Developer Management in FLOSS Projects

Theoretical Concepts and Empirical Evidence

Andreas Schilling
University of Bamberg

Diese Arbeit hat der Fakultät Wirtschaftsinformatik und Angewandte Informatik der Otto-Friedrich-Universität Bamberg als Dissertation vorgelegen.

Erstgutachter: Prof. Dr. Tim Weitzel

Zweitgutachter: Prof. Dr. Kai Fischbach

Mitglied der Promotionskommission: Prof. Dr. Guido Wirtz

Tag der Disputation: 16.07.2015

Developer Management in FLOSS Projects

Theoretical Concepts and Empirical Evidence

Andreas Schilling
University of Bamberg

Including a Foreword by Prof. Dr. Tim Weitzel

Dedicated to my family

Foreword by Prof. Dr. Tim Weitzel

The open source movement has revolutionized Information Systems development (4 out of 5 developers use projects like Git or Eclipse) and diffusion. FLOSS (Free/Libre Open Source Software) hence became a major topic for research and practice in the Information Systems community. A matured literature now offers a variety of useful insights from success factors like the importance of sustained contributions to ways of how to bridge the cultural and spatial distances in such virtual and distributed developer teams. A special issue of the Journal of the Association of Information Systems (JAIS), the flagship journal of the AIS, (Volume 11, Issue 11/12, winter 2010) summarizes that we know a lot about properties of successful FLOSS projects yet struggle to know how to achieve them. We hence face an interesting IT management challenge that, by the nature of FLOSS, requires interdisciplinary thinking: „FLOSS is a complex phenomenon that requires an interdisciplinary understanding of technical, economic, legal and socio-cultural dynamics" (Crowston/Wade, JAIS (11) ii). This is, exactly, what Dr. Schilling offers in this work.

The clever idea underlying the research of Andreas Schilling is to combine theoretical perspectives from organization sciences and HR research with the FLOSS literature to systematically identify how to find and retain FLOSS developers so that the project becomes and stays successful. He asks (and later answers): How can FLOSS projects effectively attract, integrate, and retain developers? Together with inventively gathered and sophistically analyzed data, we are offered new insights into successful FLOSS governance. Interesting results include that, in-deed, the extant literature always talks about "teams" and uses control variables like team size, yet is silent about how to attract and retain good team members or how to consider relational – not only unary – member properties. To address this issue, Dr. Schilling develops a Person-Job (P-J) and Person-Team (P-T) fit perspective that considers both, individual and relational characteristics in FLOSS projects. Empirical analyses shows that his „objective" P-T/P-J-based FLOSS success model even beats the "subjective" predictions of experienced Google project leaders concerning performance and sustained contributions. Another noteworthy insight is that there is no correlation between past academic achievement and recent developer performance. To me, the core contribution is that and how face-to-face meetings can bridge spatial and cultural distances in distributed FLOSS teams and that more diverse teams outperform all others if and only if there is at least some offline contact amongst FLOSS team members.

Overall, the thesis of Dr. Schilling offers important theoretical and empirical contributions to a significant and well established research area by adding, among others, a modern team perspective. Together with the author's fine familiarity with the FLOSS culture and the innovative data, this research is both an instructive and engaging read that is likely to influence future work on FLOSS (e.g. the Social Practice View as theoretical foundation for many fundamental FLOSS topics) and virtual teams.

„…der Schweiss FLOSS mir von den Gliedern“

(Friedrich Nietzsche, Also sprach Zarathustra, 2. Teil)

Tim Weitzel

Acknowledgement

This dissertation would not have been possible without the help and support of my supervisor, PhD committee, colleagues, friends, and family. I am deeply thankful for the help these people provided me, more than any words can express.

First, I would like to thank Prof. Dr. Tim Weitzel, who was the supervisor for my PhD thesis. It was him who introduced me to the field of open source research and the scientific research community. Moreover, his guidance and support made it possible for me to pursue my thesis by combining skills from the business and programming domain. I am also very thankful to Prof. Dr. Kai Fischbach and Prof. Dr. Guido Wirtz for joining my PhD committee and their interest in my research. Their valuable comments and advice helped me considerably improving my work.

Special thanks go to Dr. Sven Laumer for his scientific and personal guidance throughout the years of my doctorate. In the many discussions we had online as well as in person, he often provided structure to my thinking and identified interesting new research avenues. Moreover, I am grateful for his help in publishing our research work and that I could always count on him as a colleague and as a friend.

Sincere thanks goes also to Dr. Ralph Guderlei for his technical and personal guidance. I deeply appreciate the numerous Tuesday evenings and the endless hours we spend discussing technical and mathematical issues. I value his unconditional help whenever I did not know what to do and I am very thankful for his friendship.

I also like to thank my colleagues Dr. Daniel Beimborn, Wladimir Chrennikow, Thomas Friedrich, Steffen Illig, Christian Jentsch, Janina Kettenbohrer, Dr. Chritian Maier, Dr. Bernhard Moos, Oliver Posegga, Dr. André Schäfferling, Frank Schlosser, Alexander von Stetten, Thomas Tan, Christoph Weinert, Jakob Wirth, Thomas Wirtky, and Matthäus Zylka for their support and the helpful discussions with them.

Moreover, I thank Trimberg Research Academy (TRAc) and Dr. Marion Hacke for helping me making my first steps in my doctorate and assisting me on applying for a PhD scholarship. I am also very grateful to Universität Bayern e.V. for granting me the “Bayerische Eliteförderung” scholarship for the first three years of my doctorate.

I deeply thank my family for their outstanding support throughout the years of my PhD. Without their continuous support and emotional backing, I would have never been able to come this far with my doctorate in in my life. I am sincerely grateful for the love and support of my parents Sieglinde und Jürgen which made it possible for me to pursue an academic career and live my life the way I did. Moreover, I am deeply thankful for the assistance of my brother Dr. Alexander Schilling who I could count on throughout days and nights. I am also grateful for the support and backing of my grandparents Katharina und Max throughout the course of my study and the early steps in my doctorate.

Finally, I am deeply grateful for the emotional assistance and support of my girlfriend Helena Stefan. Throughout my research she always supported my work although it often meant to forgo things which should be normal in a relationship. Each moment with her is a gift and I am very glad that she came into my life.

Andreas Schilling

Zusammenfassung (German Summary)

Die Entwicklung von Free Libre Open Source Software (FLOSS) ist von hoher wirtschaftlicher und gesellschaftlicher Bedeutung. Zum Beispiel setzten mehr als 23 Prozent der 10 Millionen populärsten Websites weltweit die FLOSS ‚Wordpress‘ zur Organisation ihrer Inhalte ein (W3Techs 2015). Ein anderes Beispiel bietet die FLOSS ‚OpenStack‘, die von vielen Unternehmen zur Umsetzung innovativer Informationstechnologie (IT), wie unter anderem des Cloud Computing verwendet wird (IDG Connect 2013). Die hohe Bedeutung von FLOSS für Unternehmen beeinflusst auch die Softwareentwicklung. Eine Studie der Beratungsagentur Forester Research zeigt, dass vier von fünf Softwareentwicklern FLOSS Projekte wie ‚Git‘ oder ‚Eclipse‘ einsetzten (Forrester Research 2014). Neben der verbreiteten Nutzung von FLOSS im Unternehmenskontext nutzen auch viele private Konsumenten täglich FLOSS, wenn auch oft unbewusst. Ein Beispiel hierfür ist das weltweit am verbreitetsten Betriebssystem für Mobilfunktelefone ‚Android‘, das zu großen Teilen auf FLOSS basiert (IDC 2014; Google 2015). FLOSS Projekte spielen auch eine zentrale Rolle bei der Umsetzung der Idee des ‚Internet der Dinge‘. Diese Idee beschreibt die Vision, dass normale Gegenstände wie Kühlschränke, Raumthermostate oder Fernseher nicht nur mit dem Benutzer sondern auch miteinander intelligent interagieren (Miorandi et al. 2012). Um diese Vision zu verwirklichen, sowie um einseitige Abhängigkeiten zu vermeiden, schließen sich große Konzerne wie Bosch und Microsoft mit kleinen Unternehmen zusammen, um die Kommunikationsgrundlage für das Zusammenspiel der aktuellen und zukünftigen Geräte in Form von FLOSS zu entwickeln (Asay 2014).

Für den Erfolg von FLOSS Projekten ist es entscheidend, dass die beteiligten Individuen sich einbringen und zusammenwirken. Trotz der hohen wirtschaftlichen und sozialen Bedeutung von FLOSS legen verschiedene Studien nahe, dass viele FLOSS Projekte gar nicht oder nur unzureichend weiterentwickelt werden (Madey und Christley 2008; Chengalur-Smith et al. 2010; Fang und Neufeld 2009). Eine derart unzureichende Entwicklungsaktivität in FLOSS Projekten kann jedoch folgenschwere Konsequenzen haben, die vom Ausbleiben neuer Funktionalität bis hin zur Preisgabe persönlicher Informationen reichen können (Durumeric et a. 2014).

Dennoch ist erstaunlich wenig aufgearbeitet, wie FLOSS Projekte das Engagement und das erfolgreiche Zusammenwirken der beteiligten Entwickler beeinflussen können. Ein Grund hierfür ist, dass FLOSS Projekte oft auf dem freiwilligen Engagement der Entwickler sowie dem Gedanken der offenen Mitwirkung basieren. In Folge dessen lassen sich Lehren aus dem Organisationskontext nicht eins zu eins übertragen. Darüber hinaus ist zwar viel darüber bekannt, welche Eigenschaften erfolgreiche FLOSS Projekten haben; es ist aber unklar, wie diese Eigenschaften erreicht werden können (Crowston et al. 2012, Hahn und Zhang 2005, Hahn et al. 2008). Angesichts dieses Forschungsmangels sowie der hohen wirtschaftlichen und

sozialen Relevanz von FLOSS erarbeitet die vorliegende Dissertation geeignete Strategien und Methoden für das Entwicklermanagement in FLOSS Projekten. Angelehnt an die Kernbereiche für das Management von internationalem Personal in Unternehmen behandelt die vorliegende Dissertation die übergeordnete Forschungsfrage:

Wie können Entwickler für FLOSS Projekte effektiv (i) angeworben, (ii) integriert und (iii) gebunden werden?

Zur Behandlung dieser drei Aspekte, sowie für die Ableitung konkreter Methoden und Strategien für das effektive Management von FLOSS Entwicklern ist diese Dissertation in vier Kapitel strukturiert. Das erste Kapitel bildet das Fundament für die Ausarbeitung der folgenden Kapitel, indem es den aktuellen Stand der Forschung, sowie die Herausforderungen für die Anwerbung, Integration und Bindung von FLOSS Entwicklern zusammenfasst. Darauf aufbauend, behandelt Kapitel 2 die Identifikation geeigneter Entwickler in FLOSS Projekten. Hierzu wird, in Anlehnung an den Unternehmenskontext, die Verwendung des Person-Job (P-J) und Person-Team (P-T) Fit (Edwards 1991, Werbel und Johnson 2001) thematisiert und evaluiert. Zur Erarbeitung von Handlungsempfehlungen für das produktive Zusammenspiel von FLOSS Entwicklern wird in Kapitel 3 der Einfluss der geografischen Distanz der Entwickler zueinander und die Präsenz namhafter FLOSS Entwickler im Projekt untersucht. Für diese Untersuchungen wird unter anderem auf der Selbstbestimmungstheorie (Deci und Ryan 1985) und der Theorie der sozialen Praxis (MacIntyre 1981) aufgebaut. Abschließend evaluiert Kapitel 4 die Nutzung von Mentoring (Kram 1985), um Entwickler langfristig an FLOSS Projekte zu binden.

Die im Rahmen dieser Dissertation erarbeiteten empirischen Ergebnisse tragen vielfältig dazu bei die Anwerbung, Integration und Bindung von Entwicklern in FLOSS Projekten zu verbessern. So erweisen sich die abgeleiteten objektiven Kriterien zur Evaluierung des P-J und P-T Fit als zuverlässige Indikatoren, um den Verbleib von Entwicklern in FLOSS Projekten zu prognostizieren. Im direkten Vergleich zu den subjektiven Einschätzungen der Entwickler, erweisen sich die abgeleiteten objektiven Indikatoren sogar als deutlich zuverlässiger um den Verbleib neuer Entwickler zu prognostizieren. Die Untersuchungsergebnisse in Kapitel 3 heben die Relevanz der geographischen Distanz der FLOSS Entwicklern zueinander hervor. Konkret zeigt die Auswertung von 648 Teamkonfigurationen, dass die direkten offline Beziehungen der Entwickler zueinander darüber entscheiden, ob ihre produktive Zusammenarbeit durch räumliche und kulturelle Distanz gefördert oder behindert wird. Darüber hinaus legen die Untersuchungsergebnisse in Kapitel 3 den Schluss nahe, dass die Anwesenheit namhafter Entwickler nur begrenzt zu Steigerung der Teamproduktivität in FLOSS Projekten beiträgt. Wie eine Folgestudie in Kapitel 3 zeigt, könnte ein Grund hierfür sein, dass die Anwesenheit namhafter Entwickler nur das Vertrauen der Entwickler in ihre gegenseitigen Kompetenzen stärkt. Demgegenüber wirkt sich aber nur das Zusammengehörigkeitsempfinden der Entwickler zueinander direkt auf ihre Produktivität aus. Abschließend zeigen die Untersuchungsergebnisse im vierten Kapitel dieser Dissertation, dass Mentoring ein geeignetes Instrument ist, um die Projektbindung der FLOSS-Entwickler zu erhöhen.

Die erarbeiteten Untersuchungsergebnisse tragen auf verschiedenste Weise zur FLOSS Praxis und FLOSS Forschung bei. Die abgeleiteten objektiven Kriterien zur Messung des objektiven P-J und P-T Fit neuer Entwickler in FLOSS Projekten können beispielsweise direkt dazu eingesetzt werden Teilnehmer für den Google Summer of Code (GSoC) auszuwählen. Darüber hinaus stellen die empirischen Untersuchungsergebnisse zur Verwendung von P-J und P-T Fit in FLOSS Projekten eine Grundlage für weitergehende Forschung dar, um beispielsweise weitere objektive Indikatoren für den Projektverbleib von FLOSS Entwickler abzuleiten. Basierend auf den empirischen Untersuchungen in Kapitel 3 lassen sich verschiedene Schlüsse zur wirksamen Integration von Entwicklern in FLOSS Projekten ableiten. Aufbauend auf der Arbeit von Zhang und Venkatesh (2013) zeigen die Untersuchungsergebnisse zum Einfluss der geografischen Distanz, dass sich das Projektverhalten von FLOSS Entwicklern nur vollständig durch die gemeinsame Betrachtung ihres online und offline Kontexts erklären lässt. Des Weiteren tragen die Forschungsergebnisse in Kapitel 3 dazu bei, die Theorie der sozialen Praxis nach MacIntyre (1981) im FLOSS Kontext anzuwenden und zeigen, dass bestimmte Faktoren unterschiedliche Effekte auf das kollektive und das individuelle Verhalten von FLOSS Entwicklern haben können. Für die Organisatoren von FLOSS Projekten lassen sich aufbauend auf diesen Ergebnissen verschiedene Handlungsempfehlungen ableiten. Die Ergebnisse zum Einfluss der geographischen Distanz können konkret verwendet werden, um zu entscheiden ob und wann FLOSS Projekte offline Treffen mit ihren Entwicklern durchführen sollten. Eine weiteitere wichtige Erkenntnis für FLOSS Projekte ist, dass die Teamproduktivität nicht wesentlich durch die Anwesenheit namhafter Entwickler, wohl aber durch Maßnahmen zur Stärkung des Zusammengehörigkeitsgefühls erhöht wird. Die Studienergebnisse in Kapitel 5 legen schließlich nahe, dass Mentoring eine geeignete Maßnahme für FLOSS Projekte ist, um Entwickler langfristig zu binden. Eine Grundlage für zukünftige Forschung in dieser Richtung ist, dass Mentoring sowohl direkt als auch indirekt die Projektbindung von FLOSS Entwicklern erhöht.

Für die Hauptmotivation der Dissertation, konkrete Handlungsstrategien für die Anwerbung, Integration und Bindung von FLOSS Entwicklern abzuleiten, können abschließend folgende vier Kernempfehlungen festgehalten werden: (i) Langzeitentwickler lassen sich frühzeitig durch die vorgestellten objektiven Kriterien zur Messung von P-J und P-T Fit identifizieren, (ii) die direkten offline Beziehungen der Entwickler zueinander entscheiden darüber, ob ihr produktives Zusammenwirken durch räumliche und kulturelle Differenzen behindert oder gefördert wird, (iii) nicht die Präsenz von namhaften Entwicklern, sondern Maßnahmen zur Steigerung des Zusammengehörigkeitsgefühls der FLOSS Entwickler sollten forciert werden, um ihr produktives Zusammenspiel zu verbessern (iv) Mentoring ist für FLOSS Projekte eine geeignete Maßnahme, um die Projektbindung neuer Entwickler zu erhöhen.

Die Dissertation steht in der Tradition wissenschaftlicher Arbeiten der Wirtschaftsinformatik, indem sie geeignete Strategien für das Entwicklermanagement in FLOSS Projekten durch Kombination bestehender Erfahrungen aus dem Organisationskontext und der innovativen Anwendungsdomäne ableitet und evaluiert.

Referenzen

Asay M (2014): Developers Aren't Going to Go for Proprietary Standards. Online verfügbar unter http://readwrite.com/2014/10/17/internet-of-things-open-source-iot-developers, zuletzt geprüft am 20.04.2015.

Chengalur-Smith I, Sidorova A, Daniel S (2010): Sustainability of Free/Libre Open Source Projects: A Longitudinal Study. In: *Journal of the Association for Information Systems* 11 (11), S. 657–683.

Crowston K, Wei K, Howison J, Wiggins A (2012): Free/Libre Open-Source Software Development: What We Know and What We Do Not Know. In: *ACM Computing Surveys* 44 (2), S. 1–35.

Deci, EL, Ryan RM (1985): Intrinsic Motivation and Self-Determination in Human Behavior. *Perspectives in Social Psychology*, New York: Plenum Publishing.

Durumeric Z, Payer M, Paxson V, Kasten J, Adrian D, Halderman JA, Bailey M, Li F, Weaver N, Amann J, Beekman J (2014) The Matter of Heartbleed. *ACM Internet Measurement Conference*, S. 475–488.

Edwards JR (1991) Person-Job Fit: A Conceptual Integration, Literature Review, and Methodological Critique. Cooper CL, Robertson IT, eds. *International Review of Industrial and Organizational Psychology,* S. 283–357, New York: John Wiley and Sons Ltd.

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

Forrester Research (2014): Survey Indicates Four Out of Five Developers now Use Open Source. Online verfügbar unter http://www.zdnet.com/article/survey-indicates-four-out-of-five-developers-now-use-open-source/ zuletzt geprüft am 18.04.2015.

Google (2015): Welcome to the Android Open Source Project! Online verfügbar unter https://source.android.com/, zuletzt geprüft am 20.04.2015.

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

Hahn J, Zhang C (2005) An Exploratory Study of Open Source Projects from a Project Management Perspective *Management Information Systems Research Workshop 2005*, S. 1–27.

IDC (2014): Smartphone OS Market Share, Q4 2014. Online verfügbar unter http://www.idc.com/prodserv/smartphone-os-market-share.jsp, zuletzt geprüft am 20.04.2015.

IDG Connect (2013): Openstack: Platform of Choice for Cloud. Online verfügbar unter http://www.redhat.com/files/resources/en-opst-idg-openstack-platform-choice-cloud-infographic.pdf, zuletzt geprüft am 19.03.2015.

Kram KE. (1985): *Mentoring at work. Developmental Relationships in Organizational Life.* Lanham (MD): University Press of America.

MacIntyre AC (1981): *After Virtue. A Study in Moral Theory*. 1st ed., Notre Dame: University of Notre Dame Press.

Madey G, Christley S (2008): F/OSS Research Repositories & Research Infrastructures. In: *NSF Workshop on Free/Open Source Software Repositories and Research Infrastructures,* University of California, Irvine.

Miorandi D, Sicari S, Pellegrini F, Chlamtac I (2012) Internet of Things: Vision, Applications and Research Challenges, *Ad Hoc Networks* 10 (7), S. 1497-1516.

Werbel JD, Johnson DJ (2001) The Use of Person-Group Fit for Employment Selection: A Missing Link in Person-Environment Fit. *Human Resource Management* 40 (3), S. 227–240.

W3Techs (2015): Usage Statistics and Market Share of Content Management Systems for Websites. Online verfügbar unter http://w3techs.com/technologies/overview/content_management/all/, zuletzt geprüft am 19.04.2015.

Zhang X, Venkatesh V (2013): Explaining Employee Job Performance: The Role of Online and Offline Workplace Communication Networks. In: *Management Information Systems* Quarterly 37 (3), S. 695–722

Table of Content

Introductory Paper

Introductory Paper

Developer Management in FLOSS Projects - Theoretical Concepts and Empirical Evidence

Andreas Schilling
University of Bamberg
andreas.schilling@uni-bamberg.de

Table of Contents

List of Tables

List of Figures

List of Abbreviations

API	Application Programming Interface
CBSEM	Covariance Based Structural Equation Modeling
CMC	Computer Mediated Communication
FLOSS	Free Libre Open Source Software
FS	Free Software
GSoC	Google Summer of Code
HLM	Hierarchical Linear Model
IHRM	International Human Resources Management
II	Individualism/Collectivism Index
KDE	K Desktop Environment
MFI	Masculinity / Femininity Index
OLS	Ordinary Least Squares
OSS	Open Source Software
PDI	Power Distance Index
P-E	Person-Environment
P-J	Person-Job
PLOC	Perceived Locus of Causality
PLS	Partial Least Squares
P-T	Person-Team
SDT	Self-Determination-Theory
SEM	Structural Equation Modeling
SPV	Social Practice View
UAI	Uncertainty Avoidance Index
VCS	Version Control System

1 Introduction

Initiatives developing Free Libre Open Source Software (FLOSS)[1] create software whose code can be freely studied, modified, and shared with others (Ghosh 2002). FLOSS projects are an integral part for of the day-to-day functions of the economy (Gartner Inc. 2012). In fact the content of more than 23 percent of the 10 million most popular websites worldwide is managed using the FLOSS 'Wordpress' (W3Techs 2015). Moreover, some of the most well-known billion-dollar companies in the world, such as 'Instagram', are based entirely on FLOSS (Moody 2012). While organizations first started using FLOSS to reduce their IT spending, their motivation has since changed fundamentally and the dominant factor for the use of FLOSS has become its superior quality (Accenture Inc. 2010). This is especially the case for innovative IT functionality. A recent IDG consultancy study revealed that up to 84 percent of the surveyed organizations plan to rely on the FLOSS 'OpenStack' for cloud computing functionality (IDG Connect 2013). Organizations' high reliance on FLOSS has also influenced the way software developers work. Today, four out of five software developers rely on FLOSS such as 'Eclipse' and 'Git' (Forrester Research 2014). These results are supported by an IBM study which reveals that nearly 90 percent of surveyed IT-professionals consider FLOSS a key technology for future application development (IBM 2011). In addition to its importance for the organizational domain, FLOSS plays an important but often less known role in the lives of private consumers. In particular, the worldwide most used operating system for mobile phones, 'Android', is based in large parts on FLOSS (IDC 2014; Google 2015). Moreover, FLOSS projects provide key components for the implementation of the 'Internet of Things', which describes the vision that regular devices such as refrigerators, thermostats, and TVs interact intelligently not only with the user but also with each other (Miorandi et al. 2012). To realize this vision and avoid unilateral dependences, big corporations such as 'Bosch' and 'Microsoft' form alliances with small corporations to develop the communication bases for the intended interplay of the various devices in form of FLOSS projects (Asay 2014).

Aside from its high relevance for the economy and society, FLOSS development is the topic of significant academic research. According to Crowston et al. (2012), '*… FLOSS has moved from an academic curiosity to a mainstream focus for research.*' (Crowston et al. 2012, p. 2) during the last decade. In the assessment by Crowston et al. (2012), FLOSS projects are special because they allow researchers to study unprecedented processes for collective innovation and coordination. In a similar vein, Krogh and Spaeth (2007) note that '*research on the open source software phenomenon is an interesting example of research that combines scientific rigor with relevance*' (Krogh and Spaeth 2007, p. 241) and Aksulu and Wade (2010) conclude that FLOSS research proliferated during the last ten years across various domains and disciplines.

As in the case of organizations, the success of FLOSS projects depends heavily on the activity of the individuals involved. However, in contrast to the organizational domain, it is not fully understood how FLOSS projects should effectively manage their developer base. Based on a

[1] A precise description of the term FLOSS and its relationship to Open Source Software (OSS) is provided in Section 2.1

review of FLOSS literature, Crowston et al. (2012) conclude that it has yet to be understood how social processes and project characteristics enable an effective collaboration among FLOSS developers. In a similar vein, Hahn and Zhang (2005) conclude that very few studies examined FLOSS projects from a project management perspective. For example, FLOSS research examined the reason for individuals' engagement but not their selection of a particular FLOSS project (Hahn et al. 2008).

Understanding how to manage FLOSS developers effectively is also of high practical relevance as the overwhelming majority of FLOSS initiatives face severe challenges in terms of the activity of their developer base which can threaten their entire existence (Madey and Christley 2008). One such challenge is the slow influx of new developers. A lack of new developers severely reduces FLOSS projects' innovation and make them vulnerable to development slowdown if members leave the project (Robles et al. 2009). Another central challenge for FLOSS projects is developer turnover. Several studies suggest that many FLOSS developers are engaged in the development of not only one but rather several FLOSS projects (Lakhani and Wolf 2005, Hu and Zhao 2009, David and Shapiro 2008), which effectively puts FLOSS projects into competition for developers' interest. The consequences of developer turnover can be grave because when developers leave a FLOSS project, the project often loses its ability to maintain functionality they contributed (Robles et al. 2009). The third pivotal challenge for FLOSS projects concerns developers' effective integration. Commonly high learning barriers in FLOSS projects hinder developers from becoming effective (Adams et al. 2009, von Krogh et al. 2003). Moreover, Singh et al. (2011b) provide evidence that most developers do not advance in their learning, which essentially means that they do not increase their effectiveness.

A lack of development activity in FLOSS projects can not only lead to the absence of new features but also to much worse consequences. A recent incident which exemplifies such grave consequences for the economy as well as for society as a whole is the program bug called 'Heartbleed' in the FLOSS 'OpenSSL'. OpenSSL is a cryptographic library which aims to provide secure end-to-end communication via the Internet. A recent study estimates that OpenSSL is used by more than two thirds of all webservers worldwide to ensure secure Internet traffic (Goodin 2014). Considering this broad adoption of OpenSSL, the consequences of this programming deficit, which was found by security experts at Google, were unprecedented. Durumeric et al. (2014) estimate that up to 55 percent of the top one million websites worldwide could have been vulnerable to attacks using the Heartbleed bug. Heartbleed was disastrous not only due to its enormous reach, but also because of its devastating consequences. In essence, it allowed attackers to extract sensitive information such as login credentials from webservers without leaving a trace. One of the most severe documented attacks exploiting the Heartbleed bug was directed against the largest US hospital network and involved the theft of more than 4.5 million patient records (Finkle and Kurane 2014). Not only webservers, but also industrial control systems such as those used in power plants and wastewater management are vulnerable and potentially exploitable to the Heartbleed bug (McMillan 2014). While security patches for Heartbleed have been made available for affected webservers and industrial control systems, the reputed security expert Bruce Schneier suggests that low margin, non-upgradable, embedded systems used in smart heat meters, thermostats, and in other technologies provide a major security concern for the years to come (Schneier 2014, Berinato 2014).

Heartbleed is unique not only due to its devastating economical and societal consequences, but also because it illustrates the failure of the OpenSSL project to attract, integrate, and retain developers properly. Despite the popularity of OpenSSL, the project only attracted a handful of developers. Specifically only seven contributors were listed on the project's website until April 2014 (Stokel-Walker 2014). Even worse, most of these developers did not remain active but soon left the project after adding new functionality (OpenHub 2015). This developer fluctuation led to massive unhealthy growth of the OpenSSL codebase, with code which was neither properly finished nor maintained and sometimes even termed 'highly experimental' (Stokel-Walker 2014). Ultimately, one of the core developers admitted that there were some deficits regarding the coordination of active developers which could have affected the code quality of the project (Stokel-Walker 2014). One of the deficits identified is that the two core developers have never met in person. This dissertation examines the following overall research question in light of the critical challenges FLOSS projects face regarding managing their developer base and the identified need for further research:

How can FLOSS projects effectively
attract, integrate, and retain
developers?

The remainder of this section outlines how this dissertation is structured in order to examine this overall research question. In particular, the next subsection builds upon concepts from International Human Resource Management and derives the specific research questions which are examined in this dissertation. Thereafter, Subsection 1.2 details the research approach followed in examining the articulated research questions and Subsection 1.3 outlines the structure of the dissertation.

1.1 Research Questions

In order to provide FLOSS projects with guidance on how to manage their developer base, this dissertation builds on theories and concepts from International Human Resources Management (IHRM). FLOSS projects and organizations share the vital need for appropriate talent management but differ substantially in terms of remuneration and regulatory. Due to the critical role of IHRM for organizations' success it has been the subject of various studies in organizational literature over the last years, leading to a rich pool of strategies and concepts for talent management. Horwitz (2003) and Tarique and Schuler (2010) divided IHRM into three basic management aspects. In this conceptualization, the first management aspect is concerned with attracting employees. Specifically, this aspect focuses on identifying and recruiting new talent (Tarique and Schuler 2010). The second management aspect concerns integrating employees through effective staffing and development strategies (Tarique and Schuler 2010). Finally, the third management aspect of IHRM refers to means of enhancing employees' retention behavior (Horwitz 2003).

Building on this threefold structure of IHRM, the dissertation proposes considering three basic management aspects to understand and categorize the various challenges involved in managing developers in FLOSS projects. In the following each of the three proposed aspects for developer management in FLOSS projects is outlined and the particular research questions of this dissertation are derived.

Attraction: The influx of new developers is vital to FLOSS projects. Besides adding more manpower, new developers enhance FLOSS projects' innovation (von Krogh et al. 2003). With respect to the open participation in FLOSS projects, previous research focused so far on project- (Santos et al. 2012) and relational-aspects (Hahn et al. 2008, Hu et al. 2012) to bring in new developers. An aspect, however, which has been neglected so far is how talented FLOSS developers can be identified (Pratyush et al. 2010). But this aspect is equally important as the sheer quantity of new developers. Such understanding is not only the basis for evaluating if new developers are suited and will remain in the project; it also helps FLOSS projects to identify newcomers who are worth being trained. Finally, an understanding for the characteristics new developers should possess assists FLOSS projects' attraction efforts by identifying those individuals worth attracting. Thus, to better understand how talented FLOSS developers can be identified in FLOSS projects the dissertation examines the research question:

RQ I: How can FLOSS projects identify suitable developers?

Integration: The second proposed aspect for developer management in FLOSS projects refers to integrating members effectively into the project. Developers are often involved in various FLOSS projects so there is effectively competition for developer attention among these projects (Lakhani and Wolf 2005, Hu and Zhao 2009, David and Shapiro 2008). Moreover, most developers contribute only small amounts of code to FLOSS projects (Setia et al. 2012, Singh et al. 2011b). Thus, FLOSS projects need to figure out ways to foster individual productivity by motivating developers. At the same time, it is equally important to consider collective aspects to ensure that FLOSS developers work well with the developer team. One aspect whose role is highlighted recently for FLOSS developers' productive interplay is their geographic dispersion (Colazo and Fang 2010, Daniel et al. 2013, Hu et al. 2012). However, these studies not only examined different aspects of geographic dispersion but also came in part to different conclusions regarding its role on productive teamwork. Another shortcoming of these studies is that they did not control for FLOSS developers' offline relationships. In addition, a recent study by Hu et al. (2012) calls for further research on the role of reputed developers in FLOSS projects. Nevertheless, little is known to date on the individual and collective stimuli which result from the involvement of reputed developers. In response to previous research calling for further research on the role of geographic dispersion and the involvement of reputed developers to achieve productive teamwork, this dissertation examines the two research questions:

RQ II: How do offline distances affect FLOSS developers' productive teamwork?

RQ III: How do reputable developers affect FLOSS developers' productive teamwork?

Retention: The third basic management aspect of the proposed framework is concerned with FLOSS developers' project tenure. Developer turnover hurts FLOSS projects in two sensitive ways. On the one hand, FLOSS projects often lose the ability to maintain and understand contributed code when the contributing developers leave (Robles et al. 2009). On the other hand, long-term developers often add to FLOSS projects' quality (Jorgensen 2001) and task completion (Chengalur-Smith et al. 2010). Previous FLOSS research in this regard provide evidence that developers' knowledge building and socialization (Fang and Neufeld 2009, Singh et al. 2011b, Qureshi and Fang 2010) are key facilitators for long project tenure. Although it is not yet clear how FLOSS projects can actively intervene to promote retention. In order to propose an effective education and retention strategy for FLOSS projects, the last examined research question is:

RQ IV: Is mentoring an effective retention strategy for FLOSS developers?

Figure 1 illustrates the four specific research questions of the dissertation and their relationship to developer management in FLOSS projects. The next section outlines the research approach adopted to examine these research questions.

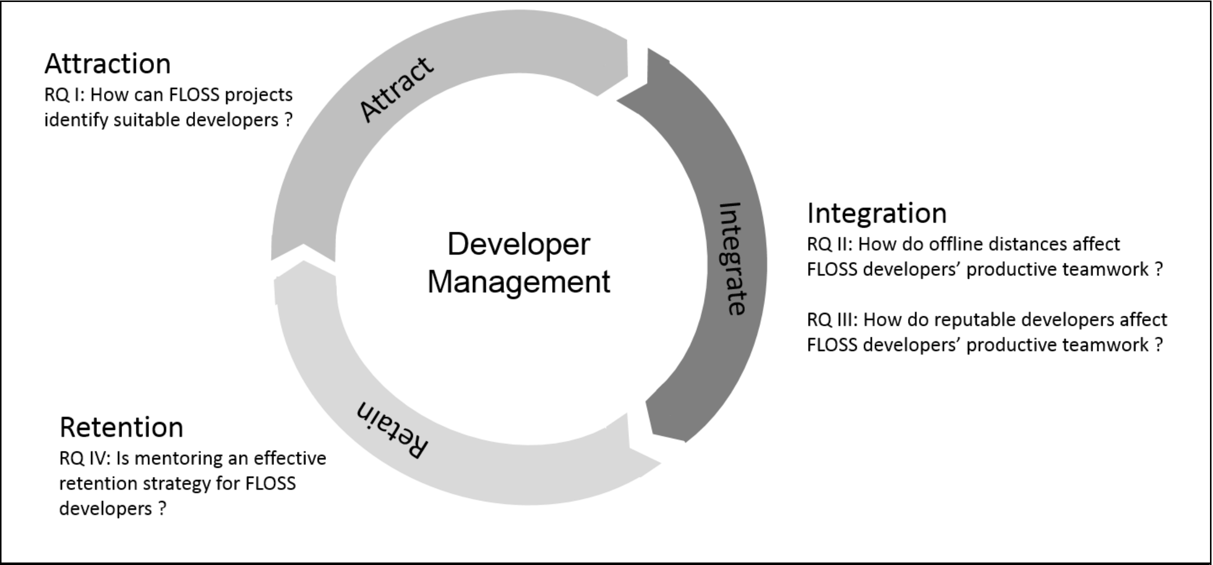

Attraction
RQ I: How can FLOSS projects identify suitable developers ?
Attract
Integrate
Retain
Developer Management
Integration
RQ II: How do offline distances affect FLOSS developers' productive teamwork ?
RQ III: How do reputable developers affect FLOSS developers' productive teamwork ?
Retention
RQ IV: Is mentoring an effective retention strategy for FLOSS developers ?

Figure 1. The four research questions of the dissertation

1.2 Research Approach

The adopted approach in the dissertation to examine the previously articulated research questions builds upon the interrelatedness between the FLOSS and the organizational domain. In particular, the pursued research approach can be characterized in four consecutive steps. Figure 2 visualizes these steps and their relation to the organizational and FLOSS domain.

The ***first step*** of the pursued research approach consists of identifying relevant theories and concepts from the organizational domain that can serve as a basis for understanding FLOSS developers' project behavior. As needed, these concepts are customized for the FLOSS domain to reflect differences in how remuneration and regulation influence teamwork in the organizational and the FLOSS domains.

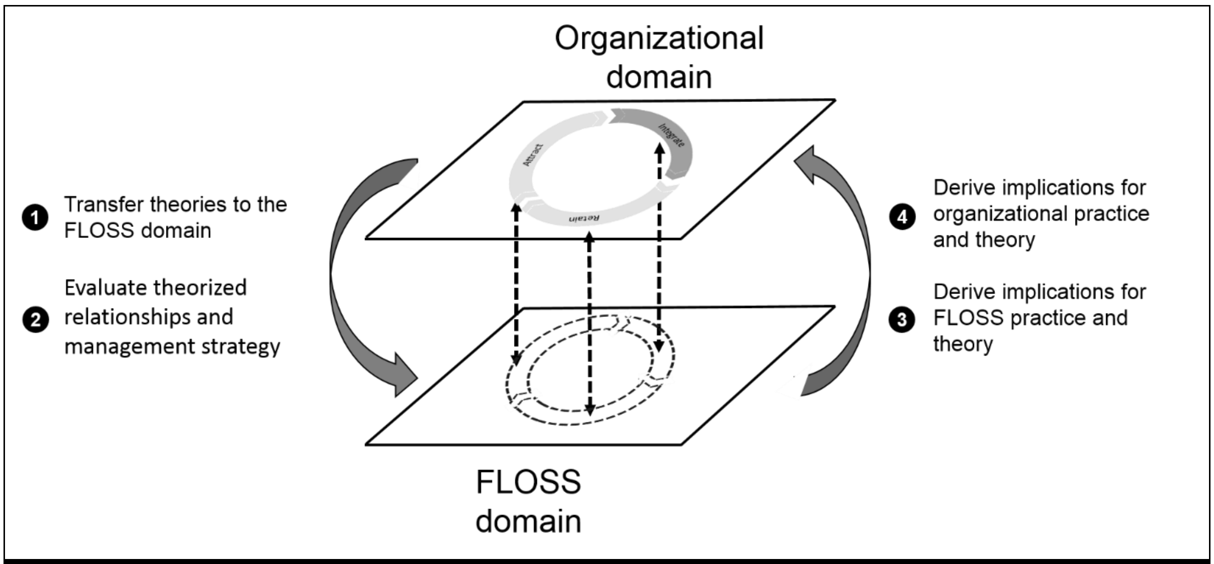

Organizational domain
Attract
Integrate
Retain
1 Transfer theories to the FLOSS domain
2 Evaluate theorized relationships and management strategy
4 Derive implications for organizational practice and theory
3 Derive implications for FLOSS practice and theory
FLOSS domain

Figure 2. Research approach of the cumulative dissertation

The ***second research step*** focuses on the empirical evaluation of relationships within the FLOSS domain. To take advantage of detailed public accessibly communication and contribution records in FLOSS projects, the dissertation analyzes principally archival records of developers' project behavior. In addition, perceptual data such as survey data was used, if appropriate, to complement the archival evaluations.

Based on the results of the performed evaluation, the ***third step*** of the adopted research approach is concerned with deriving concrete theoretical and managerial implications for the FLOSS domain. This is achieved by contrasting the findings with previous FLOSS research, delineating potential avenues for future research, and deriving concrete management advice for FLOSS projects.

Finally, the ***fourth step*** of the adopted research approach puts the derived evaluation results into context with organizational theories and strategies, contributing to organizational research in several ways. First, FLOSS teams represent an extreme case of decentralized collaboration which provides unprecedented possibilities to 'falsify' basic assumptions about software development as well as empirically evaluate and refine previously untested theories (Krogh and Spaeth 2007). Moreover, the public collaboration and communication of FLOSS developers allows teamwork behavior to be evaluated in great detail and across project boundaries, which is rarely possible in organizational settings (Singh et al. 2011a, Singh et al. 2011b). Thus, research on the functioning of FLOSS projects can provide important lessons for enhancing software development practices in organizations (Fitzgerald 2006). In addition, with the increasing consideration of knowledge workers as volunteers (Drucker 2002), FLOSS research contributes to the derivation of new management approaches for organizations.

Based on the specified research questions and the research approach followed in pursuing them, the next subsection outlines the structure of the dissertation.

1.3 Structure of the Thesis

Based on the proposed threefold framework for developer management in FLOSS projects, the cumulative dissertation is structured into four chapters. The first chapter constitutes the basis for the elaboration of the remaining chapters by providing a literature review of the status quo in FLOSS research on attracting, integrating, and retaining FLOSS developers. Based on the results of this literature review, Chapter II focuses on ways to improve the quality of developer attraction in FLOSS projects. Specifically this chapter proposes and evaluates concepts and measures used to identify developers worth attracting and training. Chapter III of the dissertation looks at means to enhance developers' integration into the FLOSS project. In particular, this chapter examines the effects of geographic distance and offline interactions among FLOSS developers on the effectiveness of their collaboration. In addition, this chapter elaborates on the effects of reputable FLOSS developers on team members' motivation and their productive interplay. Finally, Chapter IV of the dissertation proposes and evaluates the use of mentoring as a potential way to retain FLOSS developers. Figure 3 summarizes the structure of the four chapters including their relationship to the seven research articles of the dissertation.

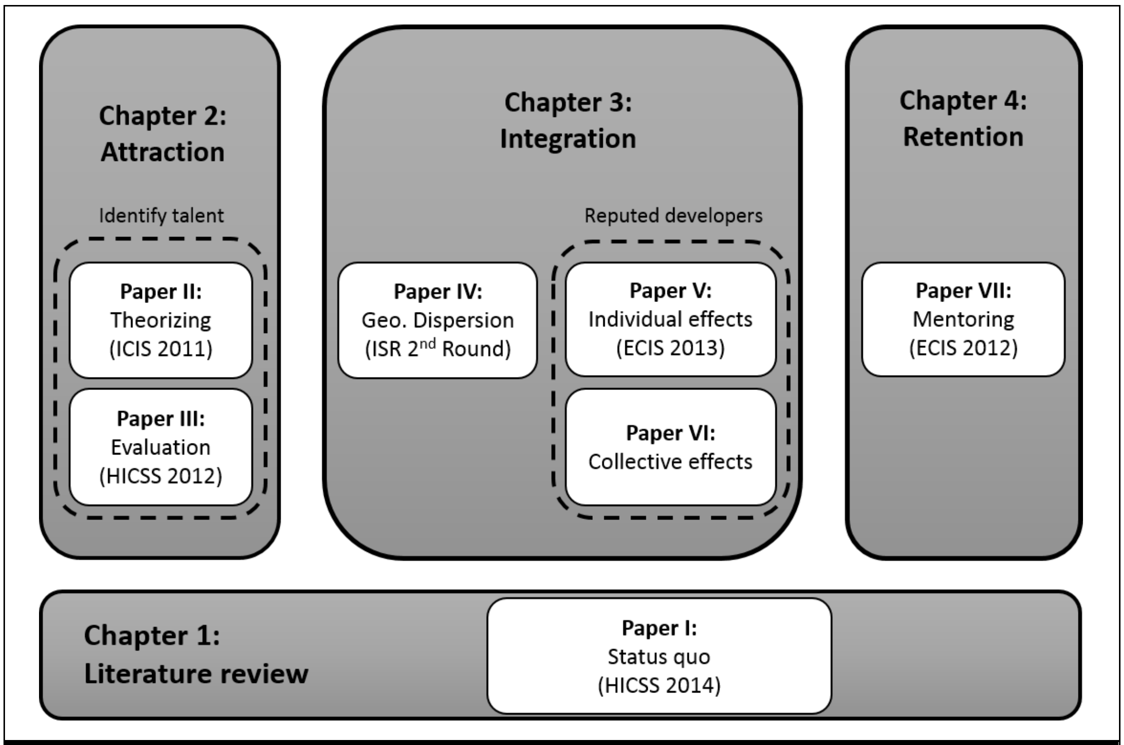

Chapter 2: Attraction
Identify talent
Paper II: Theorizing (ICIS 2011)
Paper III: Evaluation (HICSS 2012)
Chapter 3: Integration
Reputed developers
Paper IV: Geo. Dispersion (ISR 2nd Round)
Paper V: Individual effects (ECIS 2013)
Paper VI: Collective effects
Chapter 4: Retention
Paper VII: Mentoring (ECIS 2012)
Chapter 1: Literature review
Paper I: Status quo (HICSS 2014)

Figure 3. Dissertation structure

2 Theoretical Background

This section provides an overview of the various organizational theories and concepts which built the theoretic foundation for the derived strategies for attracting, integrating and retaining FLOSS developers. Prior to the presentation of the various theories and concepts, this section looks at the history and development of FLOSS. Then, the distinct challenges for attracting, integrating, and retaining FLOSS developers are outlined and the approaches taken in the dissertation to address them is described.

2.1 Free Libre Open Source Software Development

Free Libre Open Source Software (FLOSS) is an umbrella term which is used to refer to the creation of software which can be freely studied, modified, and exchanged (Ghosh 2002). With the use of this umbrella term, researchers can neglect from the often controversial differences in terms of ideology and licensing between the creation of Open Source Software (OSS) and Free Software (FS) and focus on the commonly identical development processes for the two (Crowston et al. 2012, Scacchi et al. 2006).

FLOSS development has a long history. In fact the infancy of computer programming during the 1950s is built upon the common practice of code sharing. Due to expensive and often proprietary computer hardware, software was developed almost exclusively by engineers in corporate and academic research facilities (von Krogh and von Hippel 2003). Their research background set the stage for those engineers to exchange their code routines with each other so that others could use and modify it for their specific hardware and usage configurations. However, this common sharing practice ended in the 1970s when commercial software development emerged. In contrast to the common practice of code sharing, software companies relied on software licensing and technical restrictions to prevent others from studying their code (Dixon 2004, Kavanagh 2004). As a move against these corporate practices, developers around the world created programming communities to build code which can be freely studied, modified, and distributed to others.

To this day, FLOSS is still developed through collaboration structures which are basically the same as at the beginning. The developers involved in FLOSS projects generally are not concentrated in one place but scattered around the world (Crowston et al. 2012). Moreover, FLOSS developers generally receive no direct monetary compensation from the project for their contributions. Indeed, most developers do not receive monetary compensation from any source, but rather are involved in FLOSS projects voluntarily (Fang and Neufeld 2009). In order to coordinate their working, FLOSS developers rely on computer mediated communication (CMC). The records of this CMC in the form of mailing list posts and Internet-Relay Chats are publically accessible (Hemetsberger and Reinhardt 2006). Finally, there is no formal obligation between developers and the project. Developers decide on their own which aspects they want to work on (Crowston et al. 2010). The following subsection elaborates on the distinct challenges of developer collaboration and how they complicate developer management in FLOSS projects.

2.2 Distinct Challenges for FLOSS Developer Management

Like international organizations, FLOSS projects rely on international human resources and effective talent management. However, FLOSS projects also face distinct challenges in attracting, integrating, and retaining developers. These distinct challenges make it impossible to apply existing knowledge from IHRM directly to FLOSS projects. The following paragraphs outline these distinct challenges and detail the organizational concepts used in this dissertation to address them.

A central challenge for attracting developers to FLOSS projects is the generally employed open participation practice, which means that there are no access restrictions imposed upon newcomers to contribute to the project. Although this practice is a substantial gain for knowledge sharing, it leads to an uncontrolled situation in which new developers come and go, which hampers FLOSS projects' ability to foster sustained code development. FLOSS research suggests that two factors specifically influencing FLOSS developers' sustained commitment early on are their compatibility with the project as well as with the developer team (Fang and Neufeld 2009). This compatibility of individual and relational characteristics is also central for identifying talent in organizations. Two concepts from organizational recruitment which have proven particularly effective in defining and assessing candidates' individual and relational compatibilities are Person-Job (P-J) and Person-Team (P-T) fit (see Subsection 2.3.1). Both of these concepts are based on the idea that it is neither the sole characteristics of the individual nor the sole characteristics of the organization but rather the compatibility between the two which determine individual well-being. With respect to the similarity between FLOSS projects and organizations in terms of talent identification, the dissertation relies on P-J and P-T fit to conceptualize and evaluate relevant developer characteristics. To account for those aspects in which FLOSS projects and organizations are distinct, such as monetary compensation and regulatory power, the concepts are customized for the FLOSS domain in a later step.

To integrate developers into FLOSS projects effectively, various important factors need to be considered. A key challenge in fostering individual productivity is the systemically innate dependence on FLOSS developers' self-motivation. Due to the lack of monetary rewards, FLOSS projects have to rely on other means to motivate developers to invest their time and effort. In order to understand the managerial levers which can be used to motivate FLOSS developers, the dissertation relies on the two social theories which have been used successfully to understand and examine working behavior in the organizational domain (Beadle 2006, Gagné and Deci 2005). The first of these two theories, Self-Determination Theory (SDT) by Deci and Ryan (1985) (see Subsection 2.3.2), is used to understand individuals' behavioral reaction to concrete project characteristics. SDT is based on the basic tenet that individuals strive to satisfy their innate needs for relatedness, competence, and autonomy. Based on the degree to which behavior satisfies these basic needs, Deci and Ryan (1985) distinguish five motivation forms which differ in the degree to which individuals consider themselves as self-determined. The second theory MacIntyre‘s (1981) social practice view, strives to understand the long term effects of particular characteristics of the project and environment (see Subsection 2.3.3). Compared to SDT, which focuses on FLOSS developers' current behavior, the social practice view also considers individuals' personal histories and what led them to become FLOSS developers in the first place. These two behavioral theories are the basis for evaluating the immediate motivational effects as well as the sustained teamwork gains in FLOSS projects which result from the involvement of reputable developers. Moreover, the dissertation examines how FLOSS developers' feelings of belongingness in the offline context affects their productive collaboration in FLOSS projects. In particular, it examines the degree to which offline interactions help FLOSS developers overcome the negative effects of their spatial, temporal and cultural distances.

Finally, there are various distinct challenges for retaining developers in FLOSS projects. Given the absence of monetary compensation, FLOSS projects are entirely contingent on developers' self-motivation to remain active in the project. Considering FLOSS projects often have high learning barriers which make it difficult for newcomers to contribute to the project, Singh et al. (2011b) conclude that it is common among FLOSS developers to stagnate in their learning state. A possible explanation for this is provided by Adams et al. (2009) who show that it can take up to 60 weeks for newcomers to become effective in FLOSS projects. As Singh et al. (2011b) and Fang and Neufeld (2009) point out, a key way to support FLOSS developers' knowledge building is to improve not only their coding but their project integration. A training method from the organizational domain which has proven especially effective in enhancing newcomers' competences and feelings of belongingness is mentoring (see Subsection 2.3.4). Mentoring describes a training method in which an experienced professional provides technical advice and interpersonal support to an inexperienced employee. With regard to the effectiveness of mentoring for knowledge building in the organizational domain (Hale 2000, Brashear et al. 2006) and the applicability of such training method in the FLOSS domain, the dissertation applies and evaluates its use for educating and retaining FLOSS developers.

2.3 Theory Suite

This subsection provides an overview of the various concepts and theories which are used to understand FLOSS developers' project behavior and derive concrete management advice.

2.3.1 Person-Job and Person-Team Fit

Given the open participation in FLOSS projects and the relevance of relational and individual compatibility, the dissertation proposes and evaluates the use of Person-Job fit and Person-Team Fit to identify developers who are likely to remain committed. Person-Job fit ensures that candidates are selected who have the necessary skills and abilities to accomplish the various tasks of a job. In contrast, Person-Team fit ensures that selected candidates are in alignment with the other team members.

Person-Job (P-J) and Person-Team (P-T) fit, belong to the overarching concept of Person-Environment (P-E) fit, which is based on the interactionist theory of behavior (Chatman 1989, Muchinsky and Monahan 1987) which in turn builds on the work of Lewin (1951). The basic premise of these theories is that human behavior cannot be explained fully by considering only either individual or situational characteristics, but only by combining the two (Oreg and Nov

2008). As a result, P-E fit refers to the level of congruence between the characteristics of the person and the particular context (Muchinsky and Monahan 1987).

The most common definition for Person-Job fit is by Edwards (1991) and considers it to be a twofold construct. The first component of this construct refers to the *needs-supply match*, which assesses the degree to which a person's goals, interests, and psychological needs are met through the various job characteristics (i.e. autonomy, responsibility, pay, etc.). The other component of P-J fit concerns the *demands-ability match*. This match assesses the degree to which the person possesses the abilities and skills which are required to perform the job. The particular demands of a job are commonly derived by analyzing the concrete tasks and the required level of knowledge and abilities required to complete them. Edwards (1991) supposes that a good P-J fit is not only beneficial for the organization, in terms of job performance and reduced turnover, but also for the individual, who experiences higher levels of job satisfaction and less stress. Empirical studies support this assumption. Kristof-Brown et al. (2005) reveal that P-J fit is strongly associated with individuals' level of job satisfaction and performance and even moderately high with increased job tenure. A study by Chilton et al. (2005) supports the relevance of P-J fit specifically for the context of software development indicating that software developers with higher levels of P-J fit achieve higher job performance and experience less strain.

In contrast, individuals' level of Person-Team[2] (P-T) fit refers to interpersonal characteristics. In particular, P-T fit considers individuals' supplementary fit and complementary fit to the other team members (Werbel and Johnson 2001). The *supplementary fit* refers to the degree to which the candidate shares personal characteristics (i.e. knowledge, skills, beliefs, etc.) with the other team members (Muchinsky and Monahan 1987). In contrast, candidates' *complementary fit* describes the degree to which they possess personal characteristics that are otherwise lacking in the team. According to Werbel and Johnson (2001) individuals should have both supplementary fit and complementary fit as only one of these types of fit could lead to dysfunctional teams. For example, a high degree of only supplementary fit could lead to high cohesion among team members but reduce the ability of a team to be innovative. At the same time, individuals with only complementary fit could contribute abilities otherwise lacking in the team. However, they do not possess characteristics that enable them to establish common grounds with the other team members. Individuals who have both forms of fit have been shown to have the potential to produce positive work outcomes. Kristof-Brown et al. (2005) show in their meta-analysis that P-T fit has a very strong influence on individuals' satisfaction with their coworkers and also positive effects on individual job performance and tenure. Moreover, a study by Seong et al. (2012) suggests a strong relationship between P-T fit and group performance.

There are distinct strategies for assessing the various types of fit. The two most common evaluation forms are perceived fit and actual fit. In the case of perceived fit, an individual's level of fit is assessed based on subjective impressions (Kristof-Brown et al. 2005). In contrast,

[2] Also known as Person-Group fit

actual fit is assessed through indirect measures, such as the comparison of personal and organizational characteristics.

2.3.2 Self-Determination Theory

In the absence of pecuniary rewards, it is central to understand what motivates developers to stay committed to FLOSS projects so that effective incentives can be designed to foster development activity. To understand FLOSS developers' motivation to contribute, this dissertation relies on Self-Determination-Theory (SDT).

Self-Determination-Theory (Deci and Ryan 1985), is a theoretic framework for understanding how social and contextual conditions affect individual work motivation. SDT distinguishes among distinct forms of motivation based on the degree to which individuals perceive their behavior as self-determined (Ryan and Deci 2000b). The basic assumption of SDT is that people have innate psychological needs for competence, relatedness, and autonomy which they seek to satisfy to achieve well-being.

Behavior which arises naturally through the satisfaction of these innate needs is *intrinsically motivated*. According to Deci and Ryan (2000), individuals carry out such behavior because it is in itself rewarding to them. In particular, people perceive fun and excitement when behaving in this way (Ryan and Deci 2000a). Typical examples for intrinsically motivated behavior are hobbies which individuals perform due to the fun and joy and not because of the outcomes which are associated with them.

In contrast, behavior is *extrinsically motivated* when it is not performed due to its inherent value to the individual, but due to external regulation. In SDT extrinsic motivation is not a uni-dimensional construct but comprises of various motivation forms which vary according to the degree to which individuals internalize them (Ryan and Deci 2000b). Internalization describes the process in which the individual adopts external values, attitudes, or regulations (Gagné and Deci 2005). A result to this internalization process is that the perceived locus of causality (PLOC) for the particular behavior gradually becomes internal. Specifically, SDT differentiates between the following four types of extrinsic motivation, which can be ordered along a continuum spanning from an internal to an external PLOC (Ryan and Deci 2000b).

External regulation: This motivation form classifies behavior with the lowest degree of autonomy. Individuals with such motivation behave in certain ways due to external contingencies like pecuniary rewards or punishment which are associated with it (Ryan and Deci 2000b). A typical example for this motivation form is when employees only perform a job because they get paid for it.

Introjected regulation classifies behavior which is perceived to have an external locus of causality. However, compared to externally regulated behavior, individuals with this form of motivation internalize some of the exposed regulation as their own. Typically, individuals with this form of motivation behave in a particular way in order to attain ego enhancements or avoid guilt (Ryan and Deci 2000b, Ryan and Deci 2000a). An example of this form of behavior is if an individual performs a particular job to gain self-esteem (Gagné and Deci 2005).

Identified regulation: Individuals with this motivation form identify with the value of an original externally induced behavior (Gagné and Deci 2005, Ryan and Deci 2000b). Thus, people with this form of motivation perceive an internal locus of causality for their doing. Gagné and Deci (2005) exemplify this particular form of motivation with a nurse who fully identified with her job so that she also accepts accomplishing tasks which are not interesting to her, such as bathing patients, but still necessary to achieve of the overall goal (i.e. help patients).

Integrated regulation refers to behavior which is extrinsically motivated but perceived to be completely self-determined. With this type of motivation, individuals fully integrate the originally externally posed regulations (Ryan and Deci 2000b). As with intrinsic motivation, people perceive that the particular behavior emanates from within themselves. However, contrary to intrinsic motivation, the behavior is still considered instrumentally. Gagné and Deci (2005) illustrate this motivation with the case of a nurse who considers her profession as a central aspect of her identity and thus cares for people even when she is not at work. Thus, she not only accepts but even appreciates uninteresting activities as part of providing care (Gagné and Deci 2005).

According to SDT, individuals carry out activities with greater effort and persistence if they consider them self-determined. Sheldon and Elliot (1999) support this by providing evidence that individuals with more self-determined motivation outperform and invest more effort than individuals with more controlled motivation forms. Similarly, Vansteenkiste et al. (2004) show across various learning contexts that self-determined motivation makes individuals perform better and leads to higher levels of well-being.

2.3.3 Social Practice View

The dissertation also draws on the social practice view (SPV) by MacIntyre (1981) to understand FLOSS developers' project behavior, not as a supplement to but rather as complement to SDT. In particular, the social practice view is used to derive a more holistic picture on FLOSS developers' project work.

In his influential work 'After Virtue', Alasdair MacIntyre (1981) presents the social practice view as a new theoretic framework for understanding individual behavior. MacIntyre's social practice view is part of a new form of virtue ethics and a fundamental critique of utilitarianism (Moore and Beadle 2006). Compared to classic social theories, MacIntyre's social practice view takes a much broader perspective toward understand individual behavior. For example, while SDT takes a neutral stance on the enacted behavior and focuses on the instrumental and satisfying use of it, MacIntyre's social practice view puts the particular behavior in the context of how it helps the individual to achieve excellence and unity of life (Weaver 2006). Through this holistic view, the social practice view can even explain why individuals engage in a particular behavior even if it does not result in immediate returns for them (von Krogh et al. 2012). The following paragraphs describe the notion of a social practice, the pivotal element in MacIntyre's social practice view, as well as the various constructs with which it is interwoven.

According to MacIntyre a social practice describes '*any coherent and complex form of socially established cooperative human activity through which goods internal to that form of activity* *are realized in the course of trying to achieve those standards of excellence which are appropriate to, and, partly, definitive of that form of activity*' (MacIntyre 1981, p. 187). Furthermore, a social practice is characterized by a wide and positive effect on humankind (von Krogh et al. 2012). Although MacIntyre does not elaborate on the concrete requirements regarding coherency and complexity, he provides several comparisons to understand the meaning of these necessary properties. In particular, MacIntyre (1981) points out that '*throwing a football with skill*' should not be considered a social practice, whereas '*the game of football*' should be (MacIntyre 1981, p. 187). This coarse definition of a social practice has led to several debates about what precisely constitutes a social practice (Moore and Beadle 2006).

A central part in the description of a social practice play *internal and external goods*. Following MacIntyre (1981), internal goods are only derived through pursuing a social practice and benefit all participants of a social practice. For example, in the case of portrait painting, MacIntyre (1981) describes the creation of at least two internal goods. First, there is the excellence of portrait painting which refers to the excellence of the particular product, i.e. the portrait, and the excellence in the act of painting. The second internal good refers to the good of a certain kind of life (MacIntyre 1981). This type of internal good is derived through individuals' self-reflection of their performance in the context of their life (Köhne 2012). Contrary to internal goods, *external goods* are bound to individuals and can also be attained through other means of doing (Weaver 2006). Typically external goods are pecuniary rewards and the earned fame among others for one's work (Moore and Beadle 2006). In the example of the portrait painter, such external goods could be the money earned or the fame received for the portrait.

Another difference in the two types of goods concerns their provision. Internal goods are derived through pursuing a social practice in line with '*standards of excellence*'. These standards encompass concrete behavioral and technical guidelines on how to perform the social practice. Moreover, the standards of excellence comprise of a generic element, which is the participants' will to respect the standards of excellence as well as their will to be judged based on how their performance compares to these standards (Köhne 2012, MacIntyre 1981). Continuing the example from above, the standards of excellence for portrait painting would comprise of technical guidelines related to the drawing style such as color mixing and material. In contrast, external goods are contingent on the existence of *institutions*. In MacIntyre's conceptualization, institutions are resembled through classic organizations, to which he prescribes the responsibility for acquiring money and which are structured '*in terms of power and status, and they distribute money, power and status as rewards*' (MacIntyre 1981, p. 194). The description of these institutions reflects MacIntyre's fundamental criticism of capitalist organizations which according to him 'won' over social practices (Beadle 2006). The core of MacIntyre's critique is that money, power, and status have invaded the social practice and the derivation of internal goods (Moore and Beadle 2006, MacIntyre 1994). In the example above, an art company for which the painter works would resemble an institution. MacIntyre's critique is expressed by the profit orientation of the company which could lead to the directive that employees should spend less attention to the details of deriving the product and focus primarily on the output quantity.

Finally, MacIntyre's conceptualization of a social practice and the derivation of internal goods are linked with the notion of virtues. According to MacIntyre, virtues are "*dispositions which will not only sustain practices and enable us to achieve the goods internal to practices, but which will also sustain us in the relevant kind of quest for the good*" (MacIntyre 1981, p. 218). Thus, virtues refer not only to the ability to achieve excellence in a particular social practice but also beyond the practice (Beadle 2006). MacIntyre proposes that humans strive to achieve 'unity of life' so that they can conceive their lives as a whole (Beadle 2006). In this context the development of virtues help individuals to identify those social practices which are relevant to them. According to Long (2006) honesty and courage are two examples for virtues which are relevant across various social practices.

2.3.4 Mentoring

Due to the high learning barriers in FLOSS projects, knowledge building is considered a pivotal challenge for retaining FLOSS developers longer. A possible means to address this challenge is the use of mentoring as an education and retention strategy.

Mentoring is a one-on-one teaching method in which an experienced employee (the mentor) provides technical assistance and psychological support to a less experienced individual (the protégé) (Kram 1985). Mentoring relationships have two effects on protégés. On the one hand, protégés are assisted in their actions and in learning new knowledge through the technical guidance of their mentor. On the other hand, mentors provide psychological support to their protégés in the form of counseling or friendship which in turn builds a strong interpersonal relationship between them. This interpersonal bond between the mentor and the protégé differentiates mentoring from other training methods such as classroom teaching and supervisor-employee relationships (Hale 2000). These two effects distinctly advance protégés' knowledge building. First, mentors help their protégés acquiring declarative knowledge, such as understanding facts and routines necessary for them to accomplish their job successfully, by providing technical assistance. In addition, mentoring relationships help protégés develop procedural knowledge about how to accomplish the task. This form of knowledge is very difficult to convey because it is tacit and generally arises only through practical experience. With respect to this twofold learning effect, previous evaluations suggest that mentoring is superior to other education techniques in organizations (Hale 2000). Moreover, empirical studies support that mentoring relationship enhance not only protégés' knowledge building but also their job satisfaction and their retention rates (Hale 2000, Brashear et al. 2006).

There are two basic forms of mentoring: formal and informal. In the case of formal mentoring, a concrete assignment between the mentor and the protégé is created by the organization. Thereby, formal mentoring relationships are precisely defined in terms of their learning objectives and length. In contrast, informal mentoring arises spontaneously between colleagues and has no defined content or structure. Despite their similarities as one-on-one training, formal and informal relationships should not be considered interchangeable. Rather the used mentoring form should be selected to fit the particular purpose. Formal mentoring programs should be used to educate narrowly defined learning objectives while informal mentoring relationships should be performed to foster a much broader and long-lasting development of the individual within the organization (Eby and Lockwood 2005).

Previous evaluations within the organizational domain support the effectiveness of formal as well as informal mentoring relationships. Specifically, research suggests that protégés in informal mentoring relationships achieve greater long-term behavioral change due to the investment of more personal efforts (Eby and Lockwood 2005). Moreover, formal mentorships are especially effective if protégé are satisfied with their mentor. Eby and Lockwood (2005) suggests that formal mentoring programs serve not only to meet the specified learning objectives but also to foster protégés' development within the organization. In particular, employees who have been mentored report advancements in career planning and networking opportunities (Eby and Lockwood 2005). The positive effects of formal mentoring programs are supported by empirical research by Lentz and Allen (2009). According to this study, formal mentoring benefits both the protégé and the organization. Specifically, the study reveals that protégés advance in their knowledge building and are more satisfied with their job and their intention to stay with the organization. Moreover, mentoring relationships help to alleviate negative experiences of career plateauing.

2.4 Summary

In summary, this section outlined the theoretic background for this dissertation. The first subsection provided an overview of the term FLOSS and the history of FLOSS development. Then, the distinct challenges involved in attracting, integrating, and retaining FLOSS developers are outlined. Finally, relevant organizational theories are introduced that serve as basis to address these challenges in FLOSS projects. Figure 4 below summarizes the distinct challenges, the used organizational theories, and the proposed management approaches.

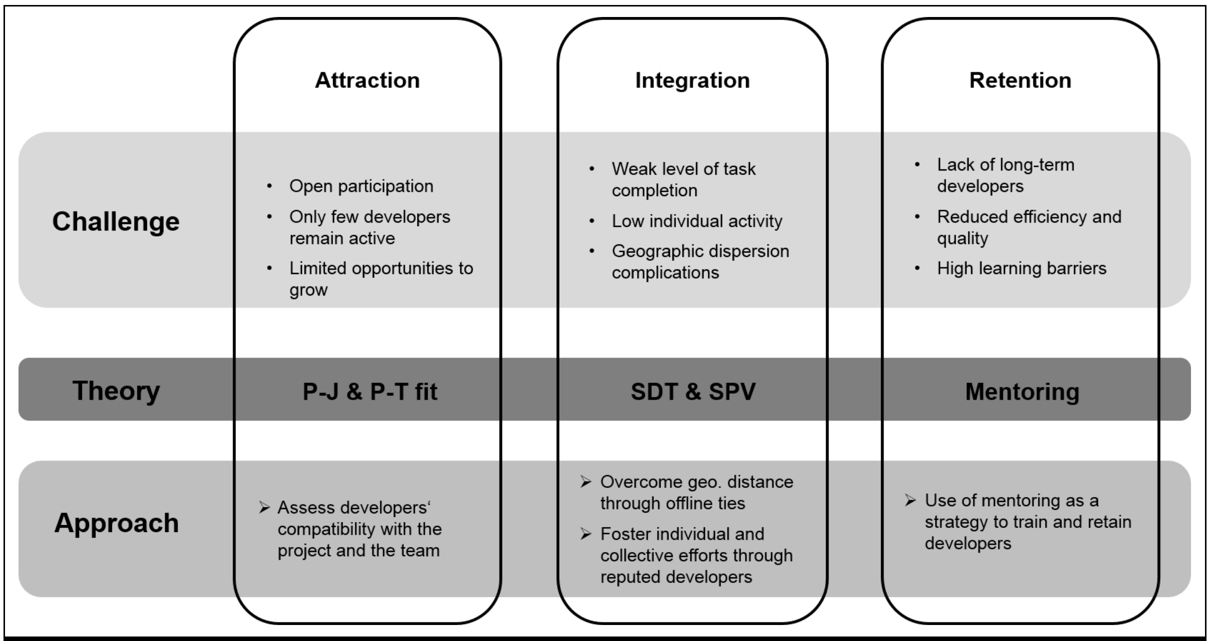

Attraction
Integration
Retention
Challenge
• Open participation
• Only few developers remain active
• Limited opportunities to grow
• Weak level of task completion
• Low individual activity
• Geographic dispersion complications
• Lack of long-term developers
• Reduced efficiency and quality
• High learning barriers
Theory
P-J & P-T fit
SDT & SPV
Mentoring
Approach
➢ Assess developers' compatibility with the project and the team
➢ Overcome geo. distance through offline ties
➢ Foster individual and collective efforts through reputed developers
➢ Use of mentoring as a strategy to train and retain developers

Figure 4: The challenges, used theories, and proposed approaches

3 Research Methodology

To address the research questions defined and derive useful advice in the three defined management areas for developer management in FLOSS projects, the dissertation combines qualitative and quantitative studies. First, in Subsection 3.1, KDE is introduced which is the evaluation context of all quantitative studies. Then, the Google Summer of Code (GSoC) event is described, which was the research setting for three of the six quantitative studies. Finally, the specifics of the performed qualitative and quantitative studies are detailed.

3.1 The K Desktop Environment

The K Desktop Environment (KDE) is a popular desktop environment system for UNIX operating systems. The next paragraph gives a short overview of the history of KDE and its relevance to organizations and private households, before the various benefits of KDE as a study context are delineated.

In October 1996 Matthias Ettrich, the founder of the KDE project, publicly announced his idea of developing a desktop environment for UNIX systems and asked for help (Ettrich 1996). Since his original mailing list post, the development of KDE has proliferated and resulted in the creation of a wide variety of FLOSS projects ranging from games to entire office suites (KDE 2011). One of the most popular KDE projects is 'KHTML', which provided the basis for most desktop and mobile web-browsers today (Netmarketshare 2015). In addition to generating great interest among developers, the user base of KDE has flourished over the years as well. Today, KDE powers many computers in a wide variety of usage scenarios. For example, nearly 52 million children in Brazil use KDE in their schools as well as around 11,000 German embassies around the world (KDE 2011).

KDE is used as the evaluation context throughout **Paper II - Paper VII**, because it provides various characteristics which make it an appropriate context for studying management aspects in FLOSS projects. One particular benefit of KDE is the ability to study a wide variety of FLOSS projects which share contextual characteristics, including programming language and development guidelines. Moreover, all KDE projects use the same development environment (e.g. IRC channels, mailing lists, version control system, etc.). This homogenous toolset substantially lowers the effort needed to extract data on KDE developers' communication and contribution behavior. In particular, the mailing lists for all KDE projects are hosted under the common domain 'org.kde' which is indexed and archived by the web-service markmail.org. Markmail provides not only a user-friendly interface to this data, but also an Application Programming Interface (API) to use the data with compiled programs. Moreover, markmail.org has indexed the mailing list '*org.kde.cvs-commits*' through 13 November 2012, which is the central place where each accepted code commit in every KDE project is published. This made it even more comfortable to extract relevant figures on KDE developers' contribution behavior and avoid problems caused by KDE projects' change of their Version Control System (VCS) (KDE Techbase 2009). Another argument which adds to KDE as a suited evaluation context for studying the various aspects of developer management concerns the huge diversity of the developer-base, ranging from Indian teenagers to 70-year-old English grandmothers who translate button descriptions (KDE 2011). Figure 5 shows the worldwide distribution of development activity for the KDE project between 1 January 2009 and 27 April 2013, which was examined in Study II. Finally, a cooperation with the project manager of the KDE Commit Digest project provided access to restricted personal data about KDE developers, such as their location information (the basis of the evaluation in Study II), and made it possible to create a broad awareness for an online survey in the KDE community (see Study V).

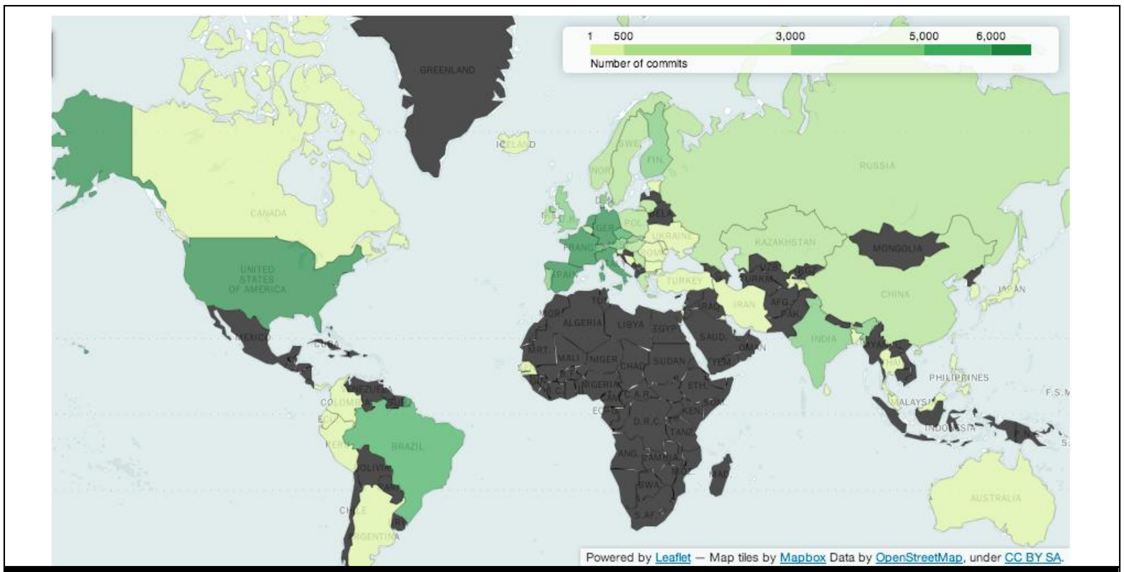


Figure 5: The worldwide distribution of KDE developers

The next section describes the annual event Google Summer of Code which is also the context of various quantitative studies on attracting, integrating and retaining FLOSS developers.

3.2 Google Summer of Code

Google Summer of Code (GSoC) is an annual event sponsored by Google in which students are awarded stipends to contribute to FLOSS projects during their summer break (Google 2015). Over the last ten years, the number of available GSoC sponsorships has more than tripled. While there were 419 stipends available to students in 2005, the most recent GSoC event in 2014 was supported by 1,307 stipends, over three times as many as in 2005 (Google 2014). In addition to receiving financial compensation, GSoC students are assisted by experienced developers who act as mentors for their project work.

The application procedure for GSoC consists of a two-step process (Google 2015). First, applicants write a project proposal in which they describe the specifics of their intended project work, including a precise schedule for its implementation. In the course of their application, GSoC students nominate also a preferred mentor for their coding project. In the second part of the application process, the particular FLOSS projects prioritize and select the GSoC proposals.

In the case of KDE, the selection process for the various GSoC applications comprises an individual as well as a community evaluation. The individual evaluation is normally performed by the nominated mentor, who reviews the proposed project and the suggested timeline and assigns a score to the proposal. After this individual review, all KDE community members are invited to vote on the project proposals. The results of these two evaluation rounds are combined to generate an overall prioritization of the GSoC proposals. Finally, based on the number of available GSoC slots for KDE, the project proposals are accepted in order of the derived prioritization.

With respect to the application and selection process, the GSoC event in KDE provides a uniquely suited study context to evaluate the relevance of P-J fit and P-T fit characteristics for predicting FLOSS developers' project permanence. The provided information in students' applications provide new ways to operationalize and evaluate fit characteristics. Furthermore, the evaluation context allows objectively derived criteria to be compared with KDE developers' subjective assessment. Finally, GSoC allows the systematic evaluation of mentoring effects in FLOSS projects. Although mentoring can also occur at other occasions, the amount of time spent on it and its structure and goals are seldom as well documented as in GSoC.

3.3 Qualitative Studies

In this dissertation several qualitative case studies were performed to understand the difficulties of developer management in FLOSS projects. Moreover, these case studies intend to derive and pre-evaluate possible strategies for addressing these difficulties.

As proposed by Eisenhardt (1989) and Yin (2009), qualitative case studies can be characterized as a research method which seeks to derive an understanding of the underlying holistic and meaningful characteristics and dynamics of real life events. As a result, qualitative case study research is an appropriate research method to examine 'how' and 'why' research questions (Yin 2009).

In order to understand the concrete problems FLOSS projects are confronted with regarding the management of their developer base, various project administrators and domain experts were interviewed. To ensure that the derived results are representative, individuals from various FLOSS projects and related organizations were selected. Table 1 provides an overview of the individuals interviewed and their involvement in FLOSS projects.

Table 1: Overview of the qualitative studies performed

Case study	Individual	FLOSS project
Case study I	Lydia Pintscher	Community Manager at KDE
Case study II	Leslie Hawthorn	Google Summer of Code Coordinator
Case study III	Brian Proffitt	Community Manager at the Linux Foundation
Case study IV	Selena Deckelmann	Main Developer of PostgreSQL
Case study V	Till Adam	Services Director KDAB
Case study VI	Jos Poortvliet	Community Manager at openSUSE
Case study VII	Michael Lauer	Project Manager OpenMoko
Case study VIII	Ian Skerrett	Marketing Director of the Eclipse Foundation

Following the recommendations of Yin (2003) the interviews were semi-structured, which means that the study-related questions were derived to fit the particular research question. The performed interviews lasted between 45 minutes and 1.5 hours and were recorded and afterwards transcribed to ease information processing and analysis.

The results of the case studies helped to understand the practical difficulties in attracting, integrating, and retaining FLOSS developers. Moreover, the results form the basis for the quantitative evaluation of various management aspects and underlying theories which are outlined in the following subsection.

3.4 Quantitative Studies

Based on the results of the qualitative studies, six quantitative studies were performed to empirically evaluate the concepts and means for effectively managing developers in FLOSS projects. In some cases, these studies are innovative, in that several measures and data extraction routines were newly developed in the course of the dissertation. To ensure the validity of these routines and measures, they have been published and discussed in related conferences prior to their use. Table 2 lists the management aspect, the research objective, and the publications in which the data extraction and measurement were originally described for each of the six empirical studies.

Table 2: Overview of the quantitative studies performed

Studies	Management aspect	Objective	Originally proposed in:	Reported in:
Study I	**Attraction**	Assess the use of P-J and P-T to identify sustained developers	**Paper I**	**Paper II**
Study II	**Integration**	(i) Evaluate the effects of spatial, temporal, and cultural distances on developers' interplay. (ii) Examine if direct offline interactions mitigate these problems.	Schilling et al. (2013)	**Paper III**
Study III	**Integration**	Assess developers' perceived motivational stimuli through working with reputable developers.	Schilling (2012)	**Paper IV**
Study IV	**Integration**	Evaluate productivity gains for FLOSS teams through the presence of reputable developers.	Schilling et al. (2014)	**Paper V**
Study V	**Integration**	Assess if and how reputable developers affect team members individual productivity	Schilling (2012)	**Paper V**
Study VI	**Retention**	Evaluate mentoring as a viable education and retention strategy.	Schilling et al. (2012)	**Paper VI**

The following subsections outline the data extraction strategy and the measurement for the examined constructs for each empirical study.

3.4.1 Study I

Study I examines new ways for FLOSS projects to identify FLOSS developers who are likely to remain active in the project. In particular, this study evaluates various measures for actual and subjective P-J and P-T fit to predict developers' project permanence. To do so, GSoC at KDE is chosen as evaluation context, because it allows to access detailed personal information of students, like their year of study which is rarely found in FLOSS projects. Moreover, GSoC is used as study context as it allows to contrast the accuracy of the newly derived measures with the employed subjective evaluation.

Data Extraction

The archival and subjective data which form the basis for this evaluation have been extracted from four distinct data sources. First, the names and email addresses of all GSoC students at KDE were extracted from the official websites for GSoC-2009 and GSoC-2010. Based on this screening process, 83 GSoC students were identified who contributed to a KDE project in 2009 or 2010 (36 students from GSoC-2009 and 47 students from GSoC-2010). Then, their contribution and conversation records were extracted using the web-service markmail.com, which indexes all KDE mailing lists and provides an API for this data. As the number of indexed mailing lists also includes the mailing list 'kde.cvs-commits' to which all accepted code commits to every KDE project get propagated, the markmail API could not only be used to reconstruct GSoC students' prior conversations but also their prior code contributions to KDE projects. With the exception of two students from GSoC-2009 and one student from GSoC-2010 all GSoC students could be associated with commits in the KDE code base repository. The three mismatches are possibly the result of students' use of different nicknames for KDE and GSoC. Thus, the overall study sample comprises 80 GSoC students. Figure 6 summarizes the details of the data extraction methods.

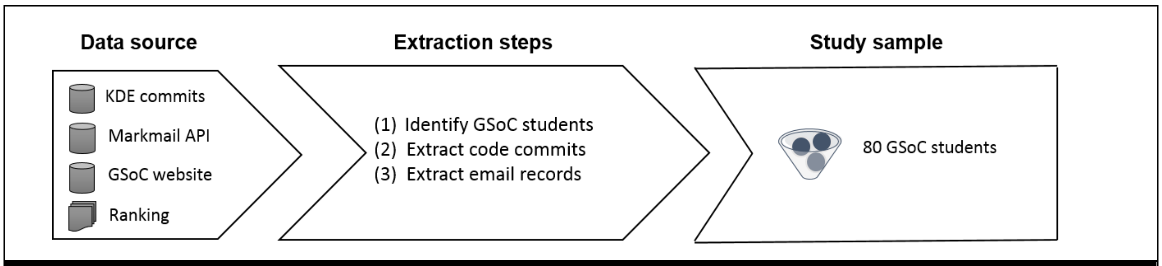


Figure 6. Data extraction in Study I

Measurement

As GSoC students formulate their project proposal according to their specific needs, needs-supply match is considered high. Therefore, the study focuses on demands-ability match for assessing GSoC students' P-J fit. In line with practices from the recruitment context, GSoC students' relevant abilities are assessed by their project expertise and project experience. To assess GSoC students' project expertise ($proj\_expertise_{i,t}$) at the beginning of GSoC, their year of study ($YoS_{i,t}$) at that time was considered. Students' project experience ($proj\_experience_i$) at the start of GSoC was assessed based on their previous engagement in KDE. Based on consultations with KDE experts, students' project experience was categorized into one of three classes. These classes roughly reflect the required efforts for the development of an add-on (< 3 commits), a small application (< 94 commits), and everything beyond that.

$$proj\_exptise_{i,t} = YoS_{i,t} \tag{1}$$

$$proj\_experience_i = \begin{cases} low & \text{if } < 3 \text{ prior\_commits} \\ mid & \text{if } 4 - 94 \text{ prior\_commits} \\ high & \text{if } > 94 \text{ prior\_commits} \end{cases} \tag{2}$$

To measure GSoC students' supplementary fit (*team_exp*$^{sup}_i$), the timespan between students' first mailing list post and the beginning of the particular GSoC event was collected. With the help of KDE administrators, the derived time was classified into the following three categories.

$$team\_exp_i^{sup} = \begin{cases} low & \text{if } < 30 \text{ prior\_days} \\ mid & \text{if } 31 - 180 \text{ prior\_days} \\ high & \text{if } > 180 \text{ prior\_days} \end{cases} \tag{3}$$

To evaluate students' complementary fit (*team_exp*$^{comp}_i$) their participation in mailing list discussions at the Bugzilla platform was considered. Bugzilla is the central platform to which KDE-related programming deficits and their solutions are posted and where discussions take place. Thus, students' complementary fit at the beginning of GSoC was determined by how frequently they engaged in programming deficit-related project discussions. With the assistance of KDE experts, the participation behavior in such problem-related discussions was classified as follows:

$$team\_exp_i^{comp} = \begin{cases} low & \text{if } < 5 \text{ prior\_posts} \\ mid & \text{if } 6 - 60 \text{ prior\_posts} \\ high & \text{if } > 60 \text{ prior\_posts} \end{cases} \tag{4}$$

KDE members' subjective evaluation of GSoC students' P-J and P-T fit was assessed by the assigned prioritization of their particular project proposal. As described in Subsection 3.2 above, GSoC proposals are evaluated in a twofold evaluation approach at KDE, first by the mentor and then through by all KDE members. Then, these results are combined to prioritize the proposals and award stipends.

In line with previous FLOSS research by Colazo and Fang (2009), the dependent variable, students' retention in the projects (*proj_ret*$_i$), was assessed based on the number of days between the end of GSoC ($D^i_{GSoCEnd}$) and their most recent code commit (D^i_t). The specific measurement of the dependent variable and the controls included in this study are summarized in Table 3.

Table 3: Measures for dependent and control variables in Study I

Construct	Based on	Sample Items
Project retention	Colazo and Fang (2009)	$proj\_ret_i = D_t^i - D_{GSoCEnd}^i$
Team size	Colazo and Fang (2009)	$team\_size^t = \mid team^t \mid$
Project size	Midha (2008)	$proj\_size^t = LoC^t$
Project age	Colazo and Fang (2009)	$proj\_age^t = NoD^t$

3.4.2 Study II

The second study in this dissertation examines the productivity effects of spatial, temporal, and cultural distances on FLOSS developers' teamwork. In addition, the study examines if offline interactions help FLOSS developers to overcome the negative effects of their geographic dispersion. The data extraction and measurement approach used in this study was originally delineated in Schilling et al. (2013).

Data Extraction

The archival records of KDE developers' collaboration behavior which build the foundation of this study were extracted in cooperation with the KDE Commit Digest project. First, the FLOSS project 'Enzyme'[3] was used to extract detailed commit statistics based on information from the VCS of each of the 65 KDE projects. Next, KDE developers' geographic locations were extracted from the KDE Commit Digest project (this project allows KDE developers to share their profiles with each other). This location data was merged with the extracted contribution data to identify KDE projects in which at least 75 percent of the submitted code commits could be assigned to developers with location information. The following six projects fulfilled this criteria and were thus selected for the study: 'KDE PIM' (a personal organizer), 'DigiKam' (a photo management suite), 'KDELibs' (cross-application libraries), 'Calligra' (an office suite), 'KDE Workspaces' (a desktop organizer), and 'Kate' (a text editor). Finally, information on KDE developers' offline meetings was extracted based on information from the central KDE website for organizing developer sprints (https://sprints.kde.org/). Since this website was not launched until April 2011 and used only hesitantly by various KDE projects at the beginning, it was also necessary to screen the websites of the examined KDE projects for information on past developer sprints. In such cases, the attendee list of previous developer sprints was reconstructed based on a blog post or a group photo.

Consistent with the observations by Kuk (2006), the development of the examined KDE projects is characterized by a high developer fluctuation. In fact, the high fluctuation leads to a new developer composition at the targeted KDE projects every week. In response to this high fluctuation and following previous research by Singh (2010), the contribution history of the six KDE projects was examined in segments of one week. In order to focus on members' collaboration process, only team configurations with at least two developers were considered.

[3] Website: '*http://enzyme-project.org*', source code: '*http://github.com/dannyakakong/Enzyme*'

Figure 7 visualizes the used sampling strategy with the four developers Mark, Carl, Joe, and Alex. Using this sampling and filtering strategy 648 team configurations (N) were derived. The various steps of the used data extraction strategy are visualized in Figure 8 below.

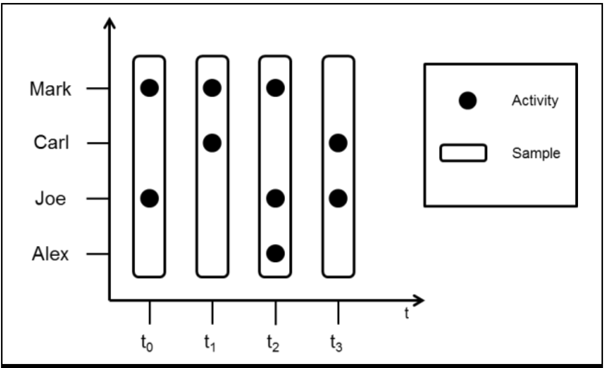


Figure 7. Sampling strategy in Study II

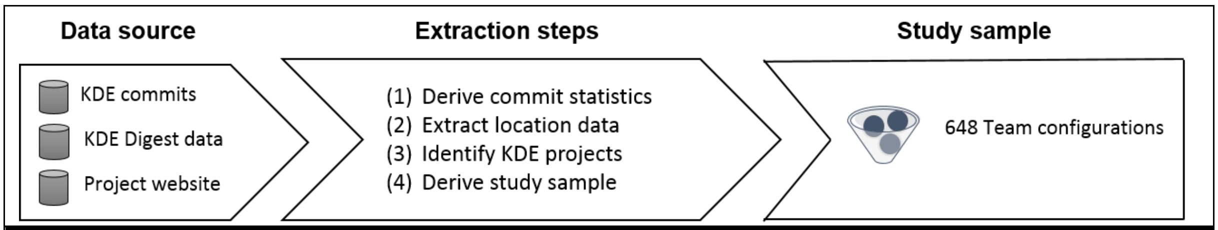


Figure 8. Data extraction in Study II

Measurement

With respect to previous work by Scellato et al. (2010) on the impact of geographic distances on interactions in online networks, the spatial distance between FLOSS developers (*spatial_dist^t*) at period *t* was not assessed in absolute terms but using the exponential decay of the distance between every two developers (*dist$_{i,j}$*). Scellato et al. (2010) recommend doing so because distance has a non-linear effect on individuals' ability to meet offline. While it makes considerable difference for individuals to meet offline if they are 1 or 1,000 miles apart from each other, it makes only a marginal difference if they are separated by 100,000 or 101,000 miles. The used measure weights the spatial distance to team members by their shares of commits in the particular period *t* (*w$_{j,t}$*) because the more individuals are involved in the particular period the more they can help other developers.

$$spatial\_dist_{i,t} = \sum_{j\in team_t \wedge i\neq j} \quad e^{-dist_{i,j}/\beta} \times w_{j,t} \tag{5}$$

$$spatial\_dist_t = \left(\sum_{i\in team_t} spatial\_dist_{i,t} \times w_{i,t}\right) \quad / \quad team\_size_t \tag{6}$$

The used measure to assess FLOSS developers' temporal distances (*temporal_dist$_t$*), considers the actual overlap in individuals' working hours. To do so, the measure uses the timestamps of every FLOSS developers' first and last commit each day to reconstruct their working hours and compute the number of overlapping hours with every other active developer each day in period *t* (*overlap$_{i,j,d}$*). In comparison, the measure used by Colazo and Fang (2010) only assesses the variance of the timestamps in FLOSS developers' first code commit each day. However, in light of the unequal work distribution in FLOSS projects (Toral et al. 2010), the sole consideration of differences in developers' stating time could lead to measurement bias. As in the case of FLOSS developers' spatial distance, the proposed measure weights members' temporal distance to each other based of their share of commits in the particular period to account for the differences in FLOSS developers' relevance at the particular period *t*.

$$temporal\_dist_{i,t} = \sum_{j \in team_t \wedge j \neq i} (\sum_{d \in t} overlap_{i,j,d}) \times w_{j,t} \tag{7}$$

$$temporal\_dist_t = (\sum_{i \in team_t} temporal\_dist_{i,t} \times w_{i,t}) \;\; / \;\; team\_size_t \tag{8}$$

The cultural distance among FLOSS developers (*cultural_dist$_t$*) was assessed based on studies by Hofstede (1980). According to Hofstede (1980) cultural differences among individuals can be assessed based on the following four criteria: (i) *Power Distance Index (PDI)*: the acceptance of unequal power distributions (ii) *Uncertainty Avoidance Index (UAI)*: the acceptance of uncertainty, (iii) *Masculinity / Femininity Index (MFI):* the dominance of masculine or feminine values in society, and (iv) *Individualism/Collectivism Index (II):* the need for individuals to integrate into groups. In line with research by Malik and Zhao (2013), FLOSS developers' cultural differences is assessed based on sum of the absolute differences of their national index scores. Although, Hofstede's research provides index scores for the countries of most developers in the evaluation sample, it does not cover some Eastern European countries. For these countries, study results by Huettinger (2008) were used, who extended Hofstede's research to Eastern European countries. As for FLOSS developers' spatial and temporal distances, the proposed measure weights a FLOSS developer's cultural distance to the other developers with respect to their share of commits (*w$_{j,t}$*) within the particular period.

$$cultural\_dist_{i,j} = \sum_{j \in team_t \wedge j \neq i} (|PDI_i - PDI_j| + |UAI_i - UAI_j| + |II_i - II_j| + |IDVI_i - IDVI_j|) \times w_{j,t} \tag{9}$$

$$cultural\_dist_t = (\sum_{i \in team_t} cultural\_dist_{i,t} \times w_{i,t}) \;\; / \;\; team\_size_t \tag{10}$$

The existence and degree of direct offline interactions among the involved FLOSS developers at the particular project in period *t*, is assessed in a two-step computation approach. In a first step, the information about attendance at the various developer sprints was used to construct a global offline relationship graph for the involved FLOSS developers. In this graph, an undirected link between developers is drawn if they attended the same coding sprint. The links in this graph are weighted by the number of previous interactions. In a second step this undirected graph was transformed into a directed graph by weighting the various connections based on the differences in interaction partners' level of expertise. Figure 9 illustrates this transformation process using an example of four developers: Anna, Mark, Carl, and Joe. In this example, Anna and Mark attended the same developer sprint. Mark is less experienced than Anna because he is new to the project. Therefore, he can benefit much more from this meeting than Anna. In turn, Anna benefits much more from meeting with Carl at some other event because he is even more experienced than she. With respect to this created relationship network, the level of directed offline interactions within a FLOSS team is assessed by calculating team members' average degree of outgoing links at period *t*.

$$\eta_{i,j,t} = expertise_{j,t} \times meetup_{i,j,t} \tag{11}$$

$$OT_t = \left(\sum_{i \in team_t} \sum_{j \in team_t \wedge j \neq i} \eta_{i,j,t}\right) \quad / \quad team\_size_t \tag{12}$$

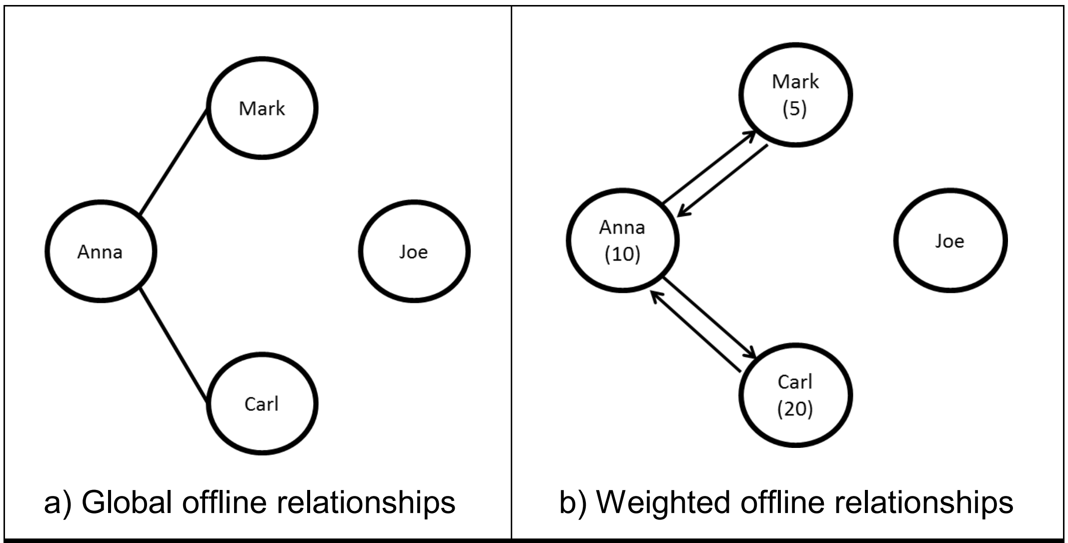

Mark
Anna
Joe
Carl
a) Global offline relationships
Mark (5)
Anna (10)
Joe
Carl (20)
b) Weighted offline relationships

Figure 9. Modeling offline social networks in Study II

In addition to the various aspects of FLOSS developers' geographic dispersion and the degree of offline interactions among them, various controls were considered. These control variables are listed in Table 4 together with their measurement and the literature on which the measurement was derived. In addition, Table 4 details the precise measurement of FLOSS developers' average team productivity, which is the dependent variable in this study.

Table 4: Measures for dependent and control variable in Study II

Construct	Based on	Measurement
Avg. team productivity	Singh et al. (2011a) Grewal et al. (2006)	$prod^t = \left(\sum_{i \in team^t} c_i^t\right) \quad / \quad team\_size^t$
Team size	Colazo and Fang (2009)	$team\_size^t = \mid team^t \mid$
Team experience	Schilling et al. (2014)	$team\_exp^t = \left(\sum_{i \in team_t} \sum_{j \in team_t \wedge j \neq i} D_{i,j,t}\right) / \ team\_size^t$
Project experience	Schilling et al. (2014)	$proj\_exp^t = \left(\sum_{i \in team^t} D_i^t\right) \quad / \quad team\_size^t$
Project size	Midha (2008)	$proj\_size^t = LoC^t$
Project age	Colazo and Fang (2009)	$proj\_age^t = NoD^t$

3.4.3 Study III

Study III examines if and how working together with reputable developers stimulates FLOSS developers' motivation to contribute to a project. This examination is performed in the context of GSoC at KDE and Gnome. GSoC was chosen as evaluation context for this study because GSoC students are generally mentored in their project work by reputable FLOSS developers. Like KDE, Gnome is a popular desktop environment for UNIX systems with a long development history and a diverse spectrum of FLOSS projects. Another advantage for the evaluation is that KDE and Gnome are two of the largest organizations which participate in GSoC. The question items used to assess the reputation of FLOSS developers' collaboration partner were originally proposed in Schilling (2012).

Data Extraction

To assess the motivation stimuli FLOSS developers perceive through working with reputable developers, a private online survey of GSoC students at KDE and Gnome was performed. For this online survey, the email addresses of all GSoC students at KDE and Gnome in 2011 were extracted from the official GSoC-2011 website and they were invited to participate in an online survey. To improve the response rate to this survey, a reminder email was sent out to all invited students who had not completed the questionnaire within two weeks after the original mailing. Overall 97 GSoC students at KDE and Gnome were invited to this online survey of whom 65 students participated in it. Figure 10 summarizes the details of the data extraction procedure used for this study.

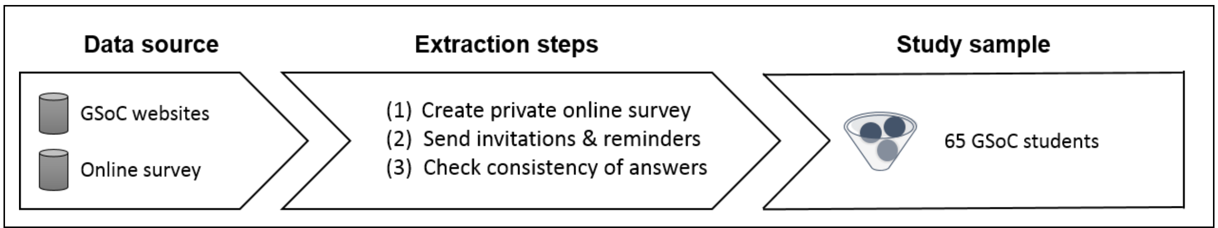

Data source
Extraction steps
Study sample
GSoC websites
Online survey
(1) Create private online survey
(2) Send invitations & reminders
(3) Check consistency of answers
65 GSoC students

Figure 10. Data extraction in Study III

Measurement

To assess the type and strength of FLOSS developers' forms of motivation, the survey adopted question items used by Sen et al. (2008) and Ke and Zhang (2010). In particular, the question items used to assess FLOSS developers' extrinsic motives were adopted from Ke and Zhang (2010). The question items used to evaluate FLOSS developers' level of intrinsic motivation were used by Sen et al. (2008). Table 5 summarizes the FLOSS literature from which the various question items were adopted and provides sample items for each construct.

FLOSS developers' community reputation was assessed based on the assumption that FLOSS projects are meritocratic i.e. FLOSS developers earn community reputation based on their project contributions. Based on this assumption and previous research by Schilling (2012), the community reputation of the developer the individuals have worked with was assessed using the following three question items: (1) 'My mentor is highly respected by other developers in the community', (2) 'Other developers know my mentor for his/her competence' and (3) 'The standing of my mentor in the community is very strong'.

Table 5. Measures for the dependent variables in Study III

Construct	Based on	Example
External motivation	Ke and Zhang (2010)	*'I am keenly aware of the income goals I have for myself if I participate in this project'*
Introjected motivation	Ke and Zhang (2010)	*'I am strongly motivated by the recognition I can earn through participating in this project'*
Identified motivation	Ke and Zhang (2010)	*'When I talk about the project, I usually say 'we' rather than 'they''*
Integrated motivation	Ke and Zhang (2010)	*'The project shares my views on open source software'*
Intrinsic motivation	Sen et al. (2008)	*'It is fun participating in this project'*

3.4.4 Study IV

Study IV examines the positive effects of the involvement of reputable developers on FLOSS teams' productivity. Therefore, the study looks beyond the motivational effects evaluated previously and focuses on the consequences which result from the involvement of reputable developers for teamwork productivity. The data extraction and a variation of the measurement used was originally published in Schilling et al. (2014).

Data Extraction

The archival records for this study were derived from two distinct data sources. As in Study II, the FLOSS project 'Enzyme' was used to derive based on the VCS in each of the 65 KDE projects detailed contribution information. In addition, Ohloh.com was queried for information on KDE developers' community reputation. Ohloh.com is a social networking site which allows FLOSS users and developers to create a profile page about themselves and exchange 'Kudos'. A 'Kudo' resembles a form of appreciation for the work or provided support (Hu et al. 2012). In order to extract the Kudo profiles for all KDE developers, the API of Ohloh.com was queried with the developer credentials (their name and the SHA-1 hash of their email address) which was extracted in the first step. In order to derive a comprehensive picture on KDE developers' community appreciation, their profile pages but also recursively the profile of each evaluator was extracted. In other words, the extracted Kudo data covers not only the evaluations of all KDE developers but also their evaluators and their evaluators and so forth. In total, this recursive lookup process resulted in the extraction of 8,195 Ohloh profiles and 34,300 Kudo relationships. Finally, the datasets on KDE developers' contribution behavior and their community endorsement were merged to identify KDE projects for which at least 75 percent of all submitted code commits between 1 January 2011 and 1 November 2013 could be attributed to developers with Ohloh profiles. The following six KDE projects passed this filtering process and were examined in this study: 'KDELibs' (cross-application libraries), 'KDE Workspaces' (a desktop organizer), ‘Calligra’ (an office suite), ‘DigiKam’ (a photo management suite), ‘KDE PIM’ (a personal organizer), ‘Plasma-Mobile’ (a desktop for mobile devices), and ‘Akonadi’ (a storage service for personal information).

As has been observed by Kuk (2006), there is high developer fluctuation in KDE projects. In fact, in the projects selected for this study, there was a new combination of active developers every week. Following the lead of Singh (2010), this great fluctuation in team compositions was handled by segmenting the development history of the considered FLOSS projects into weekly samples, as was also done in Study II. All team configurations consisting of fewer than three developers were omitted because the evaluation focuses on the developer collaboration. Based on this sampling and filtering strategy, the examined study sample comprised 749 team configurations (N). Figure 11 summarizes the data extraction steps and the transformation process for this study.

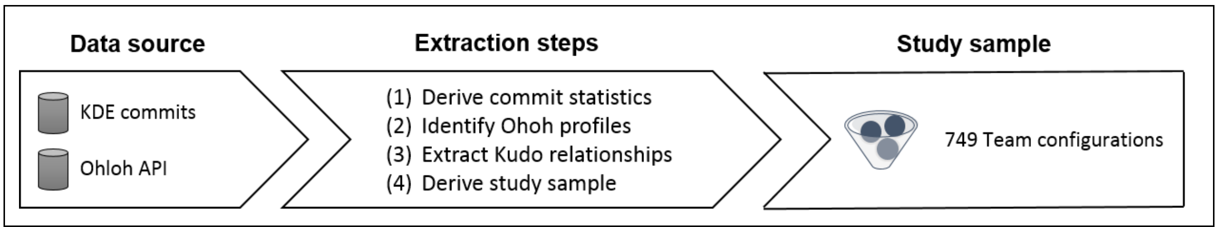
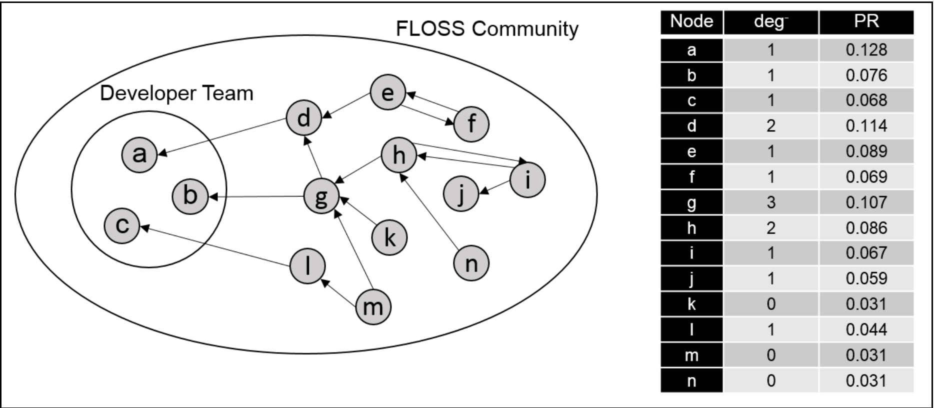


Figure 11. Extraction strategy study IV

Measurement

KDE developers’ community reputation in period *t* ($comm\_rep_i^t$) was assessed in a twofold approach. First, a global evaluation graph for each period *t* was derived based on exchanged Kudos until this period. An example of the form of this evaluation graph is presented in Figure 12 for the three KDE developers a, b, and c. Next, a rank-based measure was applied to the constructed evaluation graph to assess FLOSS developers’ community reputation in the particular period. In contrast to other network measures which treat all links in a graph as equally important, a rank-based measure distinguishes the influence of outgoing links based on the originating node’s rank. This measurement is consistent with the reputation building process in FLOSS communities, in which the positive effects of an endorsement are contingent on the evaluator’s community standing (Stewart 2005). Specifically, the PageRank algorithm by Brin and Page (1998) was used to assess FLOSS developers’ reputation because it provides two key benefits for the particular study context. First, the efficient computation of the PageRank algorithm makes it considerably easier to compute the ranks of the more than 8,000 individuals in the evaluation graph for each weak between 1 January 2011 and 1 November 2013. In addition, the computed ranks of the PageRank algorithm are very robust against reciprocal linking (Gayo-Avello 2013). Reciprocal linking describes the phenomenon that a node links to another node to get a link back. Because reciprocal linking was also discovered within the Ohloh network (Hu et al. 2012), the robustness of the PageRank algorithm against such phenomenon reduces substantially the risk of potential measurement bias. Thus, KDE developers’ community reputation in *t* was measured by computing their PageRank in the global evaluation graph at the particular period *t*. Based on this egocentric measure, the



Node	deg^-	PR
a	1	0.128
b	1	0.076
c	1	0.068
d	2	0.114
e	1	0.089
f	1	0.069
g	3	0.107
h	2	0.086
i	1	0.067
j	1	0.059
k	0	0.031
l	1	0.044
m	0	0.031
n	0	0.031

Figure 12. Evaluation graph used in Study IV

community reputation of the FLOSS developer team (*comm_rep^t^*) is assessed based on the average community endorsement of the developers involved in period *t*.

$$comm\_rep_i^t = \frac{1-d}{|\ comm^t\ |} + d \times \sum\nolimits_{\forall j \exists kudo_{j,i} \wedge j \in comm^t} \frac{comm\_rep_j^t}{kudos_j} \qquad (13)$$

$$comm\_rep^t = \left(\sum\nolimits_{i \in team^t} comm\_rep_i^t\right) \ / \ team\_size^t \qquad (14)$$

In addition to FLOSS developers' community reputation, the analysis controlled for effects of other team and project characteristics. The measures for these controls including the literature from which they were adopted are listed in Table 6. Table 6 also lists the measurement of the dependent variable of this study, which is FLOSS developers' average team productivity, and the reference it was adopted from.

Table 6. Measures for dependent and control variables in Study IV

Construct	Based on	Measurement
Team size	Colazo and Fang (2009)	$team\_size^t = \|\ team^t\ \|$
Team experience	Schilling et al. (2014)	$team\_exp^t = \left(\sum\nolimits_{i \in team^t} \sum\nolimits_{j \in team^t \wedge j \neq i} D_{i,j}^t\right) \ / \ team\_size^t$
Project experience	Schilling et al. (2014)	$proj\_exp^t = \left(\sum\nolimits_{i \in team^t} D_i^t\right) \ / \ team\_size^t$
Project size	Midha (2008)	$proj\_size^t = LoC^t$
Project age	Colazo and Fang (2009)	$proj\_age^t = NoD^t$
Team productivity	Singh et al. (2011a) Grewal et al. (2006)	$prod^t = \left(\sum\nolimits_{i \in team^t} c_i^t\right) \ / \ team\_size^t$

3.4.5 Study V

Study V examines how the involvement of reputable developers affects FLOSS developers' relationships to other team members as well as their working efforts. In order to examine these effects a public online survey was designed which was promoted in cooperation with the relaunch of the KDE Commit Digest website. In order to examine the actual as well as the subjective consequences which result from the involvement of reputable developers, this study combines perceptual and archival measurement.

Data Extraction

The data extraction for this study was performed in conjunction with the relaunch of the KDE Commit Digest website. To assess the various perceptual consequences a private online survey was compiled. At the time of the evaluation, the KDE Commit Digest project had finished a major revision of its website and was promoting it on KDE related blogs and news channels. The administrator of this project agreed to integrate a reference to the compiled online survey into the last step of the sign-up process for the project. This integration provided two key advantages for the online survey. First, the integration increased the visibility of the survey to KDE developers because it was referenced in the promotion for the KDE Commit Digest relaunch. In addition, the integration made it possible to implicitly link developers' survey to their KDE profile so their actual project behavior could be assessed. In total, 86 KDE developers participated in this survey, including six that had to be omitted due to malformed answers, resulting in a study sample of 80 KDE developers. The two data sources and the extraction steps for this study are summarized in Figure 13.

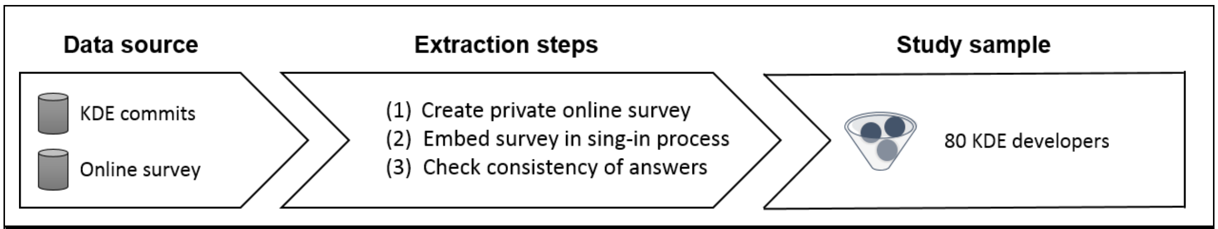

Data source
KDE commits
Online survey
Extraction steps
(1) Create private online survey
(2) Embed survey in sing-in process
(3) Check consistency of answers
Study sample
80 KDE developers

Figure 13. Data extraction in Study V

Measurement

To assess how reputable developers affect team members' relationships to each other and their working efforts, the study relies on perceptual and archival measures. To reduce measurement bias, the study relies only on measures which have already been used in previous evaluations.

In order to assess the involvement of reputable KDE developers, question items discussed in Schilling (2012) and used in Study III were selected. To evaluate FLOSS developers' cognitive and affective trust towards the team members of the particular FLOSS project, question items were adopted from Stewart and Gosain (2006) and Xu and Jones (2010). In addition, various archival measures were used to assess actual changes in KDE developers' productivity, the characteristics of the project (project size and project age), and the team (team age, team size, team experience, and project experience). Whenever possible, these archival constructs were assessed the same way as in Study II and Study IV. Table 7 summarizes the assessed constructs and the specifics of their measurement.

Table 7. Measures for dependent and control variables in Study V

Construct	Measure-ment	Based on	Measure / Sample Item
Reputation	Perceptual	Schilling (2012)	*'Some developers in this project have a strong standing in the community'*
Cognitive trust	Perceptual	Stewart and Gosain (2006)	*'I trust and respect the members of this project'*
Affective trust	Perceptual	Stewart and Gosain (2006)	*'If I share my problems with others in this project, I know they will respond constructively and caringly'*
Individual productivity	Archival	Singh et al. (2011a), Grewal et al. (2006)	$prod_i^t = c_i^t$
Team size	Archival	Colazo and Fang (2009)	$team\_size^t = \mid team^t \mid$
Individual team experience	Archival	Schilling et al. (2014)	$team\_exp_i^t = \sum_{j \in team^t \wedge j \neq i} D_{i,j}^t$
Individual proj. experience	Archival	Schilling et al. (2014)	$proj\_exp_i^t = D_i^t$
Project size	Archival	Midha (2008)	$proj\_size^t = LoC^t$
Project age	Archival	Colazo and Fang (2009)	$proj\_age^t = NoD^t$

3.4.6 Study VI

Study VI examines the effects of mentoring on KDE newcomers' project permanence and their knowledge building. This examination was performed in the context of GSoC at KDE because GSoC provides the rare occasion in which there is a documented mentoring relationship between FLOSS developers and experienced developers. The used data extraction strategy and the archival measures for this study were originally proposed and discussed in Schilling et al. (2012).

Data Extraction

For identifying mentored newcomers, the names and email addresses of all GSoC participants at KDE in 2009 and 2010 were extracted from the official GSoC websites. Then, this information was used in combination with KDE's central member directory to find those GSoC students who have been registered at KDE for no longer than four months before the start of GSoC. A similar procedure was used to identify regular newcomers to KDE. In particular, the log file of KDE's central member directory was used to identify all individuals who registered at KDE between 1 January 2010 and 1 July 2010. To ensure that the identified individuals are interested in becoming KDE developers, it was checked that they submitted at least one code commit to a KDE project during the first month after their registration. For this check the online service markmail.com was used as it indexes the mailing-list '*kde.cvs-commit*' to which each code commit to every KDE project gets published to. Based on this data extraction and filtering approach, 91 newcomers to KDE were identified for the evaluation; 41 of these newcomers have been mentored in GSoC (16 in GSoC-2009 and 25 in GSoC-2010) and 50 of them were regular (non-mentored) novices. In addition to identifying newcomers, markmail.com was used to extract all of their code commits and email records to the KDE projects which are used for assessing their learning progress. The various steps for the data extraction are illustrated in Figure 14.

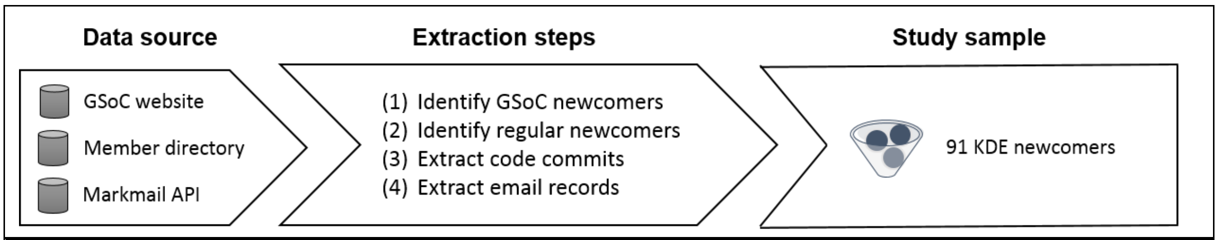


Figure 14. Data extraction in Study VI

Measurement

The level of newcomers' knowledge building, is assessed following the lead of Singh et al. (2011b), who derive an innovative learning model for the FLOSS context and show that this model is superior even to traditional learning curve models. The proposed model distinguishes three main learning states. In this model all FLOSS developers start at the lowest learning state. By engaging in learning activities (such as submitting code, or opening and participating in mailing list discussions), FLOSS developers advance into higher learning states (Singh et al. 2011b). With respect to this conceptualization, knowledge building is measured as a latent formative variable constituted by FLOSS developers' contribution and communication behavior.

Beside KDE newcomers' learning state, the study considers additional project- and team-related control values. Table 8 provides an overview of these controls and their specific measurement. The measurement used to assess FLOSS developers' project permanence is the same as in Study I.

Table 8. Measures for dependent and control variables in Study VI

Construct	Based on	Measurement
Project Retention	Colazo and Fang (2009)	$proj\_ret_i = D_t^i - D_{GSoCEnd}^i$
Team Size	Colazo and Fang (2009)	$team\_size^t = \mid team^t \mid$
Project Size	Midha (2008)	$proj\_size^t = LoC^t$
Project Age	Colazo and Fang (2009)	$proj\_age^t = NoD^t$

3.5 Evaluation Techniques

This dissertation uses various evaluation techniques to study the distinct aspects of developer behavior in FLOSS projects. The following subsections present the three evaluation techniques used in the seven research papers of this dissertation including their core assumptions and relevant measures to assess their validity.

3.5.1 Linear Model

Linear models are a fundamental modeling technique for quantitative evaluations in social science (Hanushek and Jackson 2013). A key feature of linear models is that they can be easily understood and mathematically interpreted. In addition, linear transformation can be used to express even non-linear effects in linear models (Hanushek and Jackson 2013). The following description of the different elements of linear models and Ordinary Least Squares (OLS) regression, a basic technique for estimating the parameters of linear models, paraphrases Seltman (2014)

A linear model consists of the dependent variable (*Y*) and one or more independent variables (X_i). The parameters α and β_1 to β_m are the coefficients of the linear model and express how one factor change in the independent variables affects the dependent variable. Specifically, α refers to the starting level of the dependent variable whereas β_1 to β_m are bound to the particular manifestation of the independent variable(s). Finally, the linear model includes the error term ε, which represents all latent, non-observed effects which influence the dependent variable. This error term can refer to measurement errors as well as structural errors of the performed modeling. Equation (15) depicts the population model of a linear model, which expresses the linear relationship of a dependent and independent variable for a particular population.

$$Y = a + \sum_{i=1}^{m} \beta_i X_i + \varepsilon \qquad (15)$$

Because it is generally not possible to observe an entire population, survey samples are used to estimate a particular linear model for the whole population. A common technique to estimate the coefficient(s) and the error term of a linear model is Ordinary Least Squares (OLS) regression. This method seeks to minimize the sum of the square residuals (*S*). The residuals refer to the difference between the observed instance of the dependent variable (y_i) and its assumed value according to the specified linear model. Equation (16) shows the mathematic specification of this estimation process.

$$S(b) = \sum_{i=1}^{M} y_i - (\alpha + \sum_{j=1}^{M} \beta_j x_j)^2 \to min \qquad (16)$$

OLS regression relies on several mathematical assumptions which need to be considered in order to ensure measurement validity and reliability. A core assumption is that the specified constructs have a linear relationship to each other. Other relationship forms between the dependent and independent variables can only be insufficiently uncovered or not at all. Furthermore, it is critical in linear regression that errors found are independent of each other.

The third key assumption is that the study samples are heteroscedacstic. This means that the derived study samples should not differ in terms of variance to other subpopulations. Lastly, the normality assumption supposes that the error term follows a normal distribution with expected value of zero. Despite the general relevance of these assumptions, violations of these assumptions affect the validity of a linear regression to various degrees. In particular, linear regression is somewhat robust against violations of the assumption for heteroscedascity and moderately robust against violations of the assumption of error term normality. In contrast, however, linear regression is not robust against violations of the assumption for linearity and error independence (Seltman 2014).

There are three important checks to evaluate the validity and quality of a linear model based on a concrete data sample. The first validity check is to calculate the p-values, which basically reflect the significance of various regression coefficients. In an OLS regression, this resembles a t-test with the null-hypothesis that the particular coefficient equals zero (Seltman 2014). The second important test is the check for multicollinearity. This check is used to ensure that there is no correlation between the independent variables which could bias the evaluation results. The last test is the calculation of the R^2, which is also known as the coefficient of determination. This is a test for the overall fit of the specified linear model to the observed data. It is the amount of variance in the observed data which can be attributed to the particular linear model (Seltman 2014).

In the dissertation, linear modeling and OLS regression was used in **Paper IV** and **Paper VI**. In these papers, linear modeling was used as a basic evaluation technique to examine the effects of geographic dispersion, reputable developers, and other factors on the number of commits by the developer team every week. Linear modeling was used for this evaluation context because the dependent variable is a quantitative variable. Moreover the high developer fluctuation observed in the data samples basically leads to a new developer team every week, which in turn suggests an independence of the weekly observations.

3.5.2 Proportional Hazard Model

The proportional hazard model is a common technique for survival analysis. In contrast to linear models, the outcome variable in this type of modeling is binary (e.g. survival or death of the patient). Moreover, the outcome variable is non-linearly affected by the independent variables. Thereby, the proportional hazard model examines the effects of one or more time-variant as well as time-invariant predictors on a binary outcome variable. Originating from the medical context, survival analysis and proportional hazard models are nowadays broadly used in economics as well as social science to examine the timespan between an initial event and a dichotomous event.

A common way to formulate the survival function ($S(t)$), which specifies the probability that the time T of a particular event (i.e. death) occurs after a given time t is:

$$S(t) = Pr(T > t) \tag{17}$$

Another way to model the distribution of the survival times is by using the hazard function *h(t)*. This function specifies the immediate risk that the time of the particular event *T* occurs at the observed time *t*. Equation (18) depicts the hazard function in terms of the probability that the time *T* of the particular event will occur between *t* and *Δt,* assuming that *T* did not occur until *t* (Fox and Weisberg 2011).

$$h(t) = \lim_{\Delta t \to 0} \frac{Pr[(t \leq T < t + \Delta t) \mid T \geq t]}{\Delta t} \quad (18)$$

A way to specify this hazard function further is the proportional hazard model. As the name suggests, this model assumes a proportional hazard, which means that a factor change in one of the independent variables leads to a proportional change in the hazard function. Under this assumption, the hazard function can be specified as a linear-like model consisting of various coefficients and independent variables. The first part of the model, the baseline hazard function $h_0(t)$, considers time-variant effects, including time-variant predictors. Conversely, the second part of the model comprises of the linear-like combination of time-invariant predictors. The basic notation of the proportional hazard model is specified in Equation (19).

$$h_i(t) = h_0(t) exp(\beta_1 x_{i1} + \ldots + \beta_n x_{in}) \quad (19)$$

A common technique of estimating a proportional hazard model is the Cox proportional hazard regression. This regression technique does not make any assumptions about the form of the baseline hazard function, but rather only focuses on the proportional hazard function. Despite this focus, the baseline hazard function can be estimated based on the derived covariates for the second part (Fox and Weisberg 2011). To check the proportional hazard criterion visually the log-log graph of the dependent variable and the hazard function has to be inspected. If the curves in this graph are parallel and do not cross each other, a proportional hazard can be assumed (Kleinbaum and Klein 2012). Despite the absence of further assumptions, it is beneficial to check for multicollinearity so that there is no cross-correlation among the independent variables in the observed model (Smith and Smith 2005).

To estimate the second, linear-like part of the hazard function, Cox (1972) developed a technique called partial likelihood. Since this estimation process is not based on the goal of minimizing variance of the observed data but rather on an iterative process to find the most likely coefficients, the R^2 coefficient cannot be used as a measure to assess the fit of the derived proportional hazard model. Instead, a basic test to assess the quality of a derived hazard model is the χ^2 test which compares the information explained by a concrete proportional hazard model with another proportional hazard model or the null hypothesis in which all derived coefficients are equal zero (Kleinbaum and Klein 2012).

Proportional hazard modeling in combination with Cox regression was used in **Paper III** and **Paper VII** to assess the effects of fit characteristics and mentoring on FLOSS developers' project permanence. Survival analysis was chosen for these research papers because the time already passed in the project is expected to affect the particular time when developers stop contributing to the project. Moreover, the variable of interest in this evaluation is developers' project tenure, which is a binary variable. Finally, the inspection of the log-log graphs supports the assumption of proportional hazards in the data.

3.5.3 Structural Equation Model

The third modeling technique used in the cumulative dissertation is structural equation modeling (SEM). According to Fornell (1987), SEM can be distinguished from methods of the first generation of multivariate statistic, such as multiple regressions, in that it: (i) considers multiple exogenous and endogenous variables, (ii) supports latent variables, (iii) considers measurement errors, and (iv) allows for confirmatory evaluations.

The use of latent variables is a key aspect of SEM. Latent variables describe constructs which are of theoretical interest but which are not directly observable and therefore need to be assessed indirectly using observable indicators (Bollen 1989). There are two forms of indicators: formative (or cause) indicators and reflective (or effect) indicators (Bagozzi 2011). Formative indicators cause or form the latent variable. In this case a latent variable can only be examined appropriately by considering all relevant formative indicators. The opposite is the case for reflective indicators, where the latent variable affects the indicators. Because of this inverse cause-effect relationship, the observed indicators can be a subset of all affected indicators. **Paper V** and **Paper VI** rely on reflective indicators to measure individuals' latent motivation type. An example for this is the survey item '*Participating in this project is fun*', which is one of various reflections of individuals' intrinsic motivation. Conversely, **Paper VII** follows the lead of Singh et al. (2011b) and considers FLOSS developers' contribution and collaboration behavior formative indicators for their latent learning state.

Another key feature of SEM is the ability to construct a measurement model and a structural model simultaneously (Gefen et al. 2011). This twofold construction allows errors attributable to the measurement of the various constructs to be differentiated from errors attributable to the hypothesized causal structure of the constructs. Equation (20) specifies the measurement model of a SEM in the case of reflective indicators. The reflective indicators ($X_{(m)}$) of a latent variable m are modeled as the product of the latent variable ($\xi_{(m)}$), their coefficients, which are also referred to as loadings ($\lambda_{(m)}$), and a construct specific measurement error ($\Theta_{(m)}$).

$$X_{(m)} = \xi_{(m)}\lambda_{(m)} + \Theta_{(m)} \tag{20}$$

There are various validity checks for the measurement model of a SEM. The first check assesses indicator reliability. Therefore, each indicator should load on its associated construct at a value of least 0.7 (Carmines and Zeller 1979). Moreover, the assigned indicators should explain at least half of a latent construct's variance (Chin 1998). The next validity check assesses the reliability to which a latent construct is explained through its indicators. To ensure construct reliability, each latent variable's composite reliability should be higher than 0.7 (Nunnally 1978). The third measurement check ensures discriminant validity, which refers to the distinctiveness of the various latent constructs. For discriminant validity, the average variance extracted of each construct should be higher than the quadratic correlation of that construct with any other construct (Fornell and Larcker 1981). Another check for discriminant validity is to ensure that all assigned indicators load strongest with their assigned construct and not with any other construct.

The structural model is defined as the product vector of all latent variables of its successors (Ξ) and the coefficient matrix Γ. Moreover, the structural model considers the construct specific measurement error (ε). The validity of the structural model is tested twofold. First, the significance of the hypothesized relationships can be evaluated using the bootstrapping technique. The second form of evaluation of the structural model is the calculation of the coefficient of determination (R^2). As in the case of linear regression the R^2 coefficient assess the degree to which the variance in the endogenous construct(s) are explained through the modeled constructs and hypothesized relationships (Chin 1998).

$$\xi = \Xi \cdot \Gamma + \varepsilon \qquad (21)$$

There are two common approaches to estimate a SEM which differ in their fundamental assumptions, underlying philosophy, distribution assumptions and estimation objectives. The first estimation approach is Covariance-Based Structural Equation Modeling (CBSEM). The strength of CBSEM are confirmatory model evaluations which rely on a strong conceptualization of the measurement items and modeled constructs. Therefore, CBSEM requires all measurement errors to be uncorrelated. Moreover, CBSEM requires large datasets comprising at least 200 data samples to evaluate of SEMs (Henseler et al. 2009). The second commonly used estimation technique is partial least squares (PLS). This estimation approach is variance-based and thus well suited for explanatory research. PLS does not require measurement errors to be uncorrelated and provides reliable SEM estimates even based on a relatively small amount of data samples (Chin and Newsted 1999, Henseler et al. 2009).

In line with the recommendations of Gefen et al. (2011), PLS was chosen as the estimation technique for the SEMs in **Paper V** and **Paper VI** due to the exploratory research approach in these papers. In particular, the study presented in **Paper V** examines the previously unknown effect of working with reputable developers on FLOSS developers' motivation. Similarly, **Paper VI** studies the unknown influence of the presence of reputable developers on team members' type of trust among each other. In addition to having an exploratory research focus, the two papers use PLS because the measures used were in part newly developed in the course of the dissertation and thus lack mature theoretic and empirical validation (Gefen et al. 2011).

4 Main Results

The following subsections summarizes the main results to the research questions outlined above by providing an overview of the main results of the seven research articles in the cumulative dissertation. The first subsection serves as the theoretic basis for the following subsections by summarizing the status quo on FLOSS developers' attraction, integration, and retention. The ensuing subsections describe the key findings of the six empirical studies on the three management areas for developer management in FLOSS projects.

4.1 Literature Review

4.1.1 Paper I[4]

Paper I reviews the state of research on attracting, integrating, and retaining FLOSS developers. Seven top of the class journals were screened for FLOSS-related research articles and 43 journal articles were identified which examine management aspects in FLOSS projects. These journal articles were categorized into a two dimensional concept matrix (Webster and Watson 2002). The first dimension of this concept matrix distinguishes the particular management aspect of the articles examined (i.e. attraction, integration, and retention) while the second dimension follows the recommendation of Webster and Watson (2002) and classifies the particular evaluation focus of the articles (i.e. individual-, team-, and project-centric).

Several general observations can be made based on the literature classification. First and foremost, the classification reveals that there is relatively little dedicated research on attracting and retaining FLOSS developers. This can be explained partly by the use of ambiguous measures like 'team size' which combine aspects of developer attraction with developer retention. The ambiguity of such measures makes it impossible to derive clear implications about either of the two management areas. Another general observation is that only few research articles combine aspects from more than one research perspective. Single-perspective research, however, is insufficient because research on each of the three management areas stresses the interrelation of individual, relational, and project characteristics.

Paper I reveals that most evaluations of attracting FLOSS developers took either an individual- or a project-centric research perspective. Studies with a focus on the individual highlight the relevance of extrinsic motives for FLOSS developers' initial commitment (Shah 2006, Fang and Neufeld 2009). In contrast, relevant project-based characteristics that attract new developers include a modular codebase and particular governance practices (Sen et al. 2008). Although research by Oh and Jeon (2007) and Singh et al. (2011b) indicate that team-level aspects also play a salient role in attracting developers, this aspect has received far less attention.

The performed literature classification also provides new insights into integrating developers effectively into FLOSS projects, which has been the subject of by far the most research among all three developer management tasks. Relevant means to enhance developers' project commitment include extrinsic as well as intrinsic stimuli. While, Ke and Zhang (2010) provide evidence that developers contribute more the higher they perceive their behavior to be self-determined, Roberts et al. (2006) show that there is no crowding out of intrinsic motives through extrinsic motives among FLOSS developers. Some scholars, such as Chou and He (2011), combine individual- and team-level factors and highlight that the interrelation among individual- and team-level factors. They find that project characteristics which foster

[4] Schilling A, (2014) What Do We Know About FLOSS Developers' Attraction, Retention and Commitment? A Literature Review *Proceedings of the 47th Hawaii International Conference on System Sciences (HICSS), Big Island (HI), pp. 4003 - 4012.*

developers' efforts include less restrictive code licensing, a mature codebase, and the popularity of the project.

Paper I also highlights the state of research on FLOSS developer retention. Studies with an individual focus indicate that it is especially FLOSS developers' identification with the project and their learning progress which keep them engaged in the project. Singh et al. (2011b) point out that these individual factors interrelate with group factors because FLOSS developers also learn through interacting with other team members. Project characteristics which are relevant to how long members are retained in a FLOSS project include a modular codebase and a less restrictive code licensing. Oh and Jeon (2007) suggest that team-level aspects play a salient role in retaining FLOSS developers. Their research identifies a strong herding effect among FLOSS developers, which makes their project behavior contingent on the behavior of others. However, only few papers thoroughly examined how team-level aspects affect how long FLOSS developers are retained on a project.

To summarize, **Paper I** provides an overview of the status quo on attracting, integrating, and retaining FLOSS developers. This was achieved by screening top-of-the-class journals for related research articles on FLOSS developer management and classifying them based on their management aspect and evaluation focus. Based on this categorization, **Paper I** highlights the need for dedicated research on attracting and retaining FLOSS developers and the mutual consideration of individual-, team-, and project-level aspects.

4.2 Attraction

The second chapter of this dissertation focuses on attracting developers to FLOSS projects. Building on previous evaluations, a new integrated evaluation approach is proposed and evaluated which helps to fine tune FLOSS projects' attraction efforts by identifying candidates who are most likely to remain active in the project.

4.2.1 Paper II[5]

Paper II proposes a new theoretical foundation for FLOSS projects to identify developers who are likely to remain active in the project. **Paper II** builds on previous FLOSS research, which examined the effects of individual, relational, and project characteristics on FLOSS developers' project permanence to derive concepts and criteria to assess potential candidates.

Although there are differences between the organizational and the FLOSS domain in terms of regulation and remuneration, intrinsic motivation and socialization are considered key drivers for sustained working efforts in both domains (Werbel and Johnson 2001, Fang and Neufeld 2009, Crowston et al. 2007b, von Krogh et al. 2012). Based on this commonality, **Paper II** proposes an adjusted conceptualization of the P-J and P-T fit concept for the FLOSS domain to identify developers who are likely to remain active. In line with the definition for P-J fit by

[5] Schilling A, Laumer S, Weitzel T. (2011) Is the source strong with you? A Fit Perspective to Predict Sustained Participation of FLOSS developers. *Proceedings of the 32nd International Conference on Information Systems (ICIS), Shanghai, China.*

Edwards (1991), it is proposed that FLOSS developers' level of P-J fit comprises of a *needs-supply match* as well as a *demands-ability match*. In contrast to the organizational domain, FLOSS developers are generally not attracted to FLOSS projects due to pecuniary rewards but because of concrete project features. Thus their 'needs' are much more focused towards specific contribution and implementation conditions. Consequently, the working environment and project content provide the "supply" in a FLOSS project. In assessing the demands-ability match, it is important to consider that most FLOSS projects have no dedicated demand descriptions for novices. Nevertheless, it is beneficial for newcomers to be equipped with relevant development practices and be familiar with the codebase in order to find motivation to contribute to the project in a sustained fashion (von Krogh et al. 2003, Fang and Neufeld 2009). In line with the organizational definition by Werbel and Johnson (2001), FLOSS developers' P-T fit is defined as a combination of supplementary fit and complementary fit. For the supplementary fit, **Paper II** proposes looking at the similarity of values, interests, and skills between the newcomer and the existing team members. In contrast, the complementary fit is considered the degree to which a developer has personal or technical skills which the project lacks.

In summary, **Paper II** proposes customized versions of P-J and P-T fit for the FLOSS context which can serve as a theoretical foundation to fine-tune the attraction efforts of FLOSS projects and identify talented developers. Based on existing FLOSS research, the two concepts provide a strong foundation to evaluate the individual, relational, and project-related characteristics which affect the retention behavior of FLOSS developers. Figure 15 illustrates the use of P-J and P-T fit in FLOSS projects to find developers who are likely to remain active by assessing the fit of the candidate with the project and the developer team.

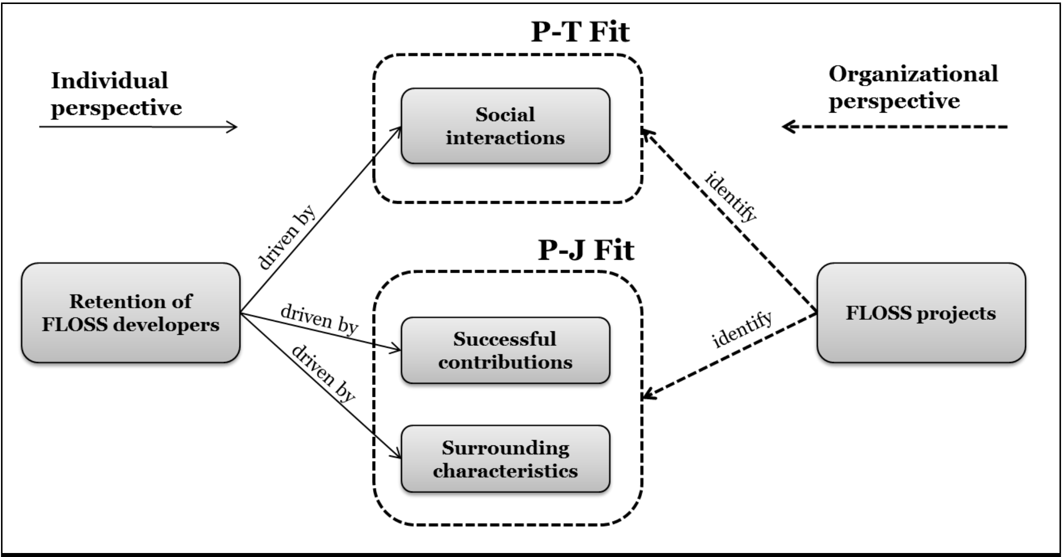


Figure 15. The use of P-T and P-J fit proposed in Paper II

4.2.2 Paper III[6]

Paper III builds on the two proposed customized concepts of P-J and P-T fit from **Paper II** and extends them in three regards. First, the paper distinguishes between objective and perceptive assessment of P-J and P-T fit. Second, it derives concrete measures to assess FLOSS developers' perceptual and objective P-J and P-T fit. Finally, it empirically evaluates these fit concepts and the measures derived based on the project permanence of 80 GSoC students at KDE (see Study I) using Cox-Regression analysis.

Paper III theorizes that there are relevant differences between the objective and perceptual assessment of P-J and P-T fit. In particular, it proposes that objective evaluation criteria provide a less biased picture of a candidate's fit with the team and the project because the assessment of actual fit is far less contingent on the assessor's experience and expertise compared to the assessment of perceived fit. In addition, **Paper III** draws on organizational literature and proposes candidates' level of relevant working experience and their year of study as useful measures of their level of P-J fit, in particular their demands-ability match. Supplementary fit is measured according to the time candidates had already been active on the FLOSS project's mailing list before joining the project. In contrast, complementary fit is measured according to candidates' contributions to bug-related discussions in the project.

The empirical evaluation in **Paper III** suggests that the subjective and the objective assessment of P-J and P-T fit help predicting FLOSS developers' project permanence. In comparison, the objective assessment of P-J and P-T fit is a much more accurate predictor for KDE developers' project permanence than the subjective evaluation. As illustrated in Figure 16b, KDE developers' subjective assessment of fit is less suited to predict the project permanence for non-extreme cases. Despite the high explanatory power of the proposed measures, actual P-J and actual P-T fit differ substantially in their ability to predict KDE developers' project permanence. For example, GSoC students' level of academic education has no effect on their project permanence. In contrast, students' level of project experience has a substantial positive effect on how long GSoC students worked on the FLOSS project (see Figure 16c). In fact, the risk of quitting is 50.6 percent lower for GSoC students who had already submitted a small patch to KDE, compared to students with no prior code contributions to KDE. Moreover, as illustrated in Figure 16d, GSoC students' supplementary fit explains to a considerable degree how long they stayed on the project. In fact, the risk of quitting is more than 64 percent lower for students who have already been active on the project's mailing list for more than a month prior to GSoC compared to students with no prior mailing list activity. However, the evaluation provides no evidence that GSoC students' complementary fit influences their project permanence.

In summary, **Paper III** advances the research presented in **Paper II** by considering the ways in which P-J and P-T fit is assessed, proposing concrete measures to assess P-J and P-T fit, and evaluating empirically the use of the two fit concepts based on GSoC students' project

[6] Schilling A, Laumer S, Weitzel T. (2012) Who Will Remain? An Evaluation of Actual Person-Job and Person-Team Fit to Predict Developer Retention in FLOSS Projects *Proceedings of the 45th Hawaii International Conference 2012 (HICSS*), Big Island (HI) pp. 3446 – 3455.

permanence. The empirical evaluation suggests that objective assessment of P-J and P-T fit predicts sustained project commitment much more accurately than subjective assessment. In particular, the objective measurement of FLOSS developers' project experience and their supplementary fit proved to be valuable predictors of their sustained project involvement.

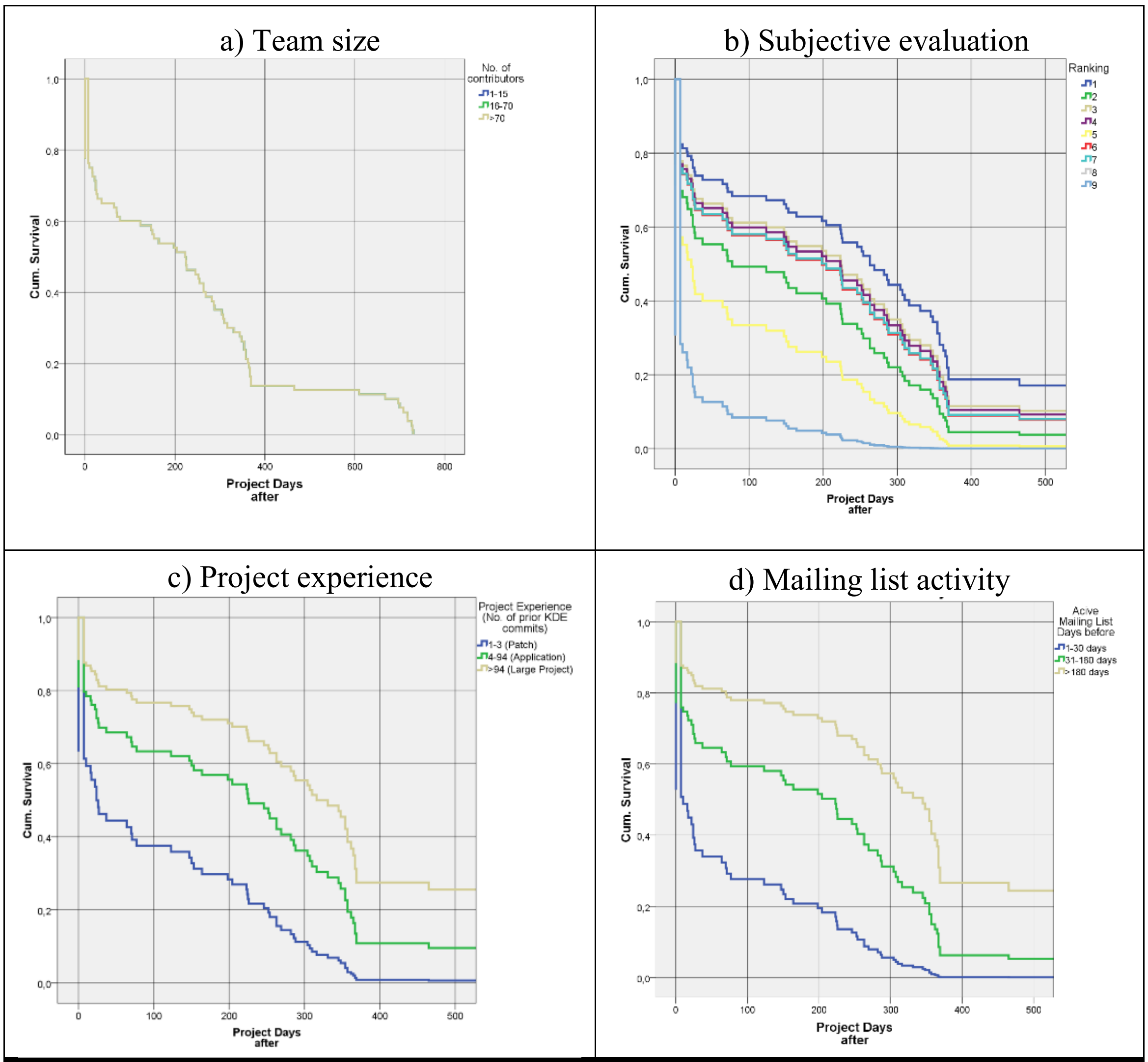


Figure 16. Results of the cox regression in Paper III

4.3 Integration

The second chapter of the dissertation examines ways to improve collaboration among FLOSS developers. **Paper IV** examines the degree to which FLOSS projects are negatively affected by spatial, temporal, and cultural distances and if these negative consequences can be overcome through direct offline interactions among FLOSS developers. In addition, **Paper V** and **Paper VI** examine the positive effects of having reputable developers involved in terms of motivating FLOSS developers and enhancing their productive collaboration.

4.3.1 Paper IV[7]

Paper IV examines the effects of FLOSS developers' geographic dispersion on their productive collaboration. In order to examine this research question, **Paper IV** proposes a new approach to consider geographic dispersion not as unitary but as multidimensional construct. It also suggests and evaluates the use of offline meetings as an effective means to overcome these problems.

Drawing on central lessons from organizational literature, Paper IV considers geographic dispersion a multi-dimensional construct consisting of spatial, temporal and cultural distances. With respect to the context low interaction in FLOSS projects, it is theorized that each of these distances has a distinct negative effect on teamwork productivity by reducing team members' level of shared knowledge and social interactions. With regard to these problems and recommendations from management literature, offline interactions are considered a potential strategy for FLOSS teams to overcome the productivity deficits caused by members' geographic dispersion. In order to evaluate the proposed effects, an empirical evaluation with 648 FLOSS teams was performed (Study II).

The results of the performed evaluation in **Paper IV**, support the notion of geographic dispersion as a multidimensional construct by showing that spatial, temporal, and cultural distances have a distinct negative effect on FLOSS developers' productive interplay. In particular, FLOSS developers' spatial and temporal distances mitigate their productive interplay moderately while cultural distances only complicate FLOSS teams' productive interplay slightly. In addition, the evaluation reveals that direct offline interactions are a viable means to overcome the negative effects of FLOSS developers' spatial and cultural distances. In particular, the analysis suggests that direct offline interactions among the involved developers reduce and even slightly reverse the negative effects of spatial and cultural distances. However, the evaluation provides no evidence that direct offline interactions contribute to overcoming the negative effects of temporal distances among FLOSS developers.

In order to understand this interaction more thoroughly and find the situations in which FLOSS developers' direct offline interactions are particularly valuable, **Paper IV** performs a post-hoc analysis. The results of this analysis show that FLOSS teams with great spatial distances among team members outperform teams with small spatial distances among team members as soon as there is a low degree of offline interaction among the involved developers (see Figure 17a). As illustrated in Figure 17b, cultural distances among FLOSS developers even become synergetic as soon as there is little direct offline interaction among the involved developers. Finally, the post-hoc analysis shows that FLOSS teams with mid-low and mid-high levels of project experience benefit most from direct offline interactions. Teams with low levels of project experience benefit also from direct offline ties but to a lesser degree. Conversely, direct offline

[7] Schilling A, Laumer S, Weitzel T. (Under Review) The Wizards of OSS - Does Developers' Geographic Dispersion Make OSS Teams More Productive? 2nd Round: *Information Systems Research (ISR)*

relationships have a negative effect on the productive interplay among team members with high project experience, as illustrated in Figure 17c.

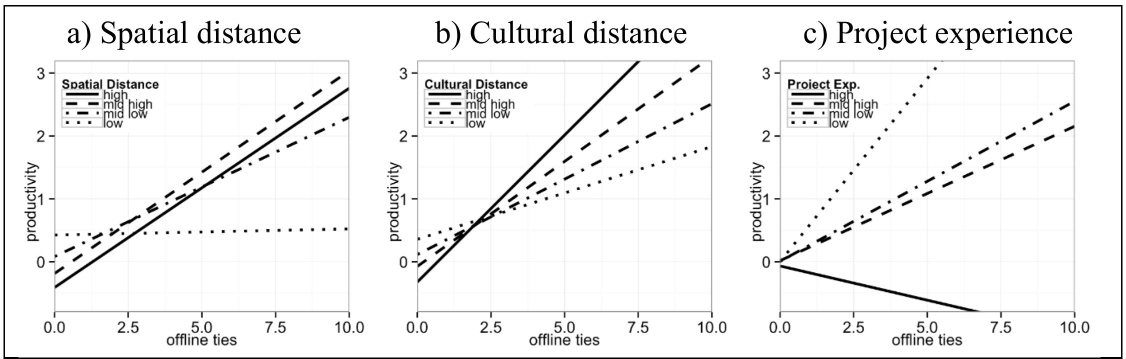


Figure 17. Visualization of the effects of geo. dispersion in Paper IV

In summary, **Paper IV** proposes and empirically evaluates the various negative effects of geographic dispersion among FLOSS developers, treating it as a multi-dimensional construct consisting of spatial, temporal, and cultural distances. The results of the empirical evaluation indicate negative effects of spatial and temporal distances, but only minor negative effects of cultural distances. Moreover, **Paper IV** theorizes and proves that the negative effects of spatial and cultural distances can be mitigated and even slightly reversed through direct offline interactions among team members. However, offline meetings have no such positive effect on team members' temporal distances.

4.3.2 Paper V[8]

Paper V examines the degree to which collaboration initiatives with reputable developers enhance FLOSS developers' motivation to contribute to a project. The examination builds on previous research by Hu et al. (2012) which highlights the stimulating effects of reputable developers on other developers' behavior.

Paper V builds on Self-Determination-Theory (Deci and Ryan 2000), theorizing that the involvement of reputable developers increases FLOSS developers' externally regulated motives by making them more visible to potential employers and to other developers within the FLOSS community. At the same time, it is proposed that reputable developers stimulate self-determined motivation forms among their collaboration partners by providing them with competent advice leading to higher autonomy in their work. To test these theorized effects, an empirical evaluation of 65 GSoC students was performed (see Study III).

The evaluation results in **Paper V** provide evidence that collaboration initiatives with reputable developers have a nuanced effect on FLOSS developers' contribution motivation. Specifically, the evaluation results suggest that working with reputable developers explains to considerable degrees FLOSS developers' self-determined motivation forms (identified, integrated, and

[8] Schilling A, Laumer S, Weitzel T. (2013) In the Spotlight - Evaluating How Celebrities Affect FLOSS Developers' Participation Motivation. *Proceedings of the 21th European Conference on Information System (ECIS), Utrecht, Netherlands.*

intrinsic motivation). However, such collaboration has little power to explain FLOSS developers' externally-regulated motivation forms. Moreover, the performed evaluation suggests that collaboration initiatives with reputable developers stimulate to considerable degrees FLOSS developers' self-determined contribution motives (identified, integrated, and intrinsic motivation). The only externally regulated motivation form which is also enhanced, is developers' introjected motivation. Even though this effect is considerable, it is the weakest of all significant effects. Figure 18 illustrates the results of the evaluation the motivational effects of working with reputable FLOSS developers.

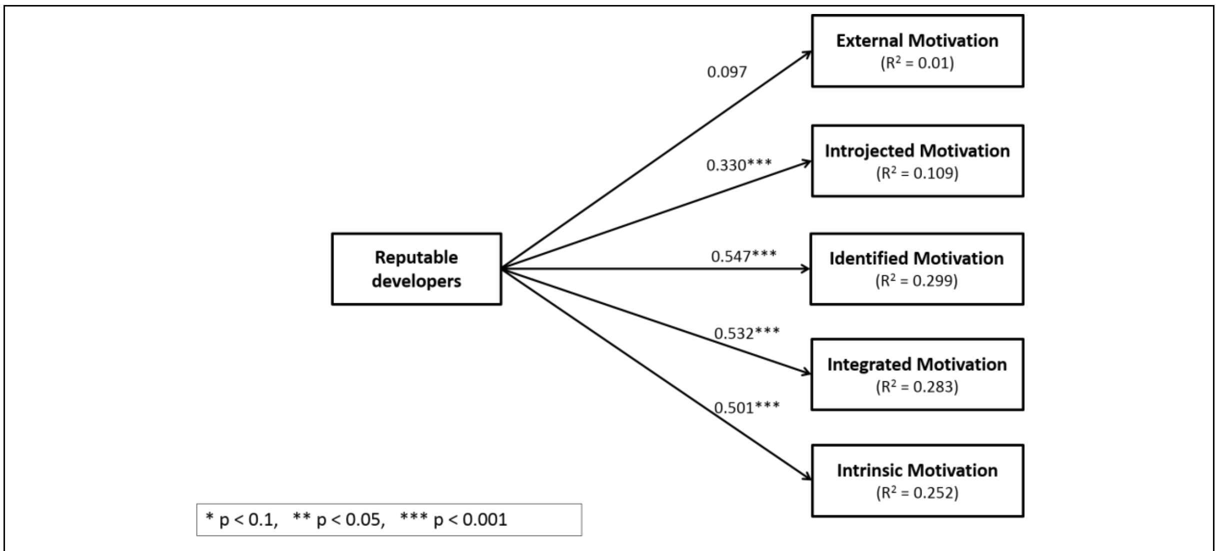


Figure 18. PLS evaluation results in Paper V

In summary, **Paper V** theorizes and evaluates the positive effects which collaboration initiatives with reputable developers have on FLOSS developers' motivation to contribute to a project. The evaluation results suggest that collaboration initiatives with reputable developers explain to considerable degrees the existence of self-determined motivation forms and have a strong positive effect on their degree. Conversely, such initiatives explain FLOSS developers' externally regulated motivation forms only marginally although they provide a considerable stimulus to FLOSS developers' introjected motivation.

4.3.3 Paper VI[9]

Paper VI advances the research of **Paper V**. Specifically, **Paper VI** theorizes that the presence of reputable developers enhances FLOSS developers' teamwork productivity by providing cognitive and affective assets to the team. In order to examine these hypothesized effects, **Paper VI** combines a structural- and an individual-centric evaluation approach.

In line with the advice from von Krogh et al. (2012) to shift away from considering FLOSS developers' motivation as the pivotal point for their project commitment, **Paper VI** builds on the social practice view by MacIntyre (1981) and theorizes that having reputable developers on

[9] Schilling A, Laumer S, Weitzel T. In Goods We Trust - Are OSS Teams With Reputable Developers More Productive?

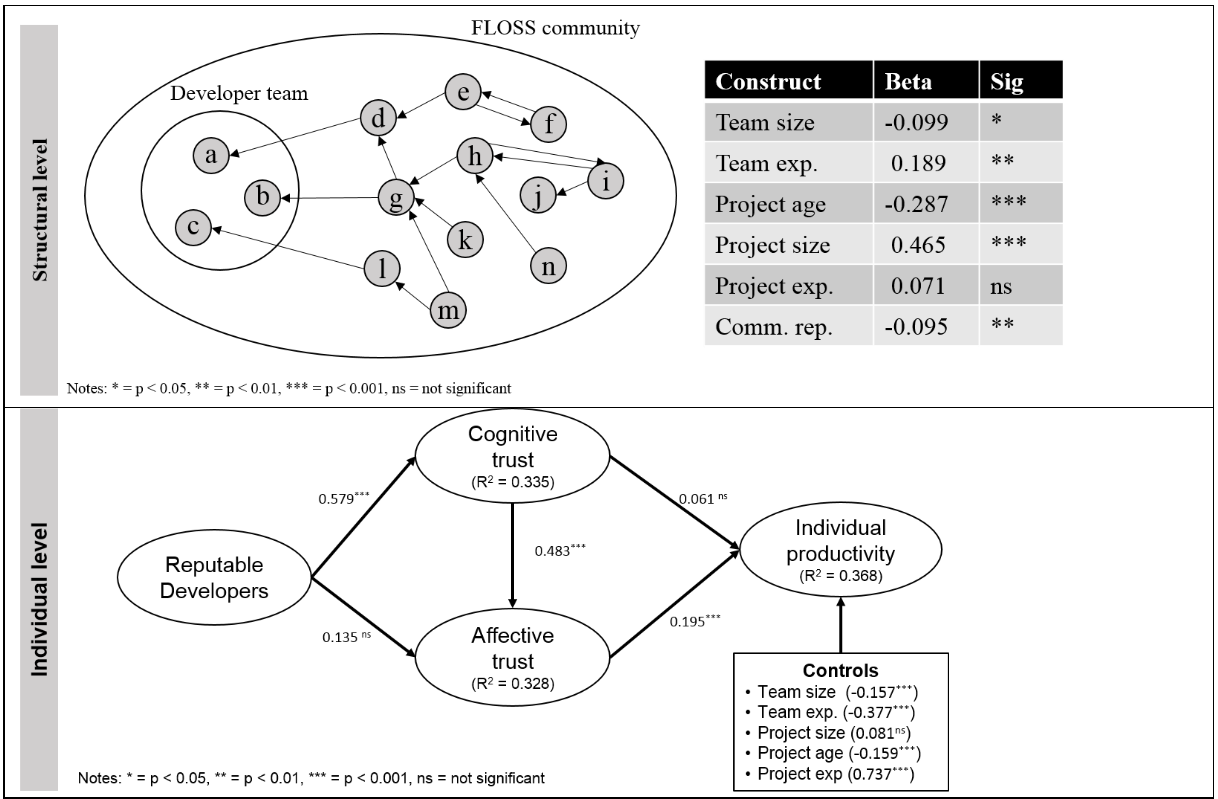
the team enhances FLOSS teams’ productivity twofold. On the one hand, **Paper VI** theorizes that reputable developers provide cognitive assets in terms of training and assistance to the FLOSS team due to their rich contribution experience. On the other hand, it is expected that their deep internalization of the FLOSS culture enables reputable developers to foster feelings of belongingness among team members by verbalizing shared goals. To evaluate the expected effects, an empirical evaluation of 749 FLOSS teams was performed (see Study IV).

The results of the empirical evaluation provide evidence that the experience of the FLOSS team as well as the project size and project age have a strong stimulating effect on FLOSS teams’ productivity. In line with the theorized effects, the results of the empirical evaluation suggests that reputable developers have a positive effect on the FLOSS teams’ productivity, however, this effect is only marginal.

To examine the reasons for this weak productivity gain, a dedicated post-hoc analysis was performed in **Paper VI.** This post-hoc analysis builds on the trust framework by McAllister (1995) and proposes that reputable FLOSS developers foster the development of cognitive as well as affective trust among team members which in turn stimulate their working efforts. To evaluate the hypothesized relationships, a dedicated evaluation of 80 FLOSS developers was performed (see Study V).

The results of this post-hoc analysis reveal that the involvement of reputable developers increases only team members’ level of cognitive trust in the FLOSS team directly. However, it is members’ level of affective trust towards the team members which directly fosters their working efforts. In addition to these effects, project experience has a strong positive effect on FLOSS developers’ individual productivity. In contrast, members’ team experience negatively affects their individual productivity. Furthermore, team size and project age have a moderate negative effect on FLOSS developers’ individual productivity. Conversely, project size has no significant effect on FLOSS developers’ individual productivity.

In summary, **Paper VI** theorizes and empirically evaluates the positive effects which reputable developers have on the collective as well as on the individual productivity of FLOSS developers. In particular, **Paper VI** provides evidence that reputable developers enhance teamwork productivity only marginally. Based on an individual centric post-hoc analysis, **Paper VI** provides a possible explanation for this effect: reputable developers only enhance developers’ level of cognitive trust towards their team. However, it is FLOSS developers’ level of affective trust in the team which directly fosters their working efforts. Figure 19 summarizes the results and the interrelation of the performed structural- and individual-centric evaluation.


Structural level
FLOSS community
Developer team
a
b
c
d
e
f
g
h
i
j
k
l
m
n
Notes: * = p < 0.05, ** = p < 0.01, *** = p < 0.001, ns = not significant
Individual level
Cognitive trust (R² = 0.335)
Reputable Developers
Affective trust (R² = 0.328)
Individual productivity (R² = 0.368)
0.579***
0.135 ns
0.483***
0.061 ns
0.195***
Controls
• Team size (-0.157***)
• Team exp. (-0.377***)
• Project size (0.081ns)
• Project age (-0.159***)
• Project exp (0.737***)
Notes: * = p < 0.05, ** = p < 0.01, *** = p < 0.001, ns = not significant

Construct	Beta	Sig
Team size	-0.099	*
Team exp.	0.189	**
Project age	-0.287	***
Project size	0.465	***
Project exp.	0.071	ns
Comm. rep.	-0.095	**

Figure 19. Structural- and individual evaluation results in Paper VI

4.4 Retention

The final chapter of this dissertation examines potential means to increase the project permanence of newcomers. Based on FLOSS literature underscoring the importance of knowledge building in retaining newcomers in projects (Fang and Neufeld 2009, Singh et al. 2011b, David and Shapiro 2008), the use of mentoring is evaluated.

4.4.1 Paper VII[10]

Paper VII examines strategies enhancing FLOSS developers' project permanence by assisting their knowledge building and socialization in the particular project. Therefore, **Paper VII** proposes and evaluates the use of mentoring as a viable education and retention strategy for FLOSS projects.

Mentoring describes a dyadic teaching method in which an experienced professional, the mentor, provides technical assistance and psychological support to an inexperienced individual, the protégé, (Kram 1985). This intense one-on-one relationship helps transfer tacit knowledge and increases protégés' work satisfaction and their intention to continue (Hale 2000, Brashear et al. 2006). With respect to these positive experiences within the organizational domain, **Paper VII** theorizes that mentoring also provides a viable education and retention strategy for FLOSS projects. In particular, it is proposed that mentoring fosters newcomers' project

[10] Schilling A, Laumer S (2012) Learning to Remain - Evaluating the Use of Mentoring for the Retention of FLOSS Developers. *Proceedings of the 20th European Conference on Information System (ECIS), Barcelona, Spain.*

permanence by enhancing knowledge building and creating interpersonal bonds with team members. An evaluation of 91 mentored and non-mentored newcomers to the KDE was performed to evaluate the supposed relationships (see Study VI).

The evaluation results in **Paper VII** support the theorized relationships. A group comparison of the levels of knowledge building among mentored and non-mentored project novices reveals that mentored novices achieved significantly higher learning states than non-mentored newcomers after a particular period of time. Moreover, a Cox proportional hazard regression shows not only a significant mediation effect between newcomers' acquired level of project knowledge and their project permanence, but also a strong direct association between mentoring and newcomers' project permanence. The evaluation also reveals that project age has a weak positive effect on newcomers' project permanence, the number of developers has a weak negative effect on it, and project size has no significant effect on newcomers' project permanence. Figure 20 illustrates the hypothesized relationships and results of the performed evaluation.

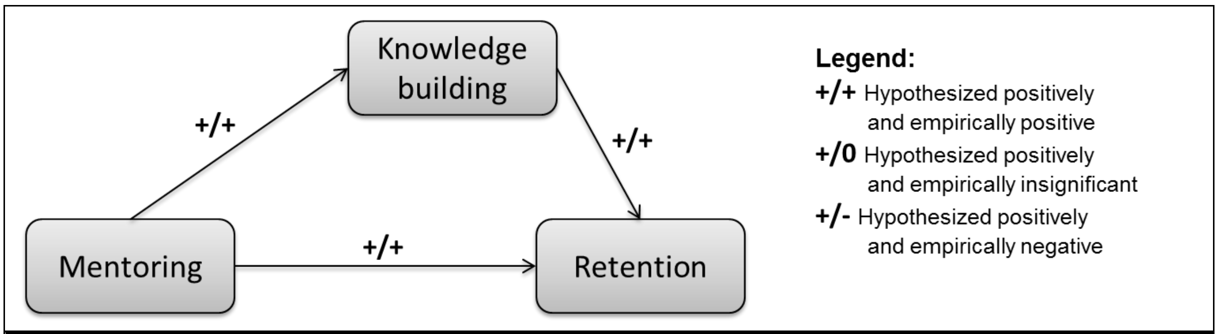

Knowledge building
+/+
+/+
Mentoring
+/+
Retention
Legend:
+/+ Hypothesized positively and empirically positive
+/0 Hypothesized positively and empirically insignificant
+/- Hypothesized positively and empirically negative

Figure 20. Evaluation results in Paper VII

In summary, **Paper VII** theorizes and empirically supports the use of mentoring as an education and as a retention strategy for FLOSS projects. The empirical evaluation suggests that newcomers to FLOSS projects who have been mentored acquire more knowledge, which in turn increased their retention behavior. In addition, the evaluation results suggest that mentoring has a direct positive effect on FLOSS developers' project permanence.

5 Limitations

The theorizing and evaluations in the seven research papers constituting the cumulative dissertation are subject to limitations, which are outlined in this section.

One limitation concerns the literature review in **Paper I**, which focused only on articles from selected top-of-the-class research journals. Articles published in other journals or in the context of conferences or books were not considered. Although the screened journals were selected from the AIS Senior Basket based on the JAIS Global Journal Ranking (Romans and Curtis 2004), it cannot be ruled out that relevant articles from other publication outlets were not considered.

Furthermore, the quantitative evaluations in **Paper III - VII** were all performed within KDE. Although KDE comprises of a wide variety of FLOSS projects, this concentration limits the ability to generalize the evaluation results. Particularly, KDE projects could differ to other FLOSS projects with regard to the team-focused collaboration of its members. Research by Howison and Crowston (2014), for example, suggests that FLOSS developers commonly work on their own, whereas previous studies in the context of KDE show a high level of collaboration among developers (Kuk 2006, Adams et al. 2009). Moreover, due to the code review process in KDE, especially newcomers need to coordinate with the maintainer of the particular module or project to get their code integrated into the project codebase. With respect to this collaboration, it could be argued that KDE projects show more similarity with virtual teams in organizations than with other FLOSS projects.

In addition, the theorizing and evaluation of FLOSS developers' project integration in **Paper IV** and **Paper VI** only considered productive teamwork, even though productive teamwork is only one of several relevant behavioral outcomes which impact effective project integration. What makes FLOSS developers' commitment special, is that it is inherently interrelated with other favorable behavioral outcomes such as learning and innovating (von Krogh et al. 2003). For example, the innovation process in FLOSS projects is not a fire-and-forget activity, but rather requires iterative refinement. This iterative refinement process is even manifested in one of the core principles of FLOSS development which is to '*release early and often*' (Raymond 1999, p. 7). Thus, FLOSS developers' productive interplay should be considered a necessary but not sufficient element for FLOSS developers' project integration.

Furthermore, the proposed multi-dimensional conceptualization of FLOSS developers' geographic dispersion in **Paper IV** is only one of several ways to define geographic dispersion. Although defining geographic dispersion in terms of spatial, temporal, and cultural distances is based on organizational literature, it is neither a complete nor an absolute conceptualization. For example, the chosen conceptualization does not consider any form of configurational aspects which also influences the effects of geographic dispersion in FLOSS projects (O'Leary and Cummings 2007).

A particular conceptual constraint concerning the combination of structural- and individual-centric research approach in **Paper VI** are the non-overlapping study samples. Although, KDE projects are alike in terms of their coding language and KDE-wide development guidelines (KDE Techbase 2014), it cannot be ruled out that individuals whose behavior was examined from a structural perspective would provide different survey replies than individuals who participated in the individual-centric survey, and vice versa. Moreover, the non-overlapping study samples for the structural- and individual-centric evaluation made it necessary to evaluate FLOSS developers' community reputation in different ways.

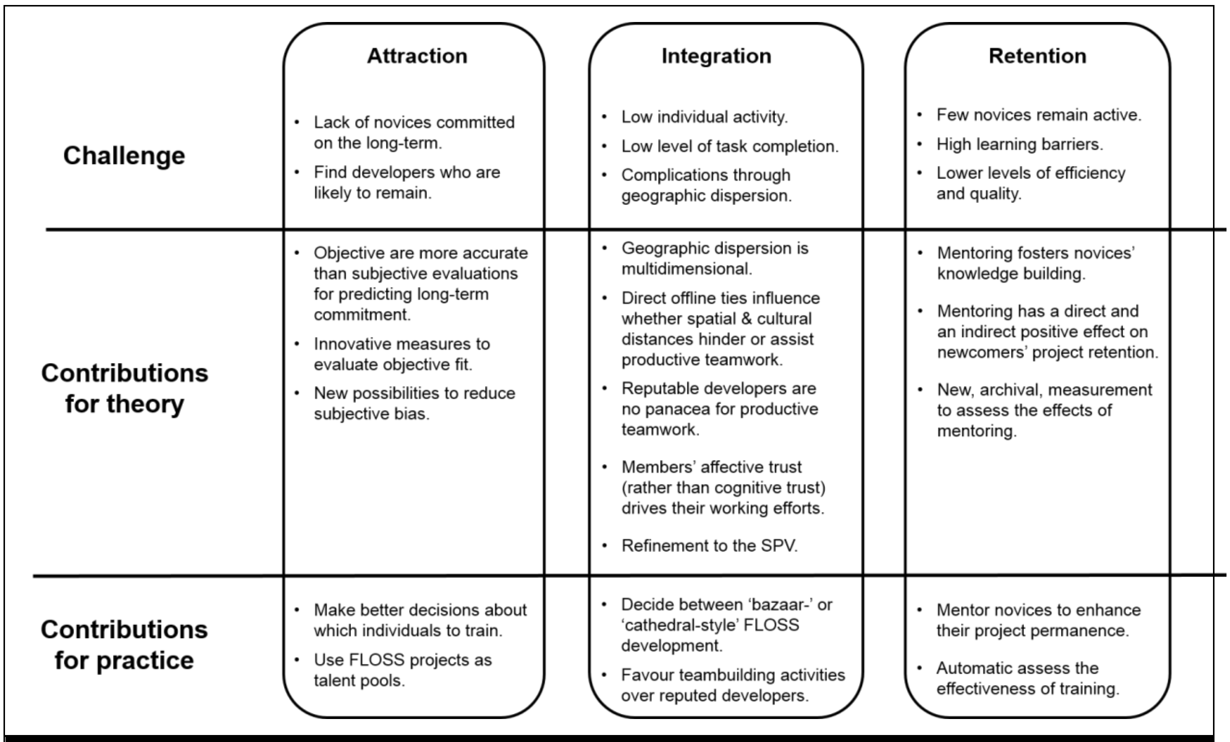
A potential concern regarding the evaluation results in **Paper III**, **Paper V**, and **Paper VII** are the monetary rewards provided to GSoC students. The desire for Google funding could lead students to elaborate their project proposals not based on their personal interests but in order to enhance their chances of getting accepted. In consequence, the needs-supply match between

GSoC candidates and their FLOSS projects in **Paper II** and **Paper III** might be lower than assumed. Similarly, it is possible that GSoC students remain active in FLOSS projects primarily to enhance their chances of getting accepted in the next GSoC event and not due to their experienced knowledge gains, which would bias the evaluation results in **Paper VII**. Likewise, it is possible that the customization of the project proposals and the desire for getting future funding could have affected GSoC students' survey behavior in **Paper V**.

Finally, the employed archival measures in **Paper IV** and **Paper VI** to assess various characteristics of FLOSS developers' geographic dispersion and their reputation in the community were newly developed in the course of this dissertation. Although the measures were developed based on previous evaluations and published in the context of related conferences prior to their use in the papers, they may be subject to conceptual and measurement bias. Furthermore, it has to be acknowledged that the evaluation in **Paper IV** and **Paper VI** focuses only on linear relationships between the dependent and independent variables and ignores curvilinear and exponential effects.

6 Contributions

The seven research articles constituting the cumulative dissertation make important theoretical and managerial contributions to FLOSS as well as the organizational domain. Figure 21 summarizes the key contributions of the research papers regarding developer management in FLOSS projects. The following two subsections describe these key contributions in more detail and discuss their implications for research and practice in the FLOSS and the organizational domain.



	Attraction	Integration	Retention
Challenge	• Lack of novices committed on the long-term. • Find developers who are likely to remain.	• Low individual activity. • Low level of task completion. • Complications through geographic dispersion.	• Few novices remain active. • High learning barriers. • Lower levels of efficiency and quality.
Contributions for theory	• Objective are more accurate than subjective evaluations for predicting long-term commitment. • Innovative measures to evaluate objective fit. • New possibilities to reduce subjective bias.	• Geographic dispersion is multidimensional. • Direct offline ties influence whether spatial & cultural distances hinder or assist productive teamwork. • Reputable developers are no panacea for productive teamwork. • Members' affective trust (rather than cognitive trust) drives their working efforts. • Refinement to the SPV.	• Mentoring fosters novices' knowledge building. • Mentoring has a direct and an indirect positive effect on newcomers' project retention. • New, archival, measurement to assess the effects of mentoring.
Contributions for practice	• Make better decisions about which individuals to train. • Use FLOSS projects as talent pools.	• Decide between 'bazaar-' or 'cathedral-style' FLOSS development. • Favour teambuilding activities over reputed developers.	• Mentor novices to enhance their project permanence. • Automatic assess the effectiveness of training.

Figure 21: Key contributions of the dissertation

6.1 Contributions to Theory

The literature review in **Paper I** identifies potential avenues for future FLOSS research. One key implication is the need to combine various evaluation levels to understand individual project behavior. FLOSS developers are exposed to various influences on the project, team and individual level, which should be considered simultaneously in order to fully understand their behavior. However, very few of the examined articles actually consider more than one concrete research perspective. Nevertheless, motivational and behavioral theories indicate that multiple research aspects should be considered simultaneously to derive a comprehensive understanding of FLOSS developers' project behavior.

Another general implication for future FLOSS research is to rely on dedicated measures in order to identify the distinct effects of the particular aspects. Many of the articles examined in the literature review rely on ambiguous measures such as 'team size', which does not account for high fluctuation among developers. This makes it impossible to tease out distinct lessons for attracting and retaining FLOSS developers.

In addition to these general recommendation for future FLOSS research, the literature review identifies specific opportunities for future research in each of the three key areas for developer management. The following subsections outline the identified research opportunities and how the remaining research papers of the dissertation addressed these research gaps.

6.1.1 Attraction

The literature review in **Paper I** identifies the need for future research on attracting FLOSS developers which combines individual, relational, and project-related factors. In order to address this need **Paper II** and **Paper III** bring these three aspects together. Moreover, the research approach in **Paper II** and **Paper III** distinguishes itself from existing studies on attracting FLOSS developers by not focusing solely on the attraction process, but rather by identifying those individuals worth attracting. As is the case in target advertising, **Paper II** and **Paper III** provide a first step toward improve efforts to attract developers to FLOSS projects by first identifying individuals worth attracting.

In order to understand how to identify developers worth attracting, **Paper II** proposes transferring the two organizational concepts P-J and P-T fit onto the FLOSS domain. In contrast to previous research on attracting developers, the proposed concepts do not consider right and wrong characteristics but instead are based on the idea that it is the congruence between individuals' needs and abilities which need to fit to the particular 'supply and demand' of the project.

Paper III extends this theoretic foundation by also considering the way in which the fit is assessed, deriving concrete measures for evaluating the two types of fit, and through practically evaluating the fit concepts within the FLOSS domain. This evaluation suggests that the objective assessment of FLOSS developers' P-J and P-T fit much more accurately predicts project permanence than their subjective fit assessment. This finding contributes twofold to FLOSS research. First, it highlights the relevance of objective evaluation criteria over individuals' perceptions for assessing newcomers' project permanence in FLOSS projects. Specifically, future research should elaborate further on the quantifying aspects of FLOSS developers' objective fit and evaluate if it takes the form of a discrete or a continuous variable. Second, the evaluation results underscore the importance of considering both individual and relational compatibility to understand sustained project behavior. Thus, future research should not consider only one of these aspects as it could be insufficient criteria for attracting developers to FLOSS projects.

Moreover, **Paper III** contributes to FLOSS literature by evaluating concrete measures for the various aspects of P-J and P-T fit. The relevance of the requirements-ability match between developers and FLOSS projects is in line with the results of previous research which considers developers' learning state a key factor for their project behavior (Singh et al. 2011b). Moreover, the evaluation results suggest that the sheer quantity of FLOSS developers' academic education is an unreliable measure of their abilities. This finding is in line with organizational literature which recommends considering the quality and not the quantity of candidates' education. In addition, the evaluation results complement previous research by Qureshi and Fang (2010) by underscoring the importance of supplementary fit between newcomers and the existing team. Moreover, the insignificant effect of individuals' complementary fit could indicate that complementary characteristics play a less relevant role for retaining FLOSS developers. Depending on whether future research confirms this insignificant relationship, this could indicate that members' differences from one another are not relevant to their ongoing project commitment.

The evaluation of P-J and P-T fit within the FLOSS domain in **Paper II** and **Paper III** provide also implications for organizational literature. Specifically, the derivation of objective measures for assessing P-J and P-T fit contribute to organizational literature. One particular contribution to organizational research concerns the derived objective assessment approach. With regard to researchers' advice to consider knowledge workers as volunteers, the evaluation results highlight the use of objective evaluation criteria for predicting sustained project commitment (Drucker 2002). Thus, the evaluation of the two fit concepts in **Paper II** and **Paper III** can provide a first step for creating new measures for assessing P-J and P-T fit within organizations, which may be applied in team staffing and recruitment decisions. In particular, the results in **Paper III** highlight the relevance of examining if the derived objective evaluation criteria also outperform subjective criteria in the organizational domain.

6.1.2 Integration

The results of the cumulative dissertation advance FLOSS literature on developers' project integration in two distinct areas. First, **Paper III** addresses the research gap identified in **Paper I** regarding the role of relational factors by examining how FLOSS developers' offline context affects their online collaboration. Specifically, **Paper III** examines if FLOSS developers' geographic dispersion negatively affects their collaboration and if direct offline interactions help them to overcome these problems. Moreover, **Paper IV** and **Paper V** contribute to FLOSS literature by examining the individual and collective effects of including reputable developers in FLOSS projects from various angles. By doing so, the two papers address an opportunity outlined in **Paper I** which is to employ a cross-perspective analysis to understand how certain phenomena interrelate with individual and collective behavior. The following paragraphs detail the concrete contributions of these articles to FLOSS and organizational literature.

The conceptualization and evaluation of geographic dispersion as a multi-dimensional construct in **Paper IV** contributes to FLOSS research in three ways. First, the multi-dimensional conceptualization of geographic dispersion can be used to build a comprehensive understanding of the influence of geographic dispersion on FLOSS development by bringing together the isolated and fragmented results of previous studies (Hu et al. 2012, Daniel et al. 2013, Colazo and Fang 2010). The results of the empirical evaluation support this multidimensional conceptualization by showing that spatial, temporal, and cultural distances explain FLOSS developers' productive interplay to considerable degrees. Second, consistent with conceptual research by Ågerfalk et al. (2005), the evaluation results reveal that spatial and cultural distances per se are neither a gain for nor a burden on effective teamwork in FLOSS projects. Rather it depends on the existence of direct interactions between FLOSS developers. If FLOSS developers have no direct offline interactions, the negative aspects of their spatial and cultural distances prevail. In this regard, the evaluation results back previous studies which suggest that spatial (Hu et al. 2012) and cultural distance (Daniel et al. 2013) hinder productive teamwork in FLOSS projects. However, as soon as there is little offline interaction among FLOSS developers, their spatial and cultural distances facilitate their productive teamwork. Irrespective of FLOSS developers' offline interactions, however, temporal distances between them complicate their productive interplay. This finding is contrary to previous research by Colazo and Fang (2010) which suggests that temporal distances increase FLOSS developers' productivity. A possible explanation for this discrepancy could lie in the different ways temporal distances among FLOSS developers is measured. While Colazo and Fang (2010) consider only differences in developers' starting time at the FLOSS project, the proposed measure in **Paper IV** assesses the actual overlap in FLOSS developers' working hours. Thus, the proposed measure is especially appropriate for the skewed work distributions which are commonly found in FLOSS projects (Toral et al. 2010). Alternatively, it could be the case that the FLOSS projects studied by Colazo and Fang (2010) are indeed more effective in coping with the negative effects of temporal distances than the KDE projects in Study II. This triggers the question for further research why the projects studied by Colazo and Fang (2010) cope better with temporal distances than KDE projects. Thirdly, **Paper IV** contributes to FLOSS literature by considering both online and offline interactions among FLOSS developers. Previous FLOSS research has focused primarily on FLOSS developers' online interactions. Although Crowston et al. (2007a) provide evidence that offline meetings are an important complement to collaborations in FLOSS projects, **Paper I** suggests that empirical studies in the FLOSS domain have neglected this aspect to date. The results presented in **Paper IV** highlight the interrelations between the FLOSS developers' offline and online contexts and call for further research to understand the interrelations of these two domains.

In addition to FLOSS literature, **Paper IV** provides several implications to organizational research. The proposed multi-dimensional conceptualization addresses a pivotal shortcoming in organizational literature which is the uni-dimensional and dichotomous differentiation of team members' geographic dispersion (Cummings et al. 2009, Hinds and Mortensen 2005, O'Leary and Cummings 2007). Therefore, the multi-dimensional conceptualization helps bringing together the isolated and fragmented findings of previous evaluations in the organizational domain and building an integrated understanding for the nuanced effects of geographic dispersion. Moreover, **Paper IV** contributes to organizational literature by highlighting the ambivalent role of spatial and cultural distances, which in turn provides an explanation for the mixed findings in organizational literature on the effects of these distance forms (O'Leary and Cummings 2007, Hinds and Mortensen 2005, Cummings et al. 2009). In addition, the evaluation results of **Paper IV** contribute to teamwork research by examining the concrete situations in which offline meetings lead to the highest value added. Thereby, the evaluation results support the advice of Siebdrat et al. (2009) to organize offline meetings especially when members are new to the team. Moreover, the evaluation suggests that offline meetings are not always a gain for teamwork productivity. In case of experienced team members, an increasing number of offline meetings even decreases their overall team productivity. Finally, **Paper IV** complements organizational literature by highlighting the need to consider both team members' online and offline interaction contexts to understand fully their behavior in the online context. Previous studies which combined these two contexts, like the work of Kirkman et al. (2004), were rather exceptional. In line with the work of Zhang and Venkatesh (2013), the performed evaluation highlight the interrelation between team members' offline and online interactions with each other.

In addition to the role of geographic dispersion, the cumulative dissertation contributes to FLOSS research by examining the positive effects which reputable developers have on team members' individual and collective productivity.

Paper VI makes a central contribution to FLOSS literature by performing a multi-level evaluation approach to examine if and how reputable developers foster productive interplay among FLOSS developers. Specifically, the multi-level evaluation approach starts with an empirical evaluation of the positive effects of reputable developers with 745 FLOSS teams. This evaluation shows that the presence of reputable developers stimulates FLOSS teams' productivity; however it does so only marginally. In order to understand this effect better, an individual centric post-hoc analysis was performed. This study revealed that a possible explanation for the marginal effect is that reputable developers only enhance members' level of cognitive trust in the team, but this type of trust has no direct effect on their individual working efforts. Instead, it is members' sense of belonging to the developer team which increases their productivity directly.

Intuitively these results appear to be contradictory to the conclusion of **Paper V** which is that collaboration initiatives increase FLOSS developers' self-determined contribution motives, which in turn are thought to lead to higher working efforts. This incompatibility, mirrors the results of an evaluation by Ke and Zhang (2010) which suggest that integrated motivation per se decreases FLOSS developers' task performance unless it is accompanied by the satisfaction of individuals' needs for autonomy, competence, and belongingness. In light of this finding, FLOSS developers' motivation may be only half of the picture, while their relationships to the other members is the other. Applied to the particular context, this could mean that the involvement of reputable developers provides the basis for increased individual efforts, but in order to unleash productive teamwork, members must feel that they belong to the developer team. Considered together, the evaluation results of **Paper V** and **Paper VI** support the general reservation of von Krogh et al. (2012) to use SDT as the theoretical basis for a comprehensive understanding of FLOSS developers' project behavior.

In contrast to the short-term orientation of SDT, the social practice view (MacIntyre 1981) provides a much broader theoretic foundation for understanding the role of individuals' past and current contribution motivation as well as their relationship to other developers. Thereby, the evaluation results in **Paper VI** not only support the social practice view but they propose some refinements to its original application in the FLOSS context (von Krogh et al. 2012). Specifically, the evaluation results in **Paper VI** challenge the idea that only one form of internal good is derived through pursuing a social practice. This is consistent with the work of MacIntyre (1981) which differentiates between at least two basic types of internal goods. The first type of internal good (i.e. the performance itself and the created product) can be derived by pursuing a social practice following the standards of excellence. In contrast, the second type of internal good requires individuals to self-reflect upon their work. This type of internal good concerns the'*related kind of life*' (MacIntyre 1981, p. 190). In light to this basic differentiation the individual focused evaluation results in **Paper VI** are plausible as they suggest that the existence of reputable developers only helps in creating one kind of internal good. In contrast, the second type of internal good, which kindles FLOSS developers' commitment, can only be derived through their self-reflection and by feeling emotionally connected to fellow team members.

An important insight that can be derived from **Paper VI** for FLOSS as well as organizational research is that team and project characteristics can have opposite effects on individual and collective behavior. One particular characteristic for which this applies is members' level of team experience. While teams with members who have worked with each other in the past are more productive, the opposite applies to individual behavior. In fact, the individual centric post-hoc analysis in **Paper VI** shows that individuals are less productive the longer they have worked with each other. An explanation for this effect could be that FLOSS developers favor their companionship over the project goals the longer they work together. As a result, the developers contribute less to the project but remain supportive and thus help other developers to become productive. From an aggregated perspective, such effect could be completely covered under the productivity gains of new developers. Thus, future research in the FLOSS and organizational domain should explicitly examine the effects of particular factors on both individual and collective behavior rather than study only one and suppose that the other is consistent with the examined one.

Moreover, the evaluation results in **Paper VI** encourage the use of the social practice view within the organizational domain. Although organizations are considered in the social practice view as classic industry corporations which govern human behavior, the picture of the workplace changed considerably. Especially with regard to the similarities between FLOSS projects and virtual teams in organizations, the performed evaluation results support the call of Beadle (2006) that the social practice view provides valuable grounds for a comprehensive understanding for employees' well-being and productivity. In contrast to motivation theories which focus on the immediate outcomes associated with individuals' behavior, MacIntyre (1981)'s theory takes a much broader view on individuals underlying ethical beliefs and long term goals. With respect to this broader theoretical foundation, MacIntyre (1981)'s social practice view could be especially valuable in terms of deriving new insights into how employees should be embedded into organizations so that they work productively and maintain a healthy work-life balance.

6.1.3 Retention

Based on the identified need in **Paper I** for more dedicated research on FLOSS developers' retention which examines the interaction of individual and team-level factors, **Paper VII** evaluates the use of mentoring as a viable retention strategy for new developers. In doing so, **Paper VII** extends previous FLOSS research which identified high learning barriers as a key inhibitor for newcomers' project permanence (Adams et al. 2009, Singh et al. 2011b). The performed evaluation in **Paper VII** support this and show that newcomers' knowledge building process is an important driver for their sustained commitment in the FLOSS project.

Moreover, the study results in **Paper VII** support mentoring as a viable education and retention strategy in FLOSS projects. In particular, the evaluation results suggest not only that mentoring helps newcomers to acquire project-related knowledge, which in turn increases their project permanence, but also that there is an additional direct positive effect on newcomers' project permanence. With respect to organizational literature (Eby and Lockwood 2005, Kram 1985), it seems likely that this direct effect can be attributed to the strong relational bond which not only fosters the transfer of knowledge but also creates a strong interpersonal relationship between the mentor and the protégé.

Finally, the examination of mentoring as a viable education and retention strategy for FLOSS projects also contributes to organization literature. Specifically, the evaluation results in **Paper VII** address the call of Parise and Forret (2008) for further research on the effects of mentoring on protégés' continuance behavior. Moreover, the derived evaluation results support previous research by Eby and Lockwood (2005) and Lentz and Allen (2009), which suggest that mentoring relationships are not only effective in conveying new knowledge but also building friendship relationships between the mentor and the protégé. In addition, the archival measurement in **Paper VII** provides a foundation for organizational research to not rely on protégés' subjective perceptions, which are often found inaccurate to assess the outcomes of mentoring relationships (Eby et al. 2004).

6.2 Contributions to Practice

6.2.1 Attraction

The research results in **Paper II** and **Paper III** have various practical implications for FLOSS projects. A concrete managerial recommendation of these papers for the process of selecting

GSoC students in KDE is to integrate the proposed objective measures for assessing P-J and P-T fit. The evaluation results in **Paper III** clearly suggest that objective measures are much more accurate predictors for sustained project commitment than the currently employed subjective evaluation process. Beyond GSoC, the proposed measures can help FLOSS projects to concentrate their training efforts on those newcomers who are likely to remain committed instead of newcomers with only a short term interest in the project. Moreover, the customized fit concepts and measures proposed in **Paper II** and **Paper III** can be used by FLOSS mangers to control the fitness of their developer base and identify the need to reach out for new developers and foster the retention of existing developers at an early stage.

In addition, **Paper II** and **Paper III** have managerial implications for organizations. Most importantly, the proposed measures in these papers provide a first step for organizations to design and employ new strategies for talent identification (Drucker 2002). Moreover, the proposed measures can be used to derive entirely new talent acquisition strategies for software companies, such as using FLOSS projects as a talent pool from which to identify recruitment candidates.

6.2.2 Integration

Regarding developers' project integration, **Paper IV** provides concrete managerial implications for enhancing team members' productive interplay in FLOSS projects. Most importantly, the evaluation results suggest that managers should not consider the effective interplay of the involved developers for granted but rather contingent upon their spatial, temporal, and cultural distances. In addition, **Paper IV** highlights the relevance of direct offline interactions among FLOSS developers for identifying the most suited development approach for FLOSS projects. If offline meetings are not possible, FLOSS projects are better off bringing together developers with little spatial, temporal, and cultural distances to each other. For such endeavors, a 'cathedral-style' development approach could be most appropriate. In such development approach, code is developed in private and only published with each software release (Raymond 1999). However, if it is possible to arrange offline meetings, project managers should favor creating a spatially and culturally dispersed developer base, which can be typically achieved through a 'bazaar-style' development approach. In addition, FLOSS projects can combine both approaches such as a 'cathedral-style' coordination between the project leader(s) and the maintainer(s) of the particular modules and a 'bazaar-style' development approach between regular developers and the maintainer(s).

In light of the common escalation and underperformance of software projects in the organizational domain (Keil and Mann 2000, Solomon 2010), the examination of productive teamwork within FLOSS projects also provides various practical lessons for the staffing and management of geographically dispersed teams in organizations. First, **Paper IV** provides evidence that organizations should consider members' ability to meet offline when staffing individuals for such teams. If offline meetings are not possible, managers should combine members with little spatial and cultural distances between them. However, if offline meetings are possible project managers should adopt the complete opposite approach and bring together members with high spatial and cultural distances. This is because these distances transform into productivity gains as soon as there is little direct offline contact between members. For all team configurations, however, managers should minimize temporal distances among team members, as they cause considerable harm to their effective interplay.

Paper V and **Paper VI** provide managerial advice for FLOSS projects regarding the supposed positive effects associated with the presence of reputable developers. Although, **Paper V** indicates that collaboration initiatives with reputable developers foster self-determined motivation forms among FLOSS developers, the multi-level research performed in **Paper VI** suggests that reputable developers should not be considered a panacea for productive teamwork. In particular, the results in **Paper VI** indicate that the presence of reputable developers can be considered a relevant but not sufficient element for productive teamwork in FLOSS projects. In fact, reputable developers only enhance members' level of cognitive trust in each other. However, this form of trust does not directly affect individuals' work efforts. Instead, it is their level of affective trust, their sense of belonging, which foster their work efforts. Thus, managers of FLOSS projects should favor dedicated team building activities which strengthen members' sense of belonging, like arranging release parties or social events, over bringing in reputable developers.

Paper VI also provides managerial implications to organizations in light of their broad use of external (e.g. Linkedin) and in-house (e.g. IBM Connections) scoring and evaluation systems. In particular, the evaluation results warn managers of relying too much on such scoring systems. Although the results of **Paper VI** indicate that reputable developers enhance teamwork productivity, this positive effect is only marginal. In comparison, Erden et al. (2014) provide evidence that the presence of reputable individuals greatly increases the equity price of the particular firm. However, while there are only few levers for firms to get attention among financial investors, **Paper VI** indicates that the productive interplay in software development teams can be fostered more effectively through dedicated team building activities which increase team members' level of affective trust in each other than by bringing in reputable developers.

6.2.3 Retention

The evaluation of mentoring as an education and knowledge building strategy for FLOSS projects has substantial implications to FLOSS practice. Previous evaluations draw a rather alarming picture of contribution behavior in FLOSS projects. Specifically Singh et al. (2011b) provide evidence that most contributors do not advance in their learning state. This is supported by Adams et al. (2009) who show that it can even take up to 60 weeks before FLOSS developers become effective. Though, many newcomers often leave the project before achieving such progress. In this situation, the evaluation results in **Paper VII** show that FLOSS projects can take active means retain their developers longer. In particular **Paper VII** supports mentoring as an effective means to assist novices' knowledge building process. An important consequence of this positive effect is that FLOSS developers show not only higher commitment regarding code development but also higher levels of project permanence. Beside the enhanced knowledge transfer, **Paper VII** suggests that mentoring also has a direct positive effect on newcomers' project permanence.

Considering organizations' reliance on education activities, **Paper VII** also contributes to the organizational domain. On the one hand, it supports managerial use of mentoring initiatives to enhance employees' education and foster their long-term project commitment. Moreover, the proposed measures provide new grounds for organizations to assess the knowledge gains of their employees. In particular, such archival measures help automatically assessing and comparing the learning gains from educational activities in organizations, which is considered a central for corporations (Gartner Inc. 2007).

7 Future Research

The results and implications of this dissertation provide new insights into developer management in FLOSS projects. Nevertheless, many questions about managing FLOSS developers remain open. The following paragraphs delineate potential avenues of future research on the various aspects of developer management in FLOSS projects. First, two general directions for future research are presented. Then, the specific directions for further elaboration on attracting, integrating, and retaining developers in FLOSS projects are delineated.

One central recommendation for future FLOSS research concerns the examination of the derived relationships and management suggestions in the context of FLOSS projects which are not related to KDE. As outlined in Section 5, a central limitation of this dissertation is its focus on KDE projects. Thus, further research is necessary which examines the derived conclusions with a well-diversified and empirically rich project sample. In this context, future research can also control for contextual differences such as differences in governance styles or programming languages.

The second general topic for future research is the development and evaluation of an integrated developer management strategy for FLOSS projects. The cumulative dissertation draws on the framework for IHRM to build an understanding and derive concrete strategies for developer management in FLOSS projects. Thus, the next step for FLOSS research is to examine the interrelations between the three management areas to build an integrated management approach. Thereby, a concrete question for further research concerns the compatibility or contradictory of means to attract, integrate, and retain FLOSS developers. For example, while the presence of reputable developers could be considered subordinate for integrating developers, it could be considered essential in terms of attracting developers (Hu et al. 2012). Thus, future research should take a holistic perspective to examine the temporal and long-term compatibility of the various theories and means to attract, integrate, and retain FLOSS developers.

Attraction

A particular area for further research on the attraction of FLOSS developers is the refinement and extension of the proposed fit concepts and measures. A key question in this context, is how FLOSS developers' level of complementary fit has no significant effect on their project permanence. Although the evaluation results in **Paper III** suggest no such influence, further research is needed to make a final decision if complementary fit should be considered inferior to supplementary fit for identifying new team members. To examine this particular aspect, further research should examine various measures for assessing candidates’ complementary fit, so that it can be differentiated between insignificant relationships which can be attributed to conceptual or measurement reasons.

Another area in which future research can extend the research in **Paper II** and **Paper III** is by considering favorable behavioral outcomes beyond FLOSS developers’ project permanence. For example, research could examine if the derived fit measures are also appropriate in finding highly innovative candidates. Although organizational research suggests that it is the case, it would be interesting to see if the various fit aspects (in particular complementary fit) and suggested measures differ in their strength to anticipate such behavioral outcomes. Future research should also consider individuals’ position in the overall interaction network in the selection process. Such research could examine if FLOSS projects should focus on finding individuals who are likely to behave favorably in the future, or rather build on the strong herding effect among FLOSS developers (Oh and Jeon 2007) and attract individuals with the highest visibility among other developers to trigger a subsequent influx of other developers.

Finally, an important research field for future research is to derive concrete strategies for actively convincing developers to become active in the FLOSS project. A central constraint which should be considered thereby is that many FLOSS projects are based on voluntary basis and therefore cannot provide pecuniary rewards to developers. Thus, future research has to design and evaluate alternative strategies for attracting developers. Possible tent poles of such as active attraction strategy could be highlighting the involvement of reputable developers (Hu et al. 2012) or the relevance of the project to the overall FLOSS community.

Integration

A key area for future examination into FLOSS developers’ project integration is the extension and refinement of the multi-dimensional consideration of geographic dispersion in FLOSS projects. Future research should also examine the relevance of configurational aspects in FLOSS projects (O'Leary and Cummings 2007). In particular, future research should examine whether there is a difference in FLOSS projects’ exposure to spatial distances if all developers work spatially distant or if it is only one developer who works apart from all others. Future studies on the effects of geographic dispersion should also take account of the actual interrelatedness of FLOSS developers’ project work. In light of the case specific degree of interrelatedness, it seems appropriate for future research to consider interrelatedness rather as a case specific variable than a generic property.

Apart from elaborating on the effects of geographic dispersion, future research should further evaluate the social practice view as a theoretical basis for FLOSS development. The broad stance of the social practice view offers a new way of looking at the effects of interactions among FLOSS developers. This perspective provided the basis for the conclusion that feelings of belongingness are much more important than the level of community reputation of the involved members. Building on this insight, future research could use the social practice view as a theoretical foundation to integrate and synthesize past and future research on FLOSS developers' project behavior. For example, it provides a holistic explanation for the relevance of affective trust (Stewart and Gosain 2006) and the importance of situated learning (Fang and Neufeld 2009). Even more, as proposed by von Krogh et al. (2012) the social practice view can be used in future research to understand the inconsistent findings of previous research which is based on SDT.

In addition to the conceptual elaboration, future research should rely on more advanced forms of evaluation. Specifically, the use of Hierarchical Linear Model (HLM) regression seems to be appropriate. This evaluation technique allows to simultaneously examine nested effects of project, group, and individual characteristics on FLOSS developers' project commitment. Thus, HLM regression helps to separate effects which affect teamwork productivity from those effects which only affect individual productivity. Such multi-level examination is especially valuable in examining the effects of reputable developers' project involvement. This is because, HLM regression provides much better ways to assess the threats to validity posed by the combination of structural and individual-centric research. Finally, HLM regression does not require distinct study samples. Therefore, HLM regression fits much better to the used sampling strategy than linear regression, at least under the assumption that some developers are permanently involved in FLOSS projects while other developers come and go (Setia et al. 2012).

Retention

Future research on retaining FLOSS developers can build on the evaluation results in **Paper VII** to build a nuanced understanding for the possibilities in FLOSS projects to foster developer retention. Thereby, it is a particular question for future research to compare the positive effects of mentoring with those of other teaching means such as developer sprints, which allow also the exchange of non-verbal communication ties (Daft and Lengel 1986). Moreover, the use of collocated training means, such as regional Linux User Groups (Bagozzi and Dholakia 2006), should be compared with virtual training settings.

Another direction for future research on retaining FLOSS developers is to elaborate on the effects of mentoring initiatives. Especially the direct link between newcomers' attendance at mentoring initiatives and their project permanence should be examined more thoroughly. The positive direct relationship between these two constructs supports the theorized positive effects of the social-bond between the mentor and the protégé, while leaving room for alternative explanations. For example, mentored students may achieve higher levels of autonomy through mentorship, which may motivate them to stay with the project. Future research should address this by examining through perceptual measures if the direct relationship between mentoring and FLOSS developers' project permanence is due to their feelings of belongingness or rather due to other reasons.

8 Conclusion

The overall goal of the cumulative dissertation is to derive theoretical concepts and empirical evidence to assist developer management in FLOSS projects. Borrowing from core structures in IHRM, the dissertation divides FLOSS developer management into three core areas:

attracting, integrating, and retaining FLOSS developers. The results of the dissertation provide distinct contributions to each of these management areas.

The dissertation derives archival measures to identify developers who are likely to remain active in the FLOSS project and who are thus worth training. The derived objective measures more accurately predict sustained project commitment than individuals' subjective assessment. Moreover, the dissertation offers two key insights into integrating developers into FLOSS projects. First, the dissertation finds that FLOSS developers' offline ties to each other significantly determine whether their spatial and cultural distances will promote or hinder their effective teamwork. In contrast, temporal distances mitigate productive teamwork regardless of developers' offline ties. Second, the dissertation reveals that the presence of reputable developers in FLOSS projects only adds marginally to productive teamwork. Specifically, the presence of reputable developers only increases the level to which team members' consider their colleagues competent. However, the involvement of reputable developers has no direct effect on team members' feelings of belongingness, which are central to stimulate their work productivity. Finally, the dissertation supports the use of mentoring as an effective means to retain FLOSS developers.

The dissertation thus provides concrete and useful advice in managing FLOSS developers which can help FLOSS projects prosper and avoid another 'Heartbleed'.

9 References

Accenture Inc. (2010) Investment in Open Source Software Set to Rise, Accenture Survey Finds, http://newsroom.accenture.com/article_display.cfm?article_id=5045#rel. Retrieved March 21, 2015

Adams PJ, Capiluppi A, Boldyreff C (2009) Coordination and Productivity Issues in Free Software: The Role of Brooks' Law. *IEEE International Conference on Software Maintenance* (IEEE), 319–328.

Ågerfalk PJ, Fitzgerald B, Holmström H, Lings B, Lundell B, Conchúir EÓ (2005) A Framework for Considering Opportunities and Threats in Distributed Software Development. In *Proceedings of the International Workshop on Distributed Software Development*, 47–61.

Aksulu A, Wade MR (2010) A Comprehensive Review and Synthesis of Open Source Research. *Journal of the Association for Information Systems* 11(11):576-656.

Asay M (2014): Developers Aren't Going to Go for Proprietary Standards. http://readwrite.com/2014/10/17/internet-of-things-open-source-iot-developers, Retrieved April 20, 2015.

Bagozzi RP (2011) Measurement and Meaning in Information Systems and Organizational Research: Methodological and Philosophical Foundations. *Management Information Systems Quarterly* 35(2):261-292.

Bagozzi RP, Dholakia UM (2006) Open Source Software User Communities: A Study of Participation in Linux User Groups. *Management Science* 52(7):1099–1115.

Beadle R (2006) MacIntyre on Virtue and Organization. *Organization Studies* 27(3):323–340.

Berinato, S (2014) Heartbleed, the Branding of a Bug, and the Internet of Things, https://hbr.org/2014/04/heartbleed-the-branding-of-a-bug-and-the-internet-of-things/ Retrieved April 18, 2015.

Bollen, KA (1989) *Structural Equations with Latent Variables*. New York: John Wiley and Sons Ltd.

Brashear TG, Bellenger DN, Boles JS, Barksdale HC (2006) An Exploratory Study of the Relative Effectiveness of Different Types of Sales Force Mentors. *Journal of Personal Selling and Sales Management* 26(1):7–18.

Brin S, Page L (1998) The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems* 30(1-7):107–117.

Carmines EG, Zeller RA (1979) *Reliability and Validity Assessment*. Beverly Hills (CA): Sage Publications

Chatman JA (1989) Matching People and Organizations Selection and Socialization in Public Accounting Firms. *Academy of Management Best Papers Proceedings*, 199–203.

Chengalur-Smith I, Sidorova A, Daniel S (2010) Sustainability of Free/Libre Open Source Projects: A Longitudinal Study. *Journal of the Association for Information Systems* 11(11):657–683.

Chilton MA, Hardgrave BC, Armstrong DJ (2005) Person-Job Cognitive Style Fit for Software Developers: The Effect on Strain and Performance. *Journal of Management Information Systems* 22(2):193-226.

Chin WW (1998) The Partial Least Squares Approach for Structural Equation Modelling. Marcoulides GA, ed. *Modern Methods for Business Research,* 295–336, Mahwah (NJ): Lawrence Erlbaum Associates.

Chin WW, Newsted PR (1999) Structural Equation Modelling Analysis with Small Samples Using Partial Least Squares. Hoyer RH, ed. *Statistical Strategies for Small Sample Research, 307-342,* Thousand Oaks (CA): Sage Publications.

Chou S, He M (2011) The Factors that Affect the Performance of Open Source Software Development - The Perspective of Social Capital and Expertise Integration. *Information Systems Journal* 21(2):195–219.

Colazo J, Fang Y (2009) Impact of License Choice on Open Source Software Development Activity. *Journal of the American Society for Information Science & Technology* 60(5):997–1011.

Colazo JA, Fang Y (2010) Following the Sun: Temporal Dispersion and Performance in Open Source Software Project Teams. *Journal of the Association for Information Systems* 11(12):684–707.

Cox DR (1972) Regression Models and Life-Tables. *Journal of the Royal Statistical Society* 34(2):187-220.

Crowston K, Heckman R, Misiolek N (2010) *Leadership in Self-Managing Virtual Teams*, Syracuse University School of Information Studies.

Crowston K, Howison J, Masango C, Eseryel U (2007a) The Role of Face-to-Face Meetings in Technology-Supported Self-Organizing Distributed Teams. *IEEE Transactions on Professional Communication* 50(3):185–203.

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

Crowston K, Wei K, Howison J, Wiggins A (2012) Free/Libre Open-Source Software Development: What We Know and What We Do Not Know. *ACM Computing Surveys* 44(2):1–35.

Cummings JN, Espinosa JA, Pickering CK (2009) Crossing Spatial and Temporal Boundaries in Globally Distributed Projects: A Relational Model of Coordination Delay. *Information Systems Research* 20(3):420–439.

Daft RL, Lengel RH (1986) Organizational Information Requirements, Media Richness and Structural Design. *Management Science* 32(5):554–571.

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

David PA, Shapiro JS (2008) Community-Based Production of Open-Source Software: What Do We Know About the Developers Who Participate? *Information Economics and Policy* 20(4):364–398.

Deci, EL, Ryan, RM (1985) *Intrinsic Motivation and Self-Determination in Human Behavior*, New York: Plenum Publishing.

Deci EL, Ryan RM (2000) The 'What' and 'Why' of Goal Pursuits Human Needs and the Self-Determination of Behavior. *Psychological Inquiry* 11(4):227–268.

Dixon, R (2004) *Open Source Software Law*, Boston: Artech House.

Drucker PF (2002) They're not Employees, They're People. *Harvard Business Review* 80(2):70–77.

Durumeric Z, Payer M, Paxson V, Kasten J, Adrian D, Halderman JA, Bailey M, Li F, Weaver N, Amann J, Beekman J (2014) The Matter of Heartbleed. *ACM Internet Measurement Conference*, 475–488.

Eby L, Butts M, Lockwood A, Simon SA (2004) Protégés Negative Mentoring Experiences: Construct Development and Nomological Validation. *Personnel Psychology* 57(2):411–447.

Eby LT, Lockwood A (2005) Protégés' and Mentors' Reactions to Participating in Formal Mentoring Programs: A Qualitative Investigation. *Journal of Vocational Behavior* 67(3):441–458.

Edwards JR (1991) Person-Job Fit: A Conceptual Integration, Literature Review, and Methodological Critique. Cooper CL, Robertson IT, eds. *International Review of Industrial and Organizational Psychology*, 283–357, New York: John Wiley and Sons Ltd.

Eisenhardt KM (1989) Building Theories from Case Study Research. *The Academy of Management Review* 14(4):532-550.

Erden Z, Klang D, Sydler R, von Krogh G (2014) How Can We Signal the Value of Our Knowledge? Knowledge-Based Reputation and Its Impact on Firm Performance in Science-Based Industries. *Long Range Planning*, 252-264, Oxford: Pergamon.

Ettrich, M (1996) New Project: Kool Desktop Environment (KDE), https://groups.google.com/forum/#!original/de.comp.os.linux.misc/SDbiV3Iat_s/zv_D_2ctS8sJ. Retrieved April 7, 2015.

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

Finkle, J, Kurane, S (2014) U.S. Hospital Breach Biggest yet to Exploit Heartbleed Bug: Expert, http://www.reuters.com/article/2014/08/20/us-community-health-cybersecurity-idUSKBN0GK0H420140820. Retrieved March 23, 2015.

Fitzgerald B (2006) The Transformation of Open Source Software. *Management Information Systems Quarterly* 30(3):587-598.

Fornell C (1987) A Second Generation of Multivariate Analysis - Classification of Methods and Implications for Marketing Research. Houston MJ, ed. *Review of Marketing, 407-450,* Chicago: American Marketing Association.

Fornell C, Larcker DF (1981) Structural Equation Models With Unobservable Variables and Measurement Error Algebra and Statistics. *Journal of Marketing Research (JMR)* 18(3):382–388.

Forrester Research (2014) Survey Indicates Four Out of Five Developers Now Use Open Source, http://www.zdnet.com/article/survey-indicates-four-out-of-five-developers-now-use-open-source/.

Fox, J, Weisberg, S (2011) *An R Companion to Applied Regression*, 2nd ed., Thousand Oaks (CA): Sage Publications.

Gagné M, Deci EL (2005) Self-Determination Theory and Work Motivation. *Journal of Organizational Behavior* 26(4):331–362.

Gartner Inc. (2007) Gartner EXP Says Organizations Must Evaluate Learning and Training Programs to Gauge Return on Investment, http://www.gartner.com/it/page.jsp?id=505592. Retrieved October 31, 2013.

Gartner Inc. (2011) Gartner Says Android to Command Nearly Half of Worldwide Smartphone Operating System Market by Year-End 2012, http://www.gartner.com/it/page.jsp?id=1622614, Retrieved October 23, 2013.

Gartner Inc. (2012) Drivers and Incentives for the Wide Adoption of Open Source Software Mark Driver, http://www.gartner.com/id=2158016, Retrieved October 25, 2013.

Gayo-Avello D (2013) Nepotistic Relationships in Twitter and Their Impact on Rank Prestige Algorithms. *Information Processing & Management* 49(6):1250–1280.

Gefen D, Rigdon EE, Straub D (2011) An Update and Extension to SEM Guidelines for Administrative and Social Science Research. *Management Information Systems Quarterly* 35(2):III-XIV.

Ghosh, R (2002) Free/Libre and Open Source Software: Survey and Study, http://flossproject.org/.

Goodin, D (2014) Critical Crypto Bug in OpenSSL Opens Two-Thirds of the Web to Eavesdropping, http://arstechnica.com/security/2014/04/critical-crypto-bug-in-openssl-opens-two-thirds-of-the-web-to-eavesdropping/. Retrieved March 24, 2015.

Google (2014) Program Information for Past Years, https://code.google.com/p/google-summer-of-code/wiki/ProgramStatistics. Retrieved April 7, 2015.

Google (2015) Google Summer of Code 2015 Frequently Asked Questions, https://www.google-melange.com/gsoc/document/show/gsoc_program/google/gsoc2015/help_page. Retrieved April 7, 2015.

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

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

Hahn J, Zhang C (2005) An Exploratory Study of Open Source Projects from a Project Management Perspective *MIS Research Workshop 2005*, Purdue University.

Hale R (2000) To Match or Mis-Match? The Dynamics of Mentoring as a Route to Personal and Organisational Learning. *Career Development International* 5(4/5):223–234.

Hanushek, EA, Jackson, JE (2013) *Statistical Methods for Social Scientists*, New York: Academic Press.

Hemetsberger A, Reinhardt C (2006) Learning and Knowledge-Building in Open-Source Communities: A Social-Experiential Approach. *Management Learning* 37(2):187–214.

Henseler J, Ringle CM, Sinkovics RR (2009) The Use of Partial Least Squares Path Modeling in International Marketing. *Advances in International Marketing* 20:277–319.

Hinds PJ, Mortensen M (2005) Understanding Conflict in Geographically Distributed Teams: The Moderating Effects of Shared Identity, Shared Context, and Spontaneous Communication. *Organization Science* 16(3):290–307.

Hofstede, G (1980) *Culture's Consequences: International Differences in Work-Related Values*. Beverly Hills (CA): Sage Publications.

Horwitz FM (2003) Finders, Keepers? Attracting, Motivating and Retaining Knowledge Workers. *Human Resource Management Journal* 13(4):23–44.

Howison J, Crowston K (2014) Collaboration Through Open Superposition: A Theory of the Open Source Way. *Management Information Systems Quarterly* 38(1):29–50.

Hu D, Zhao JL (2009) Discovering Determinants of Project Participation in an Open Source Social Network *Proceedings of the International Conference on Information Systems 2009 (ICIS 2009)*.

Hu D, Zhao JL, Chen J (2012) Reputation Management in an Open Source Developer Social Network: An Empirical Study on Determinants of Positive Evaluations. *Decision Support Systems* 53(3):526–533.

Huettinger M (2008) Cultural Dimensions in Business Life: Hofstede's Indices for Latvia and Lithuania. Baltic *Journal of Management* 3(3):359-376.

IBM (2011) Open Source Technologies Play a Key Role in Future of Application Development, https://www.ibm.com/developerworks/mydeveloperworks/blogs/techtrends/entry/open_source_technologies_play_a_key_role_in_future_of_application_development16?lang=en. Retrieved December 6, 2012.

IDG Connect (2013) Openstack: Platform of Choice for Cloud, http://www.redhat.com/files/resources/en-opst-idg-openstack-platform-choice-cloud-infographic.pdf. Retrieved March 19, 2015.

Jorgensen N (2001) Putting It All in the Trunk: Incremental Software Development in the FreeBSD Open Source Project. *Information Systems Journal* 11(4):321–336.

Kavanagh, P (2004) *Open Source Software: Implementation and Management.* New York: Elsevier.

KDE (2011) KDE Booklet Version 8, https://community.kde.org/images.community/5/5f/Kde_booklet_ver8.pdf. Retrieved April 7, 2015.

KDE Techbase (2009) MovetoGit, https://techbase.kde.org/Projects/MovetoGit. Retrieved April 7, 2015.

KDE Techbase (2014) Development Guidelines, https://techbase.kde.org/Development/Guidelines. Retrieved February 13, 2015.

Ke W, Zhang P (2010) The Effects of Extrinsic Motivations and Satisfaction in Open Source Software Development. *Journal of the Association for Information Systems* 11(12):785–808.

Keil M, Mann J (2000) Why Software Projects Escalate: An Empirical Analysis and Test of Four Theoretical Models. *Management Information Systems Quarterly* 24(4):631–664.

Kirkman BL, Rosen B, Tesluk PE, Gibson CB (2004) The Impact of Team Empowerment on Virtual Team Performance: The Moderating Role of Face-to-Face Interaction. *Academy of Management Journal* 47(2):175–192.

Kleinbaum, DG, Klein, M (2012) *Survival Analysis: A Self-Learning Text*, 3rd ed., New York: Springer.

Köhne M (2012) *The Concept of Athletic Excellence – An Inquiry into its Meaning for a Moral Judgment on the Ban on Doping*. Master Thesis, Utrecht University.

Kram, KE (1985) *Mentoring at Work: Developmental Relationships in Organizational Life*. Lanham (MD): University Press of America.

Kristof-Brown AL, Zimmerman RD, Johnson EC (2005) Consequences of Individuals' Fit at Work: A Meta-Analysis of Person-Job, Person-Organization, Person-Group, and Person-Supervisor Fit. *Personnel Psychology* 58(2):281–342.

Krogh G von, Spaeth S (2007) The Open Source Software Phenomenon: Characteristics that Promote Research. *The Journal of Strategic Information Systems* 16(3):236–253.

Kuk G (2006) Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List. *Management Science* 52(7):1031–1042.

Lakhani K, Wolf RG (2005) Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects. Feller J, Fitzgerald B, Hissam S, Lakhani KR, eds. *Perspectives on Free and Open Source Software,* 3-22, Cambridge (MA): MIT Press.

Lentz E, Allen TD (2009) The Role of Mentoring Others in the Career Plateauing Phenomenon. *Group & Organization Management* 34(3):358–384.

Lewin K (1951) Field Theory in Social Science. Cartwright D, ed. *Resolving Social Conflicts & Field Theory in Social Science,* 170-187, Washington (DC): American Psychological Association.

Long J (2006) Understanding the Role of Core Developers in Open Source Software Development. *Journal of Information Technology and Organizations* (1):75-85.

MacIntyre, AC (1981) *After Virtue: A Study in Moral Theory*, 1st ed., Notre Dame: University of Notre Dame Press.

MacIntyre AC (1994) A Partial Response to My Critics. Horton J, Mendus S, eds. *After MacIntyre: Critical Perspectives on the Work of Alasdair MacIntyre*, Notre Dame: Polity Press/ University of Notre Dame Press.

Madey G, Christley S (2008) F/OSS Research Repositories & Research Infrastructures *Workshop on Free/Open Source Software Repositories and Research Infrastructures (FOSSRRI)* ,

Malik TH, Zhao Y (2013) Cultural Distance and its Implication for the Duration of the International Alliance in a High Technology Sector. *International Business Review* 22(4):699–712.

McAllister DJ (1995) Affect- and Cognition-Based Trust as Foundations for Interpersonal Cooperation in Organizations. *Academy of Management Journal* 38(1):24–59.

McMillan, R (2014) It's Crazy What Can Be Hacked Thanks to Heartbleed, http://www.wired.com/2014/04/heartbleed_embedded/. Retrieved March 23, 2015.

Midha V (2008) Does Complexity Matter? The Impact of Change in Structural Complexity on Software Maintenance and New Developers' Contributions in Open Source Software. *Proceedings of the International Conference on Information Systems 2008 (ICIS 2008).*

Miorandi D, Sicari S, Pellegrini F, Chlamtac I (2012) Internet of Things: Vision, Applications and Research Challenges, *Ad Hoc Networks* 10(7):1497-1516.

Moody, G (2012) Another Billion-Dollar Open Source Company: Instagram, http://www.computerworlduk.com/blogs/open-enterprise/another-billiondollar-open-source-company-instagram-3569199/, Retrieved March 23, 2015.

Moore G, Beadle R (2006) In Search of Organizational Virtue in Business: Agents, Goods, Practices, Institutions and Environments. *Organization Studies* 27(3):369–389.

Muchinsky PM, Monahan CJ (1987) What is Person-Environment Congruence? Supplementary versus Complementary Models of Fit. *Journal of Vocational Behavior* 31(3):268–277.

Netmarketshare (2015) Mobile/Tablet Browser Market Share, http://www.netmarketshare.com/browser-market-share.aspx?qprid=0&qpcustomd=1&qpsp=2015&qpnp=1&qptimeframe=Y.

Nunnally, JC (1978) *Psychometric Theory*, 2nd ed., New York: McGraw-Hill.

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

O'Leary MB, Cummings JN (2007) The Spatial, Temporal, and Configurational Characteristics of Geographic Dispersion in Teams. *Management Information Systems Quarterly* 31(3):433–452.

OpenHub (2015) OpenSSL Contributors Listing, https://www.openhub.net/p/openssl/ contributors ? Retrieved March 24, 2015.

Oreg S, Nov O (2008) Exploring Motivations for Contributing to Open Source Initiatives: The Roles of Contribution Context and Personal Values. *Computers in Human Behavior* 24(5):2055–2073.

Parise MR, Forret ML (2008) Formal Mentoring Programs: The Relationship of Program Design and Support to Mentors' Perceptions of Benefits and Costs. *Journal of Vocational Behavior* 72(2):225–240.

Pratyush SN, Sherae DL, Ting-Ting CR (2010) The Impact of Person-Organization Fit on Turnover in Open Source Software Projects. *Proceedings of the International Conference on Information Systems 2010.*

Qureshi I, Fang Y (2010) Socialization in Open Source Software Projects: A Growth Mixture Modeling Approach. *Organizational Research Methods* 14(1):208–238.

Raymond, ES (1999) *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, 1st ed., Sebastopol: O'Reilly.

Roberts JA, Hann I, Slaughter SA (2006) Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects. *Management Science* 52(7):984–999.

Robles G, Gonzalez-Barahona JM, Herraiz I (2009) Evolution of the Core Team of Developers in Libre Software Projects. *6th IEEE International Working Conference on Mining Software Repositories 2009*, 167–170.

Romans D, Curtis A (2004) Global Journal Prestige and Supporting Disciplines A Scientometric Study of Information Systems Journals. *Journal of the Association for Information Systems* 5(2):29–77

Ryan, Deci (2000a) Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. *Contemporary Educational Psychology* 25(1):54–67.

Ryan RM, Deci EL (2000b) Self-Determination Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-Being. *American Psychologist* 55(1):68-78.

Santos C, Kuk G, Kon F, Pearson J (2012) The Attraction of Contributors in Free and Open Source Software Projects. *The Journal of Strategic Information Systems* 22(1):26-45.

Scacchi W, Feller J, Fitzgerald B, Hissam S, Lakhani K (2006) Understanding Free/Open Source Software Development Processes. *Software Process: Improvement and Practice* 11(2):95–105.

Scellato S, Mascolo C, Musolesi M, Latora V (2010) Distance Matters: Geo-Social Metrics for Online Social Networks. *Proceedings of the 3rd Conference on Online Social Networks,* USENIX Association.

Setia P, Rajagopalan B, Sambamurthy V, Calantone R (2012) How Peripheral Developers Contribute to Open-Source Software Development. *Information Systems Research* 23(1):144–163.

Schilling A (2012) Links to the Source - A Multidimensional View of Social Ties for the Retention of FLOSS Developers. *Proceedings of the 2012 ACM SIGMIS CPR Conference*.

Schilling A, Laumer S, Weitzel T (2012) Train and Retain - The Impact of Mentoring on the Retention of FLOSS Developers. *Proceedings of the 2012 ACM SIGMIS CPR Conference*.

Schilling A, Laumer S, Weitzel T (2013) Together but Apart - How Spatial, Temporal and Cultural Distances Affect FLOSS Developers' Project Retention. *Proceedings of the 2013 ACM SIGMIS CPR Conference*.

Schilling A, Laumer S, Weitzel T (2014) Stars Matter - How FLOSS Developers' Reputation Affects the Attraction of New Developers. *Proceedings of the 2014 ACM SIGMIS CPR Conference*.

Schneier, B (2014) Heartbleed, https://www.schneier.com/blog/archives/2014/04/heartbleed .html. Retrieved April 12, 2015.

Seltman, HJ (2014) *Experimental Design and Analysis*. Carnegie Mellon University.

Sen R, Subramaniam C, Nelson ML (2008) Determinants of the Choice of Open Source Software License. *Journal of Management Information Systems* 25(3):207–240.

Seong JY, Kristof-Brown AL, Park W, Hong D, Shin Y (2012) Person-Group Fit: Diversity Antecedents, Proximal Outcomes, and Performance at the Group Level. *Journal of Management*.

Shah SK (2006) Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development. *Management Science* 52(7):1000–1014.

Sheldon KM, Elliot AJ (1999) Goal Striving, Need Satisfaction, and Longitudinal Well-Being: The Self-Concordance Model. *Journal of Personality and Social Psychology* 76(3):482–497.

Siebdrat F, Hoegl M, Ernst H (2009) How to Manage Virtual Teams. *MIT Sloan Management Review* 50(4):63-68.

Singh PV (2010) The Small-World Effect. *ACM Transactions on Software Engineering and Methodology* 20(2):1–27.

Singh PV, Tan Y, Mookerjee V (2011a) Network Effects: The Influence of Structural Social Capital on Open Source Project Success. *Management Information Systems Quarterly* 35(4):813–829.

Singh PV, Tan Y, Youn N (2011b) A Hidden Markov Model of Developer Learning Dynamics in Open Source Software Projects. *Information Systems Research* 22(4):790–807.

Smith TC, Smith B (2005) Graphing the Probability of Event as a Function of Time Using Survivor Function Estimates and the SAS System's PROC PHREG. *Proceedings of the 13th Annual Western User's of SAS Software Conference*.

Solomon, C (2010) The Challenges of Working in Virtual Teams, http://rw-3.com/2012VirtualTeamsSurveyReport.pdf.

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

Stewart KJ, Gosain S (2006) The Impact of Ideology on Effectiveness in Open Source Software Development Teams. *Management Information Systems Quarterly* 30(2):291–314.

Stokel-Walker, C (2014) The Internet Is Being Protected By Two Guys Named Steve, http://www.buzzfeed.com/chrisstokelwalker/the-internet-is-being-protected-by-two-guys-named-st#.sl6xbbmnK. Retrieved March 19, 2015.

Tarique I, Schuler RS (2010) Global Talent Management: Literature Review, Integrative Framework, and Suggestions for Further Research. *Journal of World Business* 45(2):122–133.

Toral S, Martínez-Torres M, Barrero F (2010) Analysis of Virtual Communities Supporting OSS Projects Using Social Network Analysis. *Information and Software Technology* 52(3):296–303.

Vansteenkiste M, Simons J, Lens W, Sheldon KM, Deci EL (2004) Motivating Learning, Performance, and Persistence: The Synergistic Effects of Intrinsic Goal Contents and Autonomy-Supportive Contexts. *Journal of Personality and Social Psychology* 87(2):246–260.

von Krogh G, Haefliger S, Spaeth S, Wallin MW (2012) Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development. *Management Information Systems Quarterly* 36(2):649–676.

von Krogh G, Spaeth S, Lakhani KR (2003) Community, Joining, and Specialization in Open Source Software Innovation: A Case Study. *Research Policy* 32(7):1217–1241.

von Krogh G, von Hippel E (2003) Special Issue on Open Source Software Development. *Research Policy* 32(7):1149–1157.

W3Techs (2015) Usage Statistics and Market Share of Content Management Systems for Websites, http://w3techs.com/technologies/overview/content_management/all/. Retrieved March 19, 2015.

Weaver GR (2006) Virtue in Organizations: Moral Identity as a Foundation for Moral Agency. *Organization Studies* 27(3):341–368.

Webster J, Watson RT (2002) Analyzing the Past to Prepare for the Future Writing a Literature Review. *Management Information Systems* 26(2):XIII-XXIII.

Werbel JD, Johnson DJ (2001) The Use of Person-Group Fit for Employment Selection: A Missing Link in Person-Environment Fit. *Human Resource Management* 40(3):227–240.

Xu B, Jones DR (2010) Volunteers' Participation in Open Source Software Development: A Study from the Social-Relational Perspective. *Database for Advances in Information Systems* 41(3):69-84.

Yin, RK (2003) *Case Study Research: Design and Methods*, 3rd ed., Thousand Oaks (CA): Sage Publications.

Yin, RK (2009) *Case Study Research: Design and Methods*, 4th ed., Los Angeles (CA): Sage Publications.

Zhang X, Venkatesh V (2013) Explaining Employee Job Performance: The Role of Online and Offline Workplace Communication Networks. *Management Information Systems Quarterly* 37(3):695–722.

Chapter I:
Literature Review

Paper I

What Do We Know About FLOSS Developers' Attraction, Retention and Commitment?

A Literature Review

Andreas Schilling
University of Bamberg
andreas.schilling@uni-bamberg.de

In: Proceedings of the 47th Hawaii International Conference on System Sciences (HICSS 2014), Big Island (Hawaii), USA

The publication is available under:

http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6759097

Abstract

Free Libre Open Source Software (FLOSS) is an essential part of our daily life. Many companies and private households rely on FLOSS every day. However, the vast majority of FLOSS initiatives fail. In order to support future research and derive operational advice for FLOSS projects, this research reviews and categorizes the managerial insights from over 20 years of FLOSS research. Based on the central role of the developer base and research on human resource management, developer attraction, retention and commitment are identified as core management areas for FLOSS projects. A detailed analysis of 43 journal articles on FLOSS management identifies an extensive body, which analyses project members' commitment. In contrast, there is relatively little dedicated research on FLOSS developers' attraction and retention. Moreover, the literature review reveals that most articles use solely either an individual-, group- or project-centric research perspective although these perspectives are interrelated with each other.

Chapter II:
Developer Attraction

Paper II

Is the Source Strong with You?

A Fit Perspective to Predict Sustained Participation of FLOSS Developers

Andreas Schilling
University of Bamberg
andreas.schilling@uni-bamberg.de

Sven Laumer
University of Bamberg
sven.laumer@uni-bamberg.de

Tim Weitzel
University of Bamberg
tim.weitzel@uni-bamberg.de

In: Proceedings of the 32nd International Conference on Information Systems (ICIS 2011), Shanghai, China

The publication is available under:

http://aisel.aisnet.org/icis2011/proceedings/humancapital/3

Abstract

Despite the notable success of some Free Libre Open Source (FLOSS) projects, the overwhelming majority of FLOSS initiatives fail, mostly because of insufficient long-term participation of developers. In contrast to previous research which focuses on the individual perspective, we approach developer retention from an organizational perspective to help existing project members identify potential long-term contributors who are worth spending their time on. Methodically, we transfer two concepts from professional recruiting, Person-Job (P-J) and Person-Team (P-T) fit, to the FLOSS domain and evaluate their usage to predict FLOSS developer retention.

An empirical analysis reveals that both fit concepts are appropriate to explain FLOSS retention behavior. Looking at contributor retention in Google Summer of Code (GSoC) projects, we find a moderate correlation with P-J fit and a weak correlation with P-T fit.

Paper III

Who Will Remain?

An Evaluation of Actual Person-Job and Person-Team Fit to Predict Developer Retention in FLOSS Projects

Andreas Schilling
University of Bamberg
andreas.schilling@uni-bamberg.de

Sven Laumer
University of Bamberg
sven.laumer@uni-bamberg.de

Tim Weitzel
University of Bamberg
tim.weitzel@uni-bamberg.de

In: Proceedings of the 45th Hawaii International Conference on System Sciences (HICSS 2012), Maui (Hawaii), USA

The publication is available under:

http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6149241

Abstract

Many businesses and private households rely on Free Libre Open Source Software (FLOSS). Due to a lack of sustained contributors, however, most FLOSS projects do not survive. The early identification of developers who are likely to remain is thus an eminent challenge for the management of FLOSS initiatives. Previous research has shown that individuals' subjective assessment is often inaccurate emphasizing the need to objectively evaluate retention behavior. Consistent with the concepts Person-Job (P-J) and Person-Team (P-T) fit from the traditional recruitment literature, we derive objective measures to predict developer retention in FLOSS projects. In an analysis of the contribution behavior of former Google Summer of Code (GSoC) students we reveal that the level of development experience and conversational knowledge is strongly associated with retention. Surprisingly, our analysis reveals that students with abilities that are underrepresented in the project and students with a higher academic education do not remain considerably longer.

Chapter III:

Developer Integration

Paper IV

The Wizards of OSS - Does Developers' Geographic Dispersion Make OSS Teams more Productive?[11]

Andreas Schilling
University of Bamberg
andreas.schilling@uni-bamberg.de

Sven Laumer
University of Bamberg
sven.laumer@uni-bamberg.de

Tim Weitzel
University of Bamberg
tim.weitzel@uni-bamberg.de

Under Review (2nd review round): Information Systems Research (ISR)

[11] The term OSS has been used instead of FLOSS in order to align with related articles in the targeted journal and not to express any theoretical difference.

Abstract

This study examines the influence of geographic dispersion on the productive collaboration of Open Source Software (OSS) developers. Building on teamwork literature, we consider OSS developers' geographic dispersion as a multi-dimensional construct consisting of spatial, temporal, and cultural distances. We posit that each of these distances has a distinct negative effect on OSS developers' productive interplay by reducing their level of shared knowledge and social relatedness. As a potential strategy for OSS teams to overcome the productivity deficits of these geographic distances, we evaluate the role of direct offline interactions which take place in sprint events. The results of our empirical evaluation with 648 OSS teams largely support our research hypotheses and suggest: (*i*) geographic dispersion should be considered as a multi-dimensional construct, (*ii*) spatial, temporal, and cultural distances have a distinct negative effect on OSS teams' productivity, (*iii*) direct offline interactions between developers transform the productivity deficits of spatial and cultural distances into productivity gains for OSS teams, (*iv*) direct offline interactions are especially valuable for the productive interplay of developers who have little experience with the particular OSS project. Finally, we compare the results of our research with previous research in the organizational and OSS domain, which has generally focused only on one particular dimension of geographic dispersion. Thereby, we derive possible questions for future studies and delineate specific practical advice.

Keywords: Open Source Software, Geographically Dispersed Teamwork, Spatial Distance, Temporal Distance, Cultural Distance, Offline Interactions, Shared Knowledge, Social Cohesion

1 Introduction

Initiatives developing Open Source Software (OSS) have experienced massive growth over the past decade with OSS code production growing at exponentially high rates (Deshpande and Riehle 2008). In these OSS initiatives, developers who are scattered around the world collaborate voluntarily with each other using computer-mediated communication (CMC). With regard to the high production rates of these collaborations, OSS initiatives are considered forerunners for geographically dispersed teams within the organizational domain, which instead commonly underperform (Malone 2004, Solomon 2010). Based on this stark contrast, scholars thoroughly studied the effects of various individual- and team-level factors to understand OSS developers' productive interplay and extract relevant advice for the organizational domain. Most studies within the OSS domain, however, only examined characteristics of members' online behavior while disregarding one fundamental element of these initiatives, which is OSS developers' geographic dispersion. In comparison, the effects of team members' geographic dispersion has been examined intensively in the organizational domain where they are often seen as a substantial barrier for productive teamwork (Cummings et al. 2009). This in turn raises the question of whether OSS projects are also negatively affected by their developers' geographic dispersion and if so what can OSS projects do to overcome these challenges?

With respect to this question, we examine if OSS developers' geographic dispersion complicates code development in OSS teams. By bringing together central lessons from teamwork literature (Martins et al. 2004, Hinds and Mortensen 2005) with conceptual OSS research by Ågerfalk et al. (2005), we consider OSS developers' geographic dispersion as a multi-dimensional construct consisting of spatial, temporal, and cultural distances. Building on organizational literature and OSS studies which only consider one of these distances (Hu et al. 2012, Daniel et al. 2013), we expect that each of these distances mitigates OSS teams' productivity. Specifically, we posit that geographic dispersion hinders the effective interplay of OSS developers by complicating the formation of shared knowledge and social relationships and that the prevalent use of context-low CMC in OSS projects even amplifies this problem. Hence, our leading hypothesis is that offline interactions (which are considered to be context-rich) help OSS developers to cope with productivity deficits caused by their geographic dispersion by enhancing their abilities to build shared knowledge and interpersonal relations. Thus, by combining teamwork literature and OSS research we examine the following research question: *Do direct offline relationships among OSS developers help them to overcome the productivity deficits caused by their spatial, temporal, and cultural distances?*

The results of our research have several implications for OSS research. First, our multi-dimensional conceptualization of OSS developers' geographic dispersion in terms of their spatial, temporal, and cultural distances adds substantial explanatory power for understanding the productivity of OSS teams. Also, our conceptualization helps to bring together the findings of OSS studies which have only focused on one distance form (Hu et al. 2012, Daniel et al. 2013, Colazo and Fang 2010) and helps to derive a comprehensive picture of the effects of geographic dispersion in OSS projects. Moreover, by looking at both OSS developers' online and offline interactions, our study explicitly addresses the call by Crowston et al. (2007) for more research on the role of offline interactions among OSS developers. Finally, our evaluation results highlight the importance for OSS managers to base the staffing of their projects on developers' ability to meet offline. If offline interactions are not possible, project managers should bring together OSS developers with little spatial and cultural distances. However, if offline meetings between OSS developers are possible, project managers should adopt the completely opposite approach and bring together members with high spatial and cultural distances. Irrespective of developers' ability to meet offline, OSS managers should seek to minimize the amount of asynchronous working hours among team members to achieve productive OSS development.

In light of researchers' advice to treat knowledge workers as volunteers (Drucker 2002) our research also provides value to the organizational domain. Our study draws on the common critique that teamwork literature treats members' geographic dispersion as a unitary and dichotomous construct. In this regard, our multi-dimensional conceptualization can help to bring together some of the fragmented findings created by previous evaluations and so derive a comprehensive understanding of the positive and negative effects of members' geographic dispersion for productive teamwork. Moreover, by examining how team members' offline relationships can stimulate their online collaboration, our work extends research by Zhang and Venkatesh (2013), who recently revealed an interaction between these two domains. Finally, in

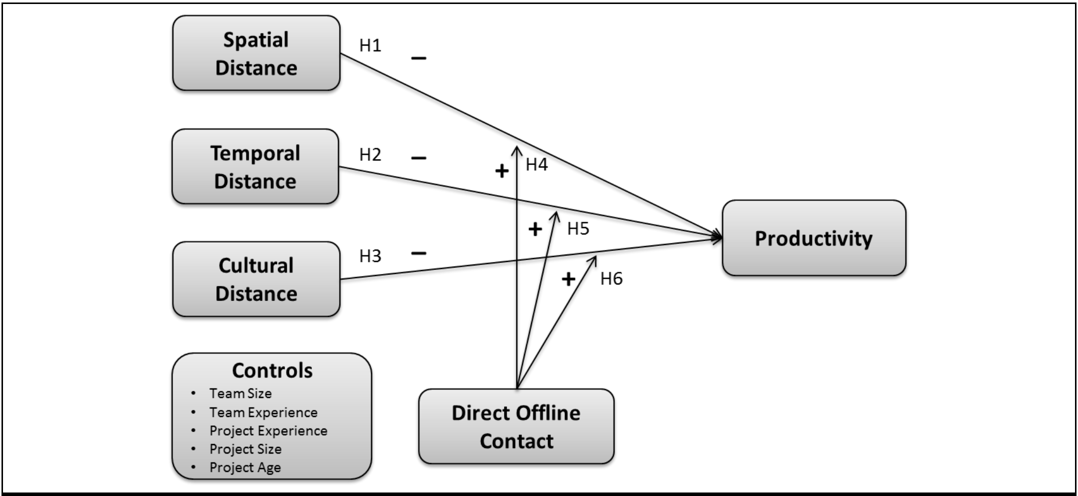


Figure 1. Research Model

the light of the common underperformance of geographically dispersed teams within corporations (Solomon 2010), our research provides several practical lessons for the staffing and management of such teams. In particular, we delineate the conditions under which geographically dispersed teams show their productivity potential and detail the conditions under which collocated teamwork should instead be used. Finally, we point out what managers of geographically dispersed teams need to watch out for.

This paper is structured as follows. In the next section, we derive our research model and formulate our research hypotheses. Thereafter, in section three, we detail the research methodology. Finally, we discuss the theoretical and managerial implications for both the OSS and the organizational domain.

2 Theory Development

To understand how OSS teams' productivity is affected by developers' geographic dispersion, we build on organizational literature and consider geographic dispersion as a multi-dimensional construct that encompasses spatial, temporal, and cultural dimensions. Without doubt geographic dispersion offers substantial gains for OSS teams' productivity such as access to a diverse set of skills and the ability to lever time-zone differences to achieve continuous code development (Colazo and Fang 2010). With regard to the prevalent use of context-low CMC, we assume, however, that the negative aspects of OSS developers' geographic dispersion prevail. Furthermore, by building on team cohesion literature, we posit that offline interactions, which are considered context rich, mitigate these productivity deficits by giving OSS developers diverse possibilities for building mutual knowledge and interpersonal relationships. Figure 1 visualizes our research model and hypotheses.

2.1 A Multi-Dimensional Model for OSS Developers' Geographic Dispersion

For a long time, organizational literature has considered geographic dispersion dichotomously and solely in terms of spatial distances (Cummings et al. 2009, O'Leary and Cummings 2007, Hinds and Mortensen 2005). However, this approach has increasingly been criticized for its inadequacy in capturing the collaboration challenges posed to geographically dispersed teams (Cummings et al. 2009, O'Leary and Cummings 2007, Hinds and Mortensen 2005). In particular the sole consideration of spatial distances seems too limited to fully understand the problems faced by geographically dispersed teams (Gibson and Gibbs 2006). Furthermore, the trend in organizations for flexible working hours and the use of time shifts fully decoupled team members' time and space constraints. Thus, when focusing only on spatial distance, it seems impossible to tease out those effects which are caused through time differences from those which are caused through differences in members' locations. As a result, recent teamwork literature recommends to distinguish between effects which are caused by differences in members' locations from those which are caused through non-overlapping working hours among members (Cummings et al. 2009, Espinosa and Carmel 2003). Apart from separating temporal and spatial constraints, organizational literature suggests that differences in team members' cultural backgrounds are another relevant factor for collaboration in geographically dispersed teams (Gibson and Gibbs 2006). Cramton (2001) provides evidence that team members from different cultures have problems in achieving a shared interpretation of concrete project incidents. With respect to the distinct effects on team members' formation of mutual knowledge and thus their collaboration, many researchers propose considering geographic dispersion as a multi-dimensional construct with spatial, temporal, and cultural dimensions (Martins et al. 2004, Hinds and Mortensen 2005, O'Leary and Cummings 2007). The second common criticism of teamwork studies concerns their dichotomous differentiation of geographic dispersion. O'Leary and Cummings (2007) show in their literature review that the overwhelming majority of teamwork studies ignores any variation or degree in team members' geographic dispersion. Instead most studies dichotomously differentiate between collocated and dispersed project teams. An exception from this common practice is an evaluation by Kirkman et al. (2006), which in turn highlights the need to consider variations of geographic dispersion. Aside from the theoretical limitations, this dichotomous differentiation mitigates the practical relevance of the derived study results as most teamwork settings in organizations are neither purely geographically dispersed nor fully collocated (Martins et al. 2004).

In the following, we draw on these central critiques to derive a conceptualization of geographic dispersion in OSS teams. With respect to organizational research by Martins et al. (2004) and Hinds and Mortensen (2005) as well as conceptual OSS research by Ågerfalk et al. (2005), we consider OSS developers' geographic dispersion as a multi-dimensional construct encompassing varying degrees of spatial, temporal, and cultural distances. As in the organizational domain, we suppose that geographic dispersion has an ambivalent effect on OSS teams' productivity. On the one hand, geographically dispersed members potentially boost team productivity not only by bringing together a diverse set of skills and experiences but also by leveraging time-zone differences to achieve continuous coding. On the other hand, however, geographic dispersion can hinder OSS developers' effective interplay by complicating the formation of shared knowledge and interpersonal relationships. With OSS developers' context-low CMC interaction, we suppose that the negative aspects of geographic dispersion prevail and thus reduce OSS teams' productivity. Next, building on organizational literature, we outline the distinct ways in which OSS developers' geographic dispersion can mitigate their productive interplay.

2.1.1 Spatial Distance

Spatially dispersed teams have substantial productivity potential as not only can they lever a global workforce but they can also bring together members with a diverse set of expertise and experiences. In practice, however, the reduced cognitive overlap among members which is caused through members' spatial distances complicates effective teamwork. According to Cramton (2001) information asymmetries constitute a severe problem for spatially dispersed teams as members often do not distinguish between information they have shared online and offline. Even worse, members often expect others to share their experiences and information. Thus, spatially dispersed team members commonly experience coordination issues which in turn undermine their effective interplay (Cummings et al. 2009). Moreover, Cramton (2001) points out that team members can interpret information asymmetries as failures of personal reliability, which in turn impedes their trust behavior. This hampers team members' effective interplay even more as trust relationships are considered a key determinant for productive teamwork (McAllister 1995). Furthermore, the negative consequences of information asymmetries are amplified through the use of CMC as it does not convey members' offline experiences and gives them only limited possibilities for informal talks to strengthen their interpersonal relationships.

Although OSS developers' interact with each other primarily online, we posit that their productive interplay is likewise affected by spatial distances. According to Crowston et al. (2007), OSS developers' often attend local meetings such as Linux User Groups (LUG), where they meet other members and listen to talks on related programming topics. While such local meetings enhance OSS developers' knowledge and expertise, they lead to unequal information distributions among them. As in the organizational domain, we expect that the resulting information asymmetries complicate OSS developers' productive interplay. Specifically we suppose that OSS developers' effective interplay is hampered through locally shared information as well as weaker interpersonal relationships between spatially dispersed developers. Hu et al. (2012) support the negative effects of spatial distances by revealing that OSS developers who live in the same city are more likely rate each other positively. Further, according to Stewart and Gosain (2006), strong interpersonal relations play a key role for OSS developers' effective collaboration. Thus, building on lessons from the organizational domain and reflecting on OSS research, we hypothesize that:

Hypothesis 1: *OSS developers' spatial distances are negatively associated with their productivity.*

2.1.2 Temporal Distance

Another critical factor for the productivity of geographically dispersed teams is the amount of overlapping working hours between members. For the sake of a uniformed terminology we refer to non-overlapping working hours among team members as temporal distance. Temporal distances can foster the performance of geographically dispersed teams by enabling members to work around the clock. Despite this gain for productivity, temporal distances often complicate productive teamwork (Espinosa et al. 2012, Cummings et al. 2009). This is because, team members have no or only limited possibilities to engage in real-time communication if they are with temporally dispersed. Instead, members are forced to rely on asynchronous CMC or arrange concrete dates to interact with each other. Both alternatives pose serious challenges for effective teamwork. Zhang and Venkatesh (2013) point out that asynchronous CMC increases the level of uncertainty among team members because long feedback loops prevent members from quickly assessing and predicting each other's behavior. Consequently, team members are more hesitant regarding their project engagement and hence reduce their working efforts which in turn lowers team productivity. Unfortunately, scheduling dedicated team meetings does not help overcoming the productivity deficits either. Because of the extra efforts associated with attending such meetings (e.g. staying up late), members often arrive at such events biased and are even more negatively loaded towards other members if the events do not result in clear decisions (Nurmi 2010). Ultimately, such negative experiences reduce individuals' working motivations and thus hamper team productivity. In light of the limited possibilities for coping with temporal distances, organizational research explicitly warns against the negative consequences for team members' productivity (Espinosa et al. 2012, Cummings et al. 2009).

We suppose that temporal distances have similar effects for OSS teams. Without question, temporal distances bear enormous productivity potential for OSS teams such as continuous development routines (Colazo and Fang 2010). However, with regard to organizational lessons, we posit that temporal distances instead complicate effective teamwork. This is because OSS developers commonly rely on real-time communication, in the form of Internet Relay Chat (IRC), to discuss new ideas and to ask for help (Bird and Nagappan). In line with organizational literature, we assume that OSS developers miss relevant information when they rely only on asynchronous CMC, as it often does not include all the conversations which took place in IRC. Moreover, we argue that synchronous CMC meetings between OSS developers do not solve the deficits caused by temporal distances, either. Instead, drawing on organizational research, we suppose that OSS developers are rather stressed and less motivated if they need to spend extra effort to come online for an IRC meeting with their colleagues, which in turn negatively affects their work efforts. Thus, building on organizational literature and the relevance of IRC for OSS developers' collaboration, we posit:

Hypothesis 2: *OSS developers' temporal distances are negatively associated with their productivity.*

2.1.3 Cultural Distance

The influence of team members' cultural backgrounds on their collaboration productivity has been a key topic in international management research for over two decades and has spawned a variety of different definitions and conceptualizations. One of the most cited definitions for culture in teamwork literature is from Hofstede (1980). A reason for this is, that Hofstede's research focuses on those aspects which affect individuals' working behavior and that his conceptualization has been evaluated and largely supported by various studies (Kirkman et al.

2006). According to Hofstede, culture can be defined as the ‘collective programming of the mind that distinguishes members of one human group from another’ (Hofstede 1980, p. 26). Individuals build this ‘collective programming’ through their experiences with others and in their social environment. To characterize cultural differences, Hofstede derived the following four sub-dimensions based on a multi-national study involving more than 115,000 IBM employees: (i) Power Distance: the acceptance of unequal power distributions, (ii) Uncertainty Avoidance: the degree to which people accept uncertainty and ambiguity, (iii) Masculinity vs. Femininity: the dominance of either masculine or feminine values within societies, (iv) Individualism vs. Collectivism: individuals’ need to integrate into groups.

For the sake of a uniformed terminology, we refer to differences in team members’ cultural backgrounds as cultural distances. Despite comprehensive research, studies within the organizational domain draw an inconsistent picture of the effects of cultural distances among team members. Although some research highlights their benefits for team members’ problem solving and creative abilities, many studies instead caution against cultural distances especially when using CMC. According to Bayazit and Mannix (2003), the underlying reason for the underperformance of culturally heterogeneous teams lies in different interpretations of information exchanged via CMC, which are caused by members’ different cultural backgrounds. Another way in which cultural differences reduce team performance is by complicating the decision processes via CMC (Gibson and Gibbs 2006, Maznevski and Chudoba 2000).

We argue that cultural differences complicate the collaboration of OSS teams in a similar way as in the organizational context. Culture has a salient influence on OSS developers’ motivation to contribute (Subramanyam and Xia 2008, Padmanabha 2007). For example, while developers from collectivistic cultures appreciate OSS projects due to ideological reasons, OSS developers from individualistic cultures are instead motivated by their wish to promote themselves (Subramanyam and Xia 2008). Even if OSS developers’ motives do not contradict each other, they may still hinder the establishment of a common ideology with shared values and norms. However, as Stewart and Gosain (2006) point out, such common ideology is a fundamental element for the productive collaboration of OSS developers. Another problem for culturally heterogeneous OSS teams is social categorization. This behavior makes OSS developers identify with those project members who share their nationality and delineate from members with other nationalities. As a result, OSS developers may selectively distribute information and help others, which in turn also mitigates team performance. In a recent study, Daniel et al. (2013) provide evidence for the supposed negative effects by showing that cultural distances indeed undermine OSS teams’ productivity. Hence, drawing on organizational literature and OSS research, we posit that:

Hypothesis 3: *OSS developers’ cultural distances are negatively associated with their productivity.*

2.2 The Moderating Role of Direct Offline Contact

As outlined above, we expect that CMC has an ambivalent effect on the productivity of geographically dispersed teams. Although CMC is without doubt the key underpinning factor for geographically dispersed teamwork, we argue that it poses a substantial barrier for the productive collaboration of geographically dispersed members. In particular, two characteristics of CMC complicate collaboration in geographically dispersed teams. These are (i) context-low interactions and (ii) members' anonymity to one another. According to organizational research, an appropriate means of overcoming these problems are offline meetings between team members (Kirkman et al. 2006). Compared to CMC, offline interactions provide a much richer context for members to articulate themselves and various possibilities to build social relationships with each other. In addition, practitioners consider offline meetings to be a key complement for collaboration in geographically dispersed teams (Siebdrat et al. 2009).

Drawing on the positive effects of offline interactions for the collaboration of organizational teams, we argue that they provide similar stimuli for the productive collaboration of geographically dispersed OSS developers. In the OSS domain, offline interactions between team members commonly occur in the context of sprint events. During these offline events OSS developers have dedicated time to talk to each other and to present their upcoming or past coding. Typically, the events end with a social activity, such as clubbing or having dinner, in which OSS developers get the possibility to build friendships and discover similarities. With regard to teamwork literature, we posit that enhanced conversations about work-related as well as other topics help OSS developers mitigate the productivity deficits caused by their spatial, temporal, and cultural distances.

2.2.1 Direct Offline Contact and Spatial Distance

Offline interactions aid the effective collaboration of spatially dispersed team members by giving them enhanced possibilities to exchange about work and non-work related topics. In comparison to members' CMC interactions, offline interactions let them put faces to names and gain personal impressions of each other. Moreover, members' abilities to visualize their thoughts through whiteboard sketches and the instant feedback of verbal and non-verbal reactions enhance the exchange of knowledge and help to avoid misunderstandings. Apart from this enhanced knowledge transfer, offline meetings ease the collaboration among spatially dispersed members by giving them possibilities for spontaneous talks. Such personal encounters often lead to off-topic conversations among team members which, in turn, lay the foundation for strong interpersonal relationships and friendships. Thus, offline interactions help spatially dispersed members overcoming central barriers for effective teamwork.

OSS studies suggest that the collaboration of spatially dispersed OSS developers similarly benefits from offline interactions. Based on observations at sprint events, Crowston et al. (2007) note that whiteboard sketches and the transfer of non-verbal cues are heavily used and appreciated among OSS developers for exchanging knowledge. Moreover, sprint attendees reported in an interview with Goth (2007) that they considered offline meetings especially important for establishing and maintaining social relationships with one another. One of the interviewed developers put it this way: ‘At our sprints, we think much more about how to build team collaboration instead of how to build a specific feature’ and ‘with the Internet, your already know what everybody is doing, and now the emphasis has shifted to the importance of conversations in the hall-way, talking about families and grabbing a beer’ (Goth 2007). With respect to organizational research and the relevance of strong interpersonal relationships for OSS teams’ productivity (Stewart and Gosain 2006), we posit that:

Hypothesis 4: *Direct offline relationships among OSS developers mitigate the productivity deficits caused by their spatial distances.*

2.2.2 Direct Offline Contact and Temporal Distance

In addition, offline interactions foster the effective collaboration of teams with temporal distances. In particular, members’ enhanced abilities to discuss and perform complex decisions help addressing central challenges faced by temporally dispersed team members. According to Maznevski and Chudoba (2000) face-to-face communication helps temporally dispersed members streamline their coordination and decision-making processes, which in turn enhances their productivity. In addition, offline meetings help teams with temporal distances become more productive by reducing the perceived level of uncertainty among members which are caused by long feedback loops in CMC. To achieve this, offline interactions not only give members the possibility to learn about each other but also enable them to monitor in person their colleagues’ team behavior and task commitment.

OSS research indicates that offline interactions have similar stimulating effects on the collaboration of temporally dispersed OSS developers. Goth (2007) considers offline meetings to be an essential means for OSS teams to overcome the challenges caused by temporal distances. In particular, the researcher highlights the possibility of efficiently performing complex team activities which require rapid member interactions. According to Crowston et al. (2007), such actions include the discussion of new ideas and making strategic decisions. Such team activities have substantial relevance for the development of OSS projects and the productivity of team members. Moreover, strong interpersonal relationships foster OSS teams’ productivity by enhancing members’ ability to cope with situations of uncertainty and help lessen feelings of isolation. Thus, by combining organizational and OSS research, we hypothesize that:

Hypothesis 5: *Direct offline relationships among OSS developers mitigate the productivity deficits caused by their temporal distances.*

2.2.3 Direct Offline Contact and Cultural Distance

Finally, offline interactions help members to cope with the productivity deficits caused by their cultural distances. In comparison to context-low CMC interactions, team members very soon become aware of relevant differences in their cultural backgrounds when interacting face-to-face. Kankanhalli et al. (2007) consider this awareness to be key for overcoming and avoiding cultural conflicts and misunderstandings among team members. Furthermore, the context rich interaction in a face-to-face environment gives team members unique verbal (e.g. different pronunciations and immediate feedback) and visual tools to express their thoughts and overcome potential misunderstandings caused by their different cultural backgrounds. Moreover, Espinosa et al. (2006) show in their study that culturally diverse team members can develop interpersonal relationships much better when communicating offline than when using CMC. These formed interpersonal relationships help members to not only work effectively work but also build a team identity which in turn helps to deescalate conflicts.

We argue that offline interactions provide similar stimuli for OSS teams with cultural distances. In particular, we suppose that the awareness of cultural differences, created in the offline space, help members working with each other online. Crowston et al. (2007) support this and state that OSS developers understand email conversations with each other much easier after they have met in person. Moreover, the researchers detail that offline meetings help OSS developers building strong interpersonal relationships, which also enhance their collaboration (Stewart and Gosain 2006). We thus hypothesize that:

Hypothesis 6: *Direct offline relationships among OSS developers mitigate the productivity deficits caused by their cultural distances.*

2.3 Control Variables

Beyond developers' geographic dispersion and their offline interactions, we account for other factors which have been shown to affect the productivity of OSS teams. Following Colazo and Fang (2010), we argue that a higher team size generally leads to a higher degree of code development in an OSS project. In addition, we take into account OSS developers' team experience, as research by Singh et al. (2011a) suggests that OSS teams perform considerably better the more often the involved developers have worked together in the past. Another characteristic which we take account of is OSS developers' level of project experience, as Singh et al. (2011b) provide evidence that OSS developers become more productive the longer they have been developing for a project. Furthermore, we account for the project size. Colazo and Fang (2010) point out that OSS projects with large codebases provide more possibilities for OSS developers to add or modify code. Finally, we control for the project age as older OSS projects are generally more mature in terms of code structure and documentation, which in turn eases OSS developers' coding (Singh et al. 2011a).

3 Research Methodology

To evaluate our research hypotheses, we examine the collaboration practices of OSS teams within the 'K Desktop Environment' (KDE). KDE is the default desktop environment on many UNIX systems and consists of various OSS projects, ranging from games to entire office suites. By focusing on KDE teams, we have the possibility of studying a diverse project spectrum within the same development context (i.e. coding conventions, tool chain, version control, etc.). In this section, we describe the details of our data collection and the measures used for our analysis.

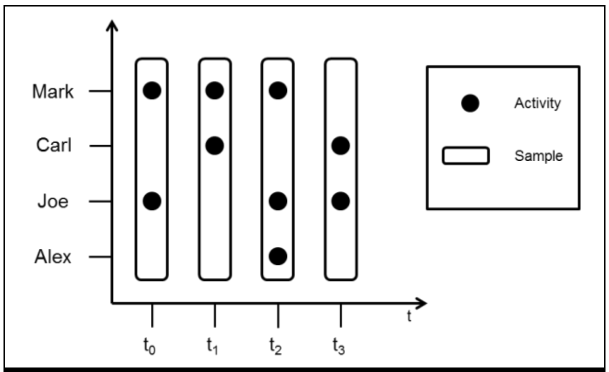


Figure 2. Deriving the Data Sample

3.1 Data Sample

We collected the data for our evaluation in three distinct extraction steps. First, we downloaded the code repositories of all 65 KDE projects and wrote programming routines which used the Version Control Systems (VCS) of these repositories to derive detailed contribution statistics on the development activity in these OSS projects. Then, in the second step, we collected information about OSS developers' geographic dispersion. To do so, we worked together with the project manager of a KDE community site where KDE developers can create a profile page on which they can also share their address information. For our research, the project manager of this site gave us full access to this location data. In the last step of our data extraction, we used these data to identify those KDE projects for which we had location information on at least 75 per cent of all issued code commits during January 1st 2009 and April 27th 2013. The six KDE projects which fulfilled this criteria are: 'KDE PIM' (a personal organizer), 'DigiKam' (a photo management suite), 'KDELibs' (cross-application libraries), 'Calligra' (an office suite), 'KDE Workspaces' (a desktop organizer), and 'Kate' (a text editor).

Based on the derived contribution statistics, we looked at the collaboration activities of these six KDE projects in detail. As in the evaluation by Kuk (2006), we noticed a very high developer fluctuation in the KDE projects. The high fluctuation essentially creates every week a new team configuration in the examined KDE projects. This allows us to study various team configurations by partitioning the derived contribution history of each KDE project into weekly samples. Because we are only interested in the collaboration of OSS developers, we filtered out all periods where none or only one developer had been active. Figure 2 visualizes our sampling strategy. In this example, Mark, Carl, Joe, and Alex are all active developers in the 'KDE PIM' project in July 2011. However, on taking a closer look one can see that over the four weeks in July 2011 there have been various permutations of inactive and active developers. Applied to the six KDE projects, this sampling strategy resulted in 648 data samples (N).

3.2 Measurement

3.2.1 Dependent Variable

With respect to previous evaluations, we assess the **productivity of OSS teams *(prod$_t$)*** based on the number of issued code commits in period *t* (Singh et al. 2011b). Code commits are a well suited indicator for this as they either add new functionality to the OSS project or modify existing code. To account for changing team sizes, we measure OSS teams' productivity using the average number of issued code commits per developer. To calculate this measure, we identify for each period *t* the set of active developers ($team_t$). Then, we sum up the number of issued code commits ($c_{i,t}$) of each developer *i* in period *t*. Finally, we divide this aggregated number through the number of developers ($team\_size_t$) in *t*.

$$prod_t = (\sum_{i \in team_t} c_{i,t}) \;\; / \;\; team\_size_t \tag{22}$$

3.2.2 Independent Variables

To measure OSS teams' **spatial distance (*spatial_dist$_t$*)**, we draw on a method which was originally proposed by Scellato et al. (2010). This method builds on the spatial distance ($dist_{i,j}$) between every two active OSS developers *i* and *j*. To calculate this distance, we use the Haversine formula (Gellert et al. 1989) that very efficiently computes the distance between two points on a sphere. Although the earth is not a perfect sphere, the Haversine formula provides an appropriate and efficient way to approximate spatial distances on earth (Sinnott 1984). Next, the proposed method uses the factor β to smooth out very long and very short distances. Finally, Scellato et al. (2010) recommend calculating the exponential decay of the resulting term because distance does not have a proportional effect on individuals' interactions. While it makes a considerable difference for individuals to meet if they work 1 or 1,000 miles apart from each other, there is only a marginal change in individuals' ability to meet with each other if they work 100,000 or 101,000 miles apart. Scellato et al. (2010) used their measure to examine the influence of spatial distances on the formation of friendship in Social Networking Sites (SNS). While the relevance of individuals' spatial distance is considered symmetric for their SNS activity, this is not necessarily the case for OSS developers. Instead, OSS developers who are highly active in a particular period can assist other developers much better than developers who only show a little commitment to the OSS project in that period. To account for this aspect, we weight an OSS developer's spatial distance to another team member (*j*) by her share of commits in the particular period ($w_{j,t}$). At the same time, we assume that team members who are more active in the particular period are more likely affected by spatial distances because they have to coordinate more with other developers. Hence, we compute the spatial distance of an OSS team as the average spatial distance weighted by developers' share of commits.

$$spatial\_dist_{i,t} = \sum_{j \in team_t \wedge i \neq j} \;\; e^{-dist_{i,j}/\beta} \times w_{j,t} \tag{23}$$

$$spatial\_dist_t = (\sum_{i \in team_t} spatial\_dist_{i,t} \times w_{i,t}) \;\; / \;\; team\_size_t \tag{24}$$

For OSS developers' **temporal distance (*temporal_dist$_{i,t}$*)**, we modify the measure as originally proposed by Colazo and Fang (2010). This measure approximates OSS developers' temporal distance based on the variance in the time of their first code commit each day. Alas, this measure is based on two important assumptions. The first assumption is that OSS developers' temporal distance to all team members is equally important. Second, the measure assumes that all project members spend similar amounts of time on the OSS project each day. However, both assumptions are not consistent with the unequal work distributions commonly found in OSS projects (Toral et al. 2010). To address these aspects, we modified the original measure of

Colazo and Fang (2010) in the following way: Rather than comparing a single point in time, we assess members' temporal distance by counting the number of overlapping working hours (*$overlap_{i,j,d}$*) between every two team members i and j on each day (*d)* in *t*. To do so, we reconstruct the daily working hours of OSS developers based on the UTC-timestamps of their first and last project commit each day. As for spatial distances, we weight OSS developers' temporal distance to team members based on their share of commits in the particular period ($w_{j,t}$). Finally, in order to assess the temporal distance for the OSS team (*$temporal\_dist_t$*), we calculate the average temporal distance per developer weighted by members' commitment in period *t*.

$$temporal\_dist_{i,t} = \sum_{j \in team_t \wedge j \neq i} \left(\sum_{d \in t} overlap_{i,j,d}\right) \times w_{j,t} \tag{25}$$

$$temporal\_dist_t = \left(\sum_{i \in team_t} temporal\_dist_{i,t} \times w_{i,t}\right) \; / \; team\_size_t \tag{26}$$

In line with our conceptualization, we draw on research by Hofstede to assess an OSS team's **cultural distances (*$cultural\_dist_t$*)**. Based on a multinational study with 116,000 IBM employees Hofstede derived national index scores for each of the four cultural dimensions. These index scores are: Power Distance Index (PDI), Uncertainty Index (UI), Masculine Index (MI), and Individuality Index (II). Following their original publication by Hofstede (1980), these index scores were evaluated and largely supported in numerous subsequent studies (Kirkman et al. 2006). Most recently, the scores have been refined by Hofstede himself (Hofstede and Minkov 2010). To evaluate OSS developers' cultural distance, we use Hofstede's national index scores in the following way: We begin by estimating KDE developers' nationality using the address details provided in their online profiles[12]. Next, we merged KDE developers' nationality with Hofstede's cultural index scores. While Hofstede's research provides index scores for most of the countries in our study sample, there are some countries, especially in Eastern Europe, which were not covered. For these countries, we draw on research by Huettinger (2008), who extended Hofstede's index scores to Eastern European countries. Finally, as proposed by Malik and Zhao (2013), we assess OSS developers' cultural differences (*$cultural\_dist_{i,j}$*) by calculating the absolute difference between their PDI, UI, MI, and II values. We assume that cultural differences are especially relevant towards those developers who are highly active in the OSS project in a given period. Therefore, we weight OSS developers' cultural differences to every other member (*j*) by her share of commits in the particular period ($w_{j,t}$). Finally, to assess cultural distance on the team-level, we average OSS developers' cultural distance weighted by their commitment in *t*.

$$cultural\_dist_{i,j} = \sum_{j \in team_t \wedge j \neq i} \left(|PDI_i - PDI_j| + |UAI_i - UAI_j| + |II_i - II_j| + |IDVI_i - IDVI_j|\right) \times w_{j,t} \tag{27}$$

$$cultural\_dist_t = \left(\sum_{i \in team_t} cultural\_dist_{i,t} \times w_{i,t}\right) \; / \; team\_size_t \tag{28}$$

[12] To ensure the validity of this approach we confirmed it using a random sample of 71 KDE developers. These KDE developers provided us with information about their location and their nationality. The results of this test support our assumption and show that our estimation matched the developers' nationality in 66 cases (90.1 per cent).

To evaluate OSS teams' **direct offline ties (*OT_t*)**, we examine developers' attendance to coding sprints of the particular KDE project. To extract this information, we first queried the central KDE community webpage for developer sprints (*https://sprints.kde.org/sprint/all*). As this webpage was not launched until April 2011, it did not cover coding sprints which occurred between January and April 2011. Moreover, not all KDE projects used this community page to organize their sprint events right after its launch, but instead waited several months before using it. Hence, we also visited the homepage of each examined KDE project and looked for information about past coding sprints. In this way we could retrieve a dedicated attendee list for the overwhelming majority of all sprint events. In the few cases in which we found no attendance list, we reconstructed it based on a labeled group photo or a blog post of the sprint. Drawing on this attendance information, we derived the network structure of OSS developers' offline relationships. To do so, we created a tie between two developers if they attended the same coding sprint. For our modeling, however, we do not consider the productivity gains of these offline meetings to be symmetrically distributed. Instead, we suppose that the productivity and coordination benefits which OSS developers can derive from their offline interactions depend on their own and their interaction partners' level of expertise. Figure 3 visualizes this modeling approach with the four developers Anna, Mark, Carl, and Joe. In this example, Anna and Mark meet at a sprint event. Mark is relatively new to the OSS project and not as experienced as Anna. Hence, he can benefit substantially from meeting Anna while she benefits only marginally from this meeting. Instead, Anna benefits much more from her meeting with Carl at some other event, because he has a much higher level of expertise than she does. To account for this asymmetric productivity gains, we model OSS developers' direct offline interactions as a directed graph. In this graph, an edge ($\eta_{i,j,t}$) between developers is weighted by the number of previous offline interactions ($meetup_{i,j,t}$) between them until period t and the level of project expertise of the developer one met ($expertise_{j,t}$). Based on this directed graph, we measure OSS teams' level of direct offline ties by calculating the average degree of outgoing links at period t.

$$\eta_{i,j,t} = expertise_{j,t} \times meetup_{i,j,t} \tag{29}$$

$$OT_t = \left(\sum_{i\in team_t} \sum_{j\in team_t \wedge j\neq i} \eta_{i,j,t}\right) \;/\; team\_size_t \tag{30}$$

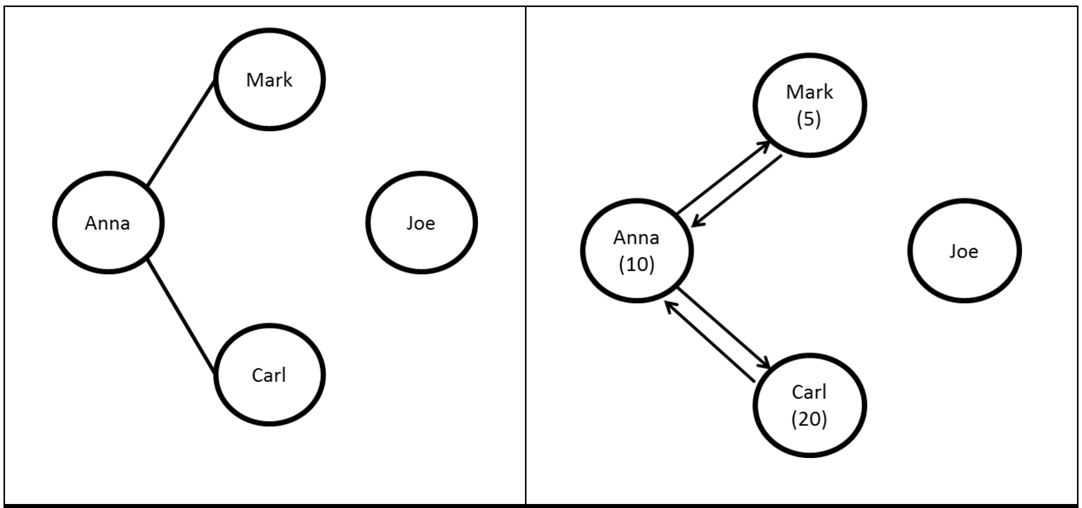


Figure 3. Modeling Resource Access in Offline Social Networks

Table 1: VIF and Cross Correlations

	VIF	1	2	3	4	5	6	7	8
1. team_size	2.99								
2. team_age	2.39	0.43***							
3. project_size	2.59	0.47***	0.41***						
4. project_age	2.03	-0.55***	-0.34***	-0.43***					
5. project_exp	2.25	-0.22***	0.48***	0.16***	0.05				
6. spatial_dist	1.84	-0.30***	-0.22***	-0.46***	0.15***	-0.28***			
7. temporal_dist	1.37	0.16***	0.20***	0.26***	-0.09*	0.05	-0.04		
8. cultural_dist	1.62	-0.31***	-0.09*	-0.13***	0.02	-0.08*	0.46***	-0,03	
9. OT	2.87	0.13***	-0.02	-0.30***	-0.19***	-0.13***	0.16***	0.22***	-0.12

Notes: * = p < 0.05, ** = p < 0.01, *** = p < 0.001

3.2.3 Control Variables

We measure OSS projects' **team size (*team_size$_t$*)**, by determining the cardinality of the developer team in period *t*. To assess **team experience (*team_exp$_t$*)** we average the number of overlapping days between every two developers until period *t* (*$D_{i,j,t}$*). Likewise, we measure developers' **project experience (*proj_exp$_t$*)** by determining the average number of days they have already been active in the OSS project (*$D_{i,t}$*). To evaluate **project size *(proj_size$_t$)***, we determine OSS projects' lines of code at the beginning of *t*. Finally, we assess **project age (*proj_age$_t$*)** by counting the number of days (*NoD_t*) between the project's inception and the beginning of *t*.

$$team\_size_t = \mid team_t \mid \tag{31}$$

$$team\_exp_t = \left(\sum_{i \in team_t} \sum_{j \in team_t \wedge j \neq i} D_{i,j,t}\right) \;/\; team\_size_t \tag{32}$$

$$proj\_size_t = LoC_t \tag{33}$$

$$proj\_exp_t = \left(\sum_{i \in team_t} D_{i,t}\right) \;/\; team\_size_t \tag{34}$$

$$proj\_age_t = NoD_t \tag{35}$$

4 Results

4.1 Descriptive Statistics and Variable Transformation

Before we started our evaluation, we checked the cross-correlations of all independent variables. Thereby, some independent variables showed a relatively strong cross-correlation, which posed the danger of multi-collinearity. To address this threat, we followed the advice of Aiken and West (1991) and mean-centered and scaled all variables. This transformation reduced the correlations between the independent variables substantially, so that most correlation coefficients are now below 0.15 (see Table 1). To ensure that the remaining correlations do not cause problems of multi-collinearity, we calculated the Variance Inflation Factor (VIF) for each independent variable. This factor measures the degree to which the variance of an independent variable is caused by collinearity. As listed in Table 1, the VIF of all the transformed variables is well below the required threshold of 10 and also lower than the recommended threshold of 4 (Greene 2003). Hence, we can ensure discriminant validity for the measures used.

4.2 Hypotheses Testing

To evaluate our research model, we compiled three regression models, which we assessed using Ordinary Least Squares (OLS) regression. Table 2 summarizes the results of these three regressions.

The baseline model which consists exclusively of the control variables already explains the productivity of OSS teams to a moderate degree ($R^2 = 0.300$). In this model, team size has no significant effect on productivity ($\beta = -0.061$, $p = 0.223$). In contrast, team experience has a weak negative effect on productivity ($\beta = -0.106$, $p = 0.029$), suggesting that OSS developers become less productive, the longer they are working with each other. Moreover, project size has a strong positive effect on productivity ($\beta = 0.402$, $p < 0.001$), indicating that the size of a project's codebase stimulates OSS developers' productivity. Project age has a weak positive effect on OSS developers' productivity ($\beta = 0.189$, $p < 0.001$). This suggests that developers are more productive in older OSS projects. Finally, developers' level of project experience has a moderate positive effect on their productivity ($\beta = 0.402$, $p < 0.001$), indicating that OSS developers with more project experience are more productive.

The second regression model adds OSS developers' spatial, temporal, and cultural distances to the control variables and has a considerably higher explanatory power for OSS teams' productivity ($R^2 = 0.513$). The results of this regression model suggest that each of the three distance forms has a distinct effect on OSS team productivity. Spatial distances between OSS developers have a moderate negative effect on team productivity ($\beta = -0.298$, $p < 0.001$). This supports hypothesis 1, which states that OSS team productivity decreases, the more developers work spatially apart from each other. Furthermore, OSS teams' temporal distances have a moderate positive effect on their work productivity ($\beta = 0.334$, $p < 0.001$). However, our measure is inverse as it considers the overlap of developers' working hours. Thus, this result, suggests a negative effect between developers' non-overlapping working hours and their productivity, which is in line with hypothesis 2. Finally, cultural distances have a weak negative effect on the productivity of OSS teams ($\beta = -0.200$, $p < 0.001$). This finding supports hypothesis 3 which states that cultural distances hinder OSS teams' productivity. Beside the influence of the three distance forms, team size has a moderate negative effect on OSS team' productivity ($\beta = -0.307$, $p < 0.001$) while team experience has no significant effect on OSS team productivity ($\beta = -0.026$, $p = 0.532$). Project size has a weak positive effect ($\beta = 0.219$, $p < 0.001$) and project age a weak positive influence ($\beta = 0.089$, $p = 0.011$) on team members' productivity. As in the baseline model, OSS developers' level of project experience has a weak positive effect on their collaboration productivity ($\beta = 0.225$, $p < 0.001$).

In the third regression model, we examine the interaction effect of OSS developers' direct offline ties on the negative effects of spatial, temporal, and cultural distances on OSS teams' productivity. This model has the highest explanatory power of all OLS regression models ($R^2 = 0.532$). As in the second regression model, spatial distances have a moderate negative effect ($\beta = -0.310$, $p< 0.001$) on OSS teams' productivity. Moreover, members' temporal and cultural distances have a moderate positive, respectively, weak negative effect on team productivity (β

Table 2: OLS Regression Results (n = 648)

	Model 1 (Baseline)		Model 2 (Geo. Dispersion)		Model 3 (Geo. Dispersion + OT)	
Control Variables						
team_size	-0.061		-0.307	***	-0.298	***
team_exp	-0.106	*	-0.026		-0.024	
project_size	0.402	***	0.219	***	0.284	***
project_age	0.189	***	0.089	*	0.172	***
project_exp	0.402	***	0.225	***	0.215	***
Geo. Dispersion						
spatial_dist			-0.298	***	-0.310	***
temporal_dist[1]			0.334	***	0.307	***
cultural_dist			-0.200	***	-0.145	***
OT					0.170	***
Interaction						
OT X spatial_dist					0.067	*
OT X temporal_dist					-0.006	
OT X cultural_dist					0.083	*
R^2	0.300		0.513		0.532	
ΔR^2			0.213	***	0.019	***

Notes: * = $p < 0.05$, ** = $p < 0.01$, *** = $p < 0.001$,
[1] = temporal_dist measures members' time overlap, hence a positive coefficient represents a negative relationship between 'temporal distance' and productivity

= 0.307, p < 0.001; β = -0.145, p < 0.001). Developers' direct offline ties have a weak positive effect on OSS teams' productivity (β = 0.170, p < 0.001). In addition, the regression analysis reveals an interaction effect between OSS developers' direct offline ties and their spatial and cultural distances. The multiplication term of developers' spatial distance and their offline ties has a weak positive effect on their productivity (β = 0.067, p = 0.028). This indicates that the negative effects of developers' spatial distances on team productivity are not only reduced but even slightly reversed through direct offline interactions between OSS developers, which supports hypothesis 4. Similarly, the interaction term between OSS developers' cultural distances and their direct offline ties has a weak positive effect on productivity (β = 0.083, p = 0.030). This finding supports hypothesis 6 by providing evidence that developers' direct offline contact mitigates and even slightly reverses the productivity deficits caused by their cultural distances. However, there is no significant interaction between OSS developers' temporal distances and their direct offline ties (β = -0.006, p = 0.628), which does not support hypothesis 5.

4.3 Post-Hoc Analysis on Offline Tie Efficacy

The results of our regression analysis suggest that direct offline relationships mitigate and even slightly reverse some of the problems associated with geographically dispersed teamwork. To achieve a better understanding of the stimulating effects of direct offline ties on OSS teams' productivity, we perform several post-hoc analyses. In particular, we seek to identify those situations in which OSS teams benefit most from direct offline relationships between developers. To do so, we examine closely the interaction effect of offline ties with spatial and cultural distances. Moreover, with respect to research by Singh et al. (2011b), we examine if there is a similar interaction effect for team members' level of project experience. For this analysis, we segment the three variables spatial distance (*spatial_dist)*, cultural distance (*cultural_dist)*, and project experience (*project_exp)* into four equally sized subgroups based on their quartiles. Next, we examine how the average team productivity in each of the different subgroups changes with an increasing level of direct offline ties among OSS developers. We refer to the formed subgroups in the following as: '*low'* (0 – 0.25 percentile), '*mid low*' (0.25 – 0.50 percentile), '*mid high*' (0.50 – 0.75 percentile), and '*high*' (0.75 – 1.00 percentile). Figure 4 depicts the interaction plots for each of these subgroups.

The effects of direct offline ties differ based on OSS teams' level of spatial distances. If there is no direct offline contact, OSS teams with '*low*' spatial distances perform best followed by teams with '*mid low*', '*mid high*' and '*high*' spatial distances. However, this picture changes substantially with the existence of direct offline ties among developers. OSS teams with '*mid high*' and '*high*' spatial distances profit the most from direct offline ties among developers and soon take the lead productivity-wise while teams with '*low*' spatial distances benefit only marginally.

A similar effect can be observed for cultural distances. In situations when developers have no direct offline contact with each other, those OSS teams with the lowest degree of cultural distances perform best, followed by teams with '*mid low*', '*mid high*', and '*high*' cultural distances. However, this order is completely reversed if there are some direct offline ties among developers. Then, the teams with the highest degree of cultural distances become the most productive ones, followed by teams with '*mid high*' and '*mid low*' degrees of cultural distances. In contrast, teams with '*low*' cultural distances perform lowest as soon as OSS developers have a low degree of direct offline ties.

Finally, we look at whether there is a similar interaction between team members' project experience and their offline ties. It turns out that when there are no direct offline ties among developers, those teams with '*mid high*' and '*high*' project experience perform best, followed by OSS teams with, '*mid low*', and '*low*' project experience. The teams with '*low*' project experience benefit most from direct offline ties, while teams with '*mid low*' and '*high*' levels of project experience also experience performance gains but to a lesser degree. Surprisingly, we observe that the productivity of OSS teams with '*high*' levels of project experience is not stimulated by developers' direct offline ties but instead decreases.

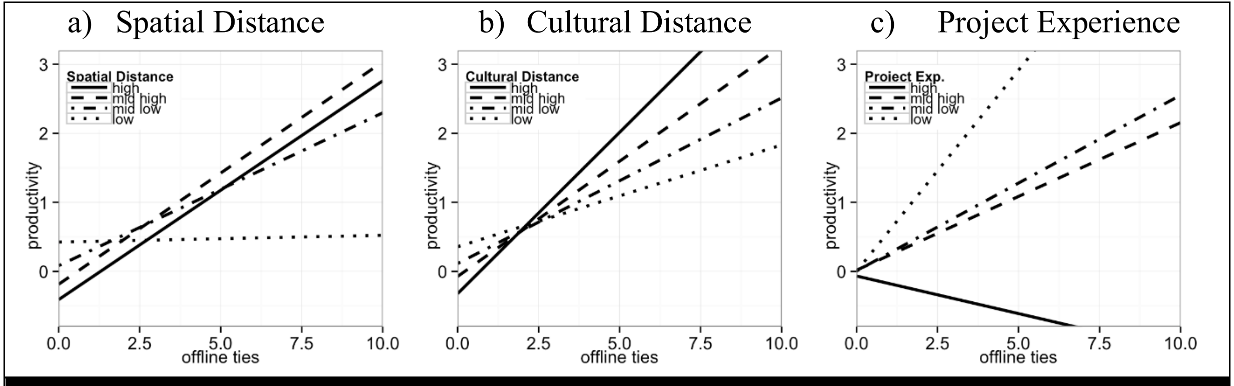


Figure 4. Visualization of the Interaction Effects

5 Discussion

With the massive growth of OSS initiatives and their geographically dispersed developer base, OSS projects are considered to be forerunners for virtual teams and global communities in general (Malone 2004). Yet, the most essential question of whether the experienced productivity gains in OSS projects can be attributed to developers' geographic dispersion or rather to a particular project means, remains unclear. In this research, we tried to open the black box of the effects posed by geographical dispersion on OSS teams' productivity. Building on organizational literature, we consider geographic dispersion as a multi-dimensional construct encompassing spatial, temporal, and cultural distances. We suppose that each of these distances hinders OSS teams' productivity in a distinct way. As a potential counter measure, we propose and evaluate the use of direct offline interactions between OSS developers. Our empirical evaluation of 648 OSS teams within KDE largely supports our research hypotheses and provides further insights into overcoming the challenges associated with OSS developers' geographic dispersion. In this section, we summarize the key findings of our research and outline theoretical and practical implications for the OSS and the organizational domain.

5.1 Key Findings

- **Geographic dispersion is a multi-dimensional construct**

Our empirical evaluation suggests that geographic dispersion should not be reduced to one particular dimension (i.e. spatial or temporal) but instead should be considered as a multi-dimensional construct. In our analyses there is little cross-correlation among OSS developers' spatial, temporal, and cultural distances, which in turn supports our assumption that each of these geographic distances has a distinct effect on OSS developers' collaboration. The high explanatory power of our second regression model ($R^2 = 0.513$), which considers OSS developers' spatial, temporal, and cultural distances and the controls, further underlines the relevance of this conceptualization. This high level of explanatory power is especially salient compared to other regression analyses in the OSS context (Daniel et al. 2013, Colazo and Fang 2010) which considered only one aspect of geographic dispersion.

- **Spatial, temporal, and cultural distances mitigate OSS teams' productivity**

Another important takeaway of our empirical analysis is that, spatial, temporal, and cultural distances, overall, hinder the productivity of OSS teams. In particular, spatial and temporal distances pose high burdens for the productive interplay of OSS developers. In comparison, cultural distances have a negative effect, too, but to a lesser degree.

- **Direct offline ties between OSS developers reverse the negative effects of spatial and cultural distances**

Our regression analysis reveals a negative interaction between OSS developers' direct offline ties and their spatial and cultural distances. Direct offline relationships not only reduce the negative effects of these distances but even transform them into positive effects for OSS teams' productivity. Alas, our analysis suggests that direct offline ties are no help for OSS teams in overcoming temporal distances.

- **OSS teams with high spatial distances outperform teams with low spatial distances as soon as there are few direct offline ties between the involved developers**

Our post-hoc analysis reveals that there is no unitary interaction between OSS developers' spatial distances and their direct offline ties. In situations when there is no offline contact, spatial distances between OSS developers go hand in hand with performance deficits. However, as soon as OSS developers establish a low degree of direct offline interactions, teams with high and mid-high spatial distances outperform teams with low and mid-low spatial distances.

- **Cultural distances become synergetic for OSS teams' productivity as soon as there is a low degree of direct offline ties among the involved OSS developers**

Another key insight of our evaluation is that OSS teams can reverse the negative effects of cultural distances as soon as developers have at least a few direct offline ties with each other. As for spatial distances, OSS teams with the less cultural distances perform better when there are no direct offline ties among OSS developers. If there is at least some direct offline interaction between developers, this picture changes completely. From this point on, the teams with more cultural distances perform best.

- **Direct offline interactions are especially valuable for OSS teams whose developers have little experience with the OSS project**

Finally, our post-hoc analysis reveals an unexpected interaction effect between OSS developers' level of project experience and their direct offline ties. Specifically, our evaluation suggests that those OSS developers with low, mid-low, and mid-high project experience benefit most from an increasing degree of direct offline ties. Conversely, offline relationships reduce the productivity of teams whose members have a high project experience. An explanation for this effect could be that once OSS developers are experts in the project they might consider offline interactions solely as social events and consequently be less motivated to actively contribute to the project than to stay only active to keep connected with each other.

5.2 Limitations

There are several limitations in our research which we would like to point out. First, there are various ways of looking at geographic dispersion in OSS teams. While we draw on central lessons from teamwork literature, we do not claim that our multi-dimensional conceptualization is complete or that it is the only way to look at geographic dispersion in OSS teams. Instead, our analysis provides grounds for thorough OSS and organizational studies on geographically dispersed teamwork. Besides complementing our conceptualization with further geographic dimensions like developers' experience within the OSS domain, future studies could use other methodological approaches for evaluating geographic distances in OSS teams. Specifically, future studies could examine if the number and size of spatial, temporal, and cultural clusters also affect the performance of OSS teams.

Secondly, we focus in our research on the role of direct offline relationships between OSS developers. We do this because we posit that it is the enhanced abilities to transfer knowledge and build social relations in face-to-face interaction which help OSS developers to overcome the productivity deficits caused by their geographic dispersion. However, by doing so, we disregard the effects of OSS developers' indirect offline relationships. In light of recent research by Zhang and Venkatesh (2013), which suggests an interaction between team members' direct and their indirect offline relationships, this shortcoming provides various opportunities for future OSS studies. Beside evaluating whether a similar interaction effect can be observed in the OSS domain, future research could build on Singh et al. (2011a) and elaborate on the effects of structural holes and of average shortest paths in OSS developers' offline interaction networks.

The third limitation concerns our conceptualization and measurement of cultural distance for which we rely on research by Hofstede (1980). While Hofstede's research is supported by various studies in the management field, it is widely used but not fully accepted (Kirkman et al. 2006). A particular critique is that it over-simplifies individuals' culture and ignores cultural differences within countries. Beside this conceptual limitation, our measure for cultural distances is based on KDE developers' location data. Although a test with a random sample of KDE developers revealed an overwhelming overlap between KDE developers' country of residence and their nationality, we cannot rule out that this estimation biased the validity of our evaluation. Future research should, therefore, complement our findings by adopting other conceptualizations and dedicated measures for cultural differences among OSS developers.

Finally, our concentration on OSS teams within KDE limits the generalizability of our research results. Although this concentration provides several advantages for our evaluation (i.e. shared conventions for coding, code submissions, collaboration, etc.), it reduces the ability to generalize our findings for other OSS projects. Future research may address this shortcoming and examine the role of geographic dispersion and direct offline interactions with a more diversified sample of OSS projects.

5.3 Implications for the OSS Domain

Our study contributes to the emerging literature which examines the role of team characteristics on OSS teams' productivity. To do so, we seek to understand if and how OSS developers' geographic dispersion affects their productive interplay. As one way to overcome the negative effects of geographic dispersion, we suggest and evaluate the use of direct offline interactions between OSS developers. To the best of our knowledge, this is the first empirical evaluation in the OSS domain which looks at geographic dispersion as a multi-dimensional construct. By doing so, our research helps to bring together the findings of prior OSS studies which focused on only one of these distances (Colazo and Fang 2010, Padmanabha 2007, Daniel et al. 2013) and helps to derive a comprehensive understanding of the nuanced effects of OSS developers' geographic dispersion. In line with our theorizing, the results of our empirical evaluation suggest that the productivity of OSS teams can be explained to a considerable degree by developers' spatial, temporal, and cultural distances. Thus, future studies should not only account for the effects of these distances when examining the productivity of OSS teams, but also examine whether similar geographic effects can be observed for the attraction and retention of OSS developers.

Our research opposes the conventional wisdom that OSS teams are immune to the negative effects of geographic dispersion. Instead, our study suggests that spatial and cultural distances have neither a solely positive nor a solely negative effect on OSS teams' productivity. In line with conceptual research by Ågerfalk et al. (2005), our evaluation supports the notion of spatial and cultural distances as both a gain and a burden for effective teamwork in OSS projects. Spatial and cultural distances provide substantial benefits for effective collaboration, such as the possibility to lever a global workforce or to combine a variety of skills and expertise, though our evaluation suggests that these productivity gains require direct offline interactions among members. If there is no offline contact between OSS developers, the negative aspects of spatial and cultural distances prevail and complicate effective teamwork. This observation backs previous OSS studies which suggest that OSS developers' spatial (Hu et al. 2012) and cultural distances (Daniel et al. 2013) constrain productive teamwork. However, with offline interactions among developers, the positive aspects of spatial and cultural distances soon outweigh the downsides and make OSS teams with those distances become most productive. While our evaluation can only attest to this interaction, future studies should examine how offline interactions unleash these productivity gains. In light of team cognition literature, it seems likely that offline interactions help OSS developers to establish a better mutual understanding and interpersonal relationships, which in turn helps them to mitigate the problems associated with spatial and cultural distances. However, future studies should look more closely into this aspect. It seems to be particularly worthwhile to examine if and how offline relationships foster the formation of cognitive and affective trust among OSS developers (McAllister 1995) and if such trust relationships transform into productivity gains. Irrespective of OSS developers' offline interactions, our evaluation suggests that temporal distances throughout have a negative effect on the productivity of OSS teams. This is contrary to study results by Colazo and Fang (2010), which instead suggest that developers' temporal distances stimulate their productive collaboration. An explanation for this discrepancy could be the use of different measures for assessing temporal distances in OSS teams. While Colazo and Fang (2010) measure temporal distance based on the variance in the time of OSS developers' first code commit each day, our measure assesses and weights the actual overlap in OSS developers' working hours. Considering the common skewed work distribution in OSS projects (Toral et al. 2010), and the associated heterogeneous coordination needs, we argue that our measure captures temporal distances in OSS teams better and recommend it to future studies. Another explanation for the discrepancy could be that the OSS teams studied by Colazo and Fang (2010) were more effective in coping with the problems of temporal distances than teams within KDE. This would raise the question of how OSS projects can overcome the negative effects of temporal distances and why KDE teams are more affected by them than other OSS teams.

Finally, our research adds to literature by bringing together OSS developers' offline and online interaction contexts. While previous evaluations by Singh et al. (2011a) stress the importance of OSS developers' online relatedness, we examine the role of their offline connectedness in achieving effective collaboration. Following Crowston et al. (2007), we argue that offline meetings provide OSS developers with a rich interaction context and social relationships which help them mitigate the productivity challenges of their geographic dispersion. As far as we know, our study offers the first empirical evaluation of the effects of OSS developers' offline relationships. In particular, our findings empirically support the assumptions of prior interview studies and reveal that direct offline contact between OSS developers indeed helps overcoming and even slightly reverses the collaboration problems caused by spatial and cultural distances. Future research may draw on our findings to study if there is a similar interplay of offline and online interactions which affects OSS developers' attraction and retention.

Beside these implications for OSS literature, our evaluation provides several managerial lessons for OSS projects. Most importantly, OSS project managers should not take the effective interplay between OSS developers for granted but instead consider it as contingent upon their spatial, temporal, and cultural distances. Moreover our evaluation stresses the need for project managers to base their staffing decisions on the ability to conduct offline meetings between the involved developers. If offline meetings are not possible, OSS projects are better off bringing together developers with little spatial and cultural distances. For such endeavors a 'cathedral-like' development approach can be appropriate if code is developed in private and only published with a software release (Raymond 1999). However, if it is possible to arrange offline meetings, project managers should favor a spatially and culturally dispersed developer base, which can typically achieved by developing code openly in a 'bazar-like' approach (Raymond 1999). In addition, OSS projects can employ a combination of both approaches, such as a 'cathedral-like' approach with a globally dispersed developer group which meets regularly offline or a 'bazar-like' approach within geographic borders such as OSS projects which are only shared within a university intranet.

5.4 Implications for the Organizational Domain

Considering the explicit recommendation from scholars to consider knowledge workers as volunteers (Drucker 2002) and the notion of OSS teams as prime examples of virtual collaboration within corporations (Malone 2004), the results of our empirical study also contribute to the organizational domain. Specifically, our study addresses a fundamental shortcoming of most teamwork studies which is the uni-dimensional and dichotomous differentiation of team members' geographic dispersion (Cummings et al. 2009, Hinds and Mortensen 2005, O'Leary and Cummings 2007). Instead, our multi-dimensional conceptualization of geographic dispersion in terms of spatial, temporal, and cultural distances offers grounds for bringing the isolated findings of previous studies together and helps to derive an integrated understanding of the distinct effects of geographic dispersion. The results of our evaluation provide empirical support to the use of this multi-dimensional conceptualization and highlight its relevance for understanding members' effective interplay. Future studies examining aspects of geographic dispersion should, therefore, employ a similar conceptualization. Beside the evaluation of other distance forms (i.e. organizational boundaries), future research could use our results as a foundation for elaborating organizational strategies on the most effective way to lever a global workforce.

In addition, we contribute to organizational research by revealing that geographic dispersion per se is neither a catalyst for nor a barrier to productive teamwork. Instead, it depends on the existence of prior offline interactions between the involved members whether their geographic dispersion is a gain or a hindrance to productive teamwork. Spatial and cultural distances pose a barrier to effective collaboration if team members have not met offline with each other. This supports organizational studies which warn of the effects of combining members from different regions and cultures. However, as soon as there is little offline interaction among the members involved, this picture is inverted and team members' spatial and cultural distances become a facilitator for team productivity. The ambivalent role of spatial and cultural distances not only offers a possible explanation for the mixed findings in previous evaluations but provides further avenues for future studies. One possible topic is the elaboration of members' cognitive and affective experiences in offline interactions which convert their spatial and cultural differences into productivity boosts. Based on such an evaluation, other studies could then examine if there is an alternative to offline interactions to help achieve similar productivity gains.

Furthermore, our study contributes to organizational literature by examining the specific situations under which offline interactions stimulate the productivity of geographically dispersed teams. Specifically, we observe that team members with little project experience benefit most from direct offline ties, which is consistent with the advice given by Siebdrat et al. (2009) to organize offline meetings when members are new to the project. Interestingly, we observe that more offline meetings are not always a gain for productive teamwork. Conversely, it turns out that offline meetings mitigate the productivity of teams whose members are highly experienced with the project. Apart from examining whether similar effects can be also observed in corporate teams, future studies should use a relational perspective to find out which team or individual behavior underlies this effect.

Finally, our study complements organizational research which predominantly focused on either team members' online or offline interactions. Prior studies which sought to combine these two domains such as research by Kirkman et al. (2004) were the exception. By drawing on research by Zhang and Venkatesh (2013), our empirical evaluation uncovers an important interaction between team members' direct offline interaction and their spatial and cultural distances which helps increase understanding of the contradictory results of previous studies. By doing so, our study supports the advice of Faraj and Johnson (2011) to consider the network type for understanding the different characteristics of individuals' interactions and reveals the complementary character of offline interactions for members' online behavior. Future studies should draw on our findings and examine whether members are aware of the distinct characteristics of their offline and online interactions and whether the observed complementary effects between both domains are noticed.

Considering the common escalation and underperformance of software projects in the organizational domain (Keil and Mann 2000, Solomon 2010), our research also offers some practical advice for the staffing and management of geographically dispersed teams in corporations. Most importantly, our evaluation stresses the importance of basing staffing decisions for such teams on the ability to arrange offline meetings. If offline meetings are not possible, managers should minimize the spatial and cultural distances between members. However, if offline interactions between team members are possible, project managers should adopt completely the opposite approach and bring together members with high spatial and cultural distances because direct offline interactions transform these distances into productivity gains. For all team configurations, it is advisable that temporal distances among members should be avoided as they considerably harm the effective interplay of team members.

6 Conclusion

In this research, we examine if and how geographic dispersion mitigates the productivity of OSS teams and propose that direct offline relationships among the involved developers can compensate for these deficits. To examine this research question, we draw on teamwork literature and consider geographic dispersion as a multi-dimensional construct encompassing spatial, temporal, and cultural distances. An empirical investigation of 648 teams developing OSS provides evidence that spatial, temporal, and cultural distances have a negative yet distinct effect on team productivity. Furthermore, our study results show that only a small degree of direct offline contact is necessary for OSS teams to transform developers' spatial and cultural distances into synergies for effective collaboration. Finally, our empirical evaluation highlights the value of offline meetings for OSS developers who have little experience with the project.

7 References

Ågerfalk, P. J., B. Fitzgerald, H. Holmström, B. Lings, B. Lundell, E. Ó. Conchúir. 2005. A framework for considering opportunities and threats in distributed software development. *In Proceedings of the International Workshop on Distributed Software Development*, 47–61.

Aiken, L. S., S. G. West. 1991. *Multiple regression*: *Testing and interpreting interactions.* Sage, Newbury Park, California.

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

Bird, C., N. Nagappan. Who? Where? What? Examining distributed development in two large open source projects *9th IEEE Working Conference on Mining Software Repositories (MSR)*, 237–246.

Colazo, J. A., Y. Fang. 2010. Following the Sun: Temporal Dispersion and Performance in Open Source Software Project Teams. *Journal of the Association for Information Systems* **11**(12) 684–707.

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

Crowston, K., J. Howison, C. Masango, U. Eseryel. 2007. The Role of Face-to-Face Meetings in Technology-Supported Self-Organizing Distributed Teams. *IEEE Transactions on Professional Communication* **50**(3) 185–203.

Cummings, J. N., J. A. Espinosa, C. K. Pickering. 2009. Crossing Spatial and Temporal Boundaries in Globally Distributed Projects: A Relational Model of Coordination Delay. *Information Systems Research* **20**(3) 420–439.

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

Deshpande, A., D. Riehle. 2008. The Total Growth of Open Source. B. Russo, E. Damiani, S. Hissam, B. Lundell, G. Succi, eds. *Open Source Development, Communities and Quality*. Springer US, Boston, MA, 197–209.

Drucker, P. F. 2002. They're not Employees, They're People. *Harvard Business Review* **80**(2) 70–77.

Espinosa, A., E. Carmel. 2003. The impact of time separation on coordination in global software teams: a conceptual foundation. *Software Process: Improvement and Practice* **7**(1).

Espinosa, J. A., J. N. Cummings, C. Pickering. 2012. Time Separation, Coordination, and Performance in Technical Teams. *IEEE Transactions on Engineering Management* **59**(1) 91–103.

Espinosa, J. A., W. DeLone, G. Lee. 2006. Global boundaries, task processes and IS project success: a field study. *Information Technology & People* **19**(4) 345–370.

Faraj, S., S. L. Johnson. 2011. Network Exchange Patterns in Online Communities. *Organization Science* **22**(6) 1464–1480.

Gellert, W., S. Gottwald, M. Hellwich, H. Kästner, H. Küstner. 1989. *The VNR concise encyclopedia of mathematics,* 2nd ed. Van Nostrand Reinhold, New York.

Gibson, C. B., J. L. Gibbs. 2006. Unpacking the Concept of Virtuality: The Effects of Geographic Dispersion, Electronic Dependence, Dynamic Structure, and National Diversity on Team Innovation. *Administrative Science Quarterly* **51**(3) 451–495.

Goth, G. 2007. Sprinting toward Open Source Development. *IEEE Software* **24**(1) 88–91.

Greene, W. H. 2003. *Econometric analysis,* 5th ed. Prentice Hall, Upper Saddle River, N.J.

Hinds, P. J., M. Mortensen. 2005. Understanding Conflict in Geographically Distributed Teams: The Moderating Effects of Shared Identity, Shared Context, and Spontaneous Communication. *Organization Science* **16**(3) 290–307.

Hofstede, G. 1980. *Culture's consequences: International differences in work-related values*.

Hofstede, G. J., M. Minkov. 2010. *Cultures and organizations*: *Software of the mind ; intercultural cooperation and its importance for survival,* 3rd ed. McGraw-Hill, New York, NY.

Hu, D., J. L. Zhao, Chen Jiesi. 2012. Reputation Management in an Open Source Developer Social Network: An Empirical Study on Determinants of Positive Evaluations. *Decision Support Systems* **53**(3) 526–533.

Huettinger, M. 2008. Cultural dimensions in business life: Hofstede's indices for Latvia and Lithuania. *Baltic Journal of Management* **3**(3).

Kankanhalli, A., B. C. Tan, K.-K. Wei. 2007. Conflict and Performance in Global Virtual Teams. *Journal of Management Information Systems* **23**(3) 237–274.

Keil, M., J. Mann. 2000. Why Software Projects Escalate: An Empirical Analysis and Test of Four Theoretical Models. *Management Information Systems Quarterly* **24**(4) 631–664.

Kirkman, B. L., K. B. Lowe, C. B. Gibson. 2006. A quarter century of Culture's Consequences: a review of empirical research incorporating Hofstede's cultural values framework. *Journal of International Business Studies* **37**(3) 285–320.

Kirkman, B. L., B. Rosen, P. E. Tesluk, C. B. Gibson. 2004. The Impact of Team Empowerment on Virtual Team Performance: The Moderating Role of Face-to-Face Interaction. *Academy of Management Journal* **47**(2) 175–192.

Kuk, G. 2006. Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List. *Management Science* **52**(7) 1031–1042.

Malik, T. H., Y. Zhao. 2013. Cultural distance and its implication for the duration of the international alliance in a high technology sector. *International Business Review* **22**(4) 699–712.

Malone, T. W. 2004. *The future of work*: *How the new order of business will shape your organization, your management style, and your life.* Harvard Business School Press, Boston, Mass.

Martins, L. L., L. L. Gilson, M. T. Maynard. 2004. Virtual Teams: What Do We Know and Where Do We Go From Here? *Journal of Management* **30**(6) 805–835.

Maznevski, M. L., K. M. Chudoba. 2000. Bridging Space Over Time: Global Virtual Team Dynamics and Effectiveness. *Organization Science* **11**(5).

McAllister, D. J. 1995. Affect-and Cognition-Based Trust as Foundations for Interpersonal Cooperation in Organizations. *Academy of Management Journal* **38**(1) 24–59, 10.2307/256727.

Nurmi, N. 2010. Work stressors related to geographic distance and electronic dependence in virtual teams. *International Journal of Business and Systems Research* **4**(3) 311.

O'Leary, M. B., J. N. Cummings. 2007. The Spatial, Temporal, and Configurational Characteristics of Geographic Dispersion in Teams. *Management Information Systems Quarterly* **31**(3) 433–452.

Padmanabha, R. 2007. FLOSS (Free/Libre Open Source Software): A Theme for Cultural Differences Study, Jindal Global Law School (JGLS), Jindal Global University (JGU), New Delhi.

Raymond, E. S. 1999. *The cathedral and the bazaar*: *Musings on Linux and open source by an accidental revolutionary,* 1st ed. O'Reilly, Sebastopol.

Scellato, S., C. Mascolo, M. Musolesi, V. Latora. 2010. Distance matters: geo-social metrics for online social networks. USENIX Association, ed. *Proceedings of the 3rd conference on Online social networks*.

Siebdrat, F., M. Hoegl, H. Ernst. 2009. How to Manage Virtual Teams. *MIT Sloan Management Review* **50**(4).

Singh, P. V., Y. Tan, V. Mookerjee. 2011a. Network Effects: The Influence of Structural Social Capital on Open Source Project Success. *Management Information Systems Quarterly* **35**(4) 813–829.

Singh, P. V., Y. Tan, N. Youn. 2011b. A Hidden Markov Model of Developer Learning Dynamics in Open Source Software Projects. *Information Systems Research* **22**(4) 790–807.

Sinnott, R. W. 1984. Virtues of the Haversine. *Sky and Telescope* **63**(3) 159.

Solomon, C. 2010. The Challenges of Working in Virtual Teams, http://rw-3.com/2012VirtualTeamsSurveyReport.pdf.

Stewart, K. J., S. Gosain. 2006. The Impact of Ideology on Effectiveness in Open Source Software Development Teams. *Management Information Systems Quarterly* **30**(2) 291–314.

Subramanyam, R., M. Xia. 2008. Free/Libre Open Source Software development in developing and developed countries: A conceptual framework with an exploratory study. *Decision Support Systems* **46**(1) 173–186.

Toral, S., M. Martínez-Torres, F. Barrero. 2010. Analysis of virtual communities supporting OSS projects using social network analysis. *Information and Software Technology* **52**(3) 296–303.

Zhang, X., V. Venkatesh. 2013. Explaining Employee Job Performance: The Role of Online and Offline Workplace Communication Networks. *Management Information Systems Quarterly* **37**(3) 695–722.

Paper V

In the Spotlight - Evaluating How Celebrities Affect FLOSS Developers’ Participation Motivation

Andreas Schilling
University of Bamberg
andreas.schilling@uni-bamberg.de

Sven Laumer
University of Bamberg
sven.laumer@uni-bamberg.de

Tim Weitzel
University of Bamberg
tim.weitzel@uni-bamberg.de

In: Proceedings of the 21st European Conference on Information System (ECIS 2013), Utrecht, Netherlands

The publication is available under:

http://aisel.aisnet.org/ecis2013_cr/208

Abstract

This study examines the influence of geographic dispersion on the productive collaboration of Open Source Software (OSS) developers. Building on teamwork literature, we consider OSS developers' geographic dispersion as a multi-dimensional construct consisting of spatial, temporal, and cultural distances. We posit that each of these distances has a distinct negative effect on OSS developers' productive interplay by reducing their level of shared knowledge and social relatedness. As a potential strategy for OSS teams to overcome the productivity deficits of these geographic distances, we evaluate the role of direct offline interactions which take place in sprint events. The results of our empirical evaluation with 648 OSS teams largely support our research hypotheses and suggest: (*i*) geographic dispersion should be considered as a multi-dimensional construct, (*ii*) spatial, temporal, and cultural distances have a distinct negative effect on OSS teams' productivity, (*iii*) direct offline interactions between developers transform the productivity deficits of spatial and cultural distances into productivity gains for OSS teams, (*iv*) direct offline interactions are especially valuable for the productive interplay of developers who have little experience with the particular OSS project. Finally, we compare the results of our research with previous research in the organizational and OSS domain, which has generally focused only on one particular dimension of geographic dispersion. Thereby, we derive possible questions for future studies and delineate specific practical advice.

Paper VI

In Goods We Trust - Are OSS Teams With Reputable Developers More Productive?[13]

Andreas Schilling
University of Bamberg
andreas.schilling@uni-bamberg.de

Sven Laumer
University of Bamberg
sven.laumer@uni-bamberg.de

Tim Weitzel
University of Bamberg
tim.weitzel@uni-bamberg.de

[13] The term OSS has been used instead of FLOSS in order to align with related articles in the targeted journal and not to express any theoretical difference.

Abstract

In this study, we examine if and how reputable developers increase the productivity of teams developing Open Source Software (OSS). Building on the social practice view of OSS development, we suppose that reputable developers stand out not only for their technical and behavioral competences but also for a deep internalization of the OSS culture. Because of this, we suppose that reputable developers increase members' technical competences and foster their sense of belonging to the OSS team. To our surprise, an empirical evaluation of 749 OSS team configurations reveals that reputable developers increase OSS teams' productivity only marginally. In order to understand the underlying reasons of this weak effect, we employed an individual-centric post-hoc analysis. The results of this dedicated post-hoc evaluation with 80 OSS developers indicate that reputable developers directly increase members' level of cognitive trust in the OSS team, but this form of trust is not directly linked to their individual productivity. Instead, it is members' level of affective trust in the OSS team which directly facilitates their individual productivity. However, this form of trust is not directly linked to the involvement of reputable OSS developers. Based on our multi-level evaluation, we propose a refinement to the definition of internal goods within the social practice view of OSS development. Moreover, our evaluation brings to the fore that the effects of team and project characteristics on individual behavior can be distinct from and even opposite of the effects on collective behavior. Finally, we point out that managers of online communities who wish to enhance effective collaboration should focus on activities which strengthen individuals' social bonds rather than bringing in reputable individuals.

Keywords: Open Source Software, Social Practice View, Reputation, Cognitive Trust, Affective Trust, Productivity, Multi-Level Evaluation.

'There is no 'I' in team but there is in win.' Michael Jordan

1 Introduction

In OSS (Open Source Software) development projects, individuals who are dispersed around the world collectively develop software by relying on computer mediated communication (CMC). During the last decade, software developed through such initiatives flourished, with code creation growing at exponential rates (Deshpande and Riehle 2008). Based on these high growth rates, researchers consider OSS development a prime example for effective knowledge sharing and virtual collaboration (von Krogh and von Hippel 2006) - a key challenge which is innate to the value creation process of most online communities (McLure Wasko and Faraj 2000). Though, a closer look reveals that most OSS projects are struggling for developer contributions (Fang and Neufeld 2009). Although, it has been shown, in this context, that reputable developers provide valuable assets for attracting new developers (Hu et al. 2012) and enhancing members' collaboration (Li et al. 2006, Kuk 2006, Schilling et al. 2013, Casaló et al. 2009) it remains to be seen if these assets also make OSS teams more productive. Such understanding, is of high practical relevance in helping managers of OSS initiative deriving specific actions to foster productive interplay among involved members.

In the following, we examine if and how reputable developers help OSS teams becoming more productive. To do so, we bring together individual- and structural-centric research approaches. We start from an individual-centric perspective to understand the collaboration benefits which can result from involving reputable OSS developers. To do so, we build on the social practice view of OSS development (von Krogh et al. 2012) and suppose that reputable developers foster members' technical competences as well as their feelings of belonging to the team. To evaluate our research hypothesis, we performed an empirical analysis of 749 OSS team configurations, based on the community endorsement of the involved developers and archival records from their previous contributions. With respect to the supposed cognitive and affective assets which reputable developers provide for OSS teams' productivity, we examine the research question: 'Are OSS teams with reputable developers more productive?'

Our work has various implications for OSS research. Foremost, our empirical evaluation of 749 OSS team configurations provides evidence that reputable developers increase OSS teams' productivity, but only to a marginal degree. Our individual-centric post-hoc evaluation with 80 OSS developers uncovers a reason for this marginal effect: having reputable OSS developers on a team only increases the level of cognitive trust among team members (their belief in their colleagues' competence), but this type of trust has no direct effect on their individual productivity. Instead, it is OSS developers' level of affective trust (their feeling of belonging) towards the OSS team which makes them more productive. However, this form of trust is not directly linked to the involvement of reputable developers. Building on this insight, we propose refining the definition of internal goods within the concept of the social practice view. In addition, our work provides lesson for managers of OSS projects such as to focus on activities which strengthen developers' personal bonds rather than bringing in reputable developers to make their teams more productive.

Moreover, our research has implications for the value creation in online communities. In particular, our multi-level research provides evidence that the effects of project and team characteristics on individual behavior can be distinct from and even opposite of the effects on collective behavior. In addition, our evaluation emphasizes the need to create a collective identity among community members to foster their engagement. Building on this insight, we recommend managers of online communities to focus on means which highlight members' shared interests rather than bringing in reputable individuals to facilitate effective collaboration.

This paper is structured as follows: After summarizing the current state of research on OSS team productivity, we develop our research hypothesis in Chapter 2. In Chapter 3, we outline our research methodology and our study of the collaboration experiences of 749 OSS teams. To better understand the results of this study, we perform a dedicated post-hoc analysis in Chapter 4. Finally, we conclude our work by discussing its implications for OSS projects, organizations and online communities.

2 Current State of Research on OSS Team Productivity

Literature on OSS team productivity is characterized by a duopolistic research approach, taking either an individual- or a structural-centric research perspective.

Individual-centric OSS studies commonly examine the particular motives that lead developers to engage in OSS projects. Most of these studies are based on Self-Determination Theory (SDT) (Ryan and Deci 2000). The basic tenet of SDT is that there are two motivation forms: intrinsic and extrinsic. Intrinsic motivation arises naturally; individuals with this motivation adopt a particular behavior not primarily for the outcomes which are associated with it but because of the satisfaction they derive from it. In contrast, behavior motivated extrinsically is adopted due to external stimuli (Ryan and Deci 2000). Various evaluations support the relevance of these different motivation forms for OSS developers' project behavior. Shah (2006) highlights that OSS developers need to perceive joy from their project work. At the same time, Roberts et al. (2006) provide evidence that monetary rewards are another effective stimulus for OSS developers' commitment. Moreover, studies provide evidence for the existence of various other extrinsic motives for OSS developers' project behavior, such as self-esteem (Ke and Zhang 2010, Stewart and Gosain 2006, Xu and Jones 2010). Despite the broad adoption of SDT in OSS literature, it is not without critics. One particular critique concerns the concrete stimulation of these motives between SDT and observations from the OSS domain. While SDT suggests that pecuniary rewards strengthen individuals' extrinsic motives and weaken their intrinsic motives, such effect cannot be observed within the OSS context (Roberts et al. 2006). Another critique concerns the basic assumption of SDT that individuals search for immediate outcomes of their behavior. However, in the OSS context, individuals engage in project work not only because of the immediate return but also due to social and ideological beliefs (Stewart and Gosain 2006). With respect to these critiques, OSS research recommends shifting away from considering OSS developers' motivation the pivotal point for their project commitment (von Krogh et al. 2012). Instead, recent studies emphasize the role of OSS developers' interactions with other team members to understand their commitment to the project. For example, Fang and Neufeld (2009) suggest that the motivations of OSS developers fluctuate and that their interactions with team members strengthen their sense of belonging to and identification with the OSS team, which in turn make them more productive. The important role of team members' relationships to each other is supported by Stewart and Gosain (2006). Despite the recent shift to OSS developers' interactions, it is unclear from an individual perspective which specific factors facilitate these interactions. According to Schilling et al. (2013), reputable OSS developers play a key role for team members' identification with and sense of belonging to the developer group. However, it remains to be seen if these positive effects also result in increased teamwork productivity.

In contrast, OSS studies taking a structural perspective abstract from individuals' perceptions and focus on the effects of project characteristics and topological aspects of team members' relationships to each other. By taking such a perspective, OSS studies have shown that the target audience, the operating system, and the software license have an important influence on OSS teams' productivity (Subramaniam et al. 2009). In addition, studies examining the structural characteristics of OSS developers' relationships to each other provide evidence for two distinct effects. In line with individual-centric studies, Singh et al. (2011a) provide evidence that strong relationships among team members foster their productive interplay. Furthermore, the authors reveal that OSS teams are most productive when their members have only moderately strong relations to developers outside their project. In their study, Grewal et al. (2006) take a closer look at these relationships and conclude that especially developers who had previously been involved in well-known OSS projects increase an OSS team's productivity. However, the way in which these studies infer relationships among OSS developers is not without criticism. In particular, Hu et al. (2012) point out that the approximation of personal relationships via membership overlaps seems to be rather weak. For example, such modeling approach does not differentiate between the resulting relationships among OSS developers if they have collaborated on a project for 100 days or only for 1 day. An empirical study by Hu and Zhao (2009) supports this critique by providing evidence that relationship networks derived through project affiliations differ substantially from networks which are based on developers' personal evaluations. While recent OSS studies have used conversational data to bridge this gap and better understand how team members' relationships to each other affect their productivity (Qureshi and Fang 2010, Singh et al. 2011b, Kuk 2006) there is to the best of our knowledge, no study which used more reliable data to examine how members' peer evaluation affect OSS teams' productivity.

This research uses a multi-level evaluation approach to examine if and how reputable developers make OSS teams more productive. To do so, we build on individual-centric research and consider OSS developers' interactions with each other central for their project commitment. Our theoretic foundation draws on the social practice view of OSS development (von Krogh et al. 2012), which takes a long-term view on individuals' project behavior and explicitly considers the effects imposed on them by their relationships to team members as well as their ethical considerations. In a second step, we evaluate our research hypothesis by performing an empirical evaluation based on 749 OSS teams, using developers' community endorsement and archival records from their previous contributions. In the next section, we detail the theoretic foundation on which we build and develop our research hypothesis.

3 Hypothesis Development

In this section, we develop our research hypothesis. We begin by outlining the social practice view of OSS development on which we build to understand the process of reputation building in OSS communities and the resulting cognitive and affective assets reputable developers provide to make OSS teams more productive.

3.1 The Social Practice View of OSS Development

In their recent work, von Krogh et al. (2012) base the social practice view of OSS development on the seminal work of Alasdair MacIntyre (1981). In contrast to motivation theories, which focus on people's search for immediate outcomes of their behavior, the proposed social practice view takes a long-term perspective on individuals' behavior and also considers the surrounding ethical aspects relevant to their behavior in the community. Another distinction to motivation theories is the assumption that individuals do not strive solely for immediate returns but also wish to achieve personal development. In the following, we present the central building blocks of MacIntyre's theory and their implications for the OSS domain as proposed by von Krogh et al. (2012).

The central building block of the theory is the social practice. According to MacIntyre (1981) a social practice is '*any coherent and complex form of socially established cooperative human activity through which goods internal to that form of activity are realized in the course of trying to achieve those standards of excellence which are appropriate to, and partly, definitive of, that form of activity*' (MacIntyre 1981, p. 187). In line with this generic definition, von Krogh et al. (2012) describe the social practice of OSS development as individuals' engagement in teams with related, relatively flat, peer-oriented, and decentralized communities. Thereby, the standards of excellence which OSS developers adhere to and apply encompass technical as well as behavioral guidelines. By pursuing a social practice, individuals generate internal and external goods. Internal goods are beneficial to all participants of the social practice as well as the community as a whole. In the context of OSS development, internal goods refer to the resulting code (if licensed appropriately), the joy of working with others and the enrichment of the OSS project with new features. In contrast, external goods are privately owned by the involved individuals. In the OSS context, such external goods are the derived individual reputation gains and the solution to one's particular software problem. Another distinction between internal and external goods concerns their provision. While internal goods are derived in the course of the social practice by adhering to the standards of excellence, external goods are provided by institutions (MacIntyre 1981). MacIntyre characterizes these institutions as organizations in which human cooperation is governed by rules and routines that exist beyond the presence and efforts of each individual (MacIntyre 1981, von Krogh et al. 2012). Applied to the OSS context, von Krogh et al. (2012) suggest considering the OSS movement itself an institution because it ensures that code contributed by developers can be studied openly by others, allowing them to derive reputation gains from their contributions. Despite their distinct provision, internal and external goods are not mutually exclusive but rather resemble two sides of the same coin. According to MacIntyre (1981) external goods cannot be acquired without the creation of internal goods and vice versa.

3.2 Reputation Building in OSS Communities

In the following, we build on the social practice view of OSS development to understand how developers build reputation in the OSS community. In contrast to previous evaluations which simply consider reputation gains as a result of OSS developers' contributions (Daniel et al. 2013, Grewal et al. 2006), we take a broader view and propose that OSS developers have to demonstrate not only their technical but also their collaboration competences in order to become reputable.

According to the social practice view of OSS development (von Krogh et al. 2012), developers enhance their community reputation through the positive evaluation of their work by members of the OSS community. Like the standards of excellence, this evaluation process applies not only to OSS developers' coding, but also to their collaboration behavior. In his comment on OSS development, Raymond (1999) points out that peers judge OSS developers not only based on their technical abilities but also considering their 'awareness and acculturation' with the OSS culture (Raymond 1999, p. 94). As a consequence, the standards of excellence are not only the basis for OSS developers' derivation of internal goods, but also define the ways in which their work is evaluated by members of the OSS community. Thus, by adhering to the standards of excellence, OSS developers produce in the course of their project work not only internal but also external goods. Because of this, we suppose that reputable developers stand out not only due to their technical and behavioral competence but also due to their internalization of the values and believes of OSS development. Another important aspect for OSS developers' reputation building is that not all community members are equally eligible to evaluate others. According to Raymond (1999) only OSS developers who are trusted to follow the code of excellence should evaluate others. The general validity of this advice for OSS communities is supported by the work of Stewart (2005). In this research, Stewart provides evidence that the reputation gains resulting from a positive evaluation are contingent on the evaluator's reputation in the OSS community.

With this take on reputation building in OSS communities, we reject the common notion that reputable OSS developers solely shine through their technical abilities and argue that their behavioral competences and their identification with the norms and beliefs of the OSS culture are also critical factors. Based on this nuanced view on reputable OSS developers, we outline in the next section their value for OSS teams' productivity.

3.3 Collaboration Assets of Reputable Developers

To understand how reputable developers can enhance OSS teams' productivity, we draw on the social practice view of OSS development and the reputation building process, which we outlined above. By doing so, we assume that reputable OSS developers are deeply familiar with the standards of excellence for OSS development. Thus, these developers know not only how to build code efficiently but also which collaboration processes facilitate productive OSS teamwork. Moreover, because these individuals enjoy a strong reputation in the OSS community, we suppose that they have the necessary power and respect to implement the necessary changes in OSS teams. Building on research by Schilling et al. (2013) and Kuk (2006), we suppose that reputable developers increase OSS teams' productivity through enhancing members' competencies and feelings of relatedness, rather than through providing short term incentives such as greater visibility. In the following, we outline the various advantages for productive teamwork resulting from having reputable developers in an OSS team.

With their rich OSS experience, reputable developers know how to use programming tools very well and implement functionality efficiently (Singh et al. 2011b). Thus, the work of such developers alone increases the productivity of an OSS team. In addition to their individual coding, we suppose that reputable OSS developers provide various *cognitive assets* for OSS teams' productivity. One way is by assisting team members. Due to their rich OSS experience, we suppose that reputable developers deeply believe in the values of sharing and caring (Stewart and Gosain 2006) and thus assist their colleagues in their project work and advise them on strategic decisions. For example, reputable developers could help their colleagues use a particular tool better or increase their skill in writing modular code, which in turn makes it easier for other developers to make additions and modifications to the codebase (Baldwin and Clark 2006). Schilling and Laumer (2012) support the positive effects of such helping behavior by providing evidence that OSS developers acquire considerably more knowledge when they are assisted by reputable developers. As Singh et al. (2011b) suggest, these knowledge gains, in turn, make members more productive because OSS developers with greater knowledge contribute more code in a given amount of time. Moreover, reputable developers can enhance the cognitive processes in OSS teams by establishing and maintaining a culture of regular work updates. By cultivating such information exchange, team members become aware of their colleagues' work and expertise. Such awareness benefits team members' effective collaboration twofold. First, it enhances their situational knowledge about the current work of their colleagues. This form of knowledge is particularly helpful for OSS developers to identify conflicts among their work at an early stage and to become aware of the problems their colleagues are struggling with. As a result, such work updates can considerably reduce the level of perceived uncertainty towards colleagues, which is considered an essential constraint for OSS developers' project commitment (Shah 2006). Second, these work updates make OSS developers aware of their colleagues' expertise, which helps them to identify the members who can assist them in their project work and thus reduce the time they are struggling with particular issues.

Beside these cognitive assets, we suppose that reputable developers provide various *affective assets* to enhance OSS teams' productivity. For example, by creating and cultivating open exchange among team members, reputable developers make their colleagues aware that they all share the wish to make the OSS project succeed. As Xu and Jones (2010) point out, the realization of such common goal can foster OSS developers' sense of belonging to and identification with the team. Moreover, we suppose that reputable developers foster this identification process by emphasizing the relevance of one's contribution for the project and its value to other members of the OSS community. Schilling et al. (2013) support this by providing evidence that reputable OSS developers help team members perceive their project work as meaningful and personally important. As Xu and Jones (2010) and Ke and Zhang (2010) suggest, such identification, leads OSS developers to increase their working efforts for the OSS project. Another way in which reputable developers can foster affective feelings among team members is by avoiding disputes from escalating. Debates between OSS developers can soon become heated and emotional (Bergquist and Ljungberg 2001), in such cases reputable developers can use their standing in the OSS community to objectify such discussion. If this is not possible, reputable developers can draw on their rich project experience to mitigate the caused damage for example by helping team members filtering out relevant critiques to their posts and ignoring destructive comments. Finally, we suppose that reputable OSS developers nurture feelings of relatedness among team members by increasing the derived enjoyment from their collaboration. This is because OSS developers derive satisfaction from acquiring and applying new knowledge (Fang and Neufeld 2009). Thus, reputable developers' assistance not only enhances members' competencies but also raises the derived satisfaction from their project work (Casaló et al. 2009). Moreover, reputable developers can pass on the OSS value of 'sharing and caring' to their team members and thus create a supportive team atmosphere. Such atmosphere not only makes it easier for team members to ask for help and increase their knowledge, but it also increases the level of satisfaction of the helper. With respect to the various cognitive and affective collaboration assets which can be levered in OSS teams through reputable OSS developers, we hypothesize that:

Hypothesis: *The involvement of reputable OSS developers increases an OSS team's productivity.*

In addition to controlling for the involvement of reputable developers, we control for various team and project characteristics which have been shown to affect OSS teams' productivity. In line with research by Colazo and Fang (2010), we suppose that a larger **team size** increases OSS teams' productivity because it enables developers to work on several aspects at the same time. Moreover, following Singh et al. (2011a), we suppose that OSS teams are more productive the more their members have worked with each other in the past. Thus, we control for OSS developers' **team experience**. Furthermore, we account for the effects of developers' **project experience**, as Singh et al. (2011b) provide evidence that OSS developers become more productive the longer they are actively contributing to an OSS project. Moreover, we control for two characteristics of the particular OSS project. Because OSS projects with bigger codebases provide more possibilities for OSS developers to add and modify code, we control for the **project size** of an OSS project. Finally, with respect to research by Singh et al. (2011a), we control for the **project age**, as code of mature OSS projects is often better documented and structured, which in turn makes it easier for developers to add and modify code (Sen et al. 2008).

4 Research Methodology

To evaluate our research hypothesis, we study the collaboration experiences of OSS teams within the 'K Desktop Environment' (KDE). KDE is a popular desktop environment for UNIX systems and includes a wide spectrum of OSS projects, such as games, organizers, and even entire office suites. By focusing on developer teams within KDE, we are able to study the collaboration experiences of a variety of OSS projects which share the same development context (i.e. programming language, licensing, tool chain, version control system, etc.). This benefits our evaluation substantially because it guarantees that the various team configurations in our study share the same 'standards of excellence' and 'institutional' characteristics in terms of the social practice view of OSS development. In the following, we outline the details of our data collection and the measures used in our evaluation.

4.1 Data Sample

The data used in our evaluation was extracted from two distinct sources. To derive statistics on KDE developers' contribution behavior, we downloaded the code repositories of all 65 KDE projects and wrote programming routines which analyzed the log files of the Version Control System (VCS) of these projects. In addition, we queried the Social Network Site (SNS) Ohloh.com for information on KDE developers' community endorsement. Using Ohloh.com, OSS users and developers can create a profile page and send each other 'Kudos', which are a form of appreciation for the work or the provided support (Hu et al. 2012). To extract this information, we queried the public Application Programming Interface (API) of Ohloh.com with developers' full name and the SHA-1 hash of the email address, which we extracted from their code commits. Moreover, we recursively extracted the Kudos received by each Kudo sender, in order to derive a comprehensive picture for the endorsement of all considered community members. In order to derive a comprehensive picture for OSS developers' community endorsement, we recursively extracted also the Kudos received by each Kudo sender. In total, this procedure led to the extraction of 8,195 Ohloh profiles and 34,300 Kudo relationships. In a second step, we identified those KDE projects for which we could assign at least 75 percent of all submitted code commits between January 1, 2011 and November 1, 2013 to developers with Ohloh accounts. This is the case for the following seven KDE projects: 'KDELibs' (cross-application libraries), 'KDE Workspaces' (a desktop organizer), 'Calligra' (an office suite), 'DigiKam' (a photo management suite), 'KDE PIM' (a personal organizer), 'Plasma-Mobile' (a desktop for mobile devices), and 'Akonadi' (a storage service for personal information).

Similar to Kuk (2006), we observe that there is a new combination of active developers at the KDE projects every week. This enables us to study various team configurations at the projects every week. Based on these team configurations and previous research by Singh (2010) we segmented our project data into weekly samples. As we are interested in OSS developers' collaboration, we filtered out all team configurations with fewer than three developers. After applying this filter, our study sample encompasses 749 team configurations (N). Figure 1 illustrates our sampling strategy using a fictitious example of four KDE developers. In this example, Mark is a frequent contributor and works one week with Carl and another week with Joe.

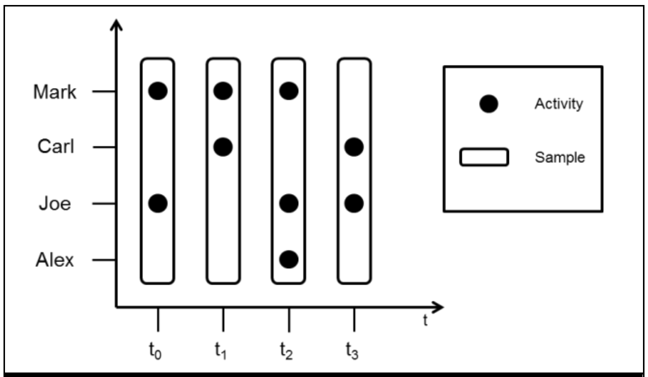


Figure 1: Sampling strategy

4.2 Measurement

4.2.1 Dependent Variable

In line with previous OSS studies, we assessed the **productivity** ***(prod^t)*** of an OSS team based on the average number of submitted code commits in period *t* (Singh et al. 2011a, Grewal et al. 2006). In order to account for changing team sizes, we computed the average number of code commits per developer. To do so, we determined those developers who were active in the project that week (*team^t*) and then aggregated the number of code commits each developer submitted that week (c_i^t). Finally, we divided the total number of commits by the number of active developers in *t* (*team_size^t*).

$$prod^t = \left(\sum\nolimits_{i \in team^t} c_i^t\right) \quad / \quad team\_size^t \tag{1}$$

4.2.2 Independent Variables

In order to assess OSS developers' **community reputation** ***(comm_rep^t)***, we transformed the extracted data from Ohloh.com in three distinct stages. In the first step, we constructed a global evaluation graph. In this directed graph, nodes represent OSS developers and edges between them resemble a positive evaluation (Kudo) of the receiving node by the sending node. Based on this evaluation graph, we computed each developer's reputation by employing a rank based measure. We do so, because a rank-based measure weights the effects of links based on the ranks of their originating node. In comparison, a degree based measure (e.g. the average in-degree), would treat all links in the graph as equally important. However, this would be inconsistent with our theorizing in which we consider that the resulting reputation gains from a positive evaluation are contingent on the community reputation of the evaluator (see Section 3.2). In particular, we rely on the PageRank algorithm proposed by Brin and Page (1998). This rank-based measure is a variant of the eigenvector centrality measure and uses the damping factor (*d*) to account for the decay in a node's influence on the ranks of subsequent nodes. This is consistent with our theorizing, as we suppose that members' endorsement by reputable OSS developers enhances their reputation, but it does not give them the same standing as that of their evaluators. Figure 2 depicts the evaluation graph, and visualizes the key difference between nodes' PageRank and their in-degree. In this example, the three nodes a, b, and c share the same in-degree, while their PageRank values differ substantially (a = 0.128, b = 0.076, c = 0.068). There are two more aspects which make the PageRank algorithm well-suited for our evaluation. First, the algorithm is a very efficient method for computing ranks in large datasets. This eases the computation of the ranks for the more than 8,000 nodes in our evaluation graph, which we calculate for each week between January 1, 2011 and November 1, 2013. Second, the PageRank algorithm is very robust against reciprocal linking (Gayo-Avello 2013). This topological characteristic describes the phenomenon that a node links to another node to get back a link from the receiver. Because reciprocal linking was also discovered in the Ohloh network (Hu et al. 2012), it is important to rely on a measure which is robust against such phenomenon. With regard to these benefits, we calculated the community reputation (*comm_rep$_i^t$*) for all members of the OSS community (*comm^t*) by computing their PageRank value at period *t*. Based on this

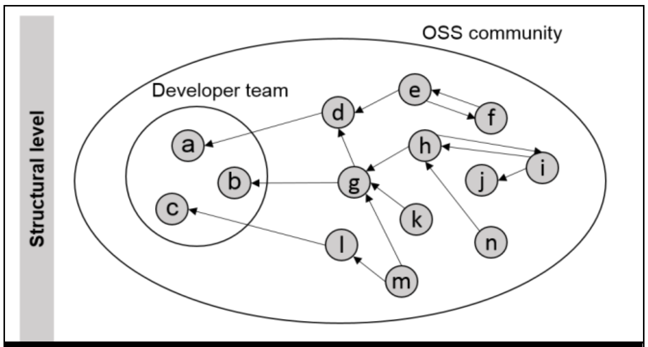


Figure 2: Evaluation graph

calculation, we assessed an OSS team's community reputation (*comm_rep^t^*) by calculating the average community reputation of the developers involved in period *t*.

$$comm\_rep_i^t = \frac{1-d}{|\ comm^t\ |} + d \times \sum\nolimits_{\forall j \exists kudo_{j,i} \wedge j \in comm^t} \frac{comm\_rep_j^t}{kudos_j} \quad (2)$$

$$comm\_rep^t = (\sum\nolimits_{i \in team^t} comm\_rep_i^t) \quad / \quad team\_size^t \quad (3)$$

4.2.3 Control Variables

We measured an OSS project's **team size (*team_size^t*)** by determining the cardinality of its set of active developers in period *t* (*team^t*). To calculate OSS developers' **team experience (*team_exp^t*),** we averaged the number of days that they have previously worked with each other ($D^t_{i,j}$) before period *t*. To assess OSS developers' **project experience (*proj_exp^t*)** we determined the average number of days (D^t_i) they had been active in the project prior to period *t*. To measure the **project size *(proj_size^t)***, we computed the total number of Lines of Code *(LoC^t)* of the OSS project's codebase in *t*. Finally, we assessed the **project age (*proj_age^t*)** based on the number of days (*NoD^t*) since the project's inception.

$$team\_size^t =|\ team^t\ | \quad (4)$$

$$team\_exp^t = (\sum\nolimits_{i \in team^t} \sum\nolimits_{j \in team^t \wedge j \neq i} D_{i,j}^t) \quad / \quad team\_size^t \quad (5)$$

$$proj\_exp^t = (\sum\nolimits_{i \in team^t} D_i^t) \quad / \quad team\_size^t \quad (6)$$

$$proj\_size^t = LoC^t \quad (7)$$

$$proj\_age^t = NoD^t \quad (8)$$

5 Results

5.1 Variable Transformation

Before we started our evaluation, we checked the cross-correlation of all independent variables. This check revealed some cross-correlations between the independent variables, which may result in multi-collinearity. To address this potential threat to validity, we followed the advice

Table 1: VIFs and cross-correlations

	VIF	1	2	3	4	5
1. Team size	1.738					
2. Team exp.	2.994	0.25***				
3. Project age	2.326	0.51***	0.47***			
4. Project size	1.722	0.38***	0.35***	0.48***		
5. Project exp.	2.819	-0.11***	0.71***	0.15***	0.23***	
6. Comm rep	1.414	-0.02	0.29***	0.10***	0.39***	0.42***
Note: * = p < 0.05, ** = p < 0.01, *** = p < 0.001						

of Aiken and West (1991) by mean-centering and scaling all variables. To ensure that the remaining cross-correlations do not bias our evaluation results, we checked the Variance Inflation Factor (VIF) for each independent variable. The VIF describes the degree to which a variable's variance is influenced by another variable. As illustrated in Table 1, the VIF values of all independent variables in our sample are far below the required threshold of 10 and also below the recommended value of 3 (Greene 2003). Thus, we can ensure discriminant validity for all variables.

5.2 Hypotheses Testing

To test our research hypotheses, we applied Ordinary Least Squares (OLS) regression analysis. Table 2 lists the results of this evaluation. Our regression model explains a substantial amount of variance in OSS teams' productivity ($R^2 = 0.278$). In particular, team size has a weak negative effect on OSS teams' productivity ($\beta = 0.099$, $p = 0.025$), which suggests that smaller OSS teams are slightly more productive than larger ones. Further, the regression analysis reveals that the project size has a strong stimulating effect on OSS teams' productivity ($\beta = 0.465$, $p < 0.001$), which in turn indicates that OSS teams working on a larger codebase are more productive than teams working on smaller codebases. In contrast, the project age has a moderate negative effect on OSS teams' productivity ($\beta = -0.287$, $p < 0.001$), which suggests that teams at younger OSS projects are more productive than when working on older projects. OSS developers' level of project experience has no significant effect on their effective collaboration ($\beta = 0.007$, $p < 0.307$). In contrast, developers' level of team experience has a moderate positive effect on OSS teams' collaboration productivity ($\beta = 0.189$, $p = 0.005$). This indicates that OSS teams are more productive the more the involved developers have worked with each other in the past. Finally, our evaluation supports our hypothesis that reputable OSS developers make OSS teams more productive. However, the revealed positive effect is marginally weak ($\beta = 0.095$, $p = 0.008$).

Table 2: OLS regression (N = 749)

Team size	-0.099	*
Team exp.	0.189	**
Project age	-0.287	***
Project size	0.465	***
Project exp.	0.071	
Comm rep	-0.095	**
Note: * = p < 0.05, ** = p < 0.01, *** = p < 0.001		

To understand why the involvement of reputable OSS developers increase the productivity of OSS teams so little, we employ in the following a detailed post-hoc analysis.

6 Post-Hoc Analysis

To better understand the results of our structural evaluation, we take in the following an individual-centric research approach and examine if reputable developers provide the supposed cognitive and affective assets for team members' collaboration and if these assets make individuals more productive. In particular, we argue that the collaboration assets which are provided by reputable OSS developers are reflected by increased levels of cognitive and affective trust among team members which, in turn, enhance their individual productivity. After describing our research model, we detail our evaluation methodology and our evaluation results.

6.1 Research Model

In the following, we build on the trust framework proposed by McAllister (1995) and argue that the supposed collaboration assets which are provided by reputable OSS developers increase team members' level of cognitive and affective trust towards their colleagues. In line with this framework and previous evaluations from the OSS context, we assume further that these two forms of trust increase team members' individual productivity. Figure 3 illustrates our research model.

Reputable Developers and Cognitive Trust

Cognitive trust arises between individuals through their rational assessment of each other's competence and through the conviction that the other will not misbehave (McAllister 1995). In the OSS context, we suppose that the involvement of reputable developers fosters such favorable assessment, for example by assisting team members in their work. Reputable OSS developers can not only prevent team members from making the same mistakes as they did, but they can also advise them in strategic decisions. Study results by Schilling and Laumer (2012) support this and suggest that team members build considerable more knowledge if they are assisted by reputable developers. We suppose that these knowledge gains do not go unnoticed and lead team members to be more confident in each other's work and thus foster their level of cognitive trust. Moreover, reputable developers can raise members' level of cognitive trust in

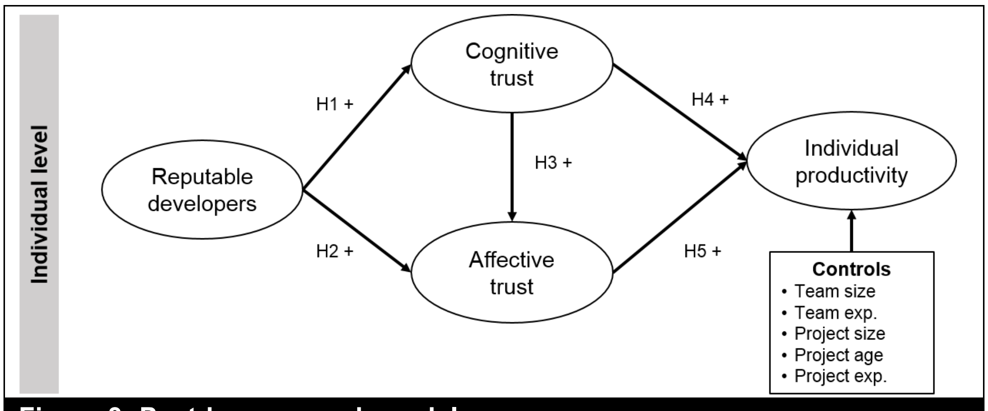


Figure 3: Post-hoc research model

the OSS team by creating and maintaining a practice of regular work updates. Compared to regular project members, reputable OSS developers are not only more familiar with the culture of sharing in OSS development but also have the necessary standing to cultivate such behavior in the team. The practice of regular work updates enables team members to see what their colleagues are currently working on and enhances their awareness for the problems their colleagues mastered in the past. As a consequence, team members are not only more confident in their colleagues' work, but also aware that they all share the goal of making the project succeed. For OSS developers, such awareness is an essential determinant for the creation of cognitive trust (Xu and Jones 2010). Finally, because of their rich project experience, reputable OSS developers can prevent their team members from having unrealistically high expectations of the work outcomes of their colleagues. Especially less experienced OSS developers are often constrained in their work by contribution barriers of the project, like a complex code architecture or intertwined modules (von Krogh et al. 2003). As a result, they are often less productive than team members with more experience. In such cases, reputable developers can draw on their collaboration experience and prevent their team members from having unrealistic expectations of novices' work, which cannot be met. With respect to the various ways reputable OSS developers can strengthen team members' perceived competence and transparency of each other's actions, we suppose that:

Hypothesis 1: Reputable OSS developers increase members' level of cognitive trust towards the OSS team.

Reputable Developers and Affective Trust

In addition, we suppose that the involvement of reputable OSS developers raises team members' level of affective trust towards each other. Affective trust stems from the emotional attachment of a 'trustor' towards the 'trustee' (McAllister 1995). One way reputable developers can foster such intense relationships between team members is through building and promoting a culture of open information exchange. This team communication fosters the awareness among members that their colleagues share the same helping values as they do. In consequence, this awareness strengthens team members' feelings of belonging to each other and, thus, their level of affective trust (Stewart and Gosain 2006). Another aspect in which an open discussion culture helps to build affective trust is through making apparent that team members share a commitment to the project's development. With this awareness, it is much more likely that team members consider themselves as belonging to the team. Xu and Jones (2010) support this by providing evidence that the perceived overlap of goals among team members is an important facilitator for the formation of affective trust. In addition, reputable developers can enhance team members' collaboration behavior both passively and actively. On the one hand, reputable developers can actively protect team members when they feel that they have been criticized unjustly or stop flame-wars from rising by deescalating mailing list disputes. Such intervention by reputable developers can help team members to look beyond such disputes and realize that they all share the common wish to make the project succeed. On the other hand, reputable developers can take passive means and exemplify good behavior, for instance in terms of caring about others. Such positive example may be followed by members and lead to a supportive group atmosphere. Stewart and Gosain (2006) support this by highlighting that team members build affective trust with each other when they perceive that the team as a whole values members' helping and sharing behavior. With respect to the different ways in which reputable developers can foster the formation of affective trust within an OSS team, we suppose that:

Hypothesis 2: Reputable OSS developers increase members' level of affective trust towards the OSS team.

Cognitive Trust as a Facilitator for Affective Trust

In line with McAllister (1995), we suppose further that cognitive trust is a facilitator for affective trust. This is because individuals need to rationally evaluate team members' integrity and competence before they can feel emotionally connected with them. In line with McAllister's research, Stewart and Gosain (2006) and Xu and Jones (2010), provide evidence that cognitive trust fosters the level of affective trust between OSS developers. Thus, building on the original framework of McAllister (1995) and with respect to previous evaluations within the OSS domain, we hypothesize that:

Hypothesis 3: OSS developers' level of cognitive trust amplifies their level of affective trust towards the OSS team.

Cognitive Trust and OSS Developers' Individual Productivity

The perceived uncertainty regarding the behavior of colleagues is a high productivity barrier for OSS developers. As Shah (2006) points out, OSS developers only commit themselves to a project if they believe the project will succeed. In this regard, we assume that cognitive trust provides the necessary foundation for OSS developers to increase their commitment. Moreover, we suppose that this form of trust fosters team members' individual productivity because it mitigates their need to monitor the work of their colleagues (McAllister 1995), which in turn gives them more time for their own coding. Another benefit of cognitive trust is that team members know whom to ask for help to quickly solve an issue. Finally, if team members have cognitive trust in each other, they are less reluctant to contact each other because they know that their colleagues want the project to succeed as much as they do, which in turn helps to avoid problems in the first place. With respect to these gains for OSS developers' individual productivity, we suppose that:

Hypothesis 4: OSS developers' level of cognitive trust in their team increases their individual productivity.

Affective Trust and OSS Developers' Individual Productivity

Moreover, we suppose that OSS developers' level of affective trust in each other fosters their individual productivity. With affective trust, there is an intense relationship between team members, which leads to the formation of a group identity (Xu and Jones 2010). With such group identity, members perceive themselves to be similar to each other and different from individuals outside the OSS team. As a consequence, OSS developers put more efforts into their project work because they do not want to disappoint their team members. Another consequence of affective trust is that team members believe that they are not being left on their own with a problem, but believe that others will care about them. This belief makes team members much more willing to share their problems with each other. As a result, team members do not delay potential problems but bring them to the fore when they occur and, by doing so, find a solution in a timely manner. In addition, affective trust fosters the satisfaction OSS developers derive from their project work, which in turn makes them willing to engage even more in the project in the future. In line with our reasoning, Xu and Jones (2010) and Stewart and Gosain (2006) support the stimulating effects of affective trust on OSS teams' productivity. With respect to the positive effects on individuals' productivity, we suppose that:

Hypothesis 5: OSS developers' level of affective trust in their team increases their individual productivity.

6.2 Research Methodology

To evaluate our post-hoc research model, we surveyed KDE developers on their interpersonal relationships towards their team members. To do so, we created an online survey to which we posted invitations at the central KDE developer mailing list and the central community webpage for KDE projects. In total, 86 KDE developers participated in our online survey. Of all responses, we had to drop six due to missing and malformed answers. To achieve high compatibility between this individual-centric study and our previous structural analysis, we used, as far as possible, the same controls and archival measures. In particular, we assessed the **project age**, **project size**, **project experience**, **team size**, **team experience**, and members' **individual productivity**, using the same archival measurement techniques as in our structural analysis in Section 5. We only modified the evaluation scope of the two experience measures from the team to the individual level by calculating them only for the particular individual.

In addition, we used perceptive measures to evaluate the involvement of reputable developers and members' level of cognitive and affective trust in their colleagues. Each of these constructs has been assessed using three question items which were answered on a five point Likert scale. To ensure the validity and reliability of the used measures, we used exclusively question items which have been used in prior evaluations. To assess the involvement of **reputable OSS developers**, we adopted three question items which have previously been used by Schilling et al. (2013). These are: *REP-1: 'Some of the developers in this project are highly respected by other developers in the community'*, *REP-2: 'Some members of this project are famous in the community'* and *REP-3: 'Some developers in this project have a strong standing in the community'*. For assessing KDE developers' level of **cognitive trust** towards the OSS team, we rely on question items which have been used by Stewart and Gosain (2006) and Xu and Jones (2010). These are: *COG-1: 'I trust and respect the members of this project'*, *COG-2: 'Members of this project team regard each other as trustworthy'* and *COG-3: 'Most members of this project are very competent and approach their work very professional'*. Likewise, we draw on these two studies to assess OSS developers' level of **affective trust** towards their team using the three question items: *AFF-1: 'If I share my problems with others in this project, I know they* *will respond constructively and caringly', AFF-2: 'Members of this project have a sharing relationship with each other. I can freely share my ideas, feelings and hopes'*, and *AFF-3: 'On this project team, I can talk freely with others about difficulties I am having and know that others are willing to listen'.*

6.3 Hypotheses Testing

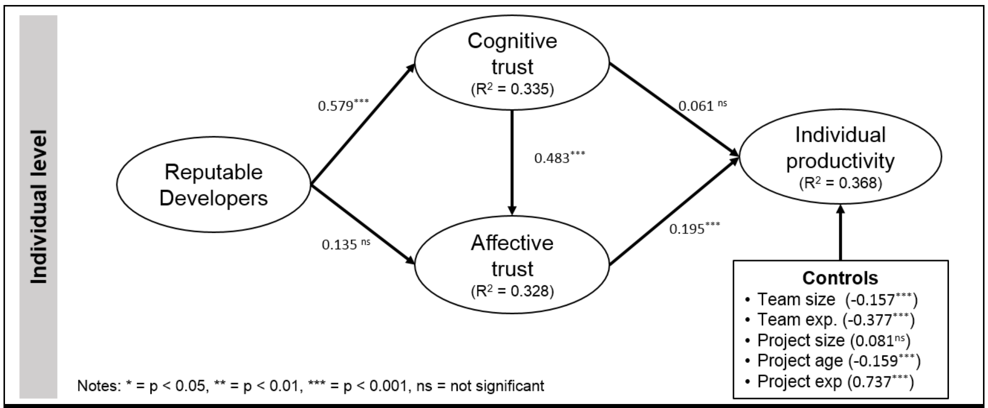
Before we started evaluating our post-hoc hypotheses, we checked the reliability and validity of our measurement model. To ensure convergent validity, we tested for each construct (*i*) the reliability of the used question items, (*ii*) its Composite Reliability (CR) and (*iii*) the Average Variance Extracted (AVE). In order to assess the reliability of the question items, we checked that they load on their associated constructs more than 0.7 (Chin 1998), which is the case for all constructs (see Table 3). Next, we computed the CR for each construct. The CR describes the degree to which a latent variable is explained through its question items. As listed in Table 3, the CR values for all constructs are above the recommended threshold of 0.7 (Chin 1998). Finally, we checked each construct's AVE, which measures the degree to which its variance is explained through the associated question items in relation to the measurement error. In our sample, the AVE of all constructs is well above the recommended value of 0.5 (Bagozzi and Yi 1988). Based on these results, we can ensure convergent validity for our measurement model.

To assess discriminant validity, we ensured that each question item loads strongest with its associated construct. Moreover, we checked that the square root of each construct's AVE is higher than the correlation between the particular construct and any other construct (Fornell and Larcker 1981). As shown in Table 3, all of our constructs pass this test. In this table, we list each construct's correlations with the other constructs and list the square roots of the AVE values on the diagonal cells.

After checking the convergent and discriminant validity of our measurement model, we compute the strength and significance of the paths in our structural model using Partial Least Squares (PLS). We used PLS for this evaluation because it requires a relatively small sample size to provide reliable results (Chin 1998). The PLS evaluation provides evidence that our model explains a substantial amount of variance in OSS developers' levels of cognitive (R^2 = 0.335) and affective trust (R^2 = 0.328) towards their team members as well as their individual productivity (R^2 = 0.368). As we supposed in Hypothesis 1, the evaluation results suggest that reputable OSS developers have a strong positive effect on members' level of cognitive trust towards their team (β = 0.579, $p < 0.001$). Conversely, we found no evidence that reputable

Table 3: Post-hoc construct consistency

	CA	CR	AVE	1	2	3	4	5	6	7	Loadings
1. Aff trust	0.81	0.88	0.72	**0.85**	0.00	0.00	0.00	0.00	0.00	0.00	0.82 - 0.87
2. Rep. devs.	0.87	0.91	0.78	0.44	**0.88**	0.00	0.00	0.00	0.00	0.00	0.82 - 0.94
3. Cog. trust	0.68	0.82	0.60	0.54	0.58	**0.77**	0.00	0.00	0.00	0.00	0.76 - 0.82
4. Project age	1.00	1.00	1.00	-0.02	0.24	0.10	**1.00**	0.00	0.00	0.00	1
5. Project size	1.00	1.00	1.00	-0.03	0.22	0.25	0.48	**1.00**	0.00	0.00	1
6. Project exp.	1.00	1.00	1.00	0.02	0.03	0.03	0.09	0.18	**1.00**	0.00	1
7. Team Exp.	1.00	1.00	1.00	-0.01	0.01	0.01	0.04	0.38	0.70	**1.00**	1
8. Team size	1.00	1.00	1.00	0.02	0.14	0.16	0.20	0.66	0.29	0.58	1


Individual level
Cognitive trust
($R^2 = 0.335$)
Reputable Developers
Affective trust
($R^2 = 0.328$)
Individual productivity
($R^2 = 0.368$)
0.579^{***}
0.135^{ns}
0.483^{***}
0.061^{ns}
0.195^{***}
Controls
• Team size (-0.157^{***})
• Team exp. (-0.377^{***})
• Project size (0.081^{ns})
• Project age (-0.159^{***})
• Project exp (0.737^{***})
Notes: * = p < 0.05, ** = p < 0.01, *** = p < 0.001, ns = not significant
Figure 4: Evaluation results

OSS developers increase members' level of affective trust in their colleagues (β = 0.135, p = 0.583), which does not support Hypothesis 2. In line with Hypothesis 3, our evaluation indicates that members' cognitive trust has a strong stimulating effect on the formation of affective trust towards their colleagues (β = 0.483, p < 0.001). Surprisingly, we find no evidence that OSS developers' level of cognitive trust in their team increases their individual productivity (β = 0.061, p = 0.583). This does not support Hypothesis 4. However, our evaluation results suggest that OSS developers' level of affective trust towards the team stimulates their individual productivity moderately (β = 0.195, p < 0.001). In addition to these effects, team size (β = -0.157, p < 0.001) and project age (β = -0.159, p < 0.001) have a weak negative effect on OSS developers' individual productivity. Furthermore, developers' project experience has a strong stimulating effect (β = 0.737, p < 0.001) while their team experience has a moderate negative effect on their individual productivity (β = -0.377, p < 0.001). Finally, our evaluation results suggest that the project size has no effect on OSS developers' individual productivity (β = 0.091, p = 0.583). Figure 4 visualizes the strength and significance of all examined relationships.

In the next section, we discuss the implications of our evaluation results for research and practice.

7 Discussion

In this research, we draw on the social practice view of OSS development (von Krogh et al. 2012) and suppose that reputable OSS developers stand out not only due to their technical and behavioral competences but also due to their internalization of the OSS culture. Because of these characteristics, we argued that reputable developers provide various assets which enhance OSS teams' productivity. Although, our structural analysis of 749 OSS teams indicates that reputable developers foster OSS teams' productivity, it reveals that their stimulating effects are marginally low. In light of the strong positive effects reputable developers have on the attraction of new developers (Hu et al. 2012) and on enhancing members' collaboration (Li et al. 2006, Kuk 2006, Schilling et al. 2013, Casaló et al. 2009), this marginal effect is even more puzzling. To understand the underlying reasons for the low productivity gains, we employed an individual-centric post-hoc analysis. The results of this post-hoc evaluation provide evidence that reputable developers increase members' belief in each other's competence, but also indicates that this form of trust is not directly linked to their individual productivity. Instead it is only members' feelings of belonging to the OSS team which have a direct effect on their individual productivity. However, these feelings are not directly linked to the involvement of reputable developers. In this section, we discuss the theoretical and managerial implications of our work for teamwork in OSS projects, organizations, and online communities. Before we do so, we summarize the key findings of our research.

7.1 Findings

- ***The involvement of reputable OSS developers has a marginal stimulating effect on OSS teams' productivity.***

Our structural analysis suggests that reputable developers increase OSS teams' productivity, but this effect is marginal. In fact, our analysis of 749 OSS teams, shows that the effect of reputable developers on OSS teams' productivity is the weakest of all controlled factors.

- ***The involvement of reputable OSS developers raises team members' level of cognitive trust, but this form of trust has no direct effect on members' individual productivity.***

As we supposed in our post-hoc research model, our evaluation suggests that the involvement of reputable OSS developers increases members' level of cognitive trust in the team. However, this form of trust has no effect on members' individual productivity. In other words, although having reputable OSS developers on a team raises the degree to which members consider each other as competent, but such favorable evaluation does not increase their individual productivity.

- ***The involvement of reputable OSS developers does not affect members' level of affective trust, but this form of trust directly facilitates members' individual productivity.***

In contrast, our post-hoc evaluation indicates that reputable OSS developers have no direct effect on team members' level of affective trust in each other. However, it is this particular form of trust which fosters members' individual productivity. Put differently, our individual-centric study suggests that it is developers' belonging to the OSS team which fosters their individual productivity, but this feeling is not directly linked to the involvement of reputable developers.

In addition, our multi-level evaluation reveals that some of the controls in our study have different effects on the individual- and team-level. While the team size and the project age exert comparable effects on developers' individual and collective productivity, this is not the case for the following three factors:

- ***OSS developers' level of project experience is a substantial stimulus for their individual productivity, but has no significant effect on OSS teams' productivity.***

Although, OSS developers' project experience is a substantial facilitator for their individual productivity, it has no measureable effect on OSS teams' productivity. This indicates that there are factors or dynamics on the team level which alleviate the productivity gains of members with extensive project experience.

- ***Developers' team experience has a strong positive effect on OSS teams' productivity, but a strong negative effect on their individual productivity when controlling for cognitive and affective trust.***

While our structural evaluation provides evidence that OSS teams are more productive the longer the involved developers have worked with each other, our individual-centric analysis undermines this contention. In fact, developers' level of team experience has a strong negative effect on their individual productivity when controlled for the effects of cognitive and affective trust.

- ***The size of an OSS project has a strong positive effect on developers' individual productivity but no effect on OSS teams' productivity.***

In contrast to our individual-centric analysis which identifies the project size as a substantial facilitator for individuals' productivity, our empirical evaluation suggests that these positive effects alleviate and even disappear on the team level.

7.2 Implications for OSS Projects

Considering that most OSS projects are struggling for developer contributions (Fang and Neufeld 2009), our research has implications for OSS research and practice.

- ***Having reputable developers involved increases OSS teams' productivity only marginally, as they increase only team members' level of cognitive but not their level of affective trust.***

Although our structural evaluation of 749 OSS teams supports our main hypothesis that reputable developers increase OSS teams' productivity, it reveals a very low stimulating effect. As our post-hoc analysis suggests, a reason for this weak effect could be that reputable developers directly increase the level to which team members consider each other competent, but this alone does not make them more productive. Instead, it is their affective feelings towards their team members which raise their individual productivity, but these feelings are not directly stimulated by having reputable developers on the team. These distinct consequences of OSS developers' types of trust in the team are in line with research by Stewart and Gosain (2006) and support the authors' reservation towards involving reputable developers to make OSS teams more productive. Moreover, these effects explain why the attraction of OSS developers, which is strongly dependent on the positive assessment of team members' competence (Shah 2006), is much more affected by the involvement of reputable OSS developers (Hu et al. 2012). However, while this explanation seems plausible, it requires future evaluation. Moreover, future research should examine if reputable developers differ in their predisposition to trust other team members compared to less reputable developers.

- ***Our research suggests a refinement to the social practice view of OSS development.***

In contrast to studies which focus on OSS developers' search for immediate outcomes of their behavior, we take a more nuanced view on their project behavior, and focus on their interactions with their team members and their ethical considerations. Building on the social practice view

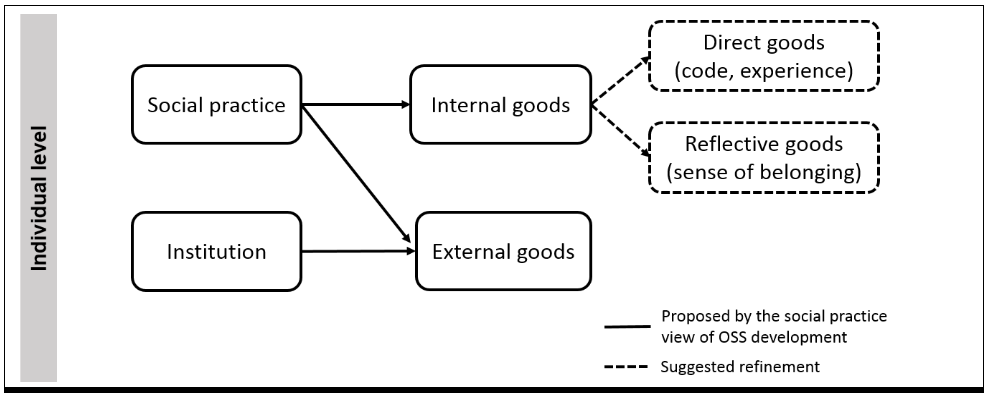

Individual level
Social practice
Internal goods
Direct goods (code, experience)
Reflective goods (sense of belonging)
Institution
External goods
Proposed by the social practice view of OSS development
Suggested refinement

Figure 5: Suggested refinement

of OSS development (von Krogh et al. 2012) and the peer evaluation process in OSS communities (Raymond 1999), we argued that reputable developers help OSS teams in assisting members in their coding and also pass on central values of the OSS culture, like 'sharing and caring'. However, our evaluation reveals that reputable developers only foster members' belief in each other's competence but not their feelings of belonging to each other. At closer look, these findings are not too surprising. In his original publication, MacIntyre (1981) distinguishes between two basic forms of internal goods. The first type of internal goods refers to the performance itself and the associated product while the second type of internal good concerns the 'related kind of life' (MacIntyre 1981, p. 190). While the first form of internal good can be derived through pursuing a social practice in line with the standards of excellence, the second type requires individual self-reflection of project work. In this regard, the distinct effects posed by reputable developers become even more plausible, as they may enhance OSS developers' competencies, but it is up to them to find overlaps between their own and their colleagues' lives. In line with MacIntyre (1981), we thus propose to distinguish within the OSS context between internal goods which can be directly obtained in the course of the social practice (i.e. code, experience) and internal goods which depend on individuals' self-reflection (i.e. sense of belonging). Figure 5 visualizes the supposed refinement of internal goods in the context of the social practice view of OSS development. This distinction is especially helpful for future research to derive new ways on how to enhance OSS developers' collaboration.

- ***Managers should favor dedicated teambuilding activities over bringing in reputable developers to enhance OSS team productivity.***

Finally, our work provides lessons for managers of OSS projects. Most importantly, our research suggests that involving reputable developers enhances OSS teams' productivity only marginally. This is because, reputable developers only increase members' cognitive trust, the belief in each other's competence, however, it is their level of affective trust, their feelings of belonging to each other, which have a direct effect on their productivity. Thus, managers of OSS teams who wish to enhance their teams' productivity should rather offer team building activities which strengthen members' interpersonal bonds, like conducting release parties or arranging social events in the context of developer sprints, rather than bringing in reputable developers.

7.3 Implications for Organizations

With respect to researchers' advice to consider knowledge workers volunteers (Drucker 2002), our work provides also implications to teams in organizations.

- ***Use of the social practice view to understand collaborations within organizations.***

In line with Beadle (2006) we suppose that the social practice view provides valuable grounds for understanding effective collaborations in organizations. One central advantage over motivation theories, which focus on the immediate outcomes associated with individuals' behavior, is that MacIntyre's theory considers individuals' ethical beliefs and their long term goals to understand their project behavior. This shift in perspective, however, requires a new understanding of designing a supportive organizational environment which much more emphasizes employees' work-life balance. In this regard, our work provides first grounds for future organizational literature not to focus only on employees' teamwork competencies but to consider also their sense of belonging. In other words, our results stress that organizational research needs to focus on why employees work and not only what they work.

- ***Managers should not rely on scoring systems in their staffing decisions and favor teambuilding activities over bringing in reputable individuals to enhance their teams' productivity.***

Finally our research provides several practical lessons for organizations. Foremost, our work adds to research on the value added of external (e.g.: LinkedIn endorsements) and in-house scoring systems (e.g.: IBM Connections). In line with the social practice view, we expect that individuals gain a positive score in these systems through making positive experiences when working together (assuming that their organizations allow them to share their work experiences openly with others internal or external to the organization). Although our research suggests that reputable individuals increase team productivity, this effect is much lower than for example their positive effect on the equity price of knowledge-intensive firms (Erden et al. 2014). While there are only few levers for firms to get such attention from financial investors, our research recommends fostering affective trust among members (such as having a beer after work) over bringing in reputable developers to enhance individuals' productive interplay.

7.4 Implications for Online Communities

As our evaluation considers the effective virtual collaboration of individuals, which is a key challenge for online communities (McLure Wasko and Faraj 2000), we delineate in the following the implications of our work for value creation in online communities.

- ***Team and project characteristics can have opposite effects on individual and collective behavior.***

A key insight of our work is the notion that the effects of project and team characteristics on individual behavior can be distinct from and even opposite of effects on collective behavior. One of these characteristics is developers' team experience, which stimulates OSS teams' productivity, but which has a strong negative effect on members' individual productivity when controlling for affective and cognitive trust. An explanation for this discrepancy could be that OSS developers favor their companionship over their project work, the longer they work together. As a result, these developers contribute less code to the OSS project but remain supportive for team members and thus help them to become more productive. From an aggregated research perspective which we used in our structural analysis, such effect would be invisible as the productivity deficits of long-term developers could be completely covered under the productivity gains of new project members. In a similar vein, such multi-level differences could provide an explanation for the distinct effects individuals' expertise has on their individual and collective behavior in online communities (McLure Wasko and Faraj 2005, 2000). Thus, future studies on online communities should be aware of such ambiguous effects and explicitly check for them.

- ***Relevance of a social practice view***

Likewise to the OSS context, the social practice view proposed by MacIntyre (1981) could provide valuable grounds for understanding value creation in online communities. In particular, its distinction between internal and external goods produced in the course of pursuing a social practice provides a suitable context for understanding the various motives which drive individuals to participate in online communities (McLure Wasko and Faraj 2000). Moreover, the social practice view helps to understand why individuals' interactions and their ethical considerations play a salient role in their community behavior. Future research adopting the social practice view in online communities can draw on our work to understand that there are only some internal goods which can be fostered with the help of others while other internal goods require individuals' self-reflection. Another central topic for future research is to identify the relevant standards of excellence for value creation in online communities.

- ***Lessons for managers of online communities***

Finally our research provides lessons for managers of online communities. Most importantly, our work indicates that involving reputable developers is not a surefire way to increase value creation in online communities. Although, our research provides evidence that reputable developers enhance individuals' productive interplay, this positive effect is marginally weak. Instead, our work supports the advice of Ren et al. (2007) to foster the emotional bonds between community members to enhance their collaboration. Such teambuilding activities could be arranged with the help of the companies involved. For example, Nike could award members of its designer community with cards to NBA playoffs (Füller et al. 2007). If such offline events are not possible, managers should seek to find online alternatives for them which emphasize members' shared interests, such as an online chat with Michael Jordan in the case of the Nike designer community.

7.5 Limitations and Future Research

There are several limitations in our research which we would like to point out. First, team members' productive collaboration is only one way to look at value creation in OSS projects, even though we argue that this aspect is inherently intertwined with other value creation processes in OSS projects, such as innovation and learning (von Krogh et al. 2003). For example, the innovation process in OSS projects is generally not a fire-and-forget activity, but requires iterative refinement. This iterative refinement is manifested in a core principle for OSS development, which is to 'release early and often' (Raymond 1999, p. 7). As a result, OSS projects' innovation process requires iterative code development. Likewise, team members' learning is linked to their own as well as other members' coding. This is because OSS developers build knowledge through actively developing code for the project or exchanging with team members, who develop for the project (Fang and Neufeld 2009, Singh et al. 2011b). Thus, with respect to the interrelation between code development and the innovation and learning processes in OSS teams, we consider members' productive interplay as a necessary but not sufficient aspect to understand value creation in OSS projects and encourage future research to build on our work to examine if involving reputable developers has similar effects on the innovation and learning processes in OSS projects.

Another potential limitation of our work is its grounding in the social practice view of OSS development. Although, the social practice view seems to be an appropriate theoretic framework to understand how social and environmental influences affect individual behavior, its use is immature and was only recently applied to the OSS domain (von Krogh et al. 2012). While our evaluation supports the general reasoning of the social practice view, we cannot judge the general appropriateness of this concept for OSS development. Moreover, we do not want to leave unmentioned that MacIntyre's theory itself is not without criticism (Beadle 2006).

Third, our combination of structural- and individual-centric evaluation approaches is constrained by non-overlapping study samples. Although we repeatedly promoted our survey through posts on the KDE developer mailing list and on the central community page, we could not achieve an overlapping study sample between our two evaluations. Although, all projects within the KDE framework share the same coding standards and governance processes, we cannot rule out that the KDE developers in our structural analysis would provide different survey replies. Moreover, due to the non-overlapping study sample, we had to rely on a perceptive measure instead of our archival measure to assess the involvement of reputable OSS developers. We did so because we could not identify the Ohloh accounts for a sufficient number of team members with whom the surveyed developers had worked. This raises the interesting question for future research if and to which degree the subjective identification of reputable developers differs from their archival assessment. Moreover, our evaluation calls for future research with overlapping samples to study the interrelation between structural- and individual-centric effects more thoroughly.

Finally, the concentration of our research on data from KDE and Ohloh.com limits the generalization of our evaluation results. By relying on data from Ohloh.com, our study is constrained to the effects of OSS developers' positive community reputation. With prevalently used peer evaluation in OSS projects, however, developers could also be affected by a negative community reputation. This raises an intriguing question for future studies: to what degree is OSS team productivity reduced by the involvement of developers with a negative reputation in the OSS community? Moreover, while our focus on KDE projects allows us to study a variety of different project and team configurations that share the same institutional characteristics and code of excellence in terms of the social practice view, it reduces our ability to transfer our findings to other OSS projects. Further, our individual-centric post-hoc analysis is constrained by the participation of only 86 KDE developers. Thus, future research should seek to examine our research based on a more diverse and quantitatively richer survey sample.

8 Conclusion

In this multi-level research, we examine if and how reputable developers foster OSS team productivity. Building on the social practice view of OSS development, we argue that reputable OSS developers enhance team members' technical competence and foster their feelings of belonging to the team. To evaluate our research hypothesis, we performed an empirical evaluation of 749 OSS teams based on the community endorsement of the involved developers and archival records of their previous contributions. Although our findings indicate that reputable OSS developers increase OSS teams' productivity, this effect is marginal. To understand the underlying reasons for this marginal productivity gain, we performed an individual-centric post-hoc analysis. The results of this analysis suggest that members of teams with reputable OSS developers indeed perceive each other to be more competent. However, this does not make them more productive. Instead, it is members' sense of belonging which makes them more productive, but such feeling is not directly linked to the involvement of reputable OSS developers.

9 References

Aiken, LS, West, SG (1991) *Multiple Regression*: *Testing and Interpreting Interactions* (Sage, Newbury Park, California).

Bagozzi RP, Yi Y (1988) On the Evaluation of Structural Equation Models. *Journal of the Academy of Marketing Science* 16(1):74–94.

Baldwin CY, Clark KB (2006) The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model? *Management Science* 52(7):1116–1127.

Beadle R (2006) MacIntyre on Virtue and Organization. *Organization Studies* 27(3):323–340.

Bergquist M, Ljungberg J (2001) The Power of Gifts: Organizing Social Relationships in Open Source Communities. *Information Systems Journal* 11(4):305–320.

Brin S, Page L (1998) The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems* 30(1-7):107–117.

Casaló LV, Cisneros J, Flavián C, Guinalíu M (2009) Determinants of Success in Open Source Software Networks. *Industrial Management & Data Systems* 109(4):532–549.

Chin WW (1998) Issues and Opinion on Structural Equation Modeling. *MIS Quarterly* 22(1): vii–xvi.

Colazo JA, Fang Y (2010) Following the Sun: Temporal Dispersion and Performance in Open Source Software Project Teams. *Journal of the Association for Information Systems* 11(12):684–707.

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

Deshpande A, Riehle D (2008) The Total Growth of Open Source. Russo B, Damiani E, Hissam S, Lundell B, Succi G, eds. *Open Source Development, Communities and Quality* (Springer US, Boston, MA), 197–209.

Drucker PF (2002) They're not Employees, They're People. *Harvard Business Review* 80(2):70–77.

Erden Z, Klang D, Sydler R, von Krogh G (2014) “How can We Signal the Value of Our Knowledge?” Knowledge-based Reputation and its Impact on Firm Performance in Science-based Industries. *Long Range Planning*.

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

Fornell C, Larcker DF (1981) Structural Equation Models With Unobservable Variables and Measurement Error Algebra and Statistics. *Journal of Marketing Research (JMR)* 18(3):382–388.

Füller J, Jawecki G, Mühlbacher H (2007) Innovation Creation by Online Basketball Communities. *Journal of Business Research* 60(1):60–71.

Gayo-Avello D (2013) Nepotistic Relationships in Twitter and Their Impact on Rank Prestige Algorithms. *Information Processing & Management* 49(6):1250–1280.

Greene, WH (2003) *Econometric analysis,* 5th ed. (Prentice Hall, Upper Saddle River, N.J).

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

Hu D, Zhao JL (2009) Discovering Determinants of Project Participation in an Open Source Social Network *Proceedings of the 30th International Conference on Information Systems (ICIS)*,

Hu D, Zhao JL, Chen J (2012) Reputation Management in an Open Source Developer Social Network: An Empirical Study on Determinants of Positive Evaluations. *Decision Support Systems* 53(3):526–533.

Ke W, Zhang P (2010) The Effects of Extrinsic Motivations and Satisfaction in Open Source Software Development. *Journal of the Association for Information Systems* 11(12):785–808.

Kuk G (2006) Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List. *Management Science* 52(7):1031–1042.

Li Y, Tan C, Teo H, Mattar AT (2006) Motivating Open Source Software Developers: Influence of Transformational and Transactional Leaderships *SIGMIS CPR '06*, 34.

MacIntyre, AC (1981) *After Virtue*: *A Study in Moral Theory,* 1st ed. (University of Notre Dame Press, Notre Dame).

McAllister DJ (1995) Affect-and Cognition-Based Trust as Foundations for Interpersonal Cooperation in Organizations. *Academy of Management Journal* 38(1):24–59.

McLure Wasko M, Faraj S (2000) "It Is What One Does": Why People Participate and Help Others in Electronic Communities of Practice. *The Journal of Strategic Information Systems* 9(2-3):155–173.

McLure Wasko M, Faraj S (2005) Why Should I Share? Examining Social Capital and Knowledge Contribution in Electronic Networks of Practice. *MIS Quarterly* 29(1).

Qureshi I, Fang Y (2010) Socialization in Open Source Software Projects: A Growth Mixture Modeling Approach. *Organizational Research Methods* 14(1):208–238.

Raymond, ES (1999) *The Cathedral and the Bazaar*: *Musings on Linux and Open Source by an Accidental Revolutionary,* 1st ed. (O'Reilly, Sebastopol).

Ren Y, Kraut R, Kiesler S (2007) Applying Common Identity and Bond Theory to Design of Online Communities. *Organization Studies* 28(3):377–408.

Roberts JA, Hann I, Slaughter SA (2006) Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects. *Management Science* 52(7):984–999.

Ryan RM, Deci EL (2000) Self-Determination Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-Being. *American Psychologist* 55(1):68.

Schilling A, Laumer S (2012) Learning to Remain - Evaluating the Use of Mentoring for the Retention of FLOSS Developers. *Proceedings of the 20th European Conference on Information System (ECIS)*,

Schilling A, Laumer S, Weitzel T (2013) In the Spotlight - Evaluating How Celebrities Affect FLOSS Developers' Participation Motivation. *Proceedings of the 21th European Conference on Information System (ECIS)*, Sen R, Subramaniam C, Nelson ML (2008) Determinants of the Choice of Open Source Software License. *Journal of Management Information Systems* 25(3):207–240.

Shah SK (2006) Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development. *Management Science* 52(7):1000–1014.

Singh PV (2010) The Small-World Effect. *ACM Trans. Softw. Eng. Methodol.* 20(2):1–27.

Singh PV, Tan Y, Mookerjee V (2011a) Network Effects: The Influence of Structural Social Capital on Open Source Project Success. *Management Information Systems Quarterly* 35(4):813–829.

Singh PV, Tan Y, Youn N (2011b) A Hidden Markov Model of Developer Learning Dynamics in Open Source Software Projects. *Information Systems Research* 22(4):790–807.

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

Stewart KJ, Gosain S (2006) The Impact of Ideology on Effectiveness in Open Source Software Development Teams. *Management Information Systems Quarterly* 30(2):291–314.

Subramaniam C, Sen R, Nelson ML (2009) Determinants of Open Source Software Project Success: A Longitudinal Study. *Decision Support Systems* 46(2):576–585.

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

von Krogh G, Spaeth S, Lakhani KR (2003) Community, Joining, and Specialization in Open Source Software Innovation: A Case Study. *Research Policy* 32(7):1217–1241.

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

Xu B, Jones DR (2010) Volunteers' Participation in Open Source Software Development: A Study from the Social-Relational Perspective. *Database for Advances in Information Systems* 41(3).

Chapter IV:
Developer Retention

Paper VII

Learning to Remain - Evaluating the Use of Mentoring For the Retention of FLOSS Developers

Andreas Schilling
University of Bamberg
andreas.schilling@uni-bamberg.de

Sven Laumer
University of Bamberg
sven.laumer@uni-bamberg.de

In: Proceedings of the 20th European Conference on Information System (ECIS 2012), Barcelona, Spain

The publication is available under:

http://aisel.aisnet.org/ecis2012/148

Abstract

In this study, we examine if and how reputable developers increase the productivity of teams developing Open Source Software (OSS). Building on the social practice view of OSS development, we suppose that reputable developers stand out not only for their technical and behavioral competences but also for a deep internalization of the OSS culture. Because of this, we suppose that reputable developers increase members' technical competences and foster their sense of belonging to the OSS team. To our surprise, an empirical evaluation of 749 OSS team configurations reveals that reputable developers increase OSS teams' productivity only marginally. In order to understand the underlying reasons of this weak effect, we employed an individual-centric post-hoc analysis. The results of this dedicated post-hoc evaluation with 80 OSS developers indicate that reputable developers directly increase members' level of cognitive trust in the OSS team, but this form of trust is not directly linked to their individual productivity. Instead, it is members' level of affective trust in the OSS team which directly facilitates their individual productivity. However, this form of trust is not directly linked to the involvement of reputable OSS developers. Based on our multi-level evaluation, we propose a refinement to the definition of internal goods within the social practice view of OSS development. Moreover, our evaluation brings to the fore that the effects of team and project characteristics on individual behavior can be distinct from and even opposite of the effects on collective behavior. Finally, we point out that managers of online communities who wish to enhance effective collaboration should focus on activities which strengthen individuals' social bonds rather than bringing in reputable individuals.

Appendix

Publications

Conference Proceedings (Peer-Reviewed)

Schilling, A., 2012. Links to the Source - A Multidimensional View of Social Ties for the Retention of FLOSS Developers, in: *Proceedings of the 2012 ACM SIGMIS CPR Conference*, Milwaukee (WI).

Schilling, A., 2014. What Do We Know about FLOSS Developers' Attraction, Retention, and Commitment? A Literature Review, in: *Proceedings of the 47th Hawaii International Conference on System Sciences (HICSS)*, Big Island (HI), pp. 4003 - 4012.

Schilling, A., Laumer, S., 2012. Learning to Remain - Evaluating the Use of Mentoring for the Retention of FLOSS Developers, in: *Proceedings of the 20th European Conference on Information System (ECIS)*, Barcelona, Spain.

Schilling, A., Laumer, S., Weitzel, T., 2011. Is the Source Strong with You? A Fit Perspective to Predict Sustained Participation of FLOSS developers, in: *Proceedings of the 32nd International Conference on Information Systems (ICIS)*, Shanghai, China.

Schilling, A., Laumer, S., Weitzel, T., 2012. Train and Retain - The Impact of Mentoring on the Retention of FLOSS Developers, in: *Proceedings of the 2012 ACM SIGMIS CPR Conference*, Milwaukee (WI).

Schilling, A., Laumer, S., Weitzel, T., 2012b. Who Will Remain? An Evaluation of Actual Person-Job and Person-Team Fit to Predict Developer Retention in FLOSS Projects, in: *Proceedings of the 45th Hawaii International Conference on System Sciences 2012 (HICSS)*, Maui (HI), pp. 3446–3455.

Schilling, A., Laumer, S., Weitzel, T., 2013a. In the Spotlight - Evaluating How Celebrities Affect FLOSS Developers' Participation Motivation, in: *Proceedings of the 21th European Conference on Information System (ECIS)*, Utrecht, Netherlands.

Schilling, A., Laumer, S., Weitzel, T., 2013b. Together but Apart - How Spatial, Temporal and Cultural Distances Affect FLOSS developers' Project Retention, in: *Proceedings of the 2013 ACM SIGMIS CPR Conference*, Cincinnati (OH).

Schilling, A., Laumer, S., Weitzel, T., 2014. Stars Matter - How FLOSS Developers' Reputation Affects the Attraction of New Developers, in: *Proceedings of the 2014 ACM SIGMIS CPR Conference*, Singapore.