Another important team characteristic is the overall experience of developers in the project team. Developers accumulate their experiences and expertise in writing code by contributing to other projects or working on their own projects. Their experiences not only help them to write clearer source code but also facilitate their ability to comprehend code (Evangelist 1984). Given their expertise, experienced developers are less likely to be affected by inconsistent programming styles in the source code when working on program comprehension and software maintenance (Binkley et al. 2013; Miara et al. 1983; Woodfield et al. 1981). This implies that the material mechanism through code comprehension and software maintenance should not be as salient when developers in the team are experienced. Style inconsistency, therefore, would not affect their comprehension of the source code and they can still work on the code without incurring too much extra effort. Thus, the activeness of experienced developers in the team is less likely to be affected by style inconsistency. Similarly, experienced developers can efficiently work on the source code with inconsistent

styles for project release. Their experiences help them comprehend the source code and fix the issues efficiently. They are more likely to make progress toward releasing projects regardless of the inconsistent programming style. This leads to the moderating effect on the relationship between style inconsistency and project release. Hence, we propose:

Hypothesis 5a: *The negative relationship between programming style inconsistency and the activeness of developers is moderated by the experience of team members such that the negative effect is weaker when developer experience is greater.*

Hypothesis 5b: *The negative relationship between programming style inconsistency and the speed of project release is moderated by the experiences of team members such that the negative effect is weaker when developer experience is greater.*

5.4.6 Project Control as Antecedent of Programming Style Inconsistency

Our previous discussions have focused on the consequences of inconsistent programming style. However, besides weakening the effects of programming style through team formation, it is also important for project teams to reduce the style inconsistency more directly. This usually can be attempted by strict project control, which is similar to what proprietary software development organizations do. The common practice for such strict project control is to enact a style guide or coding standards to team members so that team members can follow the guidelines to avoid style inconsistency. Thus, we regard project control (i.e., the use of coding standards) as an antecedent of programming style consistency in the project.

With the enactment of a coding standard, project team members will be aware of the requirements of programming style used in the project. Contributions that deviate from the coding standard may be rejected or further revised by the project owner (or core developers), and members will try to maintain the norms built in the project (Li et al. 2016). Strict project control implies efforts toward stability in collaboration among diverse individuals and forms routines in code writing for

developers to follow. In their subsequent contributions, developers would be more careful about the style they use in the source code and try to adhere to the guidelines. They would also attempt to adjust the existing coding styles in the project and then make it more consistent based on the enacted coding standard. Thus, we argue that there will be two influences of project control practices on programming style inconsistency. First, there is a short-term effect where the style inconsistency decreases after the implementation of coding standards. This is because developers may adjust the style in the existing source code to the required style and project members may also have the intention to reduce style inconsistency in the existing code after the release of coding standard. Second, there is also a longer-term effect on the subsequent contributions to the projects. The extent of programming style inconsistency will be lower in the time periods after the release of the coding standards compared with projects lacking a coding standard. Taking these together, we propose the following hypotheses:

Hypothesis 6a: *Project control is negatively associated with style inconsistency in short term.*

Hypothesis 6b: *Project control is negatively associated with style inconsistency in long term.*

5.5 Research Context and Data Collection

Our research context is open source software projects on GitHub, the largest platform for OSS development in the world. It builds on Git – a distributed version control system – to enable source code sharing and collaboration. As of June 2018, GitHub has over 28 million users and 57 million repositories (i.e., projects). In addition to codebase storage and source code sharing, GitHub provides a series of features to facilitate team collaboration, project management and social interaction (Choi et al. 2013; Yu et al. 2014). It not only supports team-based software development but also enables external developers to contribute and collaborate (Gousios et al. 2014).

On GitHub, the source codes of all public repositories are publicly accessible. Similarly, all activities including commits, code reviews and discussions are also publicly available. Such transparency enables a social coding process and developers can easily evaluate the quality of project by checking the source code and other information directly on the platform (Dabbish et al. 2012; Lee et al. 2013a). Therefore, in such an open and transparent environment, how the source code looks may be critical for projects and programming style is one of the most salient elements exposed to community members. This research site allows us to investigate the coding style of publicly visible source code and its implications on OSS collaboration.

We construct our dataset from GitHubArchive and GHTorrent (Gousios 2013). These two datasets provide a large amount of metadata from GitHub starting from 2012. They log all events including commits, issues, pull requests, releases and social interactions on all publicly accessible repositories. This enables us to track all activities and the source code of all versions across the software development life cycle. In addition to the metadata, we also extract detailed commit-level and file-level data directly from GitHub using the GitHub API (Kalliamvakou et al. 2014b). Among all the projects in our dataset (over 12 million non-forked projects), we limit our scope to projects using the JavaScript programming language which were initiated after 2012[10] as the base project pool (over 2.22 million projects). Several reasons guide this selection. First, JavaScript is the most popular language on GitHub, offering a greater number of available projects for empirical analysis. Second, JavaScript has richer stylistic elements in writing source code compared with other popular scripting languages such as Python or Ruby. For example, JavaScript uses

[10] Pre-2012 data is not available on GitHubArchive or GHTorrent; therefore, limiting the dataset to post 2012 ensures that all activities and versions are fully captured. Also, GitHub witnessed significant growth in 2012 in terms of the number of project repositories on the platform, suggesting the theoretical feasibility of using data from 2012. Nevertheless, some traceable data before 2012 are used for specific measures (e.g., developers' familiarity and experiences).

braces for program flow control, while Python and Ruby use indentation to control the structure, which reduces the number of style elements in the code. Third, it is usually difficult to compare across programming languages, especially for comparing programming styles, since each language has its own coding logic.

We constructed our data in October 2016 with all projects and related information (e.g., commits, issues and pull requests) from the inception of GitHub, and mainly focus on projects initiated between 2012 and 2014 to guarantee sufficient development duration for empirical analysis. We further excluded those individual projects and projects that are not continuously developed from our dataset,[11] consistent with the focus of our study. The identification of collaborative development projects left us with a total of 5,589 projects. To rule out the influence of potential governance mechanisms that may affect coding preferences, we did not include projects associated with organizational accounts.[12] After removing projects that have been deleted by the owner, we were left with a total of 2,281 collaborative projects for empirical analysis.

5.6 Empirical Method

5.6.1 Measures of Programming Style

One of the main challenges we need to address is how to quantify and operationalize programming style, i.e., the inconsistency of programming style. We follow the existing literature in programming languages and software engineering on the quantification of coding style, and also reference some industry standards (i.e., style guides). Specifically, we adopt the "attributes counting" (or rule/metric-based) approach for assessing style (Lee et al. 2013b; Mi et al. 2016; Smit et al. 2011a). The

[11] Although these projects are excluded from our sample, they are used to measure the variables such as team familiarity and developer experience, which are discussed in section 5.6.2.

[12] An organizational account is a type of higher level account that can govern all projects and developers together under the same organization. If a project belongs to organizational account, it may be enforced with certain governance mechanisms from the organization leading the project.

attributes are usually based on some metrics that describe the characteristics of the source code and constitute the fingerprint of developers. They mainly consist of three categories: programming format metrics, programming readability metrics and programming language metrics (Mi et al. 2016). *Programming format metrics* include those attributes related to the physical layout of the source code and the usage of white spaces. Two code segments can be lexically identical but different in format. Attributes such as indentation, alignment and braces belong to this category. *Programming readability metrics* relate to attributes that represent the degree of readability without affecting the function and efficiency of source code. Naming style and the usage of comments are examples of this category. *Programming language metrics* represent the preferences of developers to implement the functionalities in the code, such as looping structure and the usage of keywords in a specific language.

To quantify these metrics and construct variables for analysis, we first identify different scenarios for each metric and then compile a vector that captures every scenario (by indicators) in all metrics. The style vector is built at the source file level. Although there could be more fine-grained levels to construct the vector (e.g., at the method or function level), it is not meaningful to extract fingerprints at these levels due to limited code structure information. Also, it is difficult to build the vector at developer level since each file in an open source project can be edited by multiple developers. Therefore, we set our granularity at file level, but we do track the source of different programming styles (by comparing file level style inconsistency across versions). In addition, it is necessary to have enough lines of code to capture meaningful fingerprints; otherwise, the vector will be sparse (i.e., filled with many zeros) and will not guarantee a reliable measure (Mi et al. 2016). The vector is defined as $(a_1,a_2,...,a_{k_a},b_1,b_2,...,b_{k_b},c_1,c_{12},...,c_{k_c},...)$, where a_1 to a_{ka} represent the indicators for metric *a* (where each element is a scenario for metric *a*). These indicators calculate the numeric values (e.g., percentages, average numbers or binary

indicators) of each scenario. The groups *a*, *b* and *c* suggest different attributes to quantify programming style. The subscripts k_a, k_b and k_c specify the number of common scenarios for each metric. For example, if metric *a* is about the attributes of curly brackets in the code, a total of k_a scenarios (e.g., proportion of brackets at the start of source code line and proportion of brackets at the end of line) will be identified and the indicator of each scenario is calculated (see Table 5-1 in section 5.7.1 for the list of metrics adopted). Such a vector can capture the common stylistic elements in the source code and makes it easy for further operationalization. Given the potential differences on scales among metrics, all the vectors will be standardized before further calculations.

After obtaining the style vector based on the list of metrics, we then operationalize a set of measures that can represent the difference or inconsistency in programming style. We start from file-level measures and then construct project level measures for further econometric analysis. Two major groups of measures can be constructed: programming style inconsistency and number of different programming styles. At source code file level, we define *within file inconsistency* as the extent of programming style inconsistency within a source file. For each metric, the extent of concentration for some specific scenario indicators captures whether the source code style is consistent for that metric (i.e., high concentration suggests consistent style, while low concentration represents inconsistent style). Note that this is only applicable for indicators that substitute each other within a specific metric since only some indicators are able to capture within file inconsistency. For example, indicators for the proportion of different naming styles can capture within file style inconsistency, but average length of variable names cannot capture within-file inconsistency but only across-file inconsistency. The Herfindahl-Hirschman index (Harrison and Klein 2007) is used to measure the extent of concentration within a metric. Then, file level inconsistency can be aggregated to the project level. In addition to within-file inconsistency, we define across-file inconsistency as the

differences of programming styles across source code files. A file can be internally consistent in style but be different from other files. The style inconsistency between two source files can be calculated using the cosine distance between the two code fingerprint vectors. All the pairs of similarity can be aggregated at the project level. Specifically, within file inconsistency and across file inconsistency at file level and project level are specified as:

$$WithinFileInconsistency = [\sum_i^K (1 - \sum_j^{k_i} (a_{ij} / \sum_j^{k_i} a_{ij})^2)] / K$$

$$AcrossFileInconsistency = 1 - \frac{\sum_i \sum_j a_{im} b_{im}}{\sqrt{\sum_i \sum_j a_{im}^2} \sqrt{\sum_i \sum_j b_{im}^2}}$$

$$ProjectWithinFileInconsistency = \sum_i WithinFileInconsistency_i \ / \ N$$

$$ProjectAcrossFileInconsistency = \sum_i [(AcrossFileInconsistency_i{}^2 \ / \ N]$$

Following a similar specification pattern in the style vector, in the file-level definition, *a* and *b* represent the elements in the fingerprint vector with *i* denoting the metric and *j* denoting the sub-metric indicator. k_i represents the number of available indicators that substitute each other for metric *i* (note that the equation for within-file inconsistency is only applicable for complementary indicators under a specific metric) and *K* is the total number of metrics. At project level, *N* denotes the total number of files in the project and we use the standard deviation approach (Harrison and Klein 2007) to capture the deviation of programming style distances from file *i* to the mean of all files (denoted by *m*). These measures allow identifying the situations on both the within-file and across-file inconsistency in the source files.

From an empirical consideration, we compiled a customized python program based on the automatic code comprehension approach in Closure Linter[13] from Google. This approach takes a tokenization method to interpret each element in the code sequentially and stores the structure of source code in a set of tokens. Then we follow both the rules used in Mi et al. (2016) and some other important rules in

[13] Original source code of Closure Linter can be found at: https://github.com/google/closure-linter

popular JavaScript style guides such as the Airbnb JavaScript Style Guide and the Google JavaScript Style Guide.[14] For each coding style metric, we scanned the tokens generated by the source codes and obtained indicators by checking the code lines around each token.

5.6.2 Econometric Specification

We first build a project-month level econometric model to test our hypotheses on the consequences of programming style (H1 to H5). We use projects that do not have coding standards to avoid differences across those projects. We construct our key variables at the project level across calendar months. To resolve the potential simultaneity issue, the dependent variables are operationalized at the current month, while the independent variables are captured by the end of last time period (i.e., previous month). The definition and operationalization of variables are described below.

Dependent Variables. To test our hypotheses on different outcomes in open source projects, we use three sets of dependent variables at month *t* (i.e., the current month). For the *collaboration* outcomes (e.g., H1a and H1b), we use the number of new contributors (*NewDev*), and the total number of contributions or code changes by existing members (*Activeness* – *Commit, Change*) in month *t* as the dependent variables (Moqri et al. 2015). New contributors are defined as developers who contribute to the project for the first time in the current month, while existing members are those who have contributed to the project before the current month. For activeness, we measure the total contributions in two ways – For *development* outcome (e.g., H2), we consider both the number of releases in month *t* (*Release*) and a binary variable to indicate whether there are releases in a project in month *t* to

[14] See https://github.com/airbnb/javascript for Airbnb JavaScript Style Guide and https://google.github.io/styleguide/javascriptguide.xml for Google JavaScript Style Guide. For these style guides, we do not follow the exact rules defined but take the potential metric of coding preference into consideration.

capture the event of the next release (*HasRelease*). For *diffusion* (i.e., H3a and H3b), we measure attention as the number of forks or watchers of the project at month *t* (*Attention – Fork, Watch*) and code reuse as the number of new commits in the forks of the project at month *t* (*Reuse*). The event data of GitHub allow us to track these variables across time. We also note that developers may clean the source code to obtain better formatting. These commits are excluded from the operationalization of variables to avoid measurement errors.

Independent Variables. The core independent variables, as specified in hypotheses, are the measures of programming style. As discussed in section 5.6.1, we measure programming style inconsistency based on the file-level vector of metrics. Thus, the key independent variables are the project level within file inconsistency (*WithinInconsistency*) and across file inconsistency (*AcrossInconsistency*). They are captured at the end of month *t-1*. We pick the last version of source code at the month *t-1* (i.e., the source code generated by the last commit in month *t-1*) to operationalize these variables.

Moderator Variables. To test the moderating effects in H4 and H5, we construct project team level variables to capture the team familiarity and developer experience. Specifically, we measure the familiarity within the project team as the number of collaboration ties (*Collaboration*) and friendship ties (*Friendship*) of team members at the end of month *t-1*. Collaboration ties represent the collaboration experiences among the current team members in other projects. Friendship ties are measured by the follower and followee relationship of team members. For developer experience, we measure it as the number of commits a developer has contributed to other projects by the end of month *t-1* and construct the project-level variable by averaging experiences across developers (*Experience*). Given the longitudinal nature of the panel model, we also expect these moderating variables vary across time so that they can be included as control variables in all models.

Control Variables. We also include a set of control variables for both collaboration and software factors at the end of month *t-1*. Specifically, the number of issues (i.e., general discussions) (*Issues*) in month *t-1*, the number of developers (*NumDev*) by month *t-1*, total commits (*NumCommits*) by month *t-1*, total forks (*NumForks*) by month *t-1* capture the development process and stages, which can affect the subsequent collaboration, development and diffusions. These variables are tracked in the event data of the projects and can be accessed across the project history. Besides these behavioral factors in the project, the number of source code files (*NumFiles*), as well as source code complexity (*Complexity*) are the key characteristics exhibited from the source codes themselves that may affect subsequent project functioning. These software factors will be analyzed using SonarQube,[15] including McCabe's cyclomatic complexity (per function) (McCabe 1976) and maintainability index (Riaz et al. 2009). Moreover, we try to control the quality of source code since style inconsistency may be correlated with source code quality. We use the number of violations per file (*Violations*) to capture the overall quality of the source code (Avgustinov et al. 2015), which can also be calculated by static code analysis software such as SonarQube.[16] Dummy variables that account for the time trend of project (i.e., time fixed effects) will also be incorporated in the model. The main econometric models discussed above are specified as below:

$$\{NewDev_{it}, Activeness_{it}, Release_{it}\} = StyleInconsistency_{it\text{-}1} + Interactions_{it\text{-}1} + Controls_{it\text{-}1} + u_{it}$$

$$Pr(Release_{it}) = \frac{\exp(StyleInconsistency_{it\text{-}1} + Interactions_{it\text{-}1} + Controls_{it\text{-}1})}{1+\exp(StyleInconsistency_{it\text{-}1} + Interactions_{it\text{-}1} + Controls_{it\text{-}1})}$$

$$\{Attention_{it}, Reuse_{it}\} = StyleInconsistency_{it\text{-}1} + Interactions_{it\text{-}1} + Controls_{it\text{-}1} + u_{it}$$

where

[15] Details can be found at http://www.sonarqube.org/

[16] See https://rules.sonarsource.com/javascript for the rules used to capture violations in SonarQube.

$$StyleInconsistency_{it-1} = \{WithinInconsistency_{it-1}, AcrossInconsistency_{it-1}\}$$
$$Interactions_{it-1} = StyleInconsistency_{it-1} \times Familarity_{it-1} + StyleInconsistency_{it-1} \times Experience_{it-1}$$
$$Familarity_{it-1} = \{Friendship_{it-1}, Collaboration_{it-1}\}$$
$$Controls_{it-1} = Issues_{it-1} + NumDev_{it-1} + NumCommits_{it-1} + NumForks_{it-1} + NumFiles_{it-1} + Complexity_{it-1} + Violations_{it-1}$$

We use linear regressions for all the models with project-level fixed effects. For model 2 (*Release*), additional logistic regressions (which captures the timing of project release by a discrete hazard) with project fixed effects are employed.[17] We expect that programming style inconsistency will negatively impact the dependent variables, showing that inconsistency of coding style will lead to negative consequences for collaboration, development and diffusion. However, for within team collaboration (*Activeness*) and development process (*Release*), we expect that team familiarity and developer experience can positively moderate the negative effects of programming style inconsistency.

5.6.3 Econometric Model for Antecedents

The previous discussions and econometric models focus on the consequences of programming style on open source collaboration. However, as discussed in our hypotheses development, project control (i.e., the enactment of coding standards) as an antecedent of programming style inconsistency (see H6) has not been examined in the previous models. To test the related hypothesis about the use of coding standard, we adopt a different model specification that tests whether including a coding standard can reduce style inconsistency in the source code. Meanwhile, the sample used to test this effect is also different from previous models where projects with

[17] With a fixed effects specification, projects that do not have any release will be dropped in the logistic model. In addition, as our key independent variables change across time, especially between releases, we model the release in the manner of discrete choice model instead of panel survival model, which may lose significant amount of information of key variables between events.

coding standards are excluded in the analysis. All projects that conform our requirements on continuous collaboration and development are included for this analysis.

Following this model setting, we operationalize project control as a binary variable that captures the enactment of coding standard in the project repository. This can be identified by whether there are files related to the linting tools (with some specific file extensions) or style guides. Projects choose whether or not to control the coding style using standard files across time, offering a quasi-experiment setting. This can help us to evaluate the effect of coding standard on style inconsistency as a treatment effect. However, the decision on whether or not to adopt coding standards is endogenous – projects that enact the coding standards can be quite different from those without strict project control. To address this issue and enable clear identification, we further only consider projects that enact coding standards during the project's development process to estimate the before-after effect and use propensity score matching to select projects with similar characteristics but without coding standards (Caliendo and Kopeinig 2008; Rubin 2008). The specification of a Probit model to estimate the propensity score and the panel difference-in-difference model to test whether coding standards reduce style inconsistency are described as:

$$\begin{aligned} Pr(CodingStandard_{it}) = \Phi(&Issues_{it\text{-}1} + NumDev_{it\text{-}1} + NumCommits_{it\text{-}1} + NumForks_{it\text{-}1} \\ &+ NumFiles_{it\text{-}1} + Complexity_{it\text{-}1} + Violations_{it\text{-}1} \\ &+ Familarity_{it\text{-}1} + Experience_{it\text{-}1} + Time_{it\text{-}1} + StyleInconsistency_{it\text{-}1}) \\ StyleInconsistency_{it} = &CodingStandard_i + PostStandard_t \\ &+ CodingStandard_i \times PostStandard_t \\ &+ Familarity_{it\text{-}1} + Experience_{it\text{-}1} + Controls_{it\text{-}1} \end{aligned}$$

The matching procedure is performed at project-month level. For each project with coding standard included, one project month observation from projects without coding standard will be selected according to the nearest neighbor matching approach (or Mahalanobis distance matching). In addition to the control variables used in the previous models, total time since project initiation (*Time*, in month) and the current

style inconsistency in the project are used in the matching. After selecting the control group, we examine the short-term effects of enacting coding standard (H6a) by comparing the style inconsistency in the month after treatment. The average treatment effect on the treated (ATT) is calculated on the difference of style inconsistency to estimate the immediate change. Specifically, ATT is measured as the difference between the change of style inconsistency after treatment in the treated group and the change of style inconsistency after the matched project-month in the control group (i.e., a difference-in-difference estimation with single time period). To test the long-term or overall effects of enacting coding standard (H6b), we use the time periods before and after the event to estimate the difference-in-difference model (for the treatment group, the event month is the month during which the coding standard was enacted, while for the control group, the month of matched observation can be regarded as the counterfactual event month). The coefficient of the interaction term will suggest the treatment effect of enacting a coding standard in the project.

5.7 Results

5.7.1 Stylistic Metrics and Source Code Analysis

The first part of the empirical analysis is to identify the metrics for measuring programming style and create the fingerprint vector for each source code file. Although popular style guides already define a list of metrics concerning code quality and stylistic elements, researchers usually select a group of important metrics to quantify programming styles and code quality. We follow the existing software engineering literature to select the metrics for quantifying style inconsistency (Binkley et al. 2013; Boogerd and Moonen 2008; Mi et al. 2016; Mohan and Gold 2004; Smit et al. 2011a). Given our focus on the inconsistency of coding style, we choose metrics that are purely related to preference instead of code quality. Table 5-1 presents the metrics and indicators adopted in our empirical analysis. Among all these metrics, those metrics with complementary indicators (indicators that sum up to one)

were used for measuring within file style inconsistency, while all indicators were used to capture across file style inconsistency. Therefore, 22 indicators (the sources of metrics are marked in Table 5-1) were created for within file inconsistency and 62 indicators constituted the fingerprint vector for across file inconsistency.

With our customized program, for each project-month code version, all the JavaScript files in the projects were scanned and checked for stylistic metrics. More than 5 million source files in total were analyzed for the stylistic fingerprint (with our program) and code quality metrics (with SonarQube). As documented in software engineering and programming language research (Mi et al. 2016), the fingerprint extracted can only be meaningful when there are sufficient lines of code in the files. Therefore, source code files with too few lines were excluded for measuring style inconsistency. In addition, by examining file extensions used by major code checking tools (e.g., ESLint),[18] our program identified a total of 849 projects with coding standards in their project files within our selected time window. After removing these projects and a small group of projects with fatal errors in the code,[19] our sample used for the consequence model contains a total of 1,286 projects. The 849 projects are then used in the antecedent model.

[18] For instance, a project with coding standard may adopt ESLint as the code checking (i.e., linting) tool. Therefore, a configuration file of ESLint will be included in the codebase and can be identified by our code scanner.

[19] Some files in our analysis were reported with severe grammatic errors by our code scanner and SonarQube (so they cannot be properly tokenized or parsed). The errors mostly come from incomplete code so that these source code files cannot be run and extracted with meaningful information. Therefore, they are removed from our analysis.

Table 5-1. Programming Style Metrics and Indicators

Metric	Indicators
Indentation (1)	The proportion of source code lines using 2 spaces indentation. The proportion of source code lines using 4 spaces indentation. The proportion of source code lines using 8 spaces indentation. The proportion of source code lines using tab indentation. The proportion of source code lines using other indentation.
Brace Spacing (4)	The proportion open braces with spaces before. The proportion open braces with spaces after. The proportion end braces with spaces before. The proportion end braces with spaces after.
Parenthesis Spacing (4)	The proportion open parentheses with spaces before. The proportion open parentheses with spaces after. The proportion end parentheses with spaces before. The proportion end parentheses with spaces after.
Comma Spacing (2)	The proportion commas with spaces before. The proportion commas with spaces after.
Colon Spacing (2)	The proportion colons with spaces before. The proportion colons with spaces after.
Comment Spacing (2)	The proportion of multiple line comments starting with space. The proportion of single line comments starting with space.
White Space	The average length of white spaces. Whether the last line is a new blank line.
Function Name (1)	The proportion of lowercase characters in function names. The proportion of uppercase characters in function names. The proportion of number characters in function names. The proportion of underscore characters in function names.
Variable Name (1)	The proportion of lowercase characters in variable names. The proportion of uppercase characters in variable names. The proportion of number characters in variable names. The proportion of underscore characters in variable names.
Comment Style	The proportion of blank lines. The proportion of comment lines. The proportion of inline comments in all comment lines. The proportion of single line comments in all comment lines. The proportion of multiple line comments in all comment lines. The proportion of JS doc comments in all comment lines.*
Brace Style (2)	The proportion of open braces alone in line. The proportion of open braces at the beginning of line. The proportion of open braces at the end of line. The proportion of open braces at the middle of line. The proportion of end braces alone in line. The proportion of end braces at the beginning of line. The proportion of end braces at the end of line. The proportion of end braces at the middle of line.
Operator Style (1)	The proportion of dot operator at the beginning of line. The proportion of dot operator at the end of line. The proportion of dot operator at the middle of line.
Looping Structure (1)	The proportion of for in all loops. The proportion of while in all loops. The proportion of do-while in all loops.

Selection Structure (1)	The proportion of if-else in all selections. The proportion of switch-case in all selections.
Keywords Usage*	The ratio of number of keywords *try* to number of lines. The ratio of number of keywords *catch* to number of lines. The ratio of number of keywords *const* to number of lines. The ratio of number of keywords *default* to number of lines. The ratio of number of keywords *continue* to number of lines. The ratio of number of keywords *delete* to number of lines. The ratio of number of keywords *goto* to number of lines. The ratio of number of keywords *with* to number of lines. The ratio of number of keywords *package* to number of lines. The ratio of number of keywords *return* to number of lines. The ratio of number of keywords *throw* to number of lines. The ratio of number of keywords *typeof* to number of lines.
Copyright	Whether the file contains copyright information.

Notes: *These metrics or indicators are specific to the JavaScript language. The numbers in parentheses after metric names refer to the number of indicators created for within file style inconsistency from the metric. Among the metrics, *indentation*, *brace spacing*, *parenthesis spacing*, *comma spacing*, *colon spacing*, *comment spacing*, *white space*, *brace style* and *operator style* and belong to the category of *Programming Format*; *Function name*, *variable name* and *comment style* belong to the category of *Programming Readability*; *Loop structure*, *selection structure* and *keyword usage* belong to the category of *Programming Language*.

5.7.2 Results of the Consequence Model

With the 1,286 projects that do not have coding standards in their repositories, we perform econometric analysis for the consequence model following section 5.6.2. Table 5-2 presents the descriptive statistics and correlations of variables in the consequence model. High correlations were not observed among the variables and all the variables except *WithinInconsistency*, *AcrossInconsistency* and *Complexity* are log-transformed to account for their skewed distributions.

Table 5-2. Descriptive Statistics and Correlation Matrix

	Mean	SD	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
1. NewDev	0.479	1.1652	1								
2. Commit	16.99	43.068	0.31***	1							
3. Change	23755	237322	0.05***	0.15***	1						
4. Release	0.0862	0.7166	0.12***	0.17**	0.03***	1					
5. Fork	1.7314	7.6924	0.39***	0.11***	0.01	0.06***	1				
6. Watch	8.4770	37.309	0.40***	0.11***	-0.00	0.11***	0.42***	1			
7. Reuse	5.8235	24.016	0.33***	0.36***	0.02***	0.07***	0.25***	0.22***	1		
8. WithinInconsistency	0.1421	0.0292	-0.11***	-0.01	0.00	-0.04***	-0.08***	-0.11***	-0.07***	1	
9. AcrossInconsistency	0.0101	0.0247	-0.01	-0.01	0.01	0.03***	-0.02**	-0.01	-0.00	-0.27***	1
10. Issues	4.7414	19.434	0.27***	0.26***	0.03***	0.11***	0.22***	0.22***	0.28***	-0.08***	-0.00
11. NumDev	10.974	15.888	0.41***	0.14***	0.02**	0.08***	0.33***	0.36***	0.28***	-0.16***	-0.01
12. NumCommit	335.59	610.09	0.13***	0.41***	0.10***	0.18***	0.10***	0.11***	0.28***	-0.05***	-0.01
13. NumFork	35.281	139.38	0.32***	0.08***	-0.00	0.04***	0.58***	0.54***	0.24***	-0.12***	-0.01*
14. NumFile	30.965	87.870	-0.00	0.05***	0.06***	0.01	-0.02*	-0.02**	0.04***	-0.10***	0.20***
15. Complexity	7.0382	3.1718	-0.04***	0.03***	0.05***	-0.00	-0.01	-0.03***	-0.02**	0.36***	0.01
16. Violations	3300.7	9979.2	-0.02*	0.07***	0.10***	0.01	-0.03***	-0.03***	0.00	-0.00	0.07***
17.Friendship	2.2628	5.893	0.09***	0.06***	-0.00	0.02**	0.04***	0.14***	0.10***	-0.08***	-0.04***
18. Collaboration	159.88	2064.0	0.04***	-0.00	-0.00	-0.01	0.01*	-0.01	0.01	0.02*	-0.00
19. Experience	16578	48777	0.06***	0.02**	0.01	0.02**	0.05***	0.10***	0.05***	-0.07***	-0.02**

	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)
10. Issues	1									
11. NumDev	0.26***	1								
12. NumCommit	0.27***	0.44***	1							
13. NumFork	0.24***	0.60***	0.22***	1						
14. NumFile	0.00	0.01	0.11***	-0.01	1					
15. Complexity	-0.02**	-0.05***	0.03***	-0.01*	0.10***	1				
16. Violations	0.00	-0.04***	0.11***	-0.03***	0.57***	0.21***	1			
17.Friendship	0.07***	0.26***	0.19***	0.08***	-0.03***	-0.13***	0.01*	1		
18. Collaboration	-0.01	0.24***	0.07***	0.03***	0.04***	-0.01	0.01	0.18***	1	
19. Experience	0.04***	0.23***	0.09***	0.11***	-0.01	-0.07***	-0.02***	0.22***	0.22***	1

Significance Levels: * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$. N=21,768.

The results on the main effects of style inconsistency are presented in Tables 5-3 and 5-4. In Table 5-3, the number of new contributors is not affected by programming style inconsistency (Model 1) – neither within file inconsistency (β=-0.107, ns) nor across file inconsistency (β=0.185, ns) are significant. When new contributors join the team, the inconsistency of style in source code files does not seem to matter. It is possible that the new contributors care more about the popularity (supported by the significance of *NumFork*), existing team composition and their connections to the team (Hahn et al. 2008) instead of how the source code looks (all the software related variables are not significant). Therefore, H1a is not supported. In Model 2, the effects of style inconsistency on the total number of contributions are not significant for both within style inconsistency (β=-1.281, ns) and across file inconsistency (β=0.296, ns). But in Model 3, we find that within file inconsistency negatively affects the total changes (i.e., the number of lines changed) made in the source code (β=-6.812, p<0.01). Although within file inconsistency does not affecte number of contribution times, but it will decrease the amount of changes made to the source code. But for across file inconsistency, it seems that the differences of coding style across files do not induce negative effects – they do not affect developers' activeness on committing new code and certain freedom of code styles does not harm the collaboration. Therefore, H1b only receives partial support, conditional on the within file inconsistency and amount of code changes.

Table 5-3. Effects on New Contributor and Contributions

Variables	Model 1 *ln(NewDev)*	Model 2 *ln(Commit)*	Model 3 *ln(Change)*
WithinInconsistency	-0.107	-1.281	-6.812***
	(0.448)	(1.432)	(2.288)
AcrossInconsistency	0.185	0.296	0.119
	(0.281)	(1.077)	(2.339)
ln(Issues)	0.0540***	0.381***	0.804***
	(0.00536)	(0.0190)	(0.0323)
ln(NumDev)	-0.191***	0.229**	0.720***
	(0.0381)	(0.0995)	(0.138)
ln(NumCommit)	-0.0160	-0.175***	-0.542***
	(0.0120)	(0.0423)	(0.0607)
ln(NumFork)	0.0902***	-0.119***	-0.234***
	(0.0164)	(0.0462)	(0.0700)
ln(NumFile)	0.00359	-0.105*	-0.323***
	(0.0128)	(0.0554)	(0.0810)
Complexity	-0.00112	-0.00394	-0.0282
	(0.00422)	(0.0166)	(0.0223)
ln(Violations)	0.00707	0.0990***	0.436***
	(0.0103)	(0.0380)	(0.0619)
ln(Friendship)	-0.0237	-0.0226	0.0153
	(0.0170)	(0.0480)	(0.0843)
ln(Collaboration)	-0.0460***	-0.00131	-0.0608
	(0.0125)	(0.0392)	(0.0610)
ln(Experience)	-0.0124*	-0.0475**	-0.166***
	(0.00661)	(0.0219)	(0.0355)
Constant	0.612***	2.624***	7.072***
	(0.0893)	(0.272)	(0.457)
Observations	21,768	21,768	21,768
R-squared	0.041	0.083	0.059
Number of Projects	1,286	1,286	1,286

Significance levels: *** $p<0.01$, ** $p<0.05$, * $p<0.1$

Notes: Project fixed effects and time dummies are included. Robust standard errors in parentheses.

In Table 5-4, for H2, we do not find any support for project release. In Model 4 and 5, within file inconsistency does not have any effect but across file inconsistency even has a positive effect on project release. It seems that project teams do not care about the inconsistency of coding style and the freedom of collaboration may induce better functional progress. As can be noticed, project release is significantly affected by the number of issues (i.e., general discussions) posted so that project teams may attach more weight to functionalities instead of the codebase (Mallapragada et al. 2012). We also notice that the proportion of projects with

releases is quite low (a large number of projects are dropped in the conditional logit model). It is possible that how project teams leverage releases in GitHub is different from traditional open source context (Murgia et al. 2014). Model 7 to Model 9 test the effects of style inconsistency on project diffusion. In Model 6 and Model 7, it seems that style inconsistency does not affect the attention of projects. Only across file inconsistency has a partial negative effect on the number of watchers (β=-0.976, p<0.1) and the software related variables are not significant (i.e., complexity and violations). It may be because a large proportion of community members fork or watch the projects based on the initial interests instead of further contribution intention (Sheoran et al. 2014). Therefore, the source code does not play an essential role for most community users in this process. But in Model 8, we find that within file inconsistency has a negative effect on code reuse (β=-1.222, p<0.05). When reutilizing the source code for further development, the coding style will play an important role in the intention to reuse. In addition, the effect does not materialize with across file inconsistency probably because the reuse of source code may only occur with limited files. Therefore, H3a is not support and H3b is partially supported by within file style inconsistency.

Table 5-4. Effects on Release, Attention and Code Reuse

Variables	Model 4 *ln(Release)*	Model 5 *HasRelease*	Model 6 *ln(Fork)*	Model 7 *ln(Watch)*	Model 8 *ln(Reuse)*
WithinInconsistency	0.170	7.359	0.992*	-0.539	-1.222**
	(0.296)	(7.280)	(0.533)	(0.948)	(0.557)
AcrossInconsistency	1.195***	10.31***	0.273	-0.976*	-0.263
	(0.338)	(3.242)	(0.351)	(0.587)	(0.569)
ln(Issues)	0.0111***	0.355***	0.0639***	0.0726***	0.214***
	(0.00317)	(0.0903)	(0.00763)	(0.0114)	(0.00786)
ln(NumDev)	0.0221	0.564	-0.0264	0.439***	0.0408
	(0.0150)	(0.430)	(0.0477)	(0.0779)	(0.0336)
ln(NumCommit)	0.00464	0.0611	0.0401***	-0.158***	0.0138
	(0.00449)	(0.260)	(0.0151)	(0.0280)	(0.0148)
ln(NumFork)	0.00969	-0.287	0.00409		0.146***
	(0.00880)	(0.233)	(0.0283)		(0.0170)
ln(NumWatch)				0.278***	
				(0.0303)	
ln(Files)	-0.0134*	-0.350	0.0249	0.0174	-0.0105
	(0.00810)	(0.296)	(0.0198)	(0.0326)	(0.0197)
Complexity	0.00160	-0.100	-0.00351	0.00349	-0.00284
	(0.00193)	(0.0664)	(0.00495)	(0.00978)	(0.00542)
ln(Violations)	0.00706	0.356**	0.00324	0.00495	-0.0103
	(0.00706)	(0.174)	(0.0137)	(0.0241)	(0.0151)
ln(Friendship)	0.0131	0.174	-0.0311	0.101**	-0.0532***
	(0.00978)	(0.297)	(0.0264)	(0.0504)	(0.0205)
ln(Collaboration)	0.00186	0.0951	-0.0382**	-0.0375	-0.0130
	(0.00549)	(0.191)	(0.0160)	(0.0365)	(0.0148)
ln(Experience)	-0.00542*	-0.212*	-0.00994	-0.0480***	-0.000630
	(0.00298)	(0.124)	(0.00789)	(0.0149)	(0.00864)
Constant	-0.0966		0.412***	0.116	0.656***
	(0.0667)		(0.109)	(0.0906)	(0.167)
Observations	21,768	4,620	21,768	21,768	21,768
R-squared	0.023	0.128	0.045	0.196	0.054
Number of Projects	1,286	222	1,286	1,286	1,286

Significance levels: *** $p<0.01$, ** $p<0.05$, * $p<0.1$

Notes: Conditional fixed effects logit model is used for Model 5. *NumWatch* (cumulative number of watchers) is controlled when number of watcher is the dependent variable (because fork is highly correlated with watcher). Project fixed effects and time dummies are included. Robust standard errors in parentheses.

In Table 5-5 and Table 5-6, we examine the interaction effects between team familiarity and developer experience. In Table 5-5, we test H4a and H5a for the interaction effects on contribution activities. For team familiarity, we find support from friendship ties. The friendship ties positively moderate the effects of style inconsistency in Model 9 and Model 11. For total number of contributions, although friendship ties significantly shift the main effects, the main effects do not seem to be

strong. But for total amount of code change, when there are few friendship ties, within file inconsistency has a strong negative effect on contributions, and this negative effect can be significantly alleviated by team familiarity through friendship ties. And when there are more friendship ties among team members, across file inconsistency tends to be beneficial to contribution activities (the freedom of contribution is amplified with team familiarity). Similar moderating effects can be observed for collaboration ties, but only for within file inconsistency (Model 10 and Model 12). Therefore, H4a is supported with friendship ties and partially supported with collaboration ties. However, H5a is not supported and across all the models, the interaction effects between developer experiences and style inconsistency are negative. Even though we expect that experienced developers have lower costs for code comprehension, the results seem to suggest that experienced developers have stronger beliefs on programming style so that they are resistant to inconsistent styles, both within file and across files. They are less cooperative and more sensitive to style inconsistency when they have rich experiences in programming. Therefore, the findings imply that for the two mechanisms (cognitive and material), the cognitive mechanism may play a more dominant role than material mechanism (Cox and Fisher 2009; Harrison and Klein 2007; van Knippenberg and Schippers 2007).

In Table 5-6, H4b and H5b are examined. From Model 13 to Model 16, we do not find clear and significant interaction effects between team composition and style inconsistency on project releases. As we have discussed, project release may be attached with more functional improvements and can be leveraged differently by project teams in GitHub. The strategy of releasing new versions may depend more on external requirements rather than internal resources. Consequently, the effects of source code, and interaction effects with team composition, would not have an impact on the project release process. Hence, H4b and H5b are rejected.

Table 5-5. Interaction Effects on Contributions

Variables	Model 9 *ln(Commit)*	Model 10 *ln(Commit)*	Model 11 *ln(Change)*	Model 12 *ln(Change)*
WithinInconsistency	-1.895	-1.372	-8.489**	-7.183**
	(1.474)	(1.433)	(3.638)	(3.550)
AcrossInconsistency	-0.395	-0.219	-3.002	-1.608
	(0.962)	(1.109)	(3.114)	(3.373)
WithinInconsistency × ln(Friendship)	2.816**		6.825**	
	(1.187)		(2.971)	
AcrossInconsistency × ln(Friendship)	1.913*		8.413***	
	(0.986)		(3.064)	
WithinInconsistency × ln(Collaboration)		2.131***		3.555*
		(0.761)		(1.997)
AcrossInconsistency × ln(Collaboration)		1.049		0.346
		(0.855)		(2.897)
WithinInconsistency × ln(Experience)	-1.176**	-1.772***	-3.032***	-3.697***
	(0.458)	(0.541)	(1.079)	(1.337)
AcrossInconsistency × ln(Experience)	-0.803**	-1.058**	-3.499***	-3.017**
	(0.365)	(0.420)	(1.149)	(1.259)
ln(Issues)	0.379***	0.379***	0.801***	0.802***
	(0.0190)	(0.0190)	(0.0438)	(0.0437)
ln(NumDev)	0.200**	0.209**	0.632***	0.652***
	(0.0994)	(0.0992)	(0.234)	(0.236)
ln(NumCommit)	-0.171***	-0.175***	-0.526***	-0.530***
	(0.0418)	(0.0421)	(0.109)	(0.109)
ln(NumFork)	-0.114**	-0.115**	-0.221*	-0.222*
	(0.0457)	(0.0462)	(0.114)	(0.115)
ln(Files)	-0.0971*	-0.0942*	-0.309**	-0.302**
	(0.0554)	(0.0556)	(0.151)	(0.151)
Complexity	-0.00565	-0.00563	-0.0324	-0.0305
	(0.0159)	(0.0163)	(0.0360)	(0.0366)
ln(Violations)	0.0985***	0.101***	0.434***	0.433***
	(0.0380)	(0.0378)	(0.108)	(0.110)
ln(Friendship)	-0.0234	-0.0176	0.00456	0.0210
	(0.0485)	(0.0479)	(0.119)	(0.119)
ln(Collaboration)	0.00332	-0.00888	-0.0454	-0.0713
	(0.0389)	(0.0386)	(0.0952)	(0.0949)
ln(Experience)	-0.0361*	-0.0325	-0.134**	-0.131**
	(0.0213)	(0.0216)	(0.0574)	(0.0580)
Constant	2.644***	2.534***	7.152***	6.910***
	(0.272)	(0.269)	(0.702)	(0.695)
Observations	21,768	21,768	21,768	21,768
R-squared	0.084	0.084	0.060	0.060
Number of Projects	1,286	1,286	1,286	1,286

Significance levels: *** $p<0.01$, ** $p<0.05$, * $p<0.1$

Notes: Interaction terms are mean-centered to reduce collinearity issues. Project fixed effects and time dummies are included. Robust standard errors in parentheses.

Table 5-6. Interactions Effects on Project Release

Variables	Model 13 *ln(Release)*	Model 14 *ln(Release)*	Model 15 *HasRelease*	Model 16 *HasRelease*
WithinInconsistency	0.171	0.0798	7.459	7.271
	(0.301)	(0.282)	(6.947)	(6.762)
AcrossInconsistency	1.338***	1.238***	11.39***	9.387**
	(0.344)	(0.366)	(3.796)	(3.946)
WithinInconsistency × ln(Friendship)	-0.390		-2.061	
	(0.250)		(6.239)	
AcrossInconsistency × ln(Friendship)	0.221		0.0920	
	(0.251)		(3.043)	
WithinInconsistency × ln(Collaboration)		-0.345**		1.197
		(0.162)		(4.886)
AcrossInconsistency × ln(Collaboration)		0.257		4.296
		(0.187)		(4.550)
WithinInconsistency × ln(Experience)	-0.0501	0.0635	-0.586	-1.580
	(0.0680)	(0.0817)	(1.929)	(3.133)
AcrossInconsistency × ln(Experience)	0.147	0.0547	1.812	0.0116
	(0.102)	(0.133)	(2.169)	(3.035)
ln(Issues)	0.0111***	0.0110***	0.359***	0.357***
	(0.00318)	(0.00317)	(0.0898)	(0.0897)
ln(NumDev)	0.0187	0.0190	0.553	0.582
	(0.0150)	(0.0151)	(0.442)	(0.447)
ln(NumCommit)	0.00617	0.00582	0.0410	0.0371
	(0.00471)	(0.00477)	(0.257)	(0.259)
ln(NumFork)	0.00969	0.00889	-0.262	-0.281
	(0.00868)	(0.00865)	(0.234)	(0.234)
ln(Files)	-0.0142*	-0.0133*	-0.352	-0.360
	(0.00786)	(0.00796)	(0.303)	(0.285)
Complexity	0.00132	0.00120	-0.102	-0.109
	(0.00193)	(0.00198)	(0.0683)	(0.0685)
ln(Violations)	0.00845	0.00798	0.373**	0.385**
	(0.00710)	(0.00705)	(0.184)	(0.167)
ln(Friendship)	0.00988	0.00983	0.142	0.180
	(0.00891)	(0.00915)	(0.351)	(0.299)
ln(Collaboration)	0.00237	0.00442	0.102	0.0945
	(0.00544)	(0.00570)	(0.189)	(0.192)
ln(Experience)	-0.00522*	-0.00547*	-0.213*	-0.207
	(0.00304)	(0.00310)	(0.125)	(0.127)
Constant	-0.106	-0.0883		
	(0.0678)	(0.0649)		
Observations	21,768	21,768	4,620	4,620
R-squared	0.025	0.026	0.129	0.129
Number of Projects	1,286	1,286	222	222

Significance levels: *** $p<0.01$, ** $p<0.05$, * $p<0.1$

Notes: Conditional fixed effects logit model is used for Model 15 and 16. Interaction terms are mean-centered to reduce collinearity issues. Project fixed effects and time dummies are included. Robust standard errors in parentheses.

5.7.3 Results of the Antecedent Model

In the antecedent model, we use the sample of projects with coding standards and compare them to those without. In line with the quasi-experiment setting and matching procedure in §5.6.3, we focus on projects that enact coding standard during the development process. Therefore, a total of 220 projects are included in the treatment group, while projects without coding standard are used to match with treated projects. We perform both propensity score matching and Mahalanobis distance matching to ensure high quality of matching. The key covariates (variables used to calculate propensity score and Mahalanobis distance, including style inconsistency, team familiarity, developer experience, time and other control variables) between treatment and control group after matching are quite similar (Mahalanobis distance matching slightly outperforms propensity score matching in this regard), suggesting high quality of the matching procedure.

Table 5-7. Matching Analysis for Short Term Effects of Coding Standard

	Propensity Score Matching			**Mahalanobis Matching**		
	Treated	**Matched**	**Difference**	**Treated**	**Matched**	**Difference**
ΔWithinInconsistency	-0.0029	-0.0003	-0.0026**	-0.0029	0.0002	-0.0031***
ΔAcrossInconsistency	0.0009	0.0002	0.0007	0.0009	0.0002	0.0007

Significance levels: *** $p<0.01$, ** $p<0.05$, * $p<0.1$

Notes: $N=220$ in the treatment group. One to one matching is performed. A caliper of 0.1 is used for propensity score matching. All covariates are used for Mahalanobis matching. Heteroskedasticity consistent standard errors are used for statistical inference.

Table 5-7 and Table 5-8 present the matching and difference-in-difference analysis for H6a and H6b. In Table 5-7, the difference between the changes of style inconsistency in treated and matched group is significant for within file inconsistency but not significant for across file inconsistency. This suggests that after the enactment of coding standard, within file style inconsistency can quickly be reduced but across file inconsistency is less likely to be affected. Compared to within file inconsistency, across file inconsistency is more difficult to control since developers may have their own focus on certain files and do not have much attention on other files (Langlois and

Garzarelli 2008). Therefore, the developers may not be able to achieve synergy to unify coding styles across files and they may only focus on the styles of their current working files.

Table 5-8. Difference-in-Difference Analysis for Long Term Effects of Coding Standard

Variables	Model 17 *WithinInconsistency*	Model 18 *AcrossInconsistency*	Model 19 *WithinInconsistency*	Model 20 *AcrossInconsistency*
PostStandard	-0.00219***	-0.000312	-0.00171***	0.000724
	(0.000470)	(0.000888)	(0.000452)	(0.00119)
CodingStandard × *PostStandard*	-0.00184***	0.00348*	-0.00132**	0.00230
	(0.000571)	(0.00199)	(0.000549)	(0.00210)
ln(Issues)	-1.59e-05	6.00e-05	0.000159	0.000308
	(0.000171)	(0.000305)	(0.000163)	(0.000351)
ln(NumDev)	0.000215	0.000577	0.00404***	-0.00116
	(0.000699)	(0.00173)	(0.000663)	(0.00164)
ln(NumCommit)	-0.00140***	-0.000273	-0.00110***	-0.000414
	(0.000329)	(0.000777)	(0.000322)	(0.000783)
ln(NumFork)	0.000173	-7.90e-06	-0.00230***	0.000734
	(0.000364)	(0.000806)	(0.000332)	(0.000687)
ln(Files)	-0.00678***	0.00111	-0.0105***	0.00369*
	(0.000392)	(0.00168)	(0.000396)	(0.00201)
Complexity	0.000467***	-0.000345	0.000835***	-0.000268
	(0.000101)	(0.000349)	(0.000104)	(0.000393)
ln(Violations)	0.00534***	-1.93e-05	0.00718***	-0.00142
	(0.000290)	(0.00105)	(0.000307)	(0.00127)
ln(Friendship)	-0.00211***	0.000289	-0.00128***	0.000155
	(0.000407)	(0.00102)	(0.000383)	(0.00101)
ln(Collaboration)	0.000595*	0.000115	3.41e-06	0.00107
	(0.000312)	(0.000981)	(0.000290)	(0.000916)
ln(Experience)	-0.000537***	0.000306	-0.000744***	0.000112
	(0.000161)	(0.000439)	(0.000169)	(0.000463)
Constant	0.136***	0.00416	0.129***	0.00800*
	(0.00166)	(0.00404)	(0.00166)	(0.00467)
Observations	9,799	9,799	10,294	10,294
R-squared	0.110	0.030	0.157	0.035
Number of Projects	409	409	432	432

Significance levels: *** $p<0.01$, ** $p<0.05$, * $p<0.1$

Notes: *CodingStandard* (treatment group effect) is omitted due to the fixed effects specification. Model 17 and 18 follow propensity score matching. Model 19 and 20 follow Mahalanobis matching. Mahalanobis matching generates a larger control group than propensity score matching so Model 19 and 20 contain more projects. Project fixed effects and time dummies are included. Robust standard errors in parentheses.

In Table 5-8, the difference-in-difference panel model shows a similar pattern for the long term effect. The interaction effects are negatively significant for within file style inconsistency but not significant for across file inconsistency (even slightly

positive). Consistent with short term effects, the enactment of coding standards is likely to reduce within file inconsistency through project control in the long run, but it is difficult for across file inconsistency to be improved with coding standards.

5.8 Conclusions

This study investigates the role of programming style in open source collaboration. Although programming style has been documented as an important aspect in software development, existing studies have yet to study its implications on collaboration processes. The influence on collaboration is potentially more salient in the open source context, which is usually based on voluntary contribution and free style project management without strong restrictions with respect to code writing. Our study explores and examines the impacts of programming style in OSS development and how relevant team characteristics shape these relationships. We propose hypotheses on contributor, software development and community evolution perspectives, and how project control, team familiarity and developer experience serve as antecedent and moderators. To test the hypotheses, we quantify programming style inconsistency through static code analysis and a list of programming metrics. The empirical model is constructed at project-month level with project fixed effects. A quasi-experiment setup is used for understanding the usage of coding standards in terms of project control. Our empirical results suggest that programming style inconsistency negatively affects contribution activities but not other collaboration measures such as new contributors, project releases and project popularity. The negative effects mainly occur through within file style inconsistency but not through across file inconsistency. In addition, we find that team familiarity could alleviate the negative effects of style inconsistency, but developer experience would further aggravate the potential conflicts from inconsistent coding styles. The matching and difference-in-difference analysis suggest the decrease of within file inconsistency but no changes of across file inconsistency after the enactment of coding standard in both the short and long terms.

5.8.1 Theoretical Contributions

Our study contributes to several streams of literature. First, we contribute to the literature on OSS development by investigating an important aspect of regulation and control in software teams. Open source community involves developers with diverse knowledge and personal traits in programming, which may lead to different coding styles which may negatively impact the evolvement and management of development projects. Our findings suggest that there is a certain level of negative effects from inconsistent programming styles, but mainly limited to within file inconsistency and contribution activities. Developers tend to care about how the source codes look when making changes but focus more on other aspects (e.g., team composition and software functions) for other activities. We also show that a certain level of freedom for contributions (mainly captured by across file style inconsistency) is not detrimental for the development process or project evolvement. Therefore, we identify both the importance of keeping self-regulated (with consistent coding style in files) and allowing for freedom (with the existence of across file inconsistency) (Bagozzi and Dholakia 2006; Shah 2006). Our study serves as one of the first studies that offer insights on how the materiality of open source software affects developer activities and team functioning. Open source teams should leverage not only the behavioral aspects of collaboration but also the material aspects of collaboration deliverables.

Second, our study further investigates the quantification and implication of programming style in software engineering. Our measures and approaches are useful for a deeper understanding of stylistic inconsistency in OSS. Specifically, different with mixed choices in software engineering and programming language research on the selection of coding rules (Boogerd and Moonen 2008; Lee et al. 2013b), we focus on coding traces that are not objectively regarded as correct or incorrect, but are more subjectively related to developers' preferences. We also extend this stream of

literature by capturing two aspects of coding style inconsistency – within file and across files. The findings suggest differential effects of these two aspects on specific but not all collaboration outcome measures. Although software engineering research has examined several approaches for quantifying programming style and identifying the relationships between software metrics, little is known about how collaborative behaviors can be affected by software metrics such as style inconsistency and maintainability. Our study offers insights into the interaction between software factors and behavioral factors.

Third, we provide possible mitigating factors for group separation relevant to the diversity literature. Beyond culture or country diversity with documented negative effects on team performance (Harrison and Klein 2007), we explore another aspect of separation – technical norms in product development (i.e., programming style in this study) – which is more likely to be controlled and coordinated in working groups. Our study implies that in spite of the cultural diversity (Daniel et al. 2013), coordination with consistent opinions on the product (supported by consistent coding style) may be helpful for team collaboration and performance. However, in the case of programming style inconsistency, the diversity also implies the presence of multiple (and inconsistent) of work styles within the group, which not only relates to individual's understanding on the product but also impacts the cognitive intentions to collaborate with others with different work styles. Our study examines this important type of diversity which is more salient in the open source community. The findings show that it may be important for software teams to be tightly coupled within elements (or product units) (i.e., consistent within file style) but loosely coupled across elements (i.e., across file style not necessarily to be consistent). Work groups can leverage different components in the product collaboration to achieve higher efficiency. In addition, our findings on the interaction effects indicate that more familiar members and less experienced members may be less sensitive to the inconsistent opinions or preferences in group work. Therefore, team formation

mechanisms can be used to mitigate the conflicts from different styles but experiences (or potentially tenure) may make the situations worse.

5.8.2 Practical Implications

For practical implications, our study suggests that open source teams need to pay attention to the software itself (i.e., the programming style from different developers) when organizing the development process. To facilitate contribution activities, teams can either enable members to be consistent when contributing source code or enact coding standard to regulate coding styles in a formal way. The software artifact can play an essential role in shifting developers' motivations and efforts. In addition, our findings suggest that through team formation mechanisms, the consequences of style inconsistency can be mitigated by stronger connections and familiarity among developers. But project owners should also notice that more experienced developers may have stronger beliefs about programming styles so that style inconsistency needs be reduced to avoid the detrimental moderating effects from developer experience.

5.8.3 Limitations

The current study has several limitations that require further examinations. First, there may exist variations and heterogeneity across different coding style metrics, which needs a further check for the effects of different sets or categories of metrics. Second, our analysis mainly focuses on the code status at the project month level. It is also necessary to analyze the dynamic evolution of source code at more fine-grained levels (e.g., how the code evolves across commits). Third, since JavaScript may have certain unique coding grammars or features, it will be valuable to look at other programming languages to compare with the current analysis.

CHAPTER 6 CONCLUSION

6.1 Summary

Online communities for innovation have been trending in recent years due to its potential for economic value creation for organizations and society. These communities go beyond the boundaries of traditional innovation activities within organizations and democratize individuals in the wave of innovation and entrepreneurship. My dissertation focuses on this emerging phenomenon and aims to better understand the open collaboration process in these innovation communities from a group diversity perspective. Given the nature of geographical dispersion and voluntary participation, individuals from various cultural and knowledge backgrounds work and collaborate together. Group diversity serves as a suitable lens for understanding the collaboration process in such innovation communities.

The first essay investigates an open innovation community where firms use crowdsourcing in new product development. It focuses on how firms can organize the crowds in this process. We draw on the diversity literature to define different types of participants and develop hypotheses on their value contributions in the crowdsourced new product development process. Using data from 425 new product development campaigns, I test how the variety of knowledge in participants affect the collective performance of large online crowds. I find that crowd members with both diverse experience and specialized experience are helpful for the development process. I also observe a group of members with T-shaped experience in non-focal tasks may increase the collective performance. In addition, I do not find any significant effect of generalists on development duration.

The second essay examines self-organized innovation communities characterized by open collaboration. It investigates the regulation of open collaboration by conceptualizing programming style as a type of work style diversity. Hypotheses about the main effects of programming styles on collaboration,

development and diffusion are developed. I also try to explore the moderators and antecedent of programming style. Based on the software engineering literature, I measure programming style inconsistency at multiple levels. Using data and source code in software projects from GtiHub, I find that within file style inconsistency has negative effects on contribution activities such as code changes and code reuse, but across file style inconsistency does not significantly impact collaboration outcomes. The negative effects of within file inconsistency are further (positively) moderated by team familiarity but (negatively) intensified by developer experiences. In addition, the adoption of coding standard can help to reduce within file inconsistency but does not affect across file inconsistency.

In summary, my dissertation seeks to examine the open collaboration process and the management of innovation communities. I try to understand the open-form collaboration activities in both firm-oriented innovation communities and self-organized innovation communities. From a group diversity perspective, I attempt to examine group level effects of knowledge variety and work style separation on value co-creation and open collaboration in innovation communities. Overall, my dissertation presents the investigation of important types group diversity in online innovation collectives and provides insights on open collaboration and innovation activities.

6.2 Contributions and Implications

6.2.1 Theoretical Contributions

My dissertation makes several contributions to the literature. First, the essays contribute to the IS literature on online innovation communities. Although there have been several streams of research trying to understand online innovation communities, few studies empirically examine how firms organize the crowd in the innovation process. The first essay investigates a new business model on crowdsourcing for new product development and extends this stream of literature by exploring the

collaboration and value co-creation process in crowdsourcing campaigns. The new insights from the business model and research findings enrich the existing understanding on organizing the crowds using collaboration-based mode. In addition, research on team collaboration in innovation communities pay little attentions on the nature of products in the collaboration process. Existing understandings on product collaboration mostly focus on behavioral factors directly, but neglect the product itself that can reflect some hidden behavioral differences among group members. The second essay fills this gap by tapping the source code in open source communities and by revealing the role of different coding styles in the software innovation process. Understanding the role of the nature of the product in the collaboration process helps to explain and resolve more nuanced challenges in open collaboration communities. In summary, the two essays examine important phenomena about open collaboration in innovation communities and extend the related literature.

Second, my dissertation also extends the current literature on group formation in online contexts. In online groups, there are usually frequent entries and exits during the collaboration process, but only few studies capture this dynamics in online groups. The two essays, therefore, attempt to capture the dynamics of membership in online collaboration groups and the evolution of group formation. We show in a dynamic community environment, how online groups and teams can be organized and governed to achieve better efficiency and effectiveness in innovation. In addition, these two essays provide implications on how to form the online groups based members' knowledge variety, familiarity and experience (Harrison and Klein 2007; Huckman et al. 2009) using different perspectives (i.e., distribution of knowledge and programming style). Thus, my dissertation contributes to the research stream on group formation by investigating group dynamics and formation in online communities.

Third, my dissertation attempts to explore group diversity from unique perspectives and contexts. Although studies on group diversity have drawn various

findings and research implications, our knowledge on group diversity in open collaboration is still limited (Ren et al. 2015). The first essay examines diversity in large and dispersed online groups, where diverse individuals induce uncertainties in value creation and management, to extend the diversity literature into the large scale online collaboration context. Due to the lack of communication among members and IT-enabled knowledge system in online communityies, the depth of experience plays a more important role than diversity for in group performance. The second essay focuses on a specific type of diversity – individual work style diversity – which is less likely to be observed in offline work groups, in an open collaboration context. Both essays intend to enrich our knowledge on group diversity in large scaled collaboration and online collectives, where face-to-face interactions and strict control mechanisms are uncommon. Therefore, my dissertation brings group diversity into the new emerged collaboration context in innovation communities and extend the boundary of the literature.

6.2.2 Practical Implications

My dissertation also provides several practical implications for group formation in innovation communities. First, firms need to attract experienced members with both diversity and specialization into the product development process to create value. The difference between T-shaped in other task members and generalists should also be noticed for organizing participants in innovation communities. They need to cultivate community members' experience by recommending tasks that can enrich both diverse and specialized knowledge for their members. Second, software teams should pay greater attention on programming style in the source code to reduce the potential negative consequences (from within file style inconsistency). They should also be careful when leveraging team formation and governance mechanisms. They can implement coding guidelines to control coding style or form project team with familiar ones, but they need to be careful about the non-cooperative actions of

experienced developers. Third, users and leaders for innovation activities in online communities should pay attention to the nature of the product to be developed. Consistent opinions, attitudes and contribution styles on the product (particularly the consistency within the unit of product development) and norms (or routines) in the collaboration group can facilitate the innovation process even though group members may exhibit diversity in other dimensions.

REFERENCES

Ahlstrom, D. 2010. "Innovation and Growth: How Business Contributes to Society," *Academy of Management Perspectives* (24:3), pp. 11-24.

Amabile, T.M. 1983. "The Social Psychology of Creativity: A Componential Conceptualization," *Journal of Personality and Social Psychology* (45:2), p. 357.

Antorini, Y.M., Muniz, A.M., and Askildsen, T. 2012. "Collaborating with Customer Communities: Lessons from the Lego Group," *MIT Sloan Management Review* (53:3), pp. 73-79.

Arabyarmohamady, S., Moradi, H., and Asadpour, M. 2012. "A Coding Style-Based Plagiarism Detection," *International Conference on Interactive Mobile and Computer Aided Learning*, Amman, Jordan: IEEE, pp. 180-186.

Arakji, R.Y., and Lang, K.R. 2007. "Digital Consumer Networks and Producer-Consumer Collaboration: Innovation and Product Development in the Video Game Industry," *Journal of Management Information Systems* (24:2), pp. 195-219.

Archak, N., and Ghose, A. 2010. "Learning-by-Doing and Project Choice: A Dynamic Structural Model of Crowdsourcing," *Thrity First International Conference on Information Systems*, St. Louis, MO.

Archak, N., and Sundararajan, A. 2009. "Optimal Design of Crowdsourcing Contests," *Thirtieth International Conference on Information Systems*, Phoenix, AZ.

Argote, L. 2012. *Organizational Learning: Creating, Retaining and Transferring Knowledge*. Berlin, Germany: Springer.

Armstrong, D.J., and Hardgrave, B.C. 2007. "Understanding Mindshift Learning: The Transition to Object-Oriented Development," *MIS Quarterly* (31:3), pp. 453-474.

August, T., Shin, H., and Tunca, T.I. 2013. "Licensing and Competition for Services in Open Source Software," *Information Systems Research* (24:4), pp. 1068-1086.

Avgustinov, P., Baars, A.I., Henriksen, A.S., Lavender, G., Menzel, G., de Moor, O., Schäfer, M., and Tibble, J. 2015. "Tracking Static Analysis Violations over Time to Capture Developer Characteristics," *37th International Conference on Software Engineering*, Florence, Italy: IEEE Press, pp. 437-447.

Avital, M., Andersson, M., Nickerson, J., Sundararajan, A., Alstyne, M.V., and Verhoeven, D. 2014. "The Collaborative Economy: A Disruptive Innovation or Much Ado About Nothing?," *Thirty Fifth International Conference on Information Systems*, Auckland, New Zealand.

Bagozzi, R.P., and Dholakia, U.M. 2006. "Open Source Software User Communities: A Study of Participation in Linux User Groups," *Management Science* (52:7), pp. 1099-1115.

Bayazit, M., and Mannix, E.A. 2003. "Should I Stay or Should I Go? Predicting Team Members' Intent to Remain in the Team," *Small Group Research* (34:3), pp. 290-321.

Bayus, B.L. 2013. "Crowdsourcing New Product Ideas over Time: An Analysis of the Dell Ideastorm Community," *Management Science* (59:1), pp. 226-244.

Bechky, B.A. 2003. "Sharing Meaning across Occupational Communities: The Transformation of Understanding on a Production Floor," *Organization Science* (14:3), pp. 312-330.

Belsley, D.A., Kuh, E., and Welsch, R.E. 2005. *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. Hoboken, NJ: John Wiley & Sons.

Binkley, D., Davis, M., Lawrie, D., Maletic, J.I., Morrell, C., and Sharif, B. 2013. "The Impact of Identifier Style on Effort and Comprehension," *Empirical Software Engineering* (18:2), pp. 219-276.

Blau, P.M. 1977. *Inequality and Heterogeneity: A Primitive Theory of Social Structure*. New York: Free Press.

Blincoe, K., and Damian, D. 2015. "Implicit Coordination: A Case Study of the Rails OSS Project," in *Open Source Systems: Adoption and Impact*. Berlin, Germany: Springer, pp. 35-44.

Bogers, M., Afuah, A., and Bastian, B. 2010. "Users as Innovators: A Review, Critique, and Future Research Directions," *Journal of Management* (36:4), July 1, 2010, pp. 857-875.

Boh, W.F., Evaristo, R., and Ouderkirk, A. 2014. "Balancing Breadth and Depth of Expertise for Innovation: A 3M Story," *Research Policy* (43:2), pp. 349-366.

Boh, W.F., Slaughter, S.A., and Espinosa, J.A. 2007. "Learning from Experience in Software Development: A Multilevel Analysis," *Management Science* (53:8), pp. 1315-1331.

Boogerd, C., and Moonen, L. 2008. "Assessing the Value of Coding Standards: An Empirical Study," *IEEE International Conference on Software Maintenance*, Beijing, China, pp. 277-286.

Boudreau, K. 2010. "Open Platform Strategies and Innovation: Granting Access Vs. Devolving Control," *Management Science* (56:10), pp. 1849-1872.

Boudreau, K., Gaule, P., Lakhani, K.R., Riedl, C., and Woolley, A.W. 2014. "From Crowds to Collaborators: Initiating Effort & Catalyzing Interactions among Online Creative Workers." Harvard Business School Technology & Operations Management Unit Working Paper.

Boudreau, K., Lacetera, N., and Lakhani, K.R. 2011. "Incentives and Problem Uncertainty in Innovation Contests: An Empirical Analysis," *Management Science* (57:5), pp. 843-863.

Boudreau, K., and Lakhani, K.R. 2013. "Using the Crowd as an Innovation Partner," *Harvard Business Review* (91:4), pp. 60-69.

Buse, R.P., and Weimer, W.R. 2010. "Learning a Metric for Code Readability," *IEEE Transactions on Software Engineering* (36:4), pp. 546-558.

Cahalane, M., Feller, J., Finnegan, P., Hayes, J., and OReilly, P. 2014. "Leveraging Distributed Collective Intelligence: An Investigation of Solver Engagement with Innovation Challenges," *Thirty Fifth International Conference on Information Systems*, Auckland, New Zealand.

Caliendo, M., and Kopeinig, S. 2008. "Some Practical Guidance for the Implementation of Propensity Score Matching," *Journal of Economic Surveys* (22:1), pp. 31-72.

Caliński, T., and Harabasz, J. 1974. "A Dendrite Method for Cluster Analysis," *Communications in Statistics* (3:1), pp. 1-27.

Caliskan-Islam, A., Harang, R., Liu, A., Narayanan, A., Voss, C., Yamaguchi, F., and Greenstadt, R. 2015. "De-Anonymizing Programmers Via Code Stylometry," *24th USENIX Security Symposium (USENIX Security 15)*, Washington, D.C., pp. 255-270.

Cannella, A.A., Park, J.-H., and Lee, H.-U. 2008. "Top Management Team Functional Background Diversity and Firm Performance: Examining the Roles of Team Member Colocation and Environmental Uncertainty," *Academy of Management Journal* (51:4), pp. 768-784.

Capiluppi, A., Boldyreff, C., Beecher, K., and Adams, P.J. 2009. "Quality Factors and Coding Standards–a Comparison between Open Source Forges," *Electronic Notes in Theoretical Computer Science* (233), pp. 89-103.

Carte, T., and Chidambaram, L. 2004. "A Capabilities-Based Theory of Technology Deployment in Diverse Teams: Leapfrogging the Pitfalls of Diversity and Leveraging Its Potential with Collaborative Technology," *Journal of the Association for Information Systems* (5:11), pp. 448-471.

Certo, S.T., Busenbark, J.R., Woo, H.s., and Semadeni, M. 2016. "Sample Selection Bias and Heckman Models in Strategic Management Research," *Strategic Management Journal* (37:13), pp. 2639-2657.

Chesbrough, H.W., and Appleyard, M.M. 2007. "Open Innovation and Strategy," *California Management Review* (50:1), pp. 57-76.

Chiravuri, A., Nazareth, D., and Ramamurthy, K. 2011. "Cognitive Conflict and Consensus Generation in Virtual Teams During Knowledge Capture: Comparative Effectiveness of Techniques," *Journal of Management Information Systems* (28:1), pp. 311-350.

Choi, J., Ferwerda, B., Hahn, J., Kim, J., and Moon, J.Y. 2013. "Impact of Social Features Implemented in Open Collaboration Platforms on Volunteer Self-Organization: Case Study of Open Source Software Development," *2013 Joint International Symposium on Wikis and Open Collaboration (WikiSym + OpenSym 2013)*, Hong Kong, China: ACM, p. 25.

Cohen, J., Cohen, P., West, S.G., and Aiken, L.S. 2013. *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. London: Routledge.

Cox, A., and Fisher, M. 2009. "Programming Style: Influences, Factors, and Elements," *Advances in Computer-Human Interactions, 2009. ACHI'09. Second International Conferences on*: IEEE, pp. 82-89.

Cramton, C.D. 2001. "The Mutual Knowledge Problem and Its Consequences for Dispersed Collaboration," *Organization Science* (12:3), pp. 346-371.

Crowston, K., Li, Q., Wei, K., Eseryel, U.Y., and Howison, J. 2007. "Self-Organization of Teams for Free/Libre Open Source Software Development," *Information and Software Technology* (49:6), pp. 564-575.

Dabbish, L., Stuart, C., Tsay, J., and Herbsleb, J. 2012. "Social Coding in Github: Transparency and Collaboration in an Open Software Repository," *Proceedings of the 2012 ACM conference on Computer Supported Cooperative Work*, Seattle, WA: ACM, pp. 1277-1286.

Dahlin, K.B., Weingart, L.R., and Hinds, P.J. 2005. "Team Diversity and Information Use," *Academy of Management Journal* (48:6), pp. 1107-1123.

Daniel, S., Agarwal, R., and Stewart, K.J. 2013. "The Effects of Diversity in Global, Distributed Collectives: A Study of Open Source Project Success," *Information Systems Research* (24:2), pp. 312-333.

Di Gangi, P.M., and Wasko, M. 2009. "Steal My Idea! Organizational Adoption of User Innovations from a User Innovation Community: A Case Study of Dell Ideastorm," *Decision Support Systems* (48:1), pp. 303-312.

Di Gangi, P.M., Wasko, M.M., and Hooker, R.E. 2010. "Getting Customers' Ideas to Work for You: Learning from Dell How to Succeed with Online User Innovation Communities," *MIS Quarterly Executive* (9:4), pp. 213-228.

Dissanayake, I., Zhang, J., and Gu, B. 2014. "Virtual Team Performance in Crowdsourcing Contests: A Social Network Perspective," *Thirty Fifth International Conference on Information Systems*, Auckland, New Zealand.

Earley, C.P., and Mosakowski, E. 2000. "Creating Hybrid Team Cultures: An Empirical Test of Transnational Team Functioning," *Academy of Management Journal* (43:1), pp. 26-49.

Ely, R.J. 2004. "A Field Study of Group Diversity, Participation in Diversity Education Programs, and Performance," *Journal of Organizational Behavior* (25:6), pp. 755-780.

Estelles-Arolas, E., and Gonzalez-Ladron-de-Guevara, F. 2012. "Towards an Integrated Crowdsourcing Definition," *Journal of Information Science* (38:2), pp. 189-200.

Evangelist, M. 1984. "Program Complexity and Programming Style," *IEEE International Conference on Data Engineering*: IEEE, pp. 534-541.

Fang, Y., and Neufeld, D. 2009. "Understanding Sustained Participation in Open Source Software Projects," *Journal of Management Information Systems* (25:4), pp. 9-50.

Faraj, S., Jarvenpaa, S.L., and Majchrzak, A. 2011. "Knowledge Collaboration in Online Communities," *Organization Science* (22:5), pp. 1224-1239.

Feller, J., Finnegan, P., Fitzgerald, B., and Hayes, J. 2008. "From Peer Production to Productization: A Study of Socially Enabled Business Exchanges in Open Source Service Networks," *Information Systems Research* (19:4), pp. 475-493.

Fichter, K. 2009. "Innovation Communities: The Role of Networks of Promotors in Open Innovation," *R&D Management* (39:4), pp. 357-371.

Fitzgerald, B. 2006. "The Transformation of Open Source Software," *MIS Quarterly* (30:3), pp. 587-598.

Franke, N., and Shah, S. 2003. "How Communities Support Innovative Activities: An Exploration of Assistance and Sharing among End-Users," *Research Policy* (32:1), pp. 157-178.

Frantzeskou, G., Stamatatos, E., Gritzalis, S., and Katsikas, S. 2006. "Effective Identification of Source Code Authors Using Byte-Level Information," *28th International Conference on Software Engineering*: ACM, pp. 893-896.

Füller, J., Bartl, M., Ernst, H., and Mühlbacher, H. 2006. "Community Based Innovation: How to Integrate Members of Virtual Communities into New Product Development," *Electronic Commerce Research* (6:1), pp. 57-73.

Füller, J., Matzler, K., and Hoppe, M. 2008. "Brand Community Members as a Source of Innovation," *Journal of Product Innovation Management* (25:6), pp. 608-619.

Geiger, D., Seedorf, S., Schulze, T., Nickerson, R.C., and Schader, M. 2011. "Managing the Crowd: Towards a Taxonomy of Crowdsourcing Processes," *17th Americas Conference on Information Systems*, Detroit, MI.

Gläser, J. 2001. "Producing Communities' as a Theoretical Challenge," *Proceedings of The Australian Sociological Association*, Sydney, Australia, pp. 1-11.

Godfrey, M.W., and Tu, Q. 2000. "Evolution in Open Source Software: A Case Study," *International Conference on Software Maintenance*, San Jose, CA: IEEE, pp. 131-142.

Gousios, G. 2013. "The GhTorent Dataset and Tool Suite," *Proceedings of the 10th Working Conference on Mining Software Repositories*, San Francisco, CA: IEEE, pp. 233-236.

Gousios, G., Pinzger, M., and Deursen, A.v. 2014. "An Exploratory Study of the Pull-Based Software Development Model," *Proceedings of the 36th International Conference on Software Engineering*, Hyderabad, India: ACM, pp. 345-355.

Graham, P. 2004. *Hackers & Painters: Big Ideas from the Computer Age*. Boston, US: O'Reilly.

Greer, C.R., and Lei, D. 2012. "Collaborative Innovation with Customers: A Review of the Literature and Suggestions for Future Research," *International Journal of Management Reviews* (14:1), pp. 63-84.

Grewal, R., Lilien, G.L., and Mallapragada, G. 2006. "Location, Location, Location: How Network Embeddedness Affects Project Success in Open Source Systems," *Management Science* (52:7), pp. 1043-1056.

Haefliger, S., von Krogh, G., and Spaeth, S. 2007. "Code Reuse in Open Source Software," *Management Science* (54:1), pp. 180-193.

Hahn, J., and Lee, G. 2013. "Archetypes of Crowdfunders' Backing Behaviors and the Outcome of Crowdfunding Efforts: An Exploratory Analysis of Kickstarter," in: *INFORMS Conference on Information Systems and Technology*. Minneapolis, MN.

Hahn, J., Moon, J.Y., and Zhang, C. 2008. "Emergence of New Project Teams from Open Source Software Developer Networks: Impact of Prior Collaboration Ties," *Information Systems Research* (19:3), pp. 369-391.

Hann, I.-H., Roberts, J.A., and Slaughter, S.A. 2013. "All Are Not Equal: An Examination of the Economic Returns to Different Forms of Participation in Open Source Software Communities," *Information Systems Research* (24:3), 2013/09/01, pp. 520-538.

Hansen, M.T., and Von Oetinger, B. 2001. "Introducing T-Shaped Managers. Knowledge Management's Next Generation," *Harvard Business Review* (79:3), pp. 106-116, 165.

Harrison, D.A., and Klein, K.J. 2007. "What's the Difference? Diversity Constructs as Separation, Variety, or Disparity in Organizations," *Academy of Management Review* (32:4), pp. 1199-1228.

Harrison, D.A., Price, K.H., and Bell, M.P. 1998. "Beyond Relational Demography: Time and the Effects of Surface-and Deep-Level Diversity on Work Group Cohesion," *Academy of Management Journal* (41:1), pp. 96-107.

Harrison, D.A., Price, K.H., Gavin, J.H., and Florey, A.T. 2002. "Time, Teams, and Task Performance: Changing Effects of Surface-and Deep-Level Diversity on Group Functioning," *Academy of Management Journal* (45:5), pp. 1029-1045.

Hars, A., and Ou, S. 2002. "Working for Free? Motivations for Participating in Open-Source Projects," *International Journal of Electronic Commerce* (6:3), pp. 25-39.

Haythornthwaite, C. 2009. "Crowds and Communities: Light and Heavyweight Models of Peer Production," *42nd Hawaii International Conference on System Sciences*: IEEE, pp. 1-10.

He, J., Butler, B.S., and King, W.R. 2007. "Team Cognition: Development and Evolution in Software Project Teams," *Journal of Management Information Systems* (24:2), pp. 261-292.

Heckman, J.J. 1979. "Sample Selection Bias as a Specification Error," *Econometrica* (47:1), pp. 153-161.

Horwitz, S.K., and Horwitz, I.B. 2007. "The Effects of Team Diversity on Team Outcomes: A Meta-Analytic Review of Team Demography," *Journal of Management* (33:6), pp. 987-1015.

Hou, W., Li, D., and Zheng, H. 2011. "Task Design, Motivation, and Participation in Crowdsourcing Contests," *International Journal of Electronic Commerce* (15:4), pp. 57-88.

Howe, J. 2006. "The Rise of Crowdsourcing," *Wired Magazine* (14:6), pp. 1-4.

Howe, J. 2008. *Crowdsourcing: How the Power of the Crowd Is Driving the Future of Business*. New York: Random House.

Huang, Y., Singh, P., and Mukhopadhyay, T. 2012. "How to Design Crowdsourcing Contest: A Structural Empirical Analysis," *Workshop of Information Systems and Economics*, Oriando, Florida.

Huang, Y., Singh, P.V., and Srinivasan, K. 2014. "Crowdsourcing New Product Ideas under Consumer Learning," *Management Science* (60:9), pp. 2138-2159.

Huckman, R.S., Staats, B.R., and Upton, D.M. 2009. "Team Familiarity, Role Experience, and Performance: Evidence from Indian Software Services," *Management Science* (55:1), pp. 85-100.

Hwang, E., Singh, P.V., and Argote, L. 2014. "Jack of All, Master of Some: The Contingent Effect of Knowledge Breadth on Innovation," *Thirty Fifth International Conference on Information Systems*, Auckland, New Zealand.

Inbar, Y., and Barzilay, O. 2014. "Community Impact on Crowdfunding Performance." Available at SSRN: http://ssrn.com/abstract=2524910.

Jackson, S.E., and Joshi, A. 2004. "Diversity in Social Context: A Multi-Attribute, Multilevel Analysis of Team Diversity and Sales Performance," *Journal of Organizational Behavior* (25:6), pp. 675-702.

Jackson, S.E., May, K.E., and Whitney, K. 1995. "Understanding the Dynamics of Diversity in Decision-Making Teams," in *Team Effectiveness and Decision Making in Organizations*. San Fancisco: Jossey-Bass, pp. 204-261.

Jiang, L., and Wagner, C. 2014. "Structuring Time through Participation in Micro-Task Crowdsourcing: A Time Allocation Perspective," *Thirty Fifth Internaltional Conference on Information Systems*, Auckland, New Zealand.

Johnson, M.K., and Hasher, L. 1987. "Human Learning and Memory," *Annual review of psychology* (38:1), pp. 631-668.

Kalliamvakou, E., Damian, D., Singer, L., and German, D.M. 2014a. "The Code-Centric Collaboration Perspective: Evidence from Github," Technical Report DCS-352-IR, University of Victoria.

Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D.M., and Damian, D. 2014b. "The Promises and Perils of Mining Github," *Proceedings of the 11th Working conference on Mining Software Repositories*, Hyderabad, India: ACM, pp. 92-101.

Kang, K., Hahn, J., and De, P. 2012. "The Impact of Depth and/or Breadth of Experiences in Software Development Productivity," in: *Academy of Management Conference (OCIS Division)*. Boston: MA.

Kankanhalli, A., Tan, B.C.Y., and Wei, K.-K. 2006. "Conflict and Performance in Global Virtual Teams," *Journal of Management Information Systems* (23:3), pp. 237-274.

Ke, W., and Zhang, P. 2009. "Motivations in Open Source Software Communities: The Mediating Role of Effort Intensity and Goal Commitment," *International Journal of Electronic Commerce* (13:4), pp. 39-66.

Kernighan, B.W., and Plauger, P.J. 1978. *The Elements of Programming Style*. New York: McGraw-Hill.

Ketchen, D.J., and Shook, C.L. 1996. "The Application of Cluster Analysis in Strategic Management Research: An Analysis and Critique," *Strategic Management Journal* (17:6), pp. 441-458.

Koh, T.K. 2014. "Participants' Strategy in Crowd-Based Design Contests–a Prospect Theory Perspective," *Thirty Fifth International Conference on Information Systems*, Auckland, New Zealand.

Krcmar, H., Bretschneider, U., Huber, M., and Leimeister, J.M. 2009. "Leveraging Crowdsourcing: Activation-Supporting Components for It-Based Ideas Competition," *Journal of Management Information Systems* (26:1), pp. 197-224.

Lakhani, K.R., and Von Hippel, E. 2003. "How Open Source Software Works:"Free" User-to-User Assistance," *Research Policy* (32:6), pp. 923-943.

Langlois, R.N., and Garzarelli, G. 2008. "Of Hackers and Hairdressers: Modularity and the Organizational Economics of Open‐Source Collaboration," *Industry and Innovation* (15:2), pp. 125-143.

Lee, G.K., and Cole, R.E. 2003. "From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development," *Organization Science* (14:6), pp. 633-649.

Lee, M.J., Ferwerda, B., Choi, J., Hahn, J., Moon, J.Y., and Kim, J. 2013a. "Github Developers Use Rockstars to Overcome Overflow of News," *CHI 2013 on Human Factors in Computing Systems*, Paris, France: ACM, pp. 133-138.

Lee, T., Lee, J.B., and In, H.P. 2013b. "A Study of Different Coding Styles Affecting Code Readability," *International Journal of Software Engineering and Its Applications* (7:5), pp. 413-422.

Lerner, J., and Tirole, J. 2002. "Some Simple Economics of Open Source," *The Journal of Industrial Economics* (50:2), pp. 197-234.

Lettl, C., Herstatt, C., and Gemuenden, H.G. 2006. "Users' Contributions to Radical Innovation: Evidence from Four Cases in the Field of Medical Equipment Technology," *R&D Management* (36:3), pp. 251-272.

Levine, S.S., and Prietula, M.J. 2013. "Open Collaboration for Innovation: Principles and Performance," *Organization Science* (25:5), pp. 1414-1433.

Li, X., Yoo, Y., and Zhang, Z. 2016. "Searching for “Stability” in Fluidity: A Routine-Based View of Open Source Software Development Process," *Thirty Seventh International Conference on Information Systems*, Dublin, Ireland.

Lichtenthaler, U. 2011. "Open Innovation: Past Research, Current Debates, and Future Directions," *Academy of Management Perspectives* (25:1), pp. 75-93.

Lin, Y., Boh, W.F., and Goh, K.H. 2014. "How Different Are Crowdfunders? Examining Archetypes of Crowdfunders and Their Choice of Projects." Available at SSRN: http://ssrn.com/abstract=2397571.

Liu, D., Chen, L., and Xu, P. 2018. "Why Crowd Pick Different Winners from Experts: Evidence from Crowdsourcing Contests," *12th China Summer Workshop on Information Management*, Qingdao, China.

Maimon, O., and Rokach, L. 2005. *Data Mining and Knowledge Discovery Handbook*. New York: Springer.

Majchrzak, A., and Malhotra, A. 2013. "Towards an Information Systems Perspective and Research Agenda on Crowdsourcing for Innovation," *The Journal of Strategic Information Systems* (22:4), pp. 257-268.

Mallapragada, G., Grewal, R., and Lilien, G. 2012. "User-Generated Open Source Products: Founder's Social Capital and Time to Product Release," *Marketing Science* (31:3), pp. 474-492.

Malone, T.W., Laubacher, R., and Dellarocas, C. 2010. "The Collective Intelligence Genome," *IEEE Engineering Management Review* (38:3), p. 38.

Mannes, A.E. 2009. "Are We Wise About the Wisdom of Crowds? The Use of Group Judgments in Belief Revision," *Management Science* (55:8), pp. 1267-1279.

March, J.G. 1991. "Exploration and Exploitation in Organizational Learning," *Organization Science* (2:1), pp. 71-87.

McCabe, T.J. 1976. "A Complexity Measure," *IEEE Transactions on Software Engineering* (SE-2:4), pp. 308-320.

McGrath, J.E., Berdahl, J.L., and Arrow, H. 1995. *Traits, Expectations, Culture, and Clout: The Dynamics of Diversity in Work Groups*. Washington, DC, US: American Psychological Association.

Menon, N.M., Mishra, A., and Ye, S. 2017. "Beyond Related Experience: Upstream Versus Downstream Experience in Innovation Contest Platforms with Interdependent Problem Domains." Available at SSRN: http://ssrn.com/abstract=2983599.

Mi, Q., Keung, J., and Yu, Y. 2016. "Measuring the Stylistic Inconsistency in Software Projects Using Hierarchical Agglomerative Clustering," *Proceedings of the 12th International Conference on Predictive Models and Data Analytics in Software Engineering*, Ciudad Real, Spain: ACM.

Miara, R.J., Musselman, J.A., Navarro, J.A., and Shneiderman, B. 1983. "Program Indentation and Comprehensibility," *Communications of the ACM* (26:11), pp. 861-867.

Moghadam, J.B., Choudhury, R.R., Yin, H., and Fox, A. 2015. "Autostyle: Toward Coding Style Feedback at Scale," *Proceedings of the Second ACM Conference on Learning at Scale*, Vancouver, Canada: ACM, pp. 261-266.

Mohan, A., and Gold, N. 2004. "Programming Style Changes in Evolving Source Code," *Proceedings of 12th IEEE International Workshop on Program Comprehension*, Bari, Italy: IEEE, pp. 236-240.

Moqri, M., Bandyopadhyay, S., and Cheng, H. 2014. "A Contract for "Crowds"," *Thirty Fifth International Conference on Information Systems*, Auckland, New Zealand.

Moqri, M., Qiu, L., Bandyopadhyay, S., and Horowitz, I. 2015. "The Effect of 'Following' on Contributions to Open Source Communities," *Thirty Sixth International Conference on Information Systems*, Fort Worth, TX.

Murgia, A., Concas, G., Tonelli, R., Ortu, M., Demeyer, S., and Marchesi, M. 2014. "On the Influence of Maintenance Activity Types on the Issue Resolution Time," *Proceedings of the 10th International Conference on Predictive Models in Software Engineering*: ACM, pp. 12-21.

Nambisan, S., and Baron, R.A. 2010. "Different Roles, Different Strokes: Organizing Virtual Customer Environments to Promote Two Types of Customer Contributions," *Organization Science* (21:2), pp. 554-572.

Narayanan, S., Balasubramanian, S., and Swaminathan, J.M. 2009. "A Matter of Balance: Specialization, Task Variety, and Individual Learning in a Software Maintenance Environment," *Management Science* (55:11), pp. 1861-1876.

Nguyen, C., Oh, O., Kocsis, D., and Vreede, G.-J. 2013. "Crowdsourcing as Lego: Unpacking the Building Blocks of Crowdsourcing Collaboration Processes," *Thirty Fourth International Conference on Information Systems*, Milan, Italy.

Nickerson, J., Brunswicker, S., Butler, B., and Wagner, C. 2014. "The Evolution of Ideas by Crowds and Communities: Competition Vs. Cooperation," *Thirty Fifth International Conference on Information Systems*, Auckland, New Zealand.

Nonaka, I., and Takeuchi, H. 1995. *The Knowledge Creating Company: How Japanese Create the Dynamics of Innovation*. New York: Oxford University Press.

Novick, L.R. 1988. "Analogical Transfer, Problem Similarity, and Expertise," *Journal of Experimental Psychology: Learning, Memory, and Cognition* (14:3), p. 510.

Oh, H., Animesh, A., and Pinsonneault, A. 2015. "Value Co-Creation in Crowdsourcing: The Effects of Social Networks on Product Co-Development Project Success," in: *INFORM Conference on Information Systems and Technology*. Philadelphia, PA.

Oh, W., and Jeon, S. 2007. "Membership Herding and Network Stability in the Open Source Community: The Ising Perspective," *Management Science* (53:7), pp. 1086-1101.

Ohno, A. 2013. "A Methodology to Teach Exemplary Coding Style Considering Students' Coding Style Feature Contains Fluctuations," *IEEE Frontiers in Education Conference*, Oklahoma City, OK: IEEE, pp. 1908-1910.

Oman, P.W., and Cook, C.R. 1988. "A Paradigm for Programming Style Research," *ACM Sigplan Notices* (23:12), pp. 69-78.

Paulini, M., Murty, P., and Maher, M.L. 2013. "Design Processes in Collective Innovation Communities: A Study of Communication," *CoDesign* (9:2), pp. 90-112.

Pedersen, J., Kocsis, D., Tripathi, A., Tarrell, A., Weerakoon, A., Tahmasbi, N., Xiong, J., Deng, W., Oh, O., and de Vreede, G.-J. 2013. "Conceptual Foundations of Crowdsourcing: A Review of Is Research," *46th Hawaii International Conference on System Sciences*, Hawaii: IEEE, pp. 579-588.

Pelled, L.H. 1996. "Demographic Diversity, Conflict, and Work Group Outcomes: An Intervening Process Theory," *Organization Science* (7:6), pp. 615-631.

Pelled, L.H., Eisenhardt, K.M., and Xin, K.R. 1999. "Exploring the Black Box: An Analysis of Work Group Diversity, Conflict and Performance," *Administrative Science Quarterly* (44:1), pp. 1-28.

Perry-Smith, J.E., and Shalley, C.E. 2003. "The Social Side of Creativity: A Static and Dynamic Social Network Perspective," *Academy of Management Review* (28:1), pp. 89-106.

Phillips, D.J., and Zuckerman, E.W. 2001. "Middle-Status Conformity: Theoretical Restatement and Empirical Demonstration in Two Markets," *American Journal of Sociology* (107:2), pp. 379-429.

Piller, F.T., and Walcher, D. 2006. "Toolkits for Idea Competitions: A Novel Method to Integrate Users in New Product Development," *R&D Management* (36:3), pp. 307-318.

Prause, C.R., and Jarke, M. 2015. "Gamification for Enforcing Coding Conventions," *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, Bergamo, Italy: ACM, pp. 649-660.

Ramaswamy, V., and Gouillart, F.J. 2010. *The Power of Co-Creation: Build It with Them to Boost Growth, Productivity, and Profits*. New York: Simon and Schuster.

Ransbotham, S., and Kane, K.C. 2011. "Membership Turnover and Collaboration Success in Online Communities: Explaining Rises," *MIS Quarterly* (35:3), pp. 613-627.

Reed, D. 2010. "Sometimes Style Really Does Matter," *Journal of Computing Sciences in Colleges* (25:5), pp. 180-187.

Reiss, S.P. 2007. "Automatic Code Stylizing," *Proceedings of the 22th IEEE/ACM International Conference on Automated Software Engineering*, Atlanta, GA: ACM, pp. 74-83.

Ren, Y., Chen, J., and Riedl, J. 2015. "The Impact and Evolution of Group Diversity in Online Open Collaboration," *Management Science* (62:6), pp. 1668-1686.

Riaz, M., Mendes, E., and Tempero, E. 2009. "A Systematic Review of Software Maintainability Prediction and Metrics," *Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement*, Lake Buena Vista, FL: IEEE Computer Society, pp. 367-377.

Roberts, J.A., Hann, I.-H., and Slaughter, S.A. 2006. "Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects," *Management Science* (52:7), pp. 984-999.

Rubin, D.B. 2008. "Causal Inference Using Potential Outcomes: Design, Modeling, Decisions," *Journal of the American Statistical Association* (62:3), pp. 277-278.

Rulke, D.L., and Galaskiewicz, J. 2000. "Distribution of Knowledge, Group Network Structure, and Group Performance," *Management Science* (46:5), pp. 612-625.

Savage, N. 2012. "Gaining Wisdom from Crowds," *Communications of the ACM* (55:3), pp. 13-15.

Schneider, B. 1987. "The People Make the Place," *Personnel Psychology* (40:3), pp. 437-453.

Schneider, B., Goldstiein, H.W., and Smith, D.B. 1995. "The Asa Framework: An Update," *Personnel psychology* (48:4), pp. 747-773.

Shah, S.K. 2006. "Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development," *Management Science* (52:7), pp. 1000-1014.

Sheoran, J., Blincoe, K., Kalliamvakou, E., Damian, D., and Ell, J. 2014. "Understanding Watchers on Github," *Proceedings of the 11th Working Conference on Mining Software Repositories*, Hyderabad, India: ACM, pp. 336-339.

Siegel, P.A., and Hambrick, D.C. 2005. "Pay Disparities within Top Management Groups: Evidence of Harmful Effects on Performance of High-Technology Firms," *Organization Science* (16:3), pp. 259-274.

Singh, P.V. 2010. "The Small-World Effect: The Influence of Macro-Level Properties of Developer Collaboration Networks on Open-Source Project Success," *ACM Transactions on Software Engineering and Methodology* (20:2), p. 6.

Singh, P.V., and Phelps, C. 2013. "Networks, Social Influence, and the Choice among Competing Innovations: Insights from Open Source Software Licenses," *Information Systems Research* (24:3), pp. 539-560.

Singh, P.V., and Tan, Y. 2010. "Developer Heterogeneity and Formation of Communication Networks in Open Source Software Projects," *Journal of Management Information Systems* (27:3), pp. 179-210.

Singh, P.V., Tan, Y., and Mookerjee, V. 2011. "Network Effects: The Influence of Structural Social Capital on Open Source Project Success," *MIS Quarterly* (35:4), pp. 813-829.

Singh, P.V., Tan, Y., and Youn, N. 2010. "A Hidden Markov Model of Developer Learning Dynamics in Open Source Software Projects," *Information Systems Research* (22:4), pp. 790-807.

Smit, M., Gergel, B., Hoover, H.J., and Stroulia, E. 2011a. "Code Convention Adherence in Evolving Software," *27th IEEE International Conference on Software Maintenance*, Williamsburg, VA: IEEE, pp. 504-507.

Smit, M., Gergel, B., Hoover, H.J., and Stroulia, E. 2011b. "Maintainability and Source Code Conventions: An Analysis of Open Source Projects," Technical Report TR11-06, University of Alberta, Department of Computing Science,.

Soloway, E., and Ehrlich, K. 1984. "Empirical Studies of Programming Knowledge," *IEEE Transactions on Software Engineering* (SE-10:5), pp. 595-609.

Spinellis, D. 2011. "Elyts Edoc," *IEEE Software* (28:2), pp. 104-104.

Stewart, D. 2005. "Social Status in an Open-Source Community," *American Sociological Review* (70:5), pp. 823-842.

Stewart, K.J., and Gosain, S. 2006. "The Impact of Ideology on Effectiveness in Open Source Software Development Teams," *MIS Quarterly* (30:2), pp. 291-314.

Surowiecki, J. 2004. *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*. London: Little, Brown.

Taylor, A., and Greve, H.R. 2006. "Superman or the Fantastic Four? Knowledge Combination and Experience in Innovative Teams," *Academy of Management Journal* (49:4), pp. 723-740.

Terwiesch, C., and Xu, Y. 2008. "Innovation Contests, Open Innovation, and Multiagent Problem Solving," *Management Science* (54:9), pp. 1529-1543.

Tibshirani, R., Walther, G., and Hastie, T. 2001. "Estimating the Number of Clusters in a Data Set Via the Gap Statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* (63:2), pp. 411-423.

van der Vegt, G.S., and Bunderson, J.S. 2005. "Learning and Performance in Multidisciplinary Teams: The Importance of Collective Team Identification," *Academy of Management Journal* (48:3), pp. 532-547.

van Knippenberg, D., De Dreu, C.K., and Homan, A. 2004. "Work Group Diversity and Group Performance: An Integrative Model and Research Agenda," *The Journal of Applied Psychology* (89:6), pp. 1008-1022.

van Knippenberg, D., and Schippers, M.C. 2007. "Work Group Diversity," *Annual Review of Psychology* (58), pp. 515-541.

van Knippenberg, D., and van Ginkel, W.P. 2010. "The Categorization-Elaboration Model of Work Group Diversity: Wielding the Double-Edged Sword," in *The Psychology of Social and Cultural Diversity*. Oxford, UK: Wiley-Blackwell, pp. 257-280.

von Hippel, E. 2005. "Open Source Software Projects as User Innovation Networks," in *Perspectives on Free and Open Source Software*. Cambridge, MA: MIT Press, pp. 267-278.

von Hippel, E., and von Krogh, G. 2003. "Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science," *Organization Science* (14:2), pp. 209-223.

von Krogh, G., Haefliger, S., Spaeth, S., and Wallin, M.W. 2012. "Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development," *MIS Quarterly* (36:2), pp. 649-676.

von Krogh, G., and von Hippel, E. 2006. "The Promise of Research on Open Source Software," *Management Science* (52:7), pp. 975-983.

Vreede, G.-J., Briggs, R.O., and Massey, A.P. 2009. "Collaboration Engineering: Foundations and Opportunities: Editorial to the Special Issue on the Journal of the Association of Information Systems," *Journal of the Association for Information Systems* (10:3), pp. 121-137.

Weisberg, R.W. 1999. "Creativity and Knowledge: A Challenge of Theories," in *Handbook of Creativity*. Cambridge, UK: Cambridge University Press, pp. 226-250.

West, J., and Lakhani, K.R. 2008. "Getting Clear About Communities in Open Innovation," *Industry and Innovation* (15:2), pp. 223-231.

West, M.A. 2002. "Sparkling Fountains or Stagnant Ponds: An Integrative Model of Creativity and Innovation Implementation in Work Groups," *Applied Psychology* (51:3), pp. 355-387.

Williams, K.Y., and O'Reilly, C.A. 1998. "Demography and Diversity in Organizations: A Review of 40 Years of Research," *Research in Organizational Behaviors* (20), pp. 77-140.

Woodfield, S.N., Dunsmore, H.E., and Shen, V.Y. 1981. "The Effect of Modularization and Comments on Program Comprehension," *Proceedings of the 5th International Conference on Software Engineering*, San Diego, California: IEEE Press, pp. 215-223.

Wooldridge, J.M. 2010. *Econometric Analysis of Cross Section and Panel Data*. Cambridge, MA: MIT press.

Wulf, G., and Schmidt, R.A. 1997. "Variability of Practice and Implicit Motor Learning," *Journal of Experimental Psychology: Learning, Memory, and Cognition* (23:4), p. 987.

Yang, J., Adamic, L.A., and Ackerman, M.S. 2008. "Crowdsourcing and Knowledge Sharing: Strategic User Behavior on Taskcn," *Proceedings of the 9th ACM Conference on Electronic Commerce*: ACM, pp. 246-255.

Yang, Y., Chen, P.Y., and Banker, R. 2010. "Impact of Past Performance and Strategic Bidding on Winner Determination of Open Innovation Contest," *Workshop on Information Systems and Economics*, St. Louis, MO.

Yang, Y., Chen, P.Y., and Pavlou, P. 2009. "Open Innovation: An Empirical Study of Online Contests," *Thirtieth International Conference on Information Systems*, Phoenix, AZ.

Yu, Y., Yin, G., Wang, H., and Wang, T. 2014. "Exploring the Patterns of Social Behavior in Github," *Proceedings of the 1st International Workshop on Crowd-based Software Development Methods and Technologies*, Hong Kong, China: ACM, pp. 31-36.

Zhang, C., Hahn, J., and De, P. 2013. "Continued Participation in Online Innovation Communities: Does Community Response Matter Equally for Everyone?," *Information Systems Research* (24:4), pp. 1112-1130.

Zhang, S., Singh, P.V., and Ghose, A. 2017. "A Structural Analysis of the Role of Superstars in Crowdsourcing Contests." Available at SSRN: http://ssrn.com/abstract=2764553.

Zhang, Z., Yoo, Y., Wattal, S., Zhang, B., and Kulathinal, R. 2014. "Generative Diffusion of Innovations and Knowledge Networks in Open Source Projects," *Thirty Fourth International Conference on Information Systems*, Auckland, New Zealand.

Zheng, H., Li, D., and Hou, W. 2011. "Task Design, Motivation, and Participation in Crowdsourcing Contests," *International Journal of Electronic Commerce* (15:4), pp. 57-88.

Zhu, K.X., and Zhou, Z.Z. 2011. "Lock-in Strategy in Software Competition: Open-Source Software Vs. Proprietary Software," *Information Systems Research* (23:2), pp. 536-545.