The Structure of Knowledge Reuse in Open Innovation Communities

Mahmood Shafeie Zargar

Desautels Faculty of Management

McGill University, Montreal

April, 2018

A thesis submitted to McGill University

in partial fulfilment of the requirements of the degree of Doctor of Philosophy

Blessed be the gambler who lost all but the desire to gamble again…

Rumi

All chapters of this work are original contributions of its author. No other individual or entity has been directly involved in the composition of this dissertation.

Abstract

This thesis examines the dynamics and the interactions of knowledge collaboration and reuse in open innovation communities. Open communities facilitate the exchange of expert knowledge and are often cited as the primary driver of peer production and collective innovation. Yet, current research provides little insight into knowledge reuse practices in these communities and the way those practices are tied into the communal mode of collaboration. Extending the knowledge networks literature on one hand, and the online communities literature on the other hand, this thesis studies conjointly the creation and the dissolution of knowledge reuse as well as collaboration ties, in form of developer mobility across projects, in the open source community of Ruby language.

The analyses show a significant impact from the social mobility of contributors across projects on knowledge reuse, as predicted by the existing literature on knowledge networks. Also in line with the expectations, the effect of social mobility on knowledge reuse is found to be more pronounced when deep knowledge reuse is at stake, backing the argument that the direct social ties provide a high bandwidth suiting complex exchanges. But more interestingly, the results show an impact in reverse order. Knowledge reuse leads to an increase in the likelihood of subsequent creation of collaboration ties across projects. The results also provide anecdotal evidence that shallow knowledge reuse may be a more important driver for subsequent collaborations across projects, acting as a socializing force. Moreover, I find that intellectual property protection mechanisms negatively impact the creation of both collaboration ties and knowledge reuse ties across projects, as they give rise to license compatibility issues.

The findings offer an interactive view of the dependency between the social networks and the knowledge networks, in which none of the two networks can be given primacy over the other. This is in contrast with the prior work on knowledge networks that largely assumes knowledge networks as embedded within social networks. On the other hand, the findings provide an account of how the reuse of intellectual capital can drive the creation of social capital, confirming a rarely tested assumption of the theory of social capital. The study also provides preliminary evidence about the generative social mechanisms that allow the adaptive deployment of production resources under the

community model.

I used a dataset covering the entire population of 61, 921 reusable Ruby projects contributed to RubyGems.org within a period of ten years from the advent of the Ruby language (2003–2013) as the input for the analyses.

Résumé

Cette thèse examine la dynamique de la collaboration et de la réutilisation des connaissances dans les communautés d'innovation ouverte. Les communautés d'innovation facilitent l'échange des connaissances spécialisées et sont souvent citées comme le principal moteur de l'innovation collective. Or, la recherche actuelle sur les communautés ne met pas les pratiques de réutilisation des connaissances en avant, ni la façon par laquelle ces pratiques sont entre-nouées avec le mode de collaboration communautaire. En se basant sur la littérature de réseaux de connaissances d'une part et la littérature de communautés en ligne d'autre part, cette thèse étudie conjointement la création et la dissolution des liens de réutilisation de connaissances ainsi que les liens de collaboration entre les projets de la communauté open source de Ruby. Pour tester les hypothèses, j'utilise les données de l'ensemble de la population (61, 921) des projets Ruby réutilisables domiciliés sur RubyGems.org dans une période de dix ans à compter de l'avènement du langage Ruby (2003-2013).

Comme prévu par la littérature existante sur les réseaux de connaissances, les analyses démontrent l'impact significatif de la mobilité sociale des développeurs entre les projets sur la réutilisation des connaissances. L'effet de la mobilité sociale sur la réutilisation des connaissances s'accentue lorsque la réutilisation de connaissances complexes est en jeu, soutenant l'argument selon lequel les liens sociaux directs fournissent une bande passante élevée convenant aux échanges complexes. Plus important, les résultats montrent un impact dans l'ordre inverse. La réutilisation des connaissances entraîne une augmentation de la probabilité de création de liens de collaboration entre les projets. Les résultats fournissent également des preuves anecdotiques du rôle important de la réutilisation superficielle des connaissances pour les collaborations ultérieures entre les projets, agissant comme une force de socialisation. De plus, je démontre que les mécanismes de protection de la propriété intellectuelle ont un impact négatif sur la création des liens de collaboration et des liens de réutilisation entre les projets, car ils génèrent des problèmes de compatibilité entre les licences.

Les résultats offrent une vision interactive de la dépendance entre les réseaux sociaux et les réseaux de connaissances, dans laquelle aucun des deux réseaux ne gagne la primauté et la domination sur l'autre. Cela contraste avec les travaux antérieurs sur les réseaux de connaissances qui

supposent que les réseaux de connaissances sont encastrés dans les réseaux sociaux. D'autre part, les résultats fournissent un compte rendu de la façon dont la réutilisation du capital intellectuel peut conduire à la création du capital social, confirmant une hypothèse rarement testée de la théorie du capital social. Cette étude fournit également des preuves préliminaires pour expliquer les mécanismes sociaux générateurs qui permettent le déploiement adaptatif des ressources de production dans le modèle communautaire.

Acknowledgements

Most people give or get back to their schools but because of a teacher that changed their life. I am no exception. If I am here today, it is to the credit of the teachers who raised my curiousity, sparked my interest, taught me how to formulate questions and showed me how to persevere until I obtain the answer. I am grateful to the late Dr. Mohammad Ayoubi for teaching me the virtues of critical thinking, Prof. Marie-Laure Salles-Djelic for teaching me what to think about, and Prof. Samer Faraj for teaching me how to think about it. I am also deeply indebted to my brother and coach Dr. Hojjat Hosseini who taught me peseverance in the face of adversity. I would not have made it this far had I been relegated to my own limited reserve of patience.

During the course of my dissertation work I have benefited from the inputs of several faculty members, colleagues and friends in McGill and HEC communities. The courses I have attended in the joint PhD program are too many to mention here, but they have for sure left their mark on me as a researcher. The core idea of my thesis was first developed in a course led by Dr. Animesh Animesh and Dr. Kunsoo Han. Later on the feedback and support from Dr. Corey Phelps were crucial for finding the right tools and methods, as well as for refining my theoretical perpective. Most crucially, Prof. Samer Faraj, chair, was always there with the right questions and the right hints to guide me through the intricacies of the academic process.

Among my colleagues, Karla Sayegh has had an important influence on my work. Since her arrival at McGill, Karla has been both a research buddy and an older sister to me. We have spent numerous hours discussing ideas and rehearsing presentations. It is not difficult to find her traces here and there, on or in-between the lines, in this thesis.

During my doctoral studies I have recieved financial support from Fonds québécois de la recherche sur la société et la culture (FQRSC), The Rathlyn Foundation, and Mr. Hian Siang Chan. I have also relied often on McGill's excellent HPC services at the data crunching stage. I would like to thank all of the above individuals and institutions for their generous support.

This long journey would not have been possible without the extreme patience and the unwavering support of my wife and teammate for life, Agata. She is not only a great mother and a loving wife,

but also a capable copy editor. She has kept my mind at peace, my stomach happy and my writing error-free so that I can focus and thrive. She could claim the rights to a significant portion of this work — such has been her contribution.

In the end, I would like to thank my parents, Aziz and Vahideh, not only for their support during my doctoral studies, but for their unshakeable faith in me combined with their selfless support at all the stages of my studies. I can only aspire to be a parent like them one day.

Table of Contents

List of Tables

List of Figures

Introduction

This thesis examines the dynamics of knowledge collaboration and reuse in community forms of organizing. It follows in the steps of the vast body of research that investigates the association between social capital and the creation of intellectual capital. Intellectual capital, whether in the form of innovation, novel ideas or knowledge stocks, is a much sought-after source of economic value-added and competitive advantage in contemporary knowledge economies (Argote & Ingram, 2000; Boisot, 1998; Drucker, 1969; Fleming, 2001; Nelson & Winter, 1982; Schumpeter, 1934). Knowledge reuse and recombination are key in the creation of intellectual capital, a social game in which the non-redundant knowledge required to achieve novel thinking and the social influence needed to establish and deploy innovative outcomes are attained through social engagement (J. Singh & Fleming, 2010). By studying knowledge collaboration and reuse in community forms, this thesis seeks to shed light on some of the processes that build up the innovative capacities of those communities.

Organization studies have long inquired into the organizational life of knowledge: knowledge reuse and recombination, as well as knowledge collaboration and co-creation. In the last four decades, the knowledge perspective (i.e., broadly defined) has fully permeated the mainstream in organization studies, increasingly informing our perception of how modern organizations operate and create value in the knowledge economy (Grant, 1996; Spender, 1996). Starting from 1970s, the word *knowledge* has increasingly appeared in the titles of the articles published in the 20 leading journals in management, economics, psychology and sociology. The number of network studies with a focus on knowledge has also multiplied (Phelps, Heidl, & Wadhwa, 2012).

The work on the knowledge perspective covers topics from organizational learning (Brown & Duguid, 1991; Nonaka, 1994; Wenger, 2000) to innovation adoption (Burt, 1987; Coleman, Katz, & Menzel, 1957; Rogers, 1962), with a wide range of findings. One of the most important takeaways of the knowledge perspective is that knowledge, despite its immaterial nature, is both socially and geographically embedded. The focus on the capacity of the social actors to transfer and reuse knowledge has given rise to a distinct body of research on knowledge networks (Phelps et al., 2012). The studies from this line of research are unequivocal about the significance of the relationship between

the social structure and interactions, seen through the prism of social networks, and the patterns of emergence, mutation and resurgence of ideas, as reflected in knowledge networks (Inkpen & Tsang, 2005).

For instance, the social network studies of innovation diffusion and adoption have found a link between social network patterns such as cohesion and social equivalence and the trajectory of information diffusion and adoption of innovations (Burt, 1987; Coleman et al., 1957; Rogers, 1962). Studies of corporate knowledge spillovers have contributed to our understanding of firm-level knowledge localization and de-localization by demonstrating how the social mobility of the individuals across firms and loci allows the firms to curb their knowledge search limitations (Almeida & Kogut, 1999; Jaffe, Trajtenberg, & Henderson, 1993; Rosenkopf & Almeida, 2003) Finally, the studies of intra-organizational social and knowledge exchange networks have demonstrated that the relations between organizational units promote knowledge exchange, conditioned by the type of relationship and the type of knowledge sought (Hansen, 2002; Hansen, Mors, & Løvås, 2005). Put shortly, the effect of social networks on knowledge networks is well-documented, often justified by the way the different facets of social capital inhabit social networks (Nahapiet & Ghoshal, 1998).

The relationship between intellectual and social capital is not, however, uni-directional. Indeed, Nahapiet and Ghoshal (1998), as well as Brown and Duguid (1991), characterize the relation between the two forms of capital as an interrelation, or a coevolution, of knowledge and relationships, citing the importance of collective forms of knowledge in sustaining collaboration. But can one deduce that social networks are embedded in knowledge networks in the same way knowledge networks are embedded in social networks? In other words, are there good arguments to support the effect of knowledge networks on social ties and structures? While the empirical evidence is heavily tilted towards the effect of social networks on knowledge networks, I argue that it makes intuitive sense to assume the existence of a reverse relationship, given that knowledge networks define our exposure to knowledge, including knowledge about the others, their expertise and possible beneficial relationships. The empirical evidence is still modest (e.g., Phene & Tallman, 2014; Rosenkopf & Nerkar, 2001), what leaves ample room for further investigation.

But for all the research done on the knowledge perspective in the formal organizations, the mere subject of the knowledge perspective, the development of novel ideas, has been slowly but steadily slipping away from this context. The lowering of barriers to access knowledge, technology and communication has brought about a shift in the *locus of innovation*, or where the novel ideas see the light of day and where they are incubated and brought to fruition (von Hippel, 1994). Recognizing this shift, Chesbrough's popular book (2006) and the follow-up research on *open innovation* have made a case for opening up the innovation processes of the firm by highlighting the increasingly distributed forms these processes can take and advocating an innovation model in which the firm boundaries are more permeable to the exchange of knowledge and ideas (Boudreau, 2010, 2011; Chesbrough & Crowther, 2006; Laursen & Salter, 2006). Yet, with a few exceptions, the learnings of this body of research remain confined to the corporate context and therefore to the formal organizations, rarely discussing the emerging and unconventional forms of organizing.

Some authors, however, have taken a step further and conjectured that communities, and not formal organisations, are quickly becoming the main platforms for knowledge creation and therefore for innovation (Adler & Kwon, 2002; Demil & Lecocq, 2006; von Krogh & von Hippel, 2006). Communities are "collectivities of people who share a common experience, interest, or conviction; who experience a positive regard for other members; and who contribute to member welfare and collective welfare" (Sproull & Arriaga, 2007, p. 898). Communities have seemingly existed throughout the history, but the advent of electronic communication technologies has ushered into a new era where the online variant of communities can serve as a viable organizational form for scalable knowledge sharing and collaboration (Faraj, Jarvenpaa, & Majchrzak, 2011; Faraj, von Krogh, Monteiro, & Lakhani, 2016; O'Mahony & Ferraro, 2007; Wasko & Faraj, 2005). The community form, thus, is a renewed form of organizing, emergent in its contemporary form. Communities, the argument goes, carry less formal structures impinging on the exchange of ideas, and their trust-based social contract is ultimately better suited to host the collective processes of melding and recasting of ideas (Adler & Kwon, 2002; von Krogh & von Hippel, 2006). Some of the best-known technological achievements and social accomplishments of our time bear testimony to that. Linux, Wikipedia,

Apache and the other well-known products of community work have not garnered their reputation as merely working solutions coming out of ad-hoc organizations. They rather became known as they evolved into dominant solutions, with a seemingly inexhaustible degree of adaptive innovation, surpassing that of the commercial competitors. Many of the underpinnings of the global information infrastructure that supports our knowledge collaborations have their roots in the community work, and more specifically in *open innovation communities* (Fleming & Waguespack, 2007).

Open innovation communities are electronically mediated communities composed of "volunteers who work informally, attempt to keep their processes of innovation public and available to any qualified contributor, and seek to distribute their work at no charge" (Fleming & Waguespack, 2007). They provide a knowledge ecosystem with an alternative governance mechanisms relying on incentives, legal frameworks and social structures not directly comparable to those of the formal organizations. (Lerner & Tirole, 2002; Murray & O'Mahony, 2007; O'Mahony & Ferraro, 2007). Openness, loosely-knit social fabric, ad-hoc trust-based control structure, and weakness of direct incentives in communities have all prompted the scholars to speculate about the capacity of communities to host knowledge ecosystems and constitute a third mode of governance beside markets and hierarchies (Adler & Kwon, 2002; Demil & Lecocq, 2006; von Krogh & von Hippel, 2006). Nonetheless, despite the strong track record of research on open innovation communities, few studies to date have directly addressed the questions regarding knowledge creation and reuse in open innovation communities (Faraj et al., 2016; Haefliger, von Krogh, & Spaeth, 2008; Kyriakou, Nickerson, & Sabnis, 2017; Murray & O'Mahony, 2007; P. V. Singh & Phelps, 2013; Sojer & Henkel, 2010).

The early research on open innovation communities was characterized by a fascination about the possibility of highly skilled individuals working for free on publicly available projects, and thus mainly focused on the labour supply dimension of communities. Motivations, participation and contributions by individual community members to the community outcomes (e.g., forum posts, code donation, bug reports, etc.) where among the topics most frequently explored (Benbya & Belbaly, 2010; Hertel, Niedner, & Herrmann, 2003; Lerner & Tirole, 2002; Mockus, Fielding, & Herbsleb, 2002; Shah, 2006; von Krogh, Haefliger, Spaeth, & Wallin, 2012). This body of work has

shed light on an array of social, psychological and economic motives that underpin the participation of individuals in community activities. These include, but are not limited to, self-interest (Lerner & Tirole, 2002; von Krogh & von Hippel, 2006), social exchange (Faraj & Johnson, 2011; Wasko & Faraj, 2005), identification (R. P. Bagozzi & Dholakia, 2006; Hertel et al., 2003), self-enjoyment (Franke & Shah, 2003; Shah, 2006) and altruism (Hars & Ou, 2002).

The interest later shifted to a macro perspective with the study of a variety of social and structural factors that shape the participation of individuals and guarantee the sustainability and the viability of the communities. These include, though ares not limited to, community size, critical mass, and social structure (Baldwin & Clark, 2006; Butler, 2001; Peddibhotla & Subramani, 2007), community governance (Dahlander & O'Mahony, 2011; O'Mahony & Ferraro, 2007), project success (Grewal, Lilien, & Mallapragada, 2006; P. V. Singh, Tan, & Mookerjee, 2011) and corporate relations (Bonaccorsi, Giannangeli, & Rossi, 2006; Economides & Katsamakas, 2006; Fosfuri, Giarratana, & Luzzi, 2008; West & Gallagher, 2006).

This extensive focus on participation, viability and their antecedents in the published research about open innovation communities has been deemed rooted in a historically informed lack of trust among organization scholars in the viability of communal forms of organizing due to the absence of reliable mechanisms to align the individual and collective interests (O'Mahony & Ferraro, 2007). But that has detracted from addressing some of the more interesting questions about the substance of work in open innovation communities; or the dynamics of knowledge collaboration and reuse that build up the capacity of these online communities to create intellectual capital (Faraj et al., 2011; Faraj et al., 2016). Notably, little has been done to explore what gained the open innovation communities their reputation — their capacity for fomenting new ideas and incubating innovation.

Taking the knowledge perspective to the study of open innovation communities seems like a logical sequel to the extant research on online communities. This work takes a step in that direction by bringing the study of the interplay between social capital and intellectual capital to the communities. This is in line with the recent calls to cover the dynamics of knowledge collaboration (Faraj et al., 2011) and knowledge reuse (Faraj et al., 2016) in online communities, as well as with an emerging

stream of empirical work that explores the community processes that support knowledge remix and recombination as vectors of open innovation (e.g., Flath, Friesike, Wirth, & Thiesse, 2017; Kyriakou et al., 2017).

By studying online communities of innovation from a knowledge perspective, my effort is to transcend the habitual existential questions about the online community phenomenon and focus instead on the processes and the outcomes of the phenomenon. I depart from the assumption that online communities, like other institutions, are self-perpetuating and self-legitimizing, capable of carrying on with their own institutional inertia and with the objective of maintaining the excellence of their practices (von Krogh et al., 2012). Community activities can be considered self-motivating practices that ultimately grant online communities their viability as social and organizational forms. As such, I de-emphasize the questions about the way online communities obtain their resources in favour of questions about the way online communities regulate and deploy their social and intellectual resources.

With this focus, I raise several inter-related research questions. Together, the questions are intended to draw a broad picture of knowledge work in open innovation communities: *How can one measure the reuse of knowledge outcomes in open innovation communities? How do community participants reuse each other's work within the context of community projects? How do the social structure of knowledge reuse and that of collaboration interact with each other in these communities?* And finally, *how do the specificities of open innovation communities relate to those patterns of knowledge reuse and collaboration?*

By asking, and subsequently trying to answer these questions, I pursue three objectives.

The first is to extend the line of study on knowledge networks to the new context and confirm or revisit the earlier findings to establish a baseline for the relevance of the knowledge perspective in this setting. That entails testing whether the existence of links in the social network of collaborations has an impact on the creation of new links in the networks of knowledge reuse. But how do the previous findings regarding the relation between social capital and intellectual capital hold in the community settings? Do social ties have an equal bearing on the way knowledge is reused in open

innovation communities given that the barriers to knowledge exposure, exchange and reuse are lower or at least different in these communities?

The second objective of this study is to serve as an empirical probe into the less studied aspect of the relation between social and intellectual capital, namely the impact of knowledge networks on social networks. I do so by testing whether the existence of links in the network of knowledge reuse has a discernible impact on the creation of new links in the social network of collaboration. Empirically demonstrating this point is key to the main thesis defended in this work, which is *an interactive view of knowledge in newtorks*, referring to the co-evolving nature of the two types of capital.

Third, this study aims to contribute to the ongoing effort to theorize knowledge work in online communities by offering a mid-range theory of knowledge reuse practices in open innovation communities.

I subscribe to the view that studying community forms offers a unique opportunity to transcend our views beyond the limited context of formal organizations and contributes to our broader understanding of collaboration in organizations at large (Faraj et al., 2011). Yet, the particulars of the social phenomena often far outweigh what can be directly deduced from general theories, and the only way to inform our general theories with original substance is by attending to the specifics of the social phenomena and developing middle-range theories (Merton, 1968a).

Therefore, although this study clearly aims at universality and generalizability in its propositions, it also partially tells the tale of how online communities function by revealing the interactive dynamics that tie the two forms of capital together in this setting. As such, the study's propositions may at times be phenomenon-driven, although theoretically informed, and certain findings are expected to be phenomenon-dependent. By investigating the phenomenon-specific issues the hope is to learn more about the way open collaborations operate in general, but also about the mechanisms that give open innovation communities their competitive and innovative edge.

In this thesis, I particularly focus on open source communities as the empirical site. Open source communities are open innovation communities whose principal objective of activity is soft-

ware development. The work done in open source communities is, by definition, knowledge work and no phenomenon represents both collaborative and innovative aspects of online communities better than open source communities. Open source community members are composed of core and occasional developers, participating users and by-standers. They cover a range of activities, including software design, development, testing, bug reporting and documenting, as well as member-to-member help and sometimes aesthetic design (Hars & Ou, 2002).

Open source communities present a data advantage that makes them particularly suitable for the kind of questions being raised in this thesis. Most open source communities provide detailed information on projects and contributors, consecutive snapshots of project data including both the original code and the specification metadata, and a systematically compiled log of individual contributions to different projects across time. This makes it possible to respond to questions that require whole-network and dynamic samples. But apart from the scale of the data, the scope of the data is also impressive in open source communities. Most organizational studies resort to proxy variables, self-reported data or communication data in order to document collaboration. The fact that the substance of work, the contributions and the work process in open source communities are all public and in digital domain provides the unique opportunity that the work itself can be measured with no need for proxy or secondary sources.

The reuse of knowledge commonly occurs in open source communities (Haefliger et al., 2008; Sojer & Henkel, 2010). However, our knowledge of reuse behaviour, its antecedents, and its consequences in open communities remains limited. The dearth of theoretically-motivated studies in this area leaves the venue open for investigation. In this dissertation, I draw on the theory of social capital, the literature on knowledge networks, and the literature on online collaboration and online communities to formulate propositions with regard to reuse and collaboration behaviours in open innovation communities. Subsequently, I proceed to test those propositions, borrowing methods from fields as far apart as political science and computer science.

In the following pages I start by laying out the theoretical framework supporting the study. Next, I bring together the perspectives covered in the framework section to develop propositions

that I formulate as testable hypotheses. The following methods section describes the data gathering, treatment and analysis methods used in the study, as well as the findings of the analysis. In the two final sections I will discuss the findings and their implications.

Theoretical Framework

Open Source Communities as Open Innovation Communities

Participation & Collaboration in Open Source Communities. Each open source community is composed of a large number of projects, and the individual developers are usually free to contribute to the projects of their choice. Community membership and activity are based on free voluntary participation. The vast majority of open source projects are single-developer, which is, their permanent members are not more than one. To these one-man projects, collaboration means attracting temporary contributors or accepting occasional contributions from community members. While these smaller projects are typically managed by their founders, the larger projects often incorporate meritocratic and democratic elements in their governance model (Kogut & Metiu, 2001; O'Mahony & Ferraro, 2007). Collaboration also takes fuller dimensions in large projects, as different sections of the collective work fall under the supervision of different contributors and large-scale changes require several interested parties to fall in line.

Achieving this latter is not a certainty. Whether the administrators of a project keep a tight grip on its management or adhere to a participatory mode of governance, disagreement about the future of a given project, a specific feature or the mode of participation may arise from time to time. Disagreements and conflicts in the governance of such projects are not always resolved uneventfully and sometimes end up provoking hostile project forks, whereby two groups of developers start working on two diverging and competing versions of the code with no intention to re-merge. Project forks have often been pointed out in the literature as a peril for open projects and a cautionary tale about the potentially wasteful allocation of resources in community forms (Fleming & Waguespack, 2007; Kogut & Metiu, 2001).

The free, voluntary participation in open projects has often been misinterpreted as being synonymous to unpaid work. Project member surveys have drawn a different picture of what the voluntary work is composed of. Between one and two-thirds of the contributors to the large community-managed projects have sponsors and are allowed to work on the projects as a part of their employment (Lakhani & Wolf, 2005; O'Mahony, 2003). The participation is, however, free and voluntary

in the sense that there is no pecuniary compensation mechanism nor binding contract controlled through the governance structure of the projects. The decision to move in, out or between projects is only constrained by the tacit consent among individuals, as well as by the social and cultural considerations such as communal norms.

There are strong norms of reciprocity in open source communities that appear in form of a generalized gift culture (Raymond, 2001), free peer-to-peer assistance (Franke & Shah, 2003; Lakhani & von Hippel, 2003), and a pressure to contribute to the commons (O'Mahony, 2003). These norms are rooted in the programming culture and enshrined in the open licenses (Williams, 2011). They are regularly enacted and enforced in the online interactions and the discussion forums (O'Mahony, 2003). That explains to some extent the developers' urge to give back to the community by means of getting involved in community projects (Wasko & Faraj, 2000).

Open source projects have been often described as layered collectivities with concentric circles of core members, peripheral members, inactive members and simple users (Dahlander & O'Mahony, 2011). Progression within the projects has been described in terms very similar to that in communities of practice; a move from the periphery to the core driven by contributions and leading to gradual gains in legitimacy (Dahlander & O'Mahony, 2011; Wenger, 2000). This process of legitimate peripheral participation allows the newcomers to observe and the novices to improve their knowledge through practice, until they are fully socialized within the community (Wenger, 2000). A developer can be simultaneously core to some projects while peripheral to others.

Development of Intellectual Capital Under The Cumulative Regime. Most human achievements in the domains of technology and innovation are based on the accumulation and combination of relevant and original ideas.

This accumulation can only occur if the prior art is disclosed and accessible to each new generation of creators, that is, they must know about the existence of such prior art as well as having the possibility to explore and reuse it in derivative work. Furthermore, there must exist a reward mechanism, either remunerative or reputation-based, that encourages the creators to disclose their work and make it accessible to the future generations. Intellectual property regimes that secure access,

disclosure and rewards can be considered as favourable to cumulative innovation and continuity in the creation of intellectual capital (Murray & O'Mahony, 2007).

The relative importance of each of these antecedents is the subject of debates. The conventional economic theories emphasize disclosure and rewards as the most notable antecedents to the accumulation of knowledge (e.g., Dasgupta & David, 1994). From the vantage point of IP prospect theory, strong IP protection facilitates the creation and maintenance of knowledge through licensing (Gallini & Scotchmer, 2002; Kitch, 1977), and this is all the more true in the digital age given the declining cost of reproduction. Others have taken a more liberal approach to IP protection, noting that with the diminishing cost of access due to digitization, reuse is now mainly hindered by copyright laws (Benkler, 2002; Lemley, 2004; Lessig, 2004). These latter have argued that only when disclosure is complemented with the possibility to access knowledge for reuse and recombination, it may lead to accumulation, while copyright mainly restricts the access (Adler, Kwon, & Heckscher, 2008; Lessig, 2008).

One distinguishing factor about the open innovation communities is their approach to intellectual property protection, which relies on alternative assumptions about the three antecedents of cumulative innovation. While the mainstream intellectual property protection regimes have historically given primacy to remunerative rewards for the creators at the expense of access to knowledge, open innovation communities are built on the promise of free and unlimited access to knowledge for reuse, along with wide permissions for recombination. The intellectual property generated in open innovation communities is governed by open licenses.

Open licenses come in different types and flavours, with the main differences rooted in the stance the licenses take towards derivative works — whether derivative works are allowed, whether they can be commercialized, and under what terms. Common among open licenses is their support for unhindered access to knowledge for reuse. With the exception of a few minor or special-purpose licences (e.g., CC BY-ND & CC BY-NC), the large majority of open licenses also allow for modification and creation of derivative work for both commercial and non-commercial purposes. The factor that divides the range of existing licenses into two broad categories is the terms of distribution for

the derivative works, or more precisely the requirement for reciprocal licensing. Reciprocal licensing is the practice of preserving distribution rights in derivative works down the line, and thus requiring the same or similar license as the original for the copies and the adaptations ("CC v.4," 2013). Licenses with reciprocal licensing stipulations are labelled as Share Alike or Copyleft, while those without are commonly referred to as permissive licenses.

Apart from cementing the values of openness and free revealing within the community, licenses are one of the most important mechanisms the open innovation communities have put in place to guard their intellectual commons (O'Mahony, 2003). Reciprocal licensing requirements not only guarantee the accessibility of the commons to the general public, but also guarantee the maintenance of accessibility by turning the future derivative works into commons. Although the copyleft licenses do not restrict the commercial use, in effect they curb commercial appropriation of the commons by turning over the copyright law to prevent behaviour that might threaten the public availability of the commons (Stallman, 1999). One can take from the commons as much as needed, as long as one does not take *away* from the commons.

The promise of open innovation communities is that, all else equal, open access translates into lower effort needed for knowledge acquisition and generally a more efficient knowledge exchange process as compared to the proprietary contexts. This, it is argued, allows those who are best positioned to identify the issues to have access to the required knowledge to enter the process of problem-solving. The shift in the locus of problem-solving has been reported as one of the distinct features of the open innovation communities, and has been credited with distributing and democratizing innovation (Von Hippel, 2001; von Hippel, 1994, 2005).

Nevertheless, the flip side of giving primacy to access in the open access model is the lack of certainty about the reward mechanisms, and thus about the supply of workforce. As mentioned above, free voluntary contribution in open innovation communities means that there is no remunerative reward mechanisms tied into the governance structure of the community projects. In absence of those, the communities rely on reputation effects, and strong reciprocity norms to regulate the behaviours and align the individual and community interests. But no matter how strong the reputation

effects and the normative pressures, they can not entirely replace remunerative rewards. Certain open source initiatives secure the inflow of rewards by adopting dual licensing for commercial uses, by engaging into freemium economics using a razor-and-blades strategy, or by asking for donations (Fosfuri et al., 2008). Yet, the path to monetization of open source projects is not always clear, and micromanaging community participation is more likely to backfire than to improve collective performance (West, 2003; West & Gallagher, 2006). Ultimately, the community-managed projects operate as a dynamic nexus of social, economic and technical interests, and their sustenance is a function of equally dynamic renovation of this nexus.

Focusing specifically on open source communities, Haefliger et al. (2008), as well as Sojer and Henkel (2010), have noted several behavioural patterns in community-managed projects related to the expected limitations in access to human capital. The two studies have found important links between the mode of provision of human capital and the reuse practices in the communities. I will further discuss these findings under the section Reuse in Open Source Communities.

However, before narrowing down to the topic of reuse in open innovation communities, I will provide an overview of knowledge reuse and its entanglement with the different facets of social capital as presented in the extant research.

Knowledge Reuse Networks

Knowledge exchange and reuse are essential to the production of intellectual capital. To produce means "to combine materials and forces within our reach" and development is thus "defined by the carrying out of new combinations" (Schumpeter, 1934, pp. 65–66). Intellectual capital is similarly developed through the recombination of the previously known (Moran & Ghoshal, 1999). Novelty lies, therefore, in devising new assortments of the existing ideas arranged into unprecedented configurations (Fleming, 2001; Nelson & Winter, 1982). Where knowledge is distributed among different parties, exchange fast becomes a prerequisite for reuse and recombination. Since novel ideas often draw upon the expertise of different parties, their development relies on the exchanges between the parties, as well as the ability to reuse the gained knowledge.

If we conceptualize social networks as consisting of nodes and relationships among nodes (Wasserman & Faust, 1994), then knowledge exchange can be defined as observing the flow or the diffusion of knowledge from one location in the network to the other loci. Since individuals are both vehicles of knowledge and the driving force behind knowledge creation, the social movements of individuals, the social ties between them, as well as the social structure in which they are embedded, are instrumental in both knowledge exchange and knowledge creation (Abrahamson & Rosenkopf, 1997; Susarla, Oh, & Tan, 2012). For instance, social ties can bring the interested parties together and act as a conduit for knowledge exchange. The same holds at the more aggregate levels of analysis. Networks of human collaboration that connect collectivities of human actors show similar properties and equally interact with the knowledge creation capacities of these collectivities (Inkpen & Tsang, 2005).

There is an expanding body of literature on *knowledge networks* that examines the social structure of knowledge outcomes (Phelps et al., 2012). Knowledge networks and the knowledge flows they incarnate intervene at several stages of knowledge creation, including access to knowledge, knowledge transfer, knowledge reuse, and recombination. The research findings are unequivocal in their support for the positive effect of knowledge sharing on problem solving and innovative capacity. At the team level, the studies report that the development of shared mental models improves mutual learning and ultimately improves the capacity of the teams to solve complex problems in innovative ways (Akgün, Byrne, Keskin, Lynn, & Imamoglu, 2005; Austin, 2003). At the corporate level evidence is accumulating that improved interaction between firms and organizational units increases the likelihood of value creation through product innovation when it promotes resource exchange and combination (Tsai, 2001; Tsai & Ghoshal, 1998). The relationship between knowledge exchange and knowledge creation has turned out to be stable in the contexts it has been studied. The approaches to quantifying knowledge exchange are, however, multiple.

Measuring Knowledge Exchange. The literature on knowledge networks has employed a variety of different measures to quantify or qualify knowledge exchange. The choice of measures is not merely an operational issue when it comes to knowledge exchange. It is important to distinguish

between these measures as they have often been referred to interchangeably, while they correspond to phenomena that are not necessarily analogous, although partially overlapping.

The most straightforward measure of knowledge exchange can be obtained by recording the act of expressing or carrying the knowledge to the recipient. Measures such as electronic or direct exchange count (Bouty, 2000; Centola & Macy, 2007; Reagans & McEvily, 2003) or direct knowledge contribution (Faraj, Kudaravalli, & Wasko, 2015) belong to this group. Communications between network members or contributions to discussions do not always translate into knowledge exchange and even less often bring in relevant knowledge, yet communication records remain one of the best proxy measures for knowledge exchange intensity. I refer to this category of measures as "knowledge transfer" measures.

Another well-established measure for knowledge exchange can be obtained by tracking down the traces of knowledge coming from a known origin in the knowledge outcomes of a focal recipient. Measures like patent citations (Mowery, Oxley, & Silverman, 1996; Rosenkopf & Almeida, 2003; Wang, Rodan, Fruin, & Xu, 2014) and academic citations (Acedo, Barroso, Casanueva, & Galán, 2006; Taylor, Dillon, & Van Wingen, 2010) are of this kind. I refer to this category of measures as "knowledge reuse" measures because they primarily indicate whether the acquired knowledge has played a formative role in the creation of new knowledge. While knowledge reuse occurs only in a subset of knowledge exchange cases, it is safe to assume that where there is knowledge reuse, there is direct or indirect knowledge exchange.

Knowledge reuse measures are not empirically tied to communication and knowledge flows in social networks and are conceptually independent from the relational properties of social networks. The phenomenon they measure falls midway between knowledge transfer and knowledge recombination, what can be labelled as "effective knowledge transfer" — knowledge transfer that has left a tangible effect on the knowledge outcomes.

Knowledge reuse measures are not measures of knowledge recombination neither, as they do not quantify the variety of sources knowledge outcomes draw on, nor the proportion of knowledge drawn from elsewhere. An ideal measure of recombination would have to provide information on

the diversity of the combined sources and their proportional contribution to final outcomes as well. The reuse measures, however, are only one causal link away from knowledge recombination and are well suited to the study of innovative capacities. A comprehensive knowledge reuse dataset can potentially be transformed to measure recombination as well.

There are other measures related to knowledge exchange that have been operationalized in the literature. For instance there are measures of innovation adoption (Coleman et al., 1957; P. V. Singh et al., 2011). These are not knowledge exchange measures per se, although they do testify the diffusion of some type of knowledge or knowledge artefact. But given the non-rival nature of knowledge as a good, any knowledge transmission can be considered diffusion as well, and it is hard to draw a line between the two concepts. Adoption measures are different from reuse measures in that they do not focus on the innovation process, neither on innovative outcomes. Nonetheless, they are effective measures when it comes to the study of innovation diffusion.

Lastly, learning measures can be adapted to the study of knowledge exchange (Uzzi & Lancaster, 2003). The concept of learning, both at the individual and the organization level, has much overlap with the concept of knowledge exchange (Brown & Duguid, 2000), whereas learning often involves the internalization of some type of knowledge by social actors and thus goes beyond the linear process of resource exchange between the actors (Argyris, 1976).

Given the motivations and the objectives of this thesis, I mainly focuses on the reuse-type networks. The term "Knowledge Network" has been used interchangeably with the term "Knowledge Reuse Network" throughout.

Social Networks vs. Knowledge Networks

The Theory of Social Capital. The study of different patterns of social connectedness and the value that can be derived from them has given rise to the theory of social capital. Central to the theory of social capital is the thesis that "networks of relationships constitute a valuable resource for the conduct of social affairs" (Nahapiet & Ghoshal, 1998, p. 243).

The concept of social capital is inclusive of social ties as well as the structure of social networks

and the resources that can be mobilized through them (Bourdieu, 1986b; Burt, 1992; Putnam, 2001). This endows social capital with two main dimensions or facets: (1) The "relational facet" that refers to the properties of interpersonal relationships and what derives from them, and (2) the "structural facet" that refers to structural features of the network at abstraction levels above interpersonal level, including but not limited to overall network structure, local network structure, and network position (Phelps et al., 2012).

In addition to the two original facets of social capital, Nahapiet and Ghoshal (1998) have proposed a third facet of social capital referring to those resources providing shared representations and understandings among network members. Shared understandings emerge across social networks and are critical to expert collaboration and knowledge creation (Bechky, 2003; Carlile, 2004). All the facets of social capital ultimately facilitate social action.

While most authors would agree on the facets of social capital, that it may emerge at micro, meso or nano levels, opinions differ on what would be an appropriate measure for social capital that would allow us to distinguish social capital when observing a social network. This is not entirely surprising considering the teleological definition of the notion of social capital that ties the recognition of the phenomenon to the "conduct of social affairs". Inevitably, what one considers as "capital" is tied to what one considers a desirable "social affair". As a consequence, different social network concepts, sometimes of opposing nature, are often discussed under the umbrella term "social capital" (e.g., network closure vs. structural holes, or the strength of weak ties vs. the strength of strong ties).

It is this broader interpretation of the notion of social capital that I adopt when discussing the relationship between social capital and knowledge reuse. I use the term as an overarching concept to discuss a variety of structural and relational social network findings that can be consequential to knowledge reuse, and vice versa.

Social Capital as A Driver of Knowledge Reuse. There are four distinguishable elements involved in knowledge reuse: The content, the creator, the reuser and the context. Social capital, embodied in social networks, interacts in all its facets with the elements of knowledge reuse (See Table 1). These elements draw the limits of possible when it comes to reuse opportunities and

decisions, and delimit the structure of knowledge reuse networks. Each of the elements of reuse interacts with all the three, or at least the two original facets of social capital.

The Content. The concept of knowledge spans a continuum ranging from our individual understanding of the phenomena (Simon, 1991) to the written representation of the more codifiable aspects of our understanding (Polanyi, 1967). The way knowledge is transformed, exchanged and reused in the social context is closely related to the form in which it resides (Spender, 1996).

Knowledge can be conveyed directly through regular lines of communication only to the extent that it can be codified and made explicit, and only the more explicit knowledge is ready for efficient reuse and recombination. Sharing tacit knowledge often requires deeper social involvement and shared experiences (Nonaka, 1994). Therefore, certain ideas that are more amenable to codification can travel further away in social networks and are more likely to be drawn upon by potential recipients of diverse backgrounds. Otherwise, knowledge that is hard to express in explicit terms, that is difficult to teach, or that is highly system-dependent or complex may not travel far across social networks (Hansen, 1999; Zander & Kogut, 1995) and its beneficiaries will be more localized around the knowledge source where they can maintain high-bandwidth information channels (Aral & Van Alstyne, 2011).

But while strong local connectedness promises efficient knowledge exchange and improved reuse through information advantages, it has been suggested that only structurally diverse links can provide effective knowledge search and broadcasting opportunities (Granovetter, 1973). In an environment where solutions are exchanged between knowledge holders, matching the right problem with the fitting solution requires highly diverse connections, even though it comes at the cost of reduced bandwidth. Weak ties and strong ties both have their strengths, and achieving the right trade-off between the two depends on the information needs of the knowledge recipients (Aral, 2016; Aral & Van Alstyne, 2011).

Therefore, the content of knowledge reuse, or the characteristics of the knowledge being reused, interact with both relational and structural aspects of social capital. While strong ties and denser social fabrics are better suited to mitigating the costs related to complex knowledge transfer, efficient

	Facets of Social Capital		
Elements of Reuse	**Relational**	**Structural**	**Cognitive**
The Content *(Type of Knowledge)*	The Strength of Weak Ties (Granovetter, 1973) The Search-Transfer Issue (Hansen, 1999)	The Diversity-Bandwidth Trade-off (Aral & Van Alstyne, 2011) Structural Holes as Social Capital (Burt, 1992)	
The Source *(The Creator)*	Direct Reciprocity (Blau, 1964; Constant, Sproull, & Kiesler, 1996)	Generalized Reciprocity (Putnam, 2001) Matthew Effect / Preferential Attachment (Barabási et al., 2002; Merton, 1968b)	
The Recipient *(The Reuser)*	Interpersonal Trust (Misztal, 1996)	Structural Trust (Coleman, 1988)	Absorptive Capacity (Cohen & Levinthal, 1990)
The Context	Legal and economic barriers to reuse (Murray & O'Mahony, 2007)	Legal and economic barriers to reuse (Murray & O'Mahony, 2007)	

Table 1. The Facets of Social Capital vs. The Elements of Reuse

knowledge search thrives on weaker ties and sparser social structures which are better suited to bringing in thinner but non-redundant knowledge.

The Creator. The creator can facilitate knowledge reuse by transforming knowledge into a format suitable for conveying or direct consultation (Nonaka, 1994). Thus, the incentives, interests and motivations of the source ultimately impacts the propensity of knowledge reuse. But codifying knowledge and making knowledge artefacts involve high costs, and the creators may not have the motivation to bear those costs (Haefliger et al., 2008; Hansen, 2002).

Legal provisions such as copyright and other similar intellectual property protection mechanisms that give creators exclusive rights over their ideas were originally designed to provide incentives and preserve the motivations of creators to bring their creations to the public (Lessig, 2001). But not all the motivations of creators are pecuniary. The creators may very well be motivated by the other by-products of their creation, from the hedonistic pleasures associated with the process of knowledge creation to the reputation accrued by the social recognition of their work among the peers (Lakhani & Wolf, 2005; Lerner & Tirole, 2002).

Knowledge production networks are characterized by preferential attachment tendencies that lead to a rich-gets-richer situation or a Matthew Effect in favour of those with outstanding reputation (Barabási et al., 2002; Merton, 1968b). This often favours the active creators and indirectly rewards them, through mechanisms such as attracting talented collaborators or bringing about sources of revenue tied to their position in the network. Such rewards constitute an incentive to share.

Social proximity may as well overcome the unwillingness of knowledge producers to expend the additional effort required for producing a reusable knowledge artefact, given that individuals tend to invest more time and resources in their close social ties (Coleman, 1988). From social exchange theory perspective, individuals are likely to engage in social action when they perceive personal benefit, and this increases the chances of acting in direct reciprocity towards close social ties (Blau, 1964; Constant, Sproull, & Kiesler, 1996). That's one of the main reasons why social proximity has been considered to determine the ease of information transfer between the individuals (Coleman et al., 1957), and strong social ties have been found particularly effective in acquiring tacit and complex

knowledge (Barabási et al., 2002; Granovetter, 1973; Hansen, 1999).

A structural outcome of proximity of groups of people is social cohesion. Cohesive social networks with a higher degree of closure lend credibility to collective sanctions and uphold trustworthiness. They also offer the sort of social closure that is required for reputation effects to arise (Burt, 2000; Coleman, 1988). Where reputation effects are combined with effective sanctioning, social networks successfully regulate members' behaviours (Burt, 2001). Social networks may effectively instil pro-social normative behaviours such as generalized reciprocity in their members (Putnam, 2001).

The Reuser. The reuser or the recipient of knowledge is the person who appropriates the knowledge created by the others and includes it in their work. Since the reusers carry the responsibility of the outcomes of their work, the decision to reuse must be interpreted as accepting vulnerability to the work of a third party. Therefore, the reusers need to ensure the quality of work they combine with or rely on. Given that quality assurance is costly by nature and requires information about the production process that is not always accessible, both trust and reputation become defining factors for the reusers.

Trust is the belief that "results of somebody's intended action will be appropriate from our point of view" (Misztal, 1996, pp. 6–10) and has both relational and structural dimensions. Those networks that are good at enforcing generalized norms of cooperation (e.g., high degree of closure), are better at fostering trustworthiness as well (Putnam, 1993). But personal relationships also create obligations and expectations between the individuals that go beyond the generalized norms in controlling behaviour (Bourdieu, 1986a). The existence of one of these two as a short-cut quality indicator eases the choice and the task of reuse.

A palpable example for the two kinds of trust emanating from two dimensions of social capital is the choice of citations in academic publications. We generally rely on the peer review system as well as the professional norms to weed out the weaker links in the academic works, but also give particular importance to the works published or recommended by the individuals we personally know. The rich social cues we have about the individuals close to us allow us to put their choices

in perspective in a way that it reduces our uncertainty in a significant way. Similar considerations drive the reuse choices in other communities dealing with knowledge creation on a daily basis. Both structural and relational dimensions of social capital affect the reuse choices of the reusers.

Successful reuse also depends on the capacity of the reuser to appropriate the other party's knowledge outcomes. Knowledge reuse cannot be accomplished unless the recipient has the absorptive capacity to decode, make sense of, evaluate, assimilate, and exploit the knowledge. Absorptive capacity is a function of the prior relevant knowledge, as well as the general experience of the recipient (Cohen & Levinthal, 1990). A part of the social capital a person holds is the shared understanding and shared context the person develops over time throughout her exchanges and collaborations with groups (Wegner, Erber, & Raymond, 1991). This shared understanding may act as a primer for the individuals to develop their absorptive capacity on topics relevant to their exchanges, and are thus conducive to more successful reuse efforts.

The Context. The context consists of "the surroundings associated with phenomena which help to illuminate that phenomena, typically factors associated with units of analysis above those expressly under investigation" (Cappelli & Sherer, 1991, p. 56). The powerful impact of contextual factors on cause-and-effect relationship are often non-trivial, yet frequently ignored (Johns, 2006). Our current understanding of knowledge reuse and recombination practices is deeply tied to the corporate setting in which these phenomena have been studied. These settings are very particular in that they severely constrict the possibilities for knowledge exchange and reuse. It is possible to hypothesize about the potential effects of this particular context on the relationship between the variables of interest. These effects are non-trivial and the relationships uncovered in this context are susceptible to change in other contexts.

The corporate context has an effect on the development of social capital by drawing clear-cut lines between insiders and outsiders, and limiting social exchanges between in-groups and out-groups. Moreover, corporations traditionally rely on proprietary intellectual property regimes and corporate secrets to protect their knowledge resources from imitation and to ensure that the pecuniary rewards stemming from their innovations flow back to them (Nickerson & Zenger, 2004). This

involves ensuring that the social boundaries and the knowledge boundaries of the formal organization of the firm coincide, and thus altering the knowledge exchange paths and the knowledge reuse practices.

In an environment where collaboration and reuse are not primarily affected by concerns for formality and direct appropriation of rewards, the relationship between social capital and knowledge reuse, and ultimately the creation of intellectual capital, could be different (Murray & O'Mahony, 2007). For instance, the flow of human capital could be less hindered by the tedium of organizational affiliation, and the barriers to knowledge flows could be less of a function of organizational boundaries. In such a context, knowing an expert in another organization would not mean the same in terms of exceptional access and exposure to the knowledge resources of that organization.

I conclude that one may legitimately expect the legal and economic context to play at least a moderating role, if not a direct one, in the interaction between social networks and knowledge reuse.

Knowledge Reuse as A Driver of Social Capital. The literature on knowledge perspective has maintained that the relation between social capital and intellectual capital is potentially bi-directional (Brown & Duguid, 2000; Nahapiet & Ghoshal, 1998). Yet, the way the different facets of intellectual capital can drive the emergence of social capital is seldom explored. There are at least three lines of argument that testify knowledge exchange and reuse have a bearing on social capital by promoting rapprochement between the creators and reusers for further collaboration.

The first line of argument is rooted in the empirical research on firm alliances. Research in this area, mostly stemming from business economics and strategy, has viewed knowledge spillovers, or the unintentional and uncompensated exchanges of knowledge, as potential signals for convergence. Knowledge spillovers can reveal resource complementarities and as such signal opportunities for closer collaboration ties to both the originator and the recipient (Malmberg & Maskell, 2002; Phene & Tallman, 2014; Yang, Phelps, & Steensma, 2010). For instance, Rosenkopf, Metiu, and George (2001) have shown that deep knowledge exchanges among company representatives within the framework of a technical committee paves the way for subsequent alliances. And in a more recent work, Phene and Tallman (2014) have demonstrated that patent citations between firms in the biotechnology

industry increase the chance of subsequent alliance formation.

These works have articulated the mechanism behind this relationship as one related to pursuing knowledge acquisition and organizational learning. They suggest that there is always tacit knowledge associated with what is revealed through spillovers and that the best way to get access to that tacit knowledge is through close collaboration, what justifies the creation of explicit social links above and beyond benefiting from the spillovers. Moreover, the argument goes, the reuse of the revealed knowledge stock can create a shared tacit understanding of the applications of the technologies and the similarity or complementarity of knowledge stock between the actors, positively affecting the cognitive facet of social capital.

Although this literature explicitly positions co-creation and collaboration as the next stage in knowledge sharing, it seldom discusses the kind of knowledge being shared (with a few exceptions, e.g., Kogut & Zander, 1992). The focus, thus, is exclusively on know-how as the most important knowledge asset in the firms and the move from relying on spillovers to alliances is viewed as a part of the quest for tacit knowledge. The terms knowledge and know-how are often treated as synonyms and often what is intended by knowledge spillover, exchange, or reuse is actually the spillover, exchange or reuse of know-how.

However, the actionable organizational knowledge has multiple facets, only one being know-how (Cross, Sproull, Constant, & Kiesler, 2004). The second line of argument for the plausibility of a causal relationship between knowledge reuse and social capital stems from the theoretical work on the facets of organizational knowledge in the context of technological systems. Garud (1997, p. 83) identifies three facets to knowledge in this context: (1) "understanding of the principles that underlie their functioning", (2) "process employed to create them", and (3) "the uses that these technological systems serve". He calls these three facets know-why, know-how, and know-what respectively. According to Garud, these three facets have different lifecycles and properties, and must be acquired and managed differently.

Of particular interest to creation of social capital is know-what. While know-how is the result of the process of "learning-by-doing" (Arrow, 1962) and thus an outcome of the internal organizational

functions, know-what is a result of the process of "learning-by-using" (von Hippel, 1988) and attained through user-creator interactions. Know-how is often generated through the production process and can be sourced from creators in a similar or complementary area of expertise. Know-what is generated at the interface between the creators and the reusers, and as such depends on the existence of such a relationship (Garud, 1997). The knowledge of what to reuse on the side of the reuser and the knowledge of what course of development to pursue on the side of the producer can be brought as examples of what has been labelled know-what by Garud. The concept partially overlaps with the notion of "sticky information" that has been cited as the source of advantage in user innovation (von Hippel, 1994, 1998). The "sticky information" are insights about possible courses of improvement that one attains only with repetitive use of a product.

The third line of argument for the possible effects of knowledge reuse on the emergence of social capital comes from the literature on transactive memory systems in social and organizational psychology. This body of work has developed the case for a fourth facet of organizational knowledge representing the knowledge of who is good at what, that is, know-who (Austin, 2003). Know-who has been studied in different streams of research under various names, such as transactive memory (Wegner et al., 1991), referrals (Cross et al., 2004), or simply information (Kogut & Zander, 1992). Know-who is often considered an outcome of prior collaboration, since it develops among the individuals as they gain experience working with each other. The research on expertise recognition has shown that the emergence of know-who facilitates collaboration and improves the outcomes of collaboration (Libby, Trotman, & Zimmer, 1987; Littlepage & Silbiger, 1992).

But reusing the product of knowledge work of the others can also help developing certain aspects of know-who, even when direct collaboration is not in order, (1) by establishing a common perception of the expertise of the different actors and establishing roles, as well as (2) by promoting problem solving approaches that share assumptions and are consonant across the collective (Cross et al., 2004). As such, knowledge revealing and the subsequent reuse do promote dynamics that are favourable to emergence of the different facets of social capital, whether in form of social connectedness and collaboration (relational), social closure and groupiness (structural) or creation of shared

understanding (cognitive).

In short, I argued in this section that social networks interact with the knowledge networks, and that the different facets of the social capital embedded in social networks both affect and are affected by knowledge reuse. Social capital sets the opportunities, the incentives and the constraints for the actors, and is instrumental in revealing, exchange and reuse of knowledge. On the other hand, knowledge revealing and reuse can enhance social capital, including in forms that pave the way for collaboration. This is either through offering a "preview" of the explanatory and procedural knowledge (know-why and know-how) that can be attained through collaboration, or by enhancing the indexical knowledge about the available knowledge components, possible solution (know-what) and their creators (know-who).

It is this interactive view of the conditions underlying the two forms of capital, social and intellectual, that constitutes the gist of the current thesis. The context of the study, open innovation communities, provides facilities for data gathering and test of such bi-directional relationships. However, in order to achieve such a result, one needs to find an appropriate method to track and trace knowledge reuse in this context. The next section explores and structures this issue.

Code Reuse as Knowledge Reuse

One challenge of studying knowledge networks in open innovation communities is measuring knowledge exchange, as there are no pre-established measures of knowledge exchange adapted to the specificities of the context. Developing and justifying such a measure is among the objectives of this dissertation. One possible proxy for tracking knowledge exchange in open innovation communities is the commonalities between knowledge outcomes. In the case of open source communities, this can be realized by measuring code reuse. The different types of code reuse can serve as measures for different aspects of knowledge reuse. That said, the transition from code reuse to knowledge reuse warrants some theoretical justification.

Code has both functional and expressive qualities (Lessig, 2001; Mackenzie, 2006). Software code consists of instructions that are read, interpreted and executed by machines, what confers

functional quality on software. But code is also text. Programmers can read and understand code, and do often use code as a means for the exchange of ideas. Therefore, code fulfils an informative and expressive role in parallel to its functional role. The expressive quality of code becomes all the more apparent as we take note of programmers boasting with zeal about the art of coding (Ford, 2015; Mackenzie, 2006; Raymond, 2001) and courts ruling that code is protected as free speech ("Bernstein v. US Dept. of Justice," 1999; "Junger v. Daley," 2000).

Harbouring both functional and expressive qualities, software may act as an open book for some, while appearing as a sharp pencil to the others. Software's dual nature means that its functional or expressive qualities are brought to the fore by the social role it is situated in (Star & Griesemer, 1989). One may run a software utility with or without understanding how it works, and one may understand software code with or without running it. So the answer to the question "is software like a can-opener or a recipe?" (Kaplan, 1998), is often "it depends". The exception is when software is distributed in machine-readable binary form with the objective of concealing its internal logic, in which case it takes a pure functional form.

Software code is a knowledge artefact that captures and preserves our knowledge about a topic, what would be otherwise transient. Programming is the process of structuring models of the physical world, people, information or processes, and embedding them within a software system (Sutcliffe & Sutcliffe, 2002, pp. 17–24). Once captured and embedded in a software, knowledge becomes more easily and cheaply reusable. Reusing a piece of code or a software component is, therefore, a case of knowledge reuse.

The transition from software reuse to knowledge reuse is a contentious one however, as it constitutes a departure from our day-to-day experience as software users. To accept software reuse as a form of knowledge reuse is to give primacy to the expressive nature of software code, while for most of us software is simply a tool that performs. It is important, however, to emphasize that the role software source code plays within a community of software developers is fundamentally different from the role of software as an opaque and mundane tool. Our everyday perception of software tools must not prevent us from studying software as a form of knowledge representation

when the source code is provided and given primacy in action.

Of course, it would not be appropriate to claim that each instance of code reuse implies the flow of the entire knowledge embedded within the source code from the original developer to the reuser. Yet, tracking down code reuse allows us to observe the overall patterns of knowledge flow (Rosenkopf & Almeida, 2003). All reuse is bound to start with acquisition of knowledge (Sutcliffe & Sutcliffe, 2002, pp. 17–24) about reuse and reuse methods, and often continues with deeper investigation into the inner workings of the reused code. It is in this perspective, and with complete awareness of the limitations of such a postulate, that I suggest code reuse as a measure of knowledge reuse and recombination among software projects. The extent and the type of code reuse can signal the breadth and the depth of knowledge reuse. The empirical evidence testifies that code reuse occurs at least as extensively in open source software projects as it does in proprietary software projects (Haefliger et al., 2008).

Types of Code Reuse. Code reuse consists of either grafting or calling upon code from a pre-existing software, and it can be characterized along several dimensions. The literature on software development methodologies is replete with reuse metrics and models based on the different facets of code reuse. In their hunt for an effective organizational software reuse model, Frakes and Terry (1996) identified no less than 50 types of code reuse cited in the literature. Here I focus on a dimension of code reuse that is ostensibly most closely tied to the costs of code reuse.

Whitebox Code Reuse. In its simplest form, code reuse is copying and pasting code, either snippets or entire files. This is the traditional approach to code reuse, and is referred to as whitebox code reuse to denote that the text of the code is reused with no abstraction. Whitebox reuse is often practised when the source code has not been designed with reusability in mind, it does not provide the appropriate reuse interfaces or it does not strictly conform to the current needs, and thus it has to be modified and adapted to fit the purpose.

Modifying code, though, requires a high degree of familiarity with the code base, and as such it can only be convenient and cost-effective when the code is already known to the developers or when the developers have access to resources that allow them to navigate quickly through the code.

In absence of good documentation or access to the original creators, the task of integrating existing code into own code can rapidly become daunting.

Despite the potential difficulties in the way of whitebox reuse, its flexibility and relative simplicity makes it a widespread practice in open source communities. Mockus (2007) estimates the percentage of identical source files appearing in more than one open source project at above 50% of the entire code base of open source communities. Nonetheless, whitebox reuse is nowhere as pervasive as blackbox reuse in open source communities. The bulk of code reuse occurs in the form of object and function calls to software modules pre-packaged for reuse (Heinemann, Deissenboeck, Gleirscher, Hummel, & Irlbeck, 2011).

Blackbox Code Reuse. The gold standard of code reuse is modularity. "A complex system is said to exhibit modularity in design if its parts can be designed independently but will work together to support the whole" (Baldwin & Clark, 2006, p. 1117). Modularity is considered one of the triumphs of open source software design, and it has been found to incite developers to join and stay active in open source projects (Baldwin & Clark, 2006; MacCormack, Rusnak, & Baldwin, 2006). Modular architectures facilitate the management of interdependencies across functions by breaking down software into self-contained functional entities or modules. These modules are then said to be reusable in a blackbox manner. When the software is blackboxed, little attention needs to be given to its internal complexities as the time of reuse and thus one can focus on the input and output streams to and from the software instead (Latour, 1999, p. 304).

Blackbox reuse is achieved by creating cross-code references in the form of function calls to the interfaces and instantiating objects provided by the focal module for the specific purpose of reuse. Knowing about the inner workings of the reused modules may be useful for troubleshooting when resorting to blackbox reuse but is not necessary, as long as the reuse interfaces are well-documented and the focal modules produce consistent and predictable responses (Prieto-Diaz, 1993).

Blackbox reuse presents a number of advantages over whitebox reuse for the reuser. Primarily, it reduces the burden of code adaptation from the shoulders of the reuser. Moreover, the reusers won't have to deal with several modified versions of a module if they have the chance of reusing a

single standardized version. At a larger level, the widespread availability of self-contained modules ready for blackbox reuse streamlines and encourages the practice of code reuse as it reduces the barriers to reuse (Ravichandran & Rothenberger, 2003). This makes blackbox reuse the preferred method of reuse among the developers, and more recurrent compared to whitebox reuse.

There are disadvantages to blackbox reuse as well. Given that blackbox reuse relies on prefabricated interfaces, it is less versatile in nature than whitebox code reuse. Also, the design of the reused code, including the robustness and the flexibility of its interfaces, becomes a critical factor to the success of the blackbox reuse. This puts a disproportionally large burden on the shoulders of those who intend to share software solutions in the form of modules. Reusability under a blackbox reuse paradigm is proportional to the costs incurred by the originator for the specific purpose of making the modules more prone to reuse (Prieto-Diaz, 1993). This is a non-negligible category of costs and in some cases it may amount to twice as much as the development costs (Tracz, 1995).

Although blackbox and whitebox reuse are operationally distinguishable from each other, the two concepts are not mutually exclusive and there is a continuum of reuse practices falling between the two. Particularly in open source communities, the leap from one to type of reuse to the other is particularly narrow. Blackboxing in open source communities never amounts to total opacity, given that the source code is publicly available. On the other hans, if a module's public interface does not satisfy the requirements, there are always means to contribute to the focal project in order to bake into it a more inclusive interface, or otherwise to branch out and specialize the focal module for own use.

Blackbox and whitebox code reuse are interesting indicators of knowledge reuse, as they both demonstrate the ways in which a software project may draw on the know-how developed in a focal project by building functional dependency over the focal project's source code. Whitebox reuse illustrates a case closer to rich knowledge transfer, as it implies a certain degree of mastery of the code being reused, while blackbox reuse describes a mix-and-match behaviour based on the public information and public interfaces of the project, what necessitates thorough search. Therefore, I would argue that the richness of whitebox reuse necessitates more bandwidth than that needed for

blackbox reuse.

Reuse in Open Source Communities. Two previous studies have particularly focused on the question of reuse in open source communities, both finding a direct link between the limitation in the supply of human capital in the communities and the reuse practices. Haefliger et al. (2008) combine a limited-sample code analysis with contributor interviews to study the pull-side forces behind code reuse. Sojer and Henkel (2010) is a more recent survey-based study of SourceForge project contributors and their self-reported reuse practices. The findings of the two papers widely converge despite their orthogonal approaches.

First, the contributors were found to be patently aware of the limited supply of workforce and therefore employed strategies to tackle the resource limitations. For instance, they saw the reuse of the existing solutions as a time and effort-saving measure that allowed them to maximize the time spent on the core problems they intended to tackle. The individuals, in other words, were faced with the equivalent of a make or buy choice. The reuse decisions occurred starting from day one of the projects. The long-time contributors, those active in several projects and those who reported a larger social networks tended to reuse more than their less experienced, less active and less socialized counterparts. The authors attributed this tendency to possibly lower local search costs when looking for reusable artefacts, although none of the two studies had the luxury of actually using network data to study the origin of the artefacts or the social networks of the individuals.

Second, the contributors showed a tendency to overly focus on what they perceived as core issues — those particularly challenging and technically interesting tasks that allowed them to signal their abilities. That, combined with the scarcity of supplementary aids that could lift the burden of the grunt work, brought about a *tragedy of the boring* whereby mundane and uninteresting tasks remained unattended. The authors noted that, given the high cost of making code reusable that can attain twice the cost of initial development, and the tedious nature of documenting and interface development tasks, a crisis of reusability in open source would be unavoidable.

Finally, the authors found that the ad-hoc mode of contribution and the ebb-and-flow of resources led to a high variability in the code quality across projects and in the absence of objective

quality assessments, the developers had to rely on the limited existing information to judge the usefulness of a piece of code. As a consequence, personal trust and reputation became prevalent factors in the decision to reuse.

The recent debates in software development around technical debt, code smells, and code refactorability, widely echoed across the open source communities, confirm the above findings and demonstrate the ongoing efforts of the developers to pre-empt the impending crisis of reusability and to devise objective code quality assessment measures across various dimensions (Allman, 2012; Fowler & Beck, 1999; Tufano et al., 2015).

On a related note, P. V. Singh and Phelps (2013)'s study of software license choices among SourceForge project initiators, although not directly on the topic of knowledge reuse, is deeply inspired by the literature on knowledge networks and, for the first time, provides empirical evidence about the localization of knowledge in open source communities. The authors reveal that project initiators opt for licenses they have learned about through their prior social relations and collaborations. In short, the knowledge about licenses propagates from project to project through developer mobility.

Theory Development

The Effect of Social Ties on Knowledge Reuse

Two groups that have members in common can be deemed close to each other in a social network. The notion of social proximity can be simply defined as the opposite to the social network distance that separates two nodes, although the strength, the frequency and the multiplicity of the ties are also indicators of how close two actors are in a network.

Social proximity may be viewed in a static way, incorporating only the currently existing social ties, or otherwise in a dynamic way, by aggregating ties over time and including actor mobility as well as tie creation and dissolution as hints of social proximity. In the dynamic perspective, the evolving affiliations of individuals with different groups over time can be considered an indicator of the social proximity of those groups. In open innovation communities, where the only meaningful unit of analysis in between the individual and the community is the project team, we may see the mobility of individual members between projects, i.e., the project coaffiliations of the members, as a measure of social proximity between projects.

The studies of knowledge networks have drawn a direct link between network proximity and knowledge outcomes. The facilitating role of network proximity in knowledge transfer has been noted at the individual level as well as the collective level — that is, within teams, organizational units and firms (e.g., Hansen, 1999; Rosenkopf & Almeida, 2003).

The literature has pointed out several mechanisms to justify this facilitating role. Social proximity is essential in mitigating knowledge reuse costs and uncertainties, including knowledge search and transfer costs (Hansen, 1999; Rosenkopf & Nerkar, 2001; Rosenkopf & Padula, 2008). From a strict information-processing perspective, the social proximity between the actors determines the ease of information transfer between them and defines the chances of exposure to knowledge (Coleman et al., 1957).

When there are barriers driving up the costs associated with access to knowledge and transfer of knowledge, close relationships can help mitigating those costs by allowing direct access to the actors at the source or close to the source of knowledge. This becomes particularly important when

knowledge exposure is incomplete or access to knowledge is restricted, e.g., through secrecy or legal means. Streamlined knowledge transfer will in turn increase the chances of knowledge reuse.

Although the explicit monetary cost of knowledge acquisition is zero in the context of open innovation communities, I argue there are implicit costs to knowledge reuse that persist in this context and can be mitigated through the advantages brought by social proximity. For instance, the cost of finding the right solution, learning about it, and adapting it remains a valid concern despite the zero price tag of the open solutions. Therefore, even where the dominant intellectual property regime is rather permissive, social proximity must retain its facilitating role for knowledge transfer, albeit for alternative reasons.

But, more importantly, network proximity between two social groups gives members access to social capital across their groups' boundaries, what explains much of the cost-mitigating qualities of social proximity. Since the social network of the open innovation communities hinges on the participation of the individual across various projects, social proximity between project teams can be also seen as having a shared history of direct collaboration.

Social proximity and propinquity bring about mutual expectations and obligations, promoting direct reciprocity (Bourdieu, 1986a; Feldman & Newcomb, 1969; Newcomb, 1961). This is all the more true in the case of professional collaboration, where personal commitments often go beyond the contractual obligations, self interest, and professional tact (Fairtlough, 1994). This is also of particular importance in open innovation communities where the individuals do not have much of a direct pecuniary incentive to make their contributions reusable to the others (Haefliger et al., 2008). As such, they may not always be willing to go the extra mile and incur the related costs, unless compelled by their personal obligations. Having collaborators in common, thus, can cultivate favourable conditions for reuse between projects.

Moreover, given the previous findings regarding the lack of transparent quality indicators in open source communities and the role of trust as a surrogate for quality assessment in reuse decisions (Haefliger et al., 2008; Lerner & Tirole, 2002), one may reasonably expect members to rely on inter-personal trust as a form of quality assurance to direct reuse choices. There is a direct relationship

between trust and collaboration. Collaboration begets trust, and trust, in turn, breeds exchange (Putnam, 2001). The collaboration of members with different projects can be seen as establishing trust across project boundaries, paving the way for reuse.

Apart from the relational facet of social capital, the cognitive facet of social capital can also affect reuse. Proximity and collaborative work have both been associated with indirect and vicarious learning that can in turn help achieving a shared language and common codes (Bresman, 2010). Having a common language or sharing communication codes constitutes a vehicle for knowledge exchange and provides a common conceptual framework to evaluate the benefits of exchange and reuse, and thus makes reuse a more likely case.

In summary, for reasons related to relational and cognitive social capital and their effect on the implicit costs of reuse, it is likely that social proximity between projects drives the reuse decisions of their developers in open source communities. This must hold true in open innovation communities, although the the motivating mechanisms may not be identical with those under proprietary intellectual property regimes.

Hypothesis 1a *Projects with coaffiliated collaborators are more likely to establish new blackbox reuse ties than those without direct social ties.*

Hypothesis 1b *Projects with coaffiliated collaborators are more likely to establish new whitebox reuse ties than those without direct social ties.*

The effect of coaffiliation ties on the two types of reuse is not equal, though. The arguments are multiple, and they have to do both with the nature of the coaffiliation ties as well as the nature of the two reuse ties.

Social network studies have shown that the strong social ties are particularly good at embedding durable relationships, promoting trust, upholding important obligations and providing high bandwidth for knowledge transfer. On the other hand, weak ties have been seen as representing another sort of social capital, promising information diversity and novelty of resources. The strength of a tie has been defined in a variety of ways. Some scholars have interpreted strength as a synonym

for directness of the tie, defining a weak tie as a mediated tie (e.g., friends of friends) (Boissevain, 1974), while others have seen tie strength as a function of mutuality, relationship intensity and the frequency of interaction (Granovetter, 1973; Uzzi, 1997).

But apart from the technical definition, what constitutes the main difference between the strong and the weak ties is in fact the amount of time and resources being invested in the tie, subsequently offering the advantages often attributed to the strong ties (Aral & Van Alstyne, 2011; Granovetter, 1973; Hansen, 1999). The homogeneity and the high level of vested interests among strongly tied entities makes strong ties better suited to transfer of complex and system-dependent knowledge, as they ensure both the willingness and the ability to go through an exchange process that is likely to require deliberate effort (Hansen, 1999). Strong ties are also useful in instilling a sense of social closure and maintaining trust (Burt, 2001). But due the intensity of interaction between strongly tied entities and the large proportion of resources they share with each other, the knowledge embedded in strongly tied entities tends to homogenize over time and become increasingly redundant (Burt, 2000; Granovetter, 1973).

On the contrary, the weak ties represent a low-involvement or remote relationship that can be had in scores and provide search advantages given the diversity of the resources and the low-redundancy information they bring within reach (Aral & Van Alstyne, 2011; Burt, 2000; Granovetter, 1985; Hansen, 1999). It is of note that the strong ties do offer search advantages as well, but often only at the local level, as their social reach remains limited (Rosenkopf & Almeida, 2003).

In the network of open source projects a coaffiliation between two projects is a collaboration tie, and it is recorded when at least one developer contributes to both projects. In order to contribute successfully to the projects, the contributor must have used both pieces of software and spent time understanding the inner functioning of both systems, which denotes significant time investment. Developers have to be selective in their allocation of resources, and are able to contribute only to a limited number of projects simultaneously. As a consequence each project can be coaffiliated to only a few other projects. In all likelihood project coaffiliations can be considered strong ties between projects.

A first argument for the differential effects of coaffiliation on whitebox versus blackbox reuse stems from the fact that coaffiliations being strong ties, they are likely better adapted to deep knowledge reuse as strong ties tend to reduce the high processing costs typical to deep knowledge reuse.

Whitebox reuse is a case of deep knowledge reuse, as unlike blackbox reuse it requires awareness of the content of the reused code, along with a certain degree of mastery of the code and its technical environment. The intellectual assets needed for whitebox reuse may at times be more costly to acquire than rewriting the code from the scratch. Therefore, developers that have prior experience dealing with a body of code have great advantage over those with limited exposure when it comes to transplanting parts of the code into other projects. By proxy, projects that employ developers with prior experience or current involvement in other projects will find it less onerous to carry over some code from those projects. The same can not be said about blackbox reuse which relies on pubic interfaces and publicly available metadata about project APIs.

A second line of argument for the differential effect of coaffiliation ties on the two types of reuse comes from the search capacities offered by the coaffiliations.

Whitebox reuse is a way of internalizing the functionalities coming from other code bases. This is a more complex and less streamlined task as compared to blackbox reuse. Since the objective in whitebox reuse is to include the logics of the needed functionalities within the source code of the project, the developers are likely to opt for whitebox reuse only when the needed functionalities draw on domains of expertise close to theirs. Under such circumstances the local search capacities of coaffiliation ties can be particularly helpful in finding the right content (Rosenkopf & Almeida, 2003). This code content can not be easily advertised nor can be efficiently searched, yet the contributing developers deal with it on a daily basis and are aware of its intricacies as well as its purpose. Coaffiliation ties, thus, constitute a golden opportunity for the open source project teams to discover eventual alignments and complementarities with code content of the other projects.

Blackbox reuse, on the other hand, is a way of externalizing the functions that are needed in the software but that the developers do not want to directly include in their code. This means the developers are able to draw on code coming from more remote areas of expertise when practising

blackbox reuse. Therefore blackbox reuse hinges on remote search capacities that may not be as efficiently accommodated by the coaffiliation ties. Moreover, the public interfaces of the projects, what blackbox reuse builds on, can be easily documented, publicised, and thus searched using regular search engines. That reduces the dependence of the knowledge discovery process on diffusion through social networks.

Finally, whitebox reuse involves modifying one's own code and adapting it to the newly grafted code, over and above the adaptation needed when resorting to blackbox reuse. This interdependency necessitates a certain degree of trust in the quality of the grafted code (Haefliger et al., 2008). Such a trust can be gained either by controlling the quality of the actual work, or via relational or structural trust in its creators (Das & Teng, 1998). Once again, the contributors who have experience with a project are best placed to judge the worth of the code, either based on their first-hand experience with the code or relying on the social capital they have developed while working on the project.

Given the three arguments above, I propose the following hypothesis:

Hypothesis 2a *Project coaffiliation is more strongly related to creation of new reuse ties in form of whitebox reuse rather than blackbox reuse.*

The literature on the social networks has seen tie strength as a linear variable, whereby the qualities enabled by tie strength will yield more as the strength of the ties increases (Marsden & Campbell, 1984). Applying the same view to the differential effect of project coaffiliation on the two types of reuse results in the understanding that the stronger the coaffiliation tie between two projects, the higher the likelihood of emergence of whitebox reuse ties.

Yet, for reasons related to the nature of the two types of reuse I argue that the differential effect of coaffiliation on the two types of reuse should wane or tilt in favour of blackbox reuse as the strength of project coaffiliation, characterized by regular and balanced contributions of one or more developers to two projects, increases.

As it was mentioned before, the gold standard of code reuse is blackbox reuse, as it prevents redundancy and minimizes rework. Yet, in many cases whitebox reuse is the appropriate choice, given that the public interface of the reused code may not always be adapted to the use case and

the cost of development and coordination for transforming it may be too high. Whitebox reuse is also useful when the developers need to avoid external dependencies in order to prevent future uncertainties, even at the cost of generating some redundant work.

When two projects share a significant amount of their human resources, though, it is unlikely that their interfaces develops far apart from each other, not is it likely that there is an aversion to develop dependencies across project boundaries. The stronger the coaffiliation between two projects, the better their resource allocation and objectives will be synced, as little to no coordination will be needed to arrange for adaptation. With little to modify in order to attain compatibility and little to worry in case of dependency it is hard to justify the cost of rework and redundancy that whitebox reuse entails. Two very strongly coaffiliated projects, thus, must be more likely to opt for blackbox reuse rather than whitebox reuse.

Hypothesis 2b *The stronger the coaffiliation between projects, the more it is likely that the reuse ties between them occur in form of blackbox reuse rather than whitebox reuse.*

Software Licenses as A Barrier to Reuse

Open and copyleft licenses are means used effectively by the open communities to guard their commons while maintaining the result of their work open for modification and reuse (Lessig, 2006; O'Mahony, 2003). Open licenses are often much more permissive than their proprietary counterparts, but they nonetheless restrict certain aspects of modification and reuse to protect *the commons* (Lessig, 2001). The famous viral clause in GNU General Public License (GPL) that requires any derivative work of a GPL-Licensed code to carry the same license is one of those protection measures ("GPL v.3," 2007).

There are important differences in the terms of open licenses, and some are clearly not cross-compatible. This has to do with the differences in the ideals and the objectives behind each license that seek to promote a different notion of the commons which shows up in the rules of access to the intellectual capital (Lessig, 2001). While the permissive vs. share-alike dichotomy and the resulting restrictions on derivative work makes up for the bulk of the incompatibilities, other issues such as

the modalities of attribution, distribution and use can also degrade compatibility between licenses or license groups. It is a less discussed reality of the open source communities that rarely an entire community adheres to the same license. This can potentially create license fault lines within the same community affecting whether and how reuse can be achieved. Previous studies of digital content reuse have shown the mitigating effect of copyright restrictions on content reuse practices (Nagaraj, 2017).

It is interesting to trace the effect of possible legal barriers to the reuse practices within the open source communities. But more importantly, it is vital to clarify to what extent the knowledge localization effects are due to reuse restrictions imposed by licenses.

I posit that license dissimilarities have an effect on knowledge reuse across the community, as specific clauses in certain licenses may be incompatible with other licenses and may thus prevent certain projects from building on the work of the projects that are released under dissimilar licenses. This holds true for both types of reuse, but the effect on whitebox reuse will likely be more pronounces because the main source of license incompatibility, the share-alike clauses, mainly affect modification and the creation of derivative work rather than use or distribution.

Hypothesis 3a *Projects with similar licenses are more likely to draw on each other's knowledge through blackbox code reuse.*

Hypothesis 3b *Projects with similar licenses are more likely to draw on each other's knowledge through whitebox code reuse.*

The Effect of Knowledge Reuse on Social Ties

Participation in open source project has been described as a process similar to what Lave and Wenger (1991) have labelled legitimate peripheral participation. The new entrants always start at the periphery of the projects, as lurkers or simple users, and they advance as they gain a better understanding of the projects, demonstrate their proficiency with their contributions and gain legitimacy among the project members (Dahlander & O'Mahony, 2011). Although this literature does not discuss a causality link between use and participation, it definitely showcases an ordered correlation

between the two notions under the community settings. There is no reason to think that code reuse is dispensed from this rule. On the contrary, there are various arguments in support of a link between code reuse and subsequent project coaffiliations, which is, mutual developer participation.

First, over time code reuse will lead to the emergence of a common stock of know-how shared by the reusing and the reused projects. The members of a project that has been drawing on another project's work gradually grow their capacity to absorb information about the inner workings and the internal logic of the reused project. Legitimacy accrues to them as they file bug reports, open discussions on features and development roadmap and seek help from the developers of the focal project on non-trivial highly technical issues. One can see this information exchange as a form of signalling the competencies and the convergent interests (Malmberg & Maskell, 2002; Phene & Tallman, 2014; Stuart, 1998; Stuart & Podolny, 1996; Yang et al., 2010). Given that they are themselves developers and well-initiated to the social organization of open projects, it is only natural that at some point the advanced reusers realize they are better off submitting their own patches to improve the project they reuse. It is not uncommon for developers to move upstream in this way to a project they used to be a user of.

Second, knowledge reuse allows the creation of a dictionary of who knows what (know-who), when the source is annotated with authorship information. Literature on teamwork has often emphasized prior collaboration experience as a precondition for the emergence of know-who in form of transactive memory (Lewis, 2004). Transactive memory allows the emergence of a self-managed division of labour that is vital for efficient teamwork (Hollingshead, 2000). But some studies have shown that the existence and the accessibility of know-who information can stimulate collaboration at a distance, even in absence of prior shared experiences (T. A. Finholt, Sproull, & Kiesler, 2002; Hollingshead, Fulk, & Monge, 2002). Software code usually contains authorship metadata as a means of contribution and blame attribution. This means the reusers get to know who has developed what and gradually build an expertise directory. Moreover, the exchanges between the developers and the reusers (e.g., troubleshooting) augment this directory with rich social cues that will become particularly useful in case of collaboration.

Finally, the knowledge reuse that ensues subsequent to knowledge spillovers in the corporate context and free revealing in the online communities have a major role in creation of know-what by advertising the existing solutions to the wider public and actively shaping their sphere of choice for solutions, as well as attracting feedback from the recipients (Garud, 1997). Know-what, generally "represents an appreciation of the kinds of phenomena worth pursuing" (p. 81 Garud, 1997), and is often generated through a process of learning by using (von Hippel, 1988). For instance, the role of user feedback in setting the course of open source projects has been recognized since the first publications on the topic (Raymond, 2001). The importance of user feedback is partially due to the fact that the use of technology often diverges from what is intended by the designer and as such the users develop a unique perspective on it that can be conveyed in the interactions with the creator (Orlikowski, 2002; von Hippel, 1994). Depending on the context and the degree of permeability of the organizational boundaries, such user-creator interactions at the interface can lead to creation of additional social ties in form of collaboration or other co-constructive engagements.

Thus, I suggest that knowledge reuse can stimulate creation of social ties in form of member coaffiliation between projects.

Hypothesis 4a *The existence of prior reuse ties between projects in form of blackbox reuse increases the likelihood of creation of new social ties between those projects through developer coaffiliation.*

Hypothesis 4b *The existence of prior reuse ties between projects in form of whitebox reuse increases the likelihood of creation of new social ties between those projects through developer coaffiliation.*

Apart from the factors mentioned above, the reusers also have a vested interest in what they reuse in a continuous manner, as the future of their work is tied to it. But this vested interest is of different grades in the two types of reuse.

One way of characterizing the difference between whitebox and blackbox reuse is to picture blackbox reuse as a live link between two software projects, as compared to whitebox reuse which denotes a static link between a project and a snapshot of an other project, frozen in time. In other terms, blackbox reuse is dependency-creating while whitebox reuse is dependency-dissolving. An

instance of blackbox reuse is tied to the evolutions of the reused and has to adapt to them, whether it is for the good or for the bad, but it will also benefit from the progress in the code-base of the reused project. On the other hand, although whitebox reuse generally signifies a deeper, more extensive integration between two code bases, it also symbolizes the cutting of the umbilical cord that ties the reusing to the reused. Once a body of code is grafted into a new project, the future developments of the project of origin matter to a much lesser extent.

Therefore, I suggest that project teams maintain a higher degree of interest in the development of those projects they reuse in a blackbox manner as compared to those they reuse in the whitebox manner. This sustained interest is likely to serve as an additional motive for future contributions of the reusing project's team to the reused project.

Hypothesis 5 *Blackbox reuse is likely to be more strongly related to creation of new social ties in form of developer coaffiliation, compared to whitebox reuse.*

Software Licenses as A Barrier to Collaboration

In their study of open source licence adoption, P. V. Singh and Phelps (2013) found that the choice of license in open source projects is socially influenced and thus path dependent. The most influential factor determining a project's license, the authors argued, is the license chosen by the other projects in its social proximity. Project initiators tended to adopt licenses that they already knew from prior collaborations. This path dependency in license choices can lead in long term to a relative separation between the spheres of collaboration around different licenses.

Moreover, open source contributors have been found ideologically motivated and their collaboration organized around collective beliefs and common norms (Stewart & Gosain, 2006). Given the ideological roots of open source licenses, and their role in keeping outsiders at bay (O'Mahony, 2003), one may expect to observe social stratification around licenses in open source communities. In other words, license dissimilarities won't only affect code reuse across projects, but also the patterns of collaboration and affiliation with projects: It is likely to observe more collaboration between projects that share a license as compared to those who don't.

Hypothesis 6 *Projects with similar licenses are more likely to develop coaffiliations by sharing and exchanging developers.*

Methods

General Approach

There is a wealth of open data available on online communities and the activities of their members. Open source communities are no exception in this regard. Individuals leave rich digital traces behind as they participate in community activities. In many communities both live and archival activity data are open to the public. The current study relies on openly available archives of digital trace data from online communities. The data is retrieved in raw format and subsequently reformatted or reduced to analytically digestible datasets. In order to accommodate a whole-network study, the entirety of available data about the population of interest is gathered in its entirety at the data gathering stage. Sampling is thus relayed to the data manipulation stage and is conducted with regard to the requirements of the statistical methods of choice.

This study also relies in part on two categories of secondary data generated from the raw digital traces data: (1) Graph data obtained by cross-linking and triangulating raw data from multiple primary sources, and (2) simplified or classified data obtained by applying dimensionality reduction methods such as similarity hashing and topic modelling to vast amounts of unstructured text coming from primary sources.

This study uses quantitative methods, more specifically inferential statistics combined with network measures, for data analysis. Analysing large datasets begets new issues not typical when dealing with conventional datasets, and often requires adapted tools and techniques. The current study aims to be innovative in its application of research methods, yet does not come with any pretence of methodological inventions. When confronted with limitations in the mainstream econometric methods I use statistical corrections, power analysis, sensitivity analysis or a combination of those to tackle the issues engendered by the sheer size of the data or other specificities.

Research Site

Defining the boundaries of a social network is a non-obvious choice faced by the students of social network analysis, particularly when designing whole-network studies. Given that in real

world social actors and structures are rarely found in a state of disconnect, identifying a meaningful collection of social links that can be isolated and analysed as a stand-alone network necessitates a decision criteria (Marsden, 2009).

I used an event-based criteria and focused on project participation to delineate the social network of programmers that actively contribute to the development of the open source libraries of Ruby programming language. These interdependent software projects, along with the Ruby language itself, constitute the intellectual commons that hold together what is commonly referred to as the Ruby community. There are precedents in the literature for considering programming in a common language and submitting code to the same foundry reason enough to recognize the existence of a community. One can assume that the these project contributors are more likely to collaborate, communicate, establish social ties, draw on each other's work and reuse each other's code (Grewal et al., 2006; P. V. Singh & Phelps, 2013; P. V. Singh et al., 2011).

The community of Ruby language programmers is a naturally bound collective with boundaries that designate the limits of both social and knowledge exchanges, and as such is a good candidate as the target population for a whole-network social network analysis.* Ruby community strikes a fine balance between internal homogeneity and external heterogeneity. Reliance on the knowledge of the Ruby programming language alone already acts as a barrier defining the scope of both collaborative and knowledge sharing activities.

Ruby is an interpreted programming language similar to Python and is the 10th most popular programming languages on the Internet (TIOBE Software, 2016). But unlike Python whose use cases span a variety of fields from scientific research to special effects, Ruby's developer base is rather homogeneous and narrowly focused on agile web application development and deployment. Given the relative homogeneity of use cases and users, it is less likely to find social fault-lines and large isolated components within Ruby community, allowing to assume Ruby community boundaries as valid markers for defining the outer limits of a social exchange network.

The Ruby language has exceptionally strong ties to the open source movement. GitHub, the

*For a glossary of open source and Ruby-related terms see Appendix A

most popular open source code repository, has been programmed in Ruby language and is an important driving force behind Ruby community and the development of Ruby language itself. Ruby community is an open-source pure-player. Virtually all available reusable Ruby libraries (i.e., *Gems* in Ruby lingo) either use an open source license or are unlicensed, that is, the large majority of the knowledge artefacts the community produces and draws on are accessible through community outlets.

Ruby community is among the more centralized open source communities. Apart from a handful of exceptions, the entirety of 60,000+ reusable Ruby gems are centrally hosted in separate bundles on a single software repository called RubyGems. Gems uploaded to RubyGems are directly accessible for installation through Ruby's internal package manager, also called RubyGems. RubyGems package manager and website endow the community with a mechanism to ensure that Ruby software packages are exposed to and accessible for community members and Ruby developers in general.

Data Gathering

I used RubyGems' gem database as the starting point of my data gathering. For practical purposes I limit my conception of ruby community to *all the individuals that have contributed to Ruby commons by way of contributing to the gems hosted on RubyGems*. RubyGems is a software package repository where the developers upload their work only if they deem it useful for the community as a stand-alone package and in its current form. This makes RubyGems different from source code repositories (e.g., SourceForge, GitHub, etc.) where the development versions of projects are hosted. By analogy RubyGems can be considered the AppStore of Ruby community.

Earlier studies of open source communities have noted difficulties in establishing cut-off thresholds for data gathering due to the high degree of skewness in member participation and project activity within and across communities (Dahlander & O'Mahony, 2011; Hars & Ou, 2002; Mockus et al., 2002). Old questions such as the role of lurkers in computer-mediated interactions and whether they can be counted in as participants (Eveland & Bikson, 1987; T. Finholt & Sproull, 1990; Pickering & King, 1995) are still subject of debates and a source of inconsistency among social studies of online

communities.

Using contributions to the community's software repository as the criterion for inclusion in the community solves several practical issues in the data gathering process by filtering out much of the contentious data with no need to tackle the above questions. The RubyGems repository only provides data on reusable projects and their developers, and therefore it effectively eliminates wide swaths of inactive and non-functional developers, lurkers and early stage learners, underdeveloped and unripe projects, abandoned project forks, unmodified duplicates and one-time trials and tests. This data gathering strategy provides a non-arbitrary and effective way of focusing data on those participants and projects who have made at least one notable contribution to the commons of the community. I contend that the strategy taken here is superior to the habitual method of starting from code repositories and discussion forums, and then devising ways to eliminate the irrelevant data points from the dataset.

I fetched the data regarding 361,482 distinct versions of 61,921 Ruby gems submitted or migrated to the RubyGems repository over the ten-year lifespan of the community (pre-2003 to 2013). I continued gathering data on project activity until August 2013, but stopped taking in the new projects at the beginning to 2013. I also gathered data about the 50,303 identifiable developers involved in the projects as well as their contributions over time (about 2.9 millions). This data came from three different sources:

RubyGems. Each release of a gem (i.e., a *package* in Ruby lingo) has a profile page on RubyGems website that links to the downloadable bundle of that specific release (See Figure 1). Each downloadable package contains a specification file with metadata about the gem, its technical dependencies and its developers. The package bundles downloaded from RubyGems also include the source code for that specific version of the software, and some times a license file.

The specification file is compiled by the developers of the package using a semi-automatic package deployment tool and it provides two categories of information: (1) The spec file includes some general data about the package including the title, the version, the release date, and the license. It also often includes a link back to the source code repository of the project on GitHub and a list

Figure 1. A RubyGems Profile Page

of contributors involved in that version of the gem. These data items are self-reposted and thus not always reliable. (2) The specs also describe the runtime and development time requirements of the package, including a list of all other packages it directly depends on in order to function properly. This data is highly reliable since Ruby's own package manager uses it during the installation process to ensure that the technical requirements of the package are met.

Scraping RubyGems constituted the first round of data gathering for this project.

GitHub. GitHub is a web-based code repository hosting platform build on Git version control system, and as such provides a complete log of the development of each project, including the chronology of contributions. Under Git the unit of contribution is *commit*. Each commit is supposed to make a meaningful and self-contained modification to the code, and the developers are discouraged from committing unfinished changes or breaking down one unit of change across several commits. The second round of data gathering consisted of following the links back to the code repositories of the projects on GitHub in order to obtain detailed information about the development activities of the projects.

The close ties between Ruby language and GitHub shows itself also in the Ruby community's choice of code repository. Apart from a handful of projects, almost all the other projects that declare their source code repository are hosted and developed on GitHub. Yet, not all project specifications identify a code repository, and about half of gems could be linked back to their source code. Manual searches and error correction marginally improved the access rate, and ultimately 38,799 gems could be linked back to 38,339 unique source code repositories. In cases where several gems were hosted on the same repository, they were considered parts of the same project and their data were merged.

I obtained the Git versioning database of each project, including the source code, along with project, contributor and commit metadata from GitHub. GitHub also tracks project forks and can indicate whether a project is a derivative of another project. I also gathered these data points while scraping the GitHub project data.

Git. Finally, I scraped the full commit authoring logs from the Git database of each project. While the Git database provides its own version of the contributor metadata, the uniqueness of

contributor profiles across projects can be assured only when the contributor metadata from Git can be linked to those from GitHub, where the users need a universal GitHub account to submit their code. Out of 2,907,546 commits in the database, 2,439,059 of them could be linked to GitHub's unique author IDs and therefore be used to construct a project affiliation dataset.

Table 2 summarizes the data items gathered through the three data gathering stages of this project, along with the corresponding data sources.

Data Source	Data Item
RubyGems	Functional Dependencies
	License Information
	Package Authorship Information
	Package Release Dates
	Complete Package Source Codes
	Link to Source Code Repository (GitHub)
GitHub	Project Ownership Information
	Fork and Branch Information
	GitHub User Information for Authors and Committers
	Full Source Code Evolution Log in Form of Git Repository
Git	The Content of Each Commit (Code and Message)
	Modifications Introduced in Each Commit
	Author Information
	Committer Information
	Commit and Authoring Dates

Table 2. Data items and their sources

Data Transformation

The raw data from online digital traces is not necessarily structured in an analysis-friendly format upon gathering, and often considerable transformation is required to bring the data to a stage where it can be fed into conventional data analysis processes. Transformation, juxtaposition, integration and triangulation of the data from these three sources allowed creating the three graph datasets needed to test the hypotheses. In the following paragraphs I give a brief description of

these datasets, and the steps taken to produce them. For an overview of the data gathering and transformation process see Figure 2.

The network of blackbox reuse between projects (functional reuse network). I used the functional dependency data from package specifications to construct the technical network of functional reuse for the Ruby community. Functional dependence occurs when a project calls a function developed in another project, thus requiring the other package for proper functioning. Functional dependences change as a project's codebase and functionalities evolve. I considered direct functional dependence of two packages, or the existence of function calls between packages, as a code reuse tie between the packages. This yielded a dynamic and directed network of functional reuse ties between projects (See Figure 3).

The network of whitebox reuse between projects (code clone network). I constructed a second network of reuse based on a different measure and a different conceptualization of the notion of reuse. Earlier in this essay I made a theoretical distinction between blackbox and whitebox reuse. This second network of reuse is intended to capture that theoretical difference. Apart from theoretical justification, though, developing an alternative measure of reuse serves a methodological double-purpose as well. Quantifying the notions of knowledge reuse and transfer is a contentious task, and any potential indicator can be argued to be a proxy variable at best. The best empirical solution to ascertain that the observed empirical patterns are rooted in the measured construct and not the measurement, is to rely on more than one measurement. Given the central place of the notion of reuse in this study, it seemed natural to develop a parallel reuse measure as a way of ascertaining the construct validity of the proposed models.

This second reuse measure quantifies the amount of code cloning or duplication between the code bases of different projects, in other words keeping track of the copy-and-pastes of the developers. The measure is calculated using similarity hashing and matching methods adopted from computer science literature. Like cryptographic hashing functions, the similarity hashing algorithms are dimensionality reduction techniques — they increase the comparability of complex data by representing it

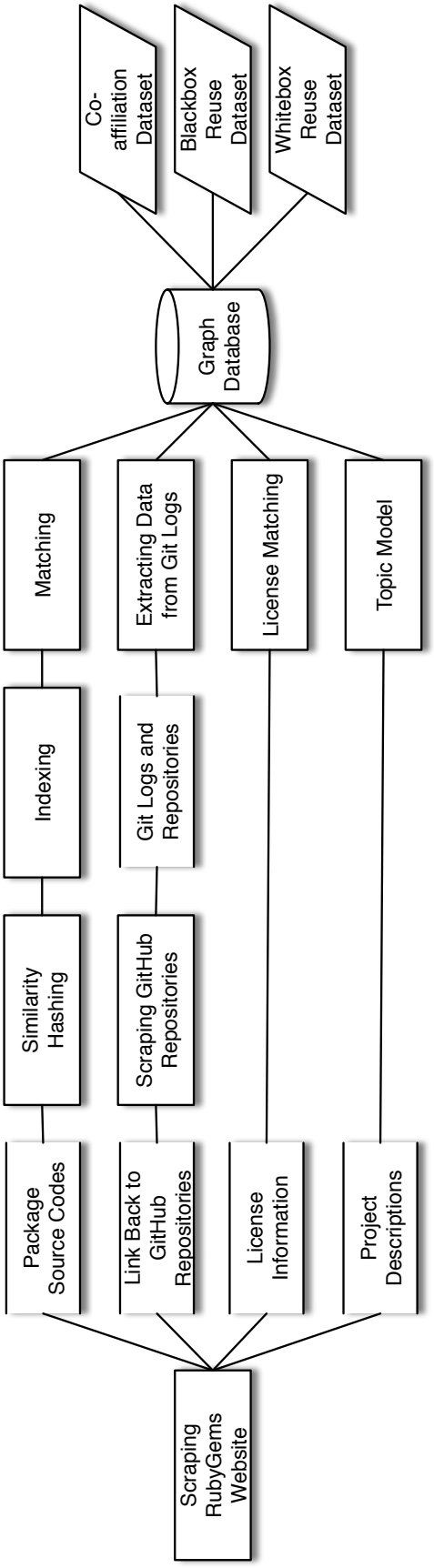


Figure 2. Data gathering and transformation flowchart

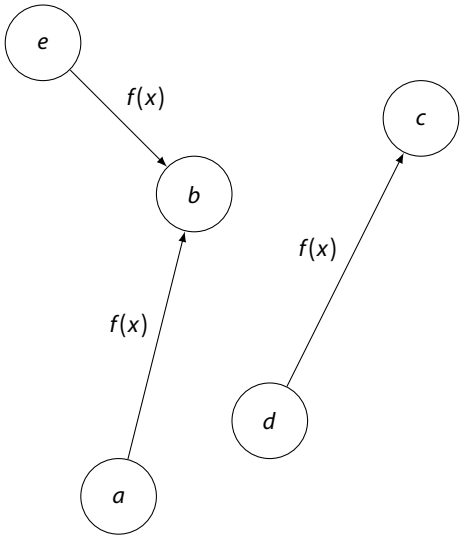


Figure 3. Network of blackbox reuse (functional reuse)

in less dimensional spaces. But while cryptographic hashes are sensitive to the slightest of changes in the data they represent, and can only be used for detecting exact matches, similarity hashes (or simhashes) allow detection of partial matches between chunks of data as they retain the information order and sequence within the structure of the hash. Simhashes are widely used in the domains where the amount of data generated necessitates mechanized ways to perform comparisons. Some major applications of simhashing are in computer forensics, copyright enforcement and anti-plagiarism software, spam detection, and image recognition (Gayoso Martínez, Hernández Álvarez, & Hernández Encinas, 2014). Software source code is a type of text, therefore one can make use of simhashing methods to detect partial cloning and similarities between source codes.

Since none of the off-the-shelf programs for clone detection could provide the accuracy and the speed required for the purpose of this study, I developed my own clone detection software. The program hinges on two well-known and widely used text-processing algorithms: (1) I used the *SpamSum* simhashing algorithm, a context triggered piecewise hash (CTPH) method originally proposed by Andrew Tridgell as a spam detection technique, to fold large bodies of code into short and comparable hashes with partial matching possibility (Kornblum, 2006). (2) The subsequent matching of hashes was carried out using a customized Levenshtein edit distance calculation algorithm. Levenshtein distance is a string metric that measures the differences between two character sequences by

quantifying the minimum number of single-character edits needed to transform one string to the other (Levenshtein, 1966; Navarro, 2001).

The program first transforms the Ruby code into an Abstract Syntax Tree composed of intermediary S-expressions (a.k.a., sexprs or sexps) in order to eliminate notational differences in coding such as line wrapping and indentation styles, space characters and punctuation, identifier names, and alternative but equivalent control structures (Baxter, Yahin, Moura, Sant'Anna, & Bier, 1998). Then it extracts the nested code structures within a body of code based on the syntax tree, and uses SpamSum to hash the corresponding S-expressions whenever the structure contains more than a certain threshold of instructions. These hashes are then loaded in an in-memory b-tree structure for fast search and retrieval (Bayer & McCreight, 1972). Then the program does a pairwise comparison of all the hashes belonging to code structures of comparable sizes and kinds (i.e., loops versus loops and functions versus functions) by calculating a similarity ratio based on the edit distance of pairs of hashes. Hashes with a similarity score above 70% are retained as cloning matches. Once the search step is over, the matches are compared and all but the largest match between each pair of files are filtered out. The matching score and the size of the matched code are then retained for use as the basis for constructing the network of whitebox reuse.

There are different types of cloning, referring to the degree of faithfulness to the original code. The simplest case of code cloning is copy and paste of source code with minor layout modifications (Type-1 clone). In other cases the reuser may decide to bring non-syntactic changes to the code, adapting the identifiers (e.g., variable names) and literals (e.g., UI messages) to their liking (Type-2). A reuser may also decide to remove or add syntactic elements to the code in order to adapt it to the new context of reuse (Type-3). Finally, the original code may serve only as an inspiration, while the adopter completely changes the implementation (Type-4) (Roy, Cordy, & Koschke, 2009). The similarity hashing method used here allows for detecting the bulk of type-1 to type-3 clones (Kornblum, 2006).

Marking code cloned across projects yielded a dynamic, directed, and weighted network as it is possible to track down the amount and the direction of cloning (See Figure 4). The dynamic aspect

of the clone network is limited, though, in that one cannot be sure when a project stopped using the code copied from another project, given that code modifications disfigure the cloned code over time and make it impossible to detect them. In other words, the edges of this network are timestamped with a creation date, but no dissolution date can be identified.

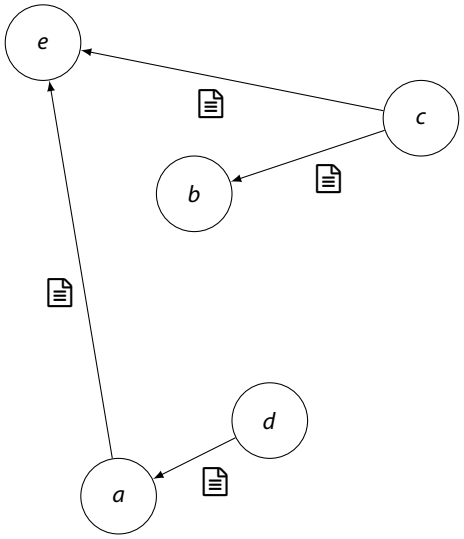


Figure 4. Network of whitebox reuse (code clone network)

The network of developer-project affiliations (co-affiliation network). Since open source projects are open for contribution, the emergence of an affiliation network is expected (Grewal et al., 2006). Affiliation networks are bipartite networks depicting participation of actors in common events (Borgatti & Halgin, 2011). I reconstructed the network of community collaborations by marking project affiliations of developers based on their involvement in projects. In this case each developer is an actor and each project can be considered an event (See Figure 5).

I constructed this network by integrating all project contributions with identified developers, based on the project commit logs of Git and GitHub. The resulting network provides granular edge properties for developer-project affiliations (e.g., the exact dates, the volume of contribution, etc.). Once more, this network is both dynamic and directed, as we can track down the time and infer the direction of movements of the developers between the projects based on the timing of the movements.

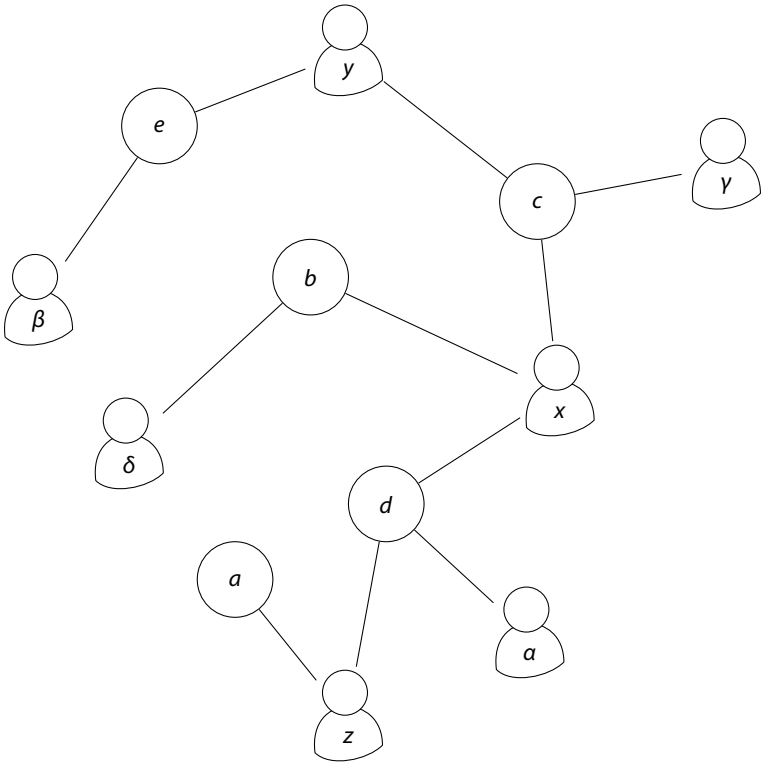
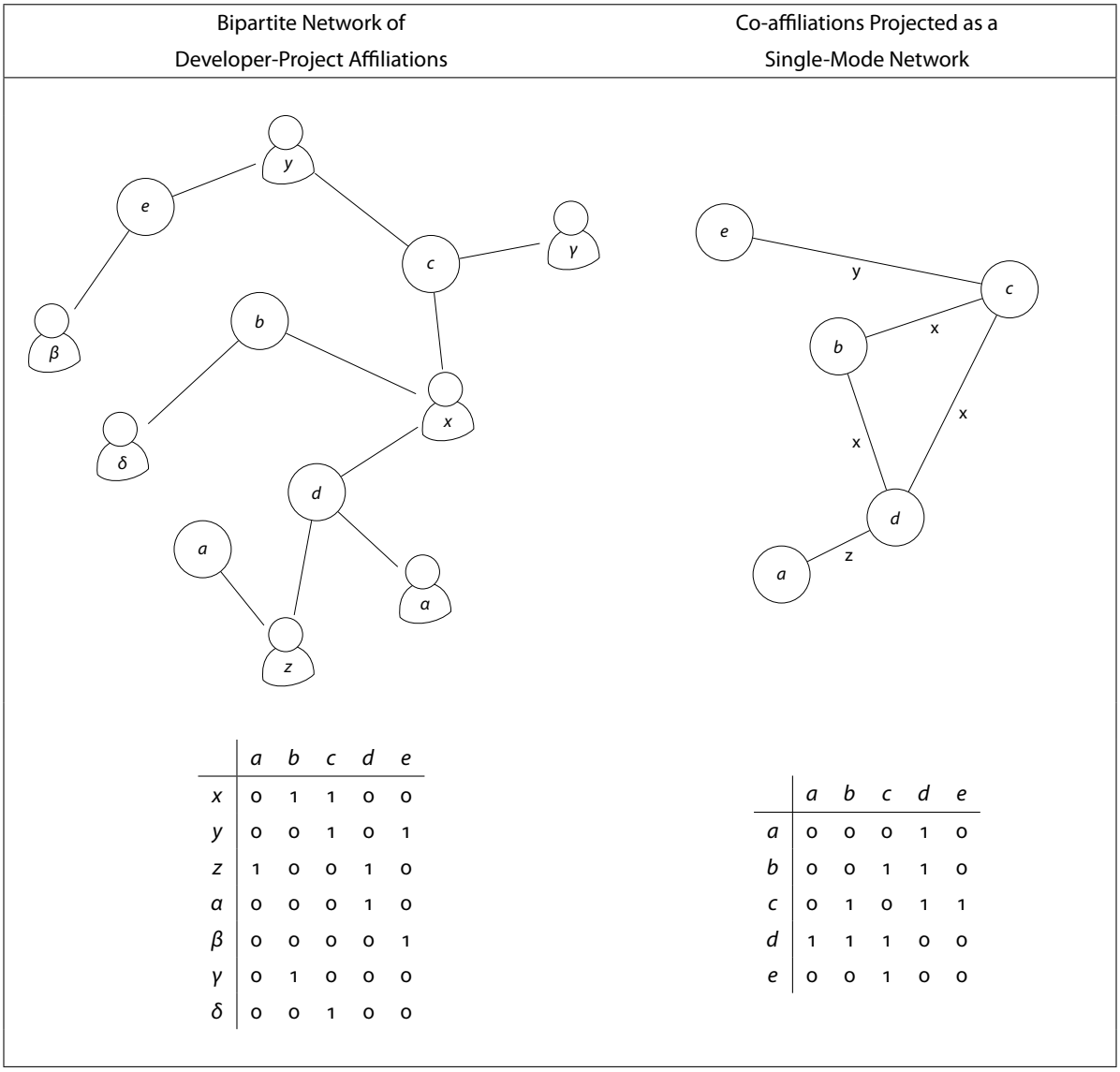


Figure 5. Bipartite network of developer-project affiliations

I also made an effort to complement this dataset using the self-reported package authorship information available from RubyGems spec files, but the data turned out to be highly censored, as well as being much less granular.

Affiliation networks like the one described here are often projected as single-mode networks for analysis purposes. A two-mode network can be projected into two distinct single-mode networks: (1) a network of actors, and (2) a network of events . Given that social network studies often focus on ties between individuals, it is customary to project two-mode networks into an inter-actor network (Borgatti & Halgin, 2011).

Nonetheless in the current study, the focus is on project-level ties, making the inter-project network depicting project-developer coaffiliations the more suitable of the two possible projections. The simplest projection results in a network of projects with binary edges (See Figure 6). Given that in the second stage of data gathering I have obtained several edge properties for developer-project affiliations (i.e., frequency of contribution, volume of contribution, etc.), I had to resort to customized



	a	*b*	*c*	*d*	*e*
x	0	1	1	0	0
y	0	0	1	0	1
z	1	0	0	1	0
α	0	0	0	1	0
β	0	0	0	0	1
γ	0	1	0	0	0
δ	0	0	1	0	0

	a	*b*	*c*	*d*	*e*
a	0	0	0	1	0
b	0	0	1	1	0
c	0	1	0	1	1
d	1	1	1	0	0
e	0	0	1	0	0

Figure 6. Projecting the bipartite network of developer-project affiliations as a single-mode network of coaffiliations

projection methods so as to preserve and aggregate the data from the original bipartite network.

Measures

All hypotheses in this proposal investigate project-level or project-network-level phenomena with the majority of the dependent, independent and control variables pointing to dyadic measures such as ties, tie properties and node similarity scores. Some project-level variables, such as centrality, age and size also enter the analysis. Since all the analyses are done at the project or project-dyad level, all network data at lower levels of analysis (e.g., individuals) has to be aggregated into project-level prior to running the analyses.

Dyad-level Measures.

Reuse Ties (Blackbox and Whitebox). I have gathered data on two types of reuse tie between projects: (1) *Functional Reuse*, derived from the network of blackbox reuse. (2) *Code Clone*, derived from the network of whitebox reuse. Reuse ties are directed, that is, project a reusing project b (i.e., $a \bullet\!\!\rightarrow b$) is distinguishable from project b reusing project a (i.e., $b \bullet\!\!\rightarrow a$). Reuse ties are also dynamic, that is, they come into existence when a project starts reusing another project, and they dissolve when the developers decide to forego that instance of reuse, whether by internally developing the functionality, removing the functionality or relying on a third project to provide the functionality.

The *Blackbox Reuse* variable is of binary nature, referring to the existence of a functional dependency or lack thereof, between two projects. This limitation has to do with the data source I use to generate the blackbox reuse variable. The RubyGems spec files only mention if a dependency exists, and have no statistics on the number of cross-project function calls. The direction of functional reuse ties can be known with certainty as it clearly appears in the project metadata. It is also rather easy to infer when a functional reuse tie comes into existence and when it dissolves by comparing the dependency lists of the different versions of a project.

The *Whitebox Reuse* variable can be quantified in a more fine-grained manner using information on the amount of reuse obtained from the clone detection algorithm (e.g., lines of code). Yet, for several practical reasons, including accuracy issues, I use the dichotomized version of the variable,

simply indicating whether the source code of a project contains instructions copied from another project or not. The direction of the code clone ties had to be inferred by identifying the first instance, or the origin, of a code snippet, and then considering the other, posterior instances as clones of the original. While given this formulation it is easy to estimate the time of creation of a code clone tie, there is no reasonable way to estimate the dissolution of a code clone tie.

The reuse tie variables are pivotal to all the hypotheses of this study, constituting either the dependent variable or the main independent variable. When using the reuse measures as dependent variables, I have preserved the direction of the tie. But when using them as independent variables, I have treated them as undirected (i.e., $a \bullet\!\!-\!\!\bullet\ b$). A combination of theoretical and operational concerns have brought about this special setup. As far as the theoretical arguments are concerned, all the hypotheses of this study can be tested using an undirected operationalization of the reuse ties. Yet, many of the important control variables require a directed operationalization of the dependent dyadic variable (i.e., controlling whether the centrality of the target project has an effect on reuse). This special treatment of the reuse variables must be assumed throughout the analysis section.

Project Coaffiliation. The project coaffiliation variable derives from the network of project coaffiliations. As detailed in the previous section, the project coaffiliation network is a project-level projection of the developer-project affiliation network. The project affiliations, in turn, are calculated based on the individual contributions to the projects. The developers are considered as affiliated with a project only as long as they contribute to the project. Project Coaffiliation ties, therefore, can potentially be weighted ties. They are also dynamic, and potentially directed (i.e., Did developer x start on project a and then moved to project b or vice versa?). The projection from developer-project affiliation to project coaffiliation is not a simple projection of a binary network then.

For the purpose of this projection one has to define and delimit the construct of project coaffiliation. For instance, a coaffiliation may be seen as simultaneous affiliation in two or more projects for a minimal period of time (strict definition) or having contributions to two or more projects during a specific window of observation (relaxed definition). At its simplest, the coaffiliation variable may take binary values indicating whether there has been any overlap between the body of developers of

two projects. At a more advanced level, the coaffiliation variable may also incorporate a score based on the number of coaffiliated developers, the instances of contribution or the amount of contribution to the coaffiliated projects. Project coaffiliation is also a time-dependant phenomenon and has to be treated as such when included in the analysis.

I used the number of instances of code contribution to a project (i.e., code commits) as the weight for the developer-project affiliations. Commit count has often been used in the literature as an indicator of the degree of involvement of the developers in the projects. The weight of the coaffiliation between the two projects — the projection of two developer-project affiliations — has to signal the degree of contribution to each project, but also the equality of contribution to the two projects. A coaffiliation tie based on a highly asymmetric contribution record in two projects must be scored down.

I calculate the coaffiliation weight for each project dyad with a commonly affiliated developer as the inequality-adjusted mean of the developer's contribution to the two projects. I opted for a simple yet effective Atkinson inequality measure (Atkinson, 1970), widely used in the economics literature, to adjust for inequality. first developed his measure to tackle the insufficiencies of some summary inequality measures such as Gini. The Atkinson measure includes a parameter (ϵ) indicating the degree of inequality-aversion. I use the common parameter value of 1, in which case the the Atkinson measure can be simplified:

$$A_\varepsilon(y_1, \ldots, y_N) = \begin{cases} 1 - \frac{1}{\mu}\left(\frac{1}{N}\sum_{i=1}^{N} y_i^{1-\varepsilon}\right)^{1/(1-\varepsilon)} & \text{for } 0 \leq \epsilon \neq 1 \\ 1 - \frac{1}{\mu}\left(\prod_{i=1}^{N} y_i\right)^{1/N} & \text{for } \varepsilon = 1, \end{cases}$$

I used the Atkinson measure to adjust the mean contribution to each coaffiliated project dyad. When more than one developer was simultaneously involved in the two projects I combined the contributions before calculating the weight. The same procedure was repeated for each project dyad with common developers. I allowed a maximum one-year lag between two affiliations to count as the basis for a coaffiliation tie.

The coaffiliation tie variable is also pivotal to all the hypotheses in this study. I include the coaffiliation tie both as a continuous variable (with tie weight) and a binary variable in Hypothesis 2b. I use the natural logarithm of the weight calculated above given that its values tend to be over-dispersed. I use a dichotomized version of the variable in the rest of the hypotheses, as they do not make claims about the strength of the coaffiliation ties.

I take the order of contribution in the projects as an indicator of the direction of movement across projects. Just like for the reuse variables, I consider coaffiliation as directed when it is a dependent variable in the analysis, and as undirected when it is an independent variable.

License Similarity. The question of reuse goes hand in hand with the right to reuse. The majority of open source software projects are not in public domain, and are released under one or more licenses with specific constraints. Any research looking into reuse practices is thus expected to take the effect of these licences into account.

During the last few years the MIT License has grown to become the dominant license adopted by the new projects in the Ruby community (See Figure 7), but there is still enough variability in the license choices to justify seeking a possible effect when it comes to accessing and adopting intellectual property commons.

Fewer than a quarter of gems publicize their software license within the package specification file. For the others, the licensing information has to be found in the project source files. I used a dedicated license detection tool called ScanCode to detect the project licensing information (Ombredanne, Yang, Balusa, et al., 2017). ScanCode includes the signature for more than 2,000 open source licenses and is able to detect them duly.

I ran ScanCode both the on the RubyGems sources and the GitHub sources of all the projects, and then weeded out the false positives manually (e.g., The license of an included font taken for the license of the whole package). ScanCode identified 150 licenses in use in the Ruby libraries belonging to 77 different license families. Whenever a project was released under more than one license, I retained all the licenses. License changes across project versions were marginal and negligible.

ScanCode also includes the license category of each detected license, referring to the reuse lim-

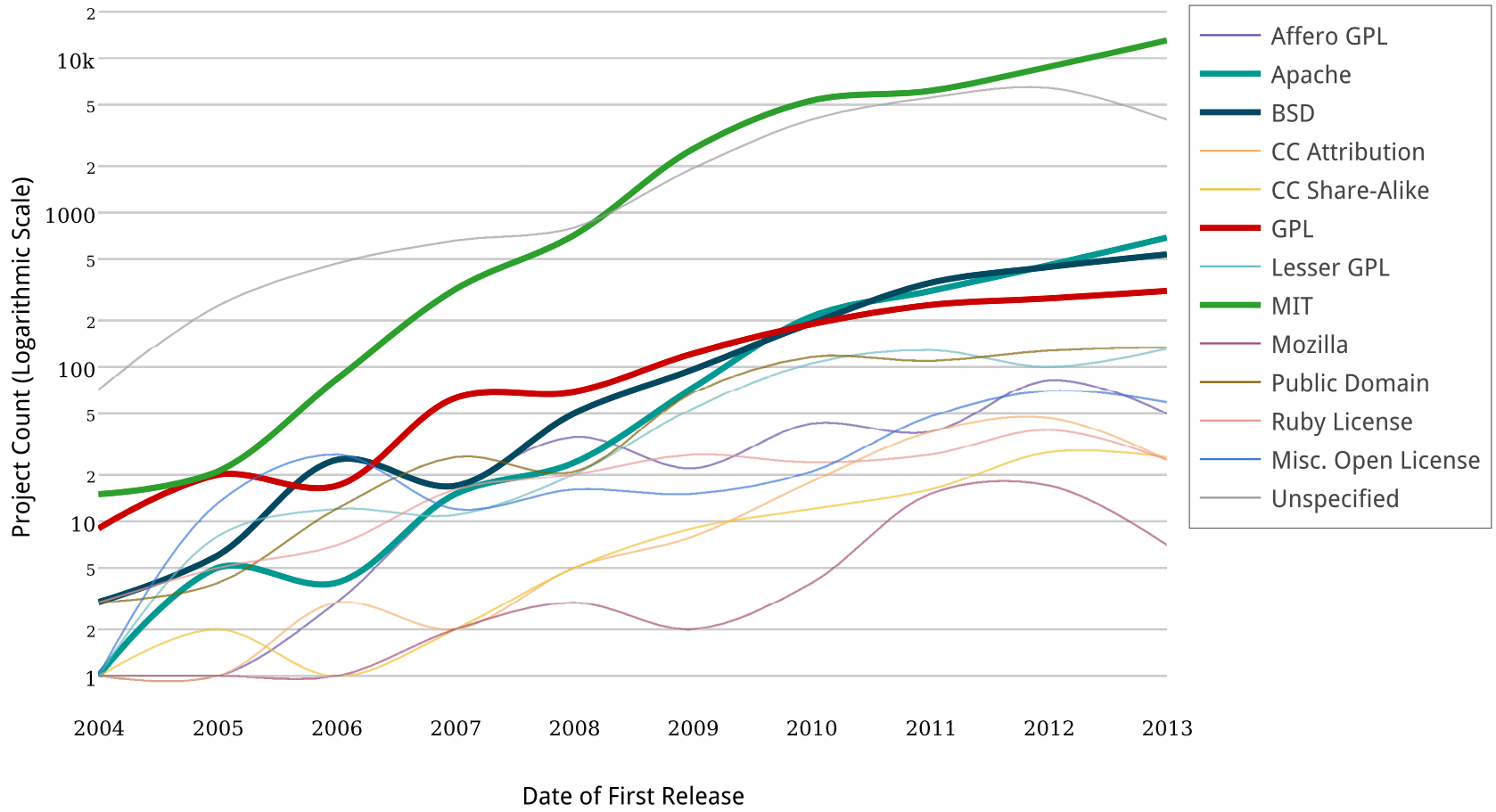


Figure 7. Evolution of license adoption in Ruby community (2003–2013)*†

*Only well-known licenses have been included in the chart.

†2013 figures are based on the first eight months of the year.

itations of that license, and mainly based on Linux Foundation's SPDX license matching guidelines*.

Family	Category	Licenses (examples)	Count
MIT	Permissive	mit, x11, fsf-mit, x11-fsf, mit-modern, x11-lucent, etc.	27,393
GPL	Copyleft	gpl, gpl-1.0, gpl-2.0, gpl-3.0, agpl-3.0, gpl-1.0-plus, etc.	1,926
BSD	Permissive	bsd-new, bsd-intel, bsd-original, bsd-simplified, etc.	1,354
Apache	Permissive	apache-1.1, apache-2.0, apache-due-credit	1,345
GPL	Copyleft Limited	lgpl, lgpl-2.0, lgpl-2.1, lgpl-3.0, gpl-2.0-font, gpl-2.0-bison, etc.	826
Public Domain	Public Domain	cc-pd, sax-pd, cco-1.0, json-pd, fsf-free, unlicense, wtfpl-2.0, etc.	629
Ruby	Copyleft Limited	ruby, ruby-2011	264
ISC	Permissive	isc	118
Artistic	Copyleft Limited	artistic-1.0, artistic-2.0, artistic-perl-1.0	70
Creative Commons	Copyleft Limited	cc-by-sa-2.0, cc-by-sa-2.5, cc-by-sa-3.0	61
Creative Commons	Permissive	cc-by-2.0, cc-by-2.5, cc-by-3.0, cc-by-2.0-uk	57
Mozilla	Copyleft Limited	mpl, mpl-1.0, mpl-1.1, mpl-2.0	45
zlib	Permissive	zlib	38
Creative Commons	Free Restricted	cc-by-nc-3.0, cc-by-nd-3.0, cc-by-nc-nd-2.0, cc-by-nc-nd-3.0, etc.	30
Eclipse	Copyleft Limited	epl-1.0	24

Table 3. Top license families, sample licenses and corresponding categories

It is important to note that discrepancy in both license family and license category can affect code reusability. While the licenses within a certain category are often deemed more compatible with each other (e.g., permissive licenses), this is not always the case (e.g., copyleft licenses). And although the licenses of the same family are often deemed more compatible with each other (e.g., all BSD licenses), certain license families do not follow the rule, as they have members across different categories (e.g., Creative Commons or GPL).

I created a license similarity matrix for dyads of projects by marking the project dyads published with licenses of the same family and the same category as having similar licenses (See Table 3). The *License Similarity* variable, which is a static dyad-level variable is extracted from this matrix. Although license family and license category could enter the analysis as separate variables, the potential added value would be submerged by the high degree of collinearity between the two variables (r = .85).

License similarity enters the analysis as a control variable in all the hypotheses, and constitutes

*See https://spdx.org/

the dependent variable in Hypotheses 3a, 3b, and 6.

Description Similarity. The reuse choice is ultimately a subjective choice, revolving around the actual needs of a project and its developers. One may argue that not all projects are equally exposed to the risk of reuse at each decision point, and that the risk of reuse hinges on the content and the functionalities of the reusing project and the reuse candidates. Software projects within the same semantic sphere, targeting neighbouring functionalities, platforms, or audiences, may potentially be more prone to drawing on each other. The same argument holds for the developer affiliations of the projects: Projects with close topics are more likely to attract the same developers.

Prior studies of open source communities have controlled this unobserved source of heterogeneity either by restricting their dataset to a localized set of closely interrelated projects (e.g., Grewal et al., 2006) or have used project categories, as defined by foundries like SourceForge, as a control variable (e.g., Hahn, Moon, & Zhang, 2008). But these categories often do not reflect the content and functionalities of the software as much as they reflect the choices of convenience of the distribution platforms (Faraj & Azad, 2012).

An alternative approach would be to use the topic modelling techniques with the project descriptions and allow the categories emerge from the descriptive text composed by the developers of the program. An appropriate method for this purpose would be Latent Semantic Analysis (LSA). LSA refers to a set of topic modelling and text classification methods that is gaining popularity in social sciences, from social psychology to information systems (Deerwester et al., 1990). Extensions of LSA have been used for document similarity detection and clustering, but also to understand the latent structure of a corpus of texts (Evangelopoulos, Zhang, & Prybutok, 2010).

LSA assumes that the meaning of a text is related to the patterns of inclusion and exclusion of terms from it, and that the terms with similar meaning will occur in the same texts. The algorithm uses a factorization method called singular value decomposition (SVD) to find a lower-ranked approximation of the original term occurrence matrix of the documents (document-term matrix), identifying and quantifying term similarities in the process. LSA can be combined with the cosine similarity method to obtain a measure of document similarity.

Cosine similarity is a measure of similarity between two vectors. If A_i and B_i are components of the vectors A and B, the cosine similarity of the vectors can calculated as:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

One can calculate a simple measure of similarity between two documents by setting A and B as the term frequency vectors of the two documents, but using the LSA vectors offers a superior solution, given that LSA mitigates the issues of synonymy and polysemy in similarity detection (Deerwester et al., 1990).

In order to obtain a measure of similarity of the project descriptions, I ran the cleaned and stop-worded project descriptions through a Latent Semantic Analysis package called "gensim" (Řehůřek & Sojka, 2010). I refrained from stubbing and stemming the terms as in technical documents derivative words tend to have specialized meanings. I set the number of topics to the recommended number of 300, optimal for cases in which the number of topics is not previously known (Bradford, 2008). Then I used the resulting LSI matrix to calculate the pairwise cosine similarity score of all the project descriptions. This yields a number within the range of -1 to 1 for each pair of projects.

Description Similarity is used as a control variable in all hypotheses.

Node-level Measures.

Project Indegree Centrality. The patterns of reuse in the Ruby community show great levels of inequality, revealing itself through richly connected hubs in the network diagram (See Figure 8). The degree distribution plot of the reuse network shows a power law distribution (See Figure 9) that in turn signals the scale-free nature of the reuse network (Barabási & Albert, 1999). Therefore, there might be a "Matthew effect", or "preferential attachment" at work in project reuse, whereby those projects that are more recurrently reused, are more likely to gain new reusers as well (Albert & Barabási, 2001).

I use the natural logarithm of the indegree centrality of the reuse target in the blackbox reuse

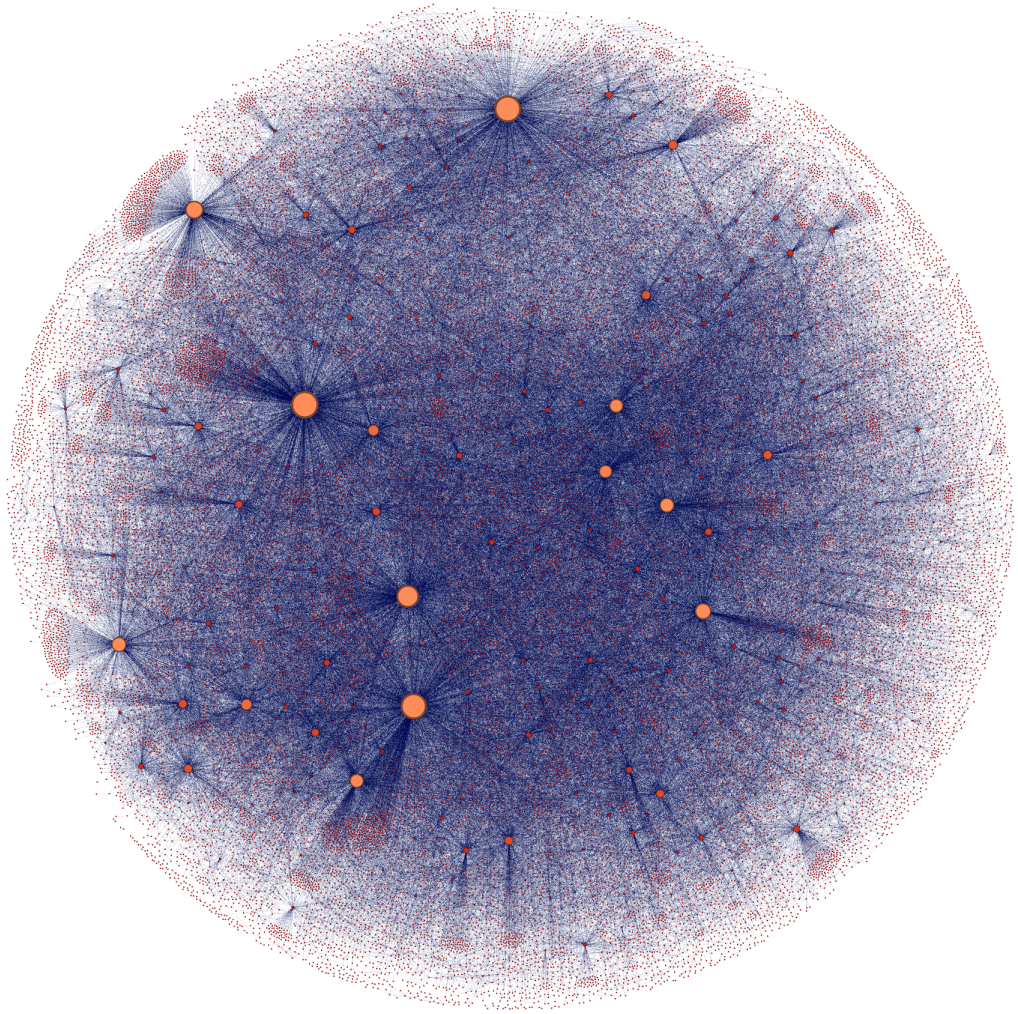


Figure 8. Functional reuse in Ruby community (2003–2013)*

* for projects with more than one day of development activity

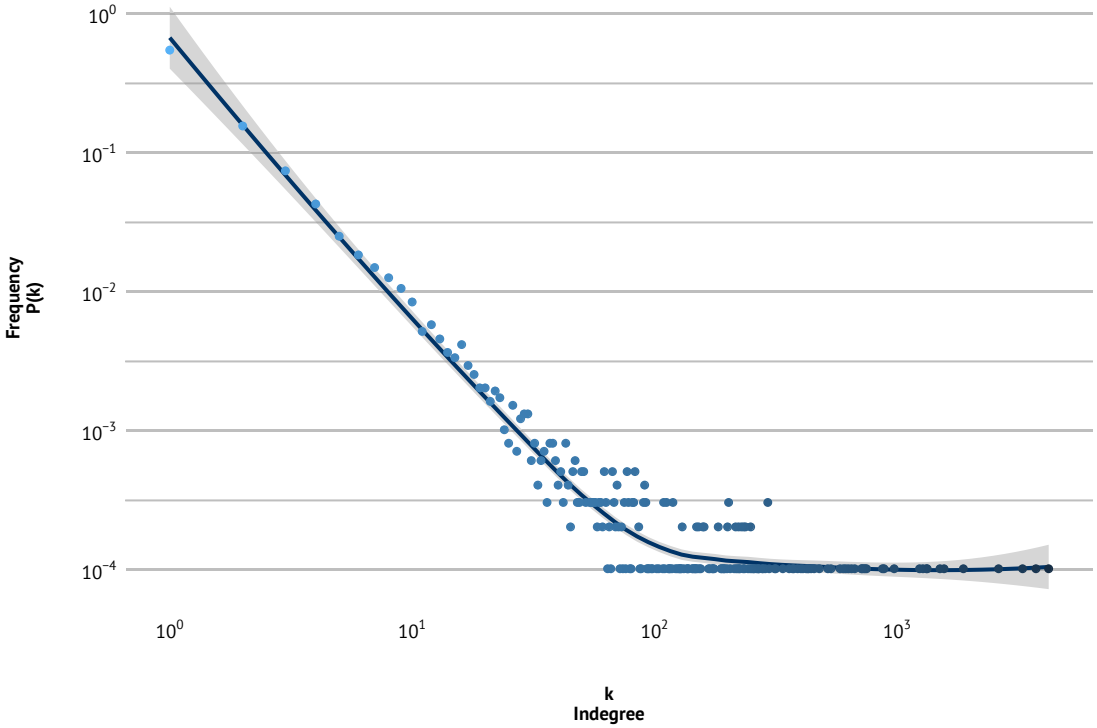


Figure 9. Log-Log degree distribution plot for the reuse network

network as a control variable to capture the potential effects of such preferences on the reuse choices. The variable is used in all models whose dependant variable is a reuse measure (Hypotheses 1–3).

Project Contributors. Although the project coaffiliation network of the Ruby community cannot be easily described as scale-free (See Figure 10), it is still characterized by a high degree of inequality. Time is a scarce resource for the developers and each developer can only work on a few projects simultaneously. It is conceivable that the developers show a tendency towards contributing to the projects that already attract a considerable amount of participation.

The *Contributors* variable is supposed to control for this effect where coaffiliation is the dependent variable (Hypotheses 4–6). The measure pertains to the destination project of the coaffiliation link — or the project that the developer joined the second. *Contributors* is the number of contributors that have contributed to the project so far, and it is logged to control the over-dispersion.

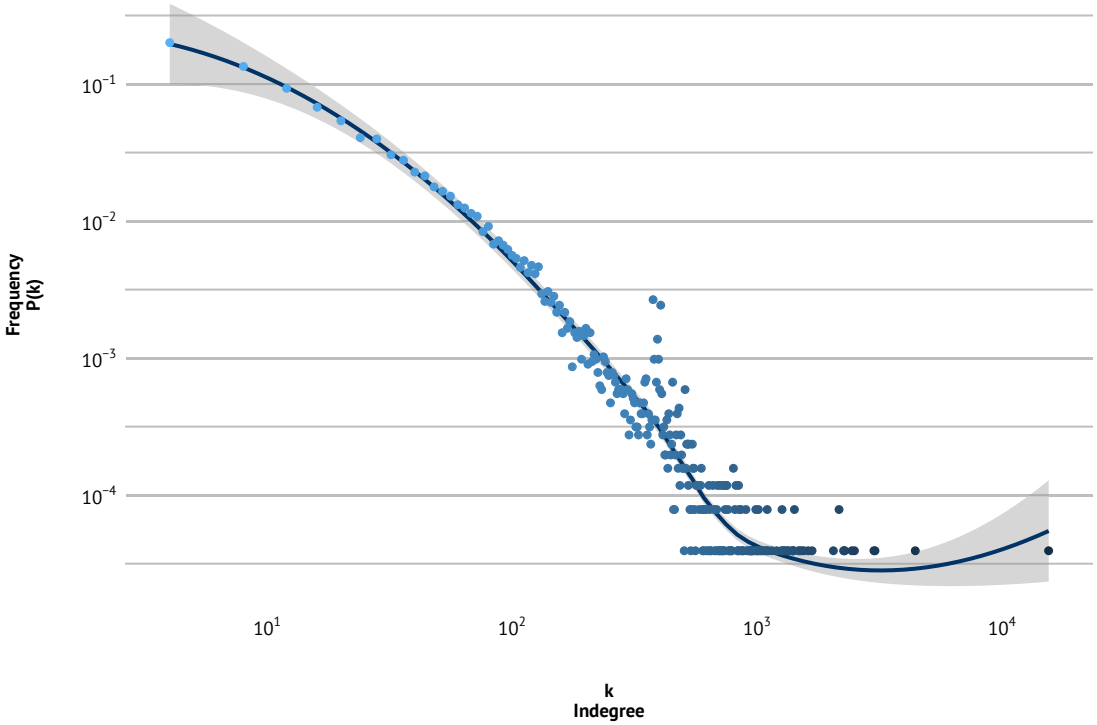


Figure 10. Log-Log degree distribution plot for the coaffiliation network

Project Activity. *Recent Activity* is calculated as the natural logarithm of the total number of contributions during the twelve months prior to the point of measurement. It is included as a control variable in all the regressions for all hypotheses to control for the activity state of the project. The developers have a preference for active projects when it comes to decisions about reusing the project (i.e., updated code) or contributing to it (i.e., usefulness and continuity).

Project Size and Age. Project age and size enter the analysis as proxy variables for project maturity. They are control variables in all the hypotheses. They relate to the reused project when reuse is the dependent variable, and to the destination in the coaffiliation link when coaffiliation is the dependent variable. *Project Size* is the natural logarithm of the size of the project in kilobytes. *Project Age* is the age of the project in years.

Analysis

All the hypothesis in this study intend to infer the effect of one type of edge on another in networks composed of tens of thousands of nodes and tens or hundreds of thousands of edges. These are very sparse networks with extremely low density and a considerable number of structural zeros (i.e., a few billions). The ideal method to test the hypotheses in the current study would be either binomial and/or multinomial logistic regressions. But there is a well-known downward bias in coefficient and probability estimations of logistic regressions when dealing with finite sample or rare event data — the phenomena in which non-events far outnumber the events. While the number of observations is sufficiently large in this study, the sparsity of events versus non-events condemns the conventional logistic regressions to suffer from this known bias.

Moreover, Including all the structural zeros in the regression analysis is not operationally feasible with the current resource constraints of the statistics software packages. Most of those structural zeros contain little surplus information as they refer to isolate projects that never enter in a reuse or coaffiliation relation with the other projects.

G. King and Zeng (2001b) propose a combination of case-control stratified sampling design (i.e., downsampling) and penalized logistic regression in order to alleviate the above concerns. The first step consists of selective sampling based on the value of the dependent variable, sampling all the ones and one or more equally-sized samples of the zeros. Next, they suggest a logistic regression method, called Rare Event Logit, to obtain corrected estimates and robust standard errors based on the sampled data. The authors themselves have used the method to analyse network data from international conflicts (G. King & Zeng, 2001a).

Sub-sampling. Given that King's method calls for sampling based on the values of the dependent variable, each different dependent variable necessitated drawing a separate sub-sample from the data. As such, Hypotheses 1a and 3a use one sub-sample (H1a sub-sample), Hypotheses 1b and 3b rely on another sub-sample (H1b sub-sample) Hypotheses 2a and 2b make use of yet another sub-sample (H2 sub-sample) while Hypotheses 4a to 6 share a separate sub-sample (H4 sub-sample).

Sample	DV	Hypotheses	o Multiplier	Size
H1a	Blackbox Reuse	H1a & H3a	50x	948,705
H1b	Whitebox Reuse	H1b & H3b	200x	586,538
H2	Reuse Type (Multinomial)	H2a & H2b	50x	1,095,893
H4	Coaffiliation	H4a, H4b, H5 & H6	80x	4,239,665

Table 4. Sub-sample descriptions

In order to get as close as possible to the notion of causality, and to remain faithful to the spirit of the hypotheses, I have conducted the sub-sampling with embedded time-ordering. For each hypothesis I take the year 2012 as the observation period in which to seek variability on the dependent variables.

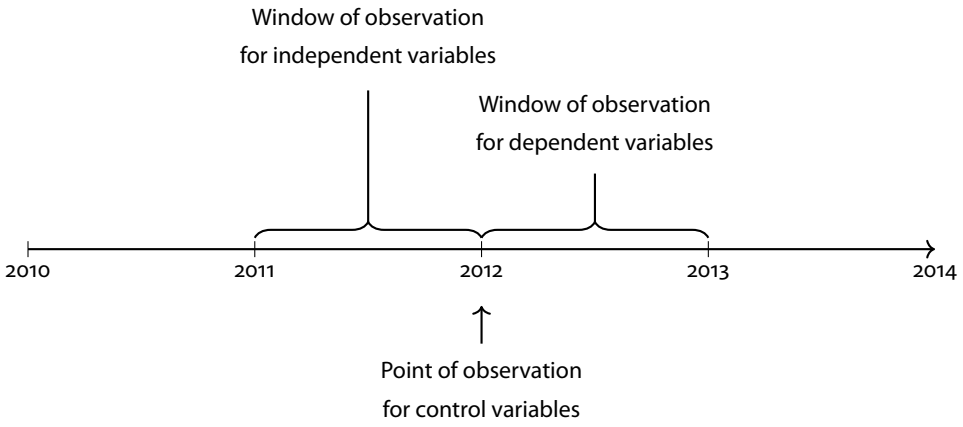


Figure 11. Observation timeline for the sub-sampling step

I also define a window of observation of one year for the independent variables (See Figure 11). For instance, for H1a I made a sample of zero and ones for new functional reuse ties during the DV observation period (Jan. 2012–Dec. 2012). Then I looked for cases of coaffiliation within the twelve months prior to the date of functional reuse. As a result, the longest possible interval between coaffiliation (IV) and reuse (DV) is less than or equal to twelve months.

I focused on this specific period for sampling, because it is the period for which the data contains the highest number of reuse observations. The occurrences of reuse are more scarce than those of coaffiliation in the data, and therefore it is a priority to obtain a large enough sample that prevents

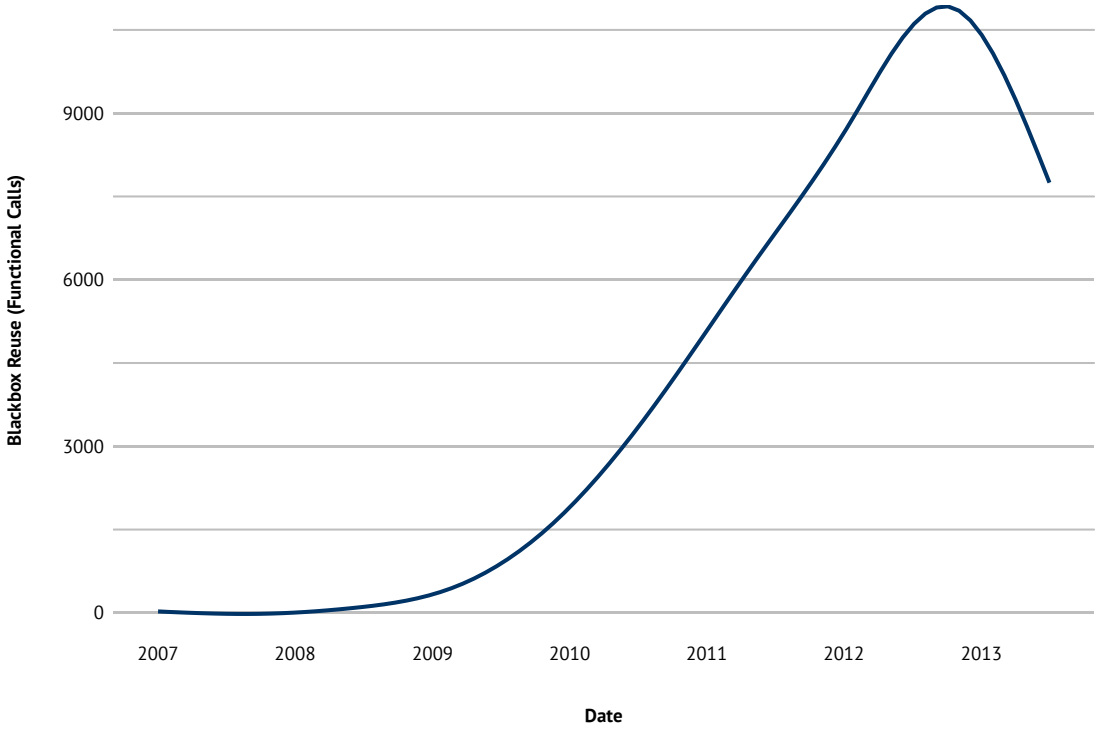


Figure 12. Functional reuse in Ruby community

instability or total separation in the regressions. The reuse observations start to drop at the beginning of 2013, because the data does not include projects created after that date (See Figure 12).

I measured the control variables at the cut-off point between the IV and DV observation windows. I have also set a one-year limit for the lifespan of developer-project affiliations. This means I consider the affiliation as dissolved if it does not get reactivated through a new contribution within the space of one year. To the best of my knowledge this is a more stringent criterion than the convention in the field, whereby the social ties are assumed to linger on for a span of four years after the last contact (P. V. Singh & Phelps, 2013; Uzzi & Spiro, 2005).

Finally, I made "zero" samples several times the size of "one" samples in the stratified sampling process. According to G. King and Zeng (2001a) a five-fold "zero" sample constitutes the sweet spot for information gain versus the hassle of additional data gathering or analysis. But given that in this study I deal with sparse variables both in IV and DV roles, I observed significant improvements in estimation stability as I increased the sample sizes. Therefore I retained the large samples, specially

since zero samples came at near-zero cost (See Table 4).

In all samples I had to discard the *Rails* project. Rails is the flagship project of the Ruby community and is foundational to most use cases of the Ruby language. This makes Rails highly central to many cases of collaboration and reuse in the community, but at a level of magnitude disproportional to the other projects. Rails is an outlier in every aspect, with reuse and coaffiliation figures tens of times those of the other leading projects. This destabilized or biased the estimates, what warranted discarding the project from the samples.

The descriptive statistics and the correlation tables for the four sub-samples are included in tables 5 to 12.

Models. I used *Rare Events Logistic* regressions (relogit), as suggested by G. King and Zeng (2001b) to estimate the models proposed in all the hypotheses, save for 2a and 2b. The relogit estimates are more resistant to the potential biases introduced by the sparsity of the events as well as the DV-based sampling.

The relogit regression resembles the standard logistic regression in both its stochastic and systematic components:

$$Y_i \sim \text{Bernoulli}(\pi_i)$$

$$\pi_i = \frac{1}{1 + \exp(-x_i\beta)}$$

Relogit, though, offers two methods for correcting the bias in the constant term that selecting on the dependent variable may introduce.

- The *prior correction* method makes adjustments directly to the intercept term, by taking into consideration the difference between the fraction of events in the sample and the fraction of events in the population. If τ is the true fraction of events in the population, $\bar{y}$ the sample's fraction of events, $\hat{\beta}_0$ the uncorrected intercept term and β_0 the corrected intercept:

$$\beta_0 = \hat{\beta}_0 - \ln\left[\left(\frac{1-\tau}{\tau}\right)\left(\frac{\bar{y}}{1-\bar{y}}\right)\right]$$

Statistic	Mean	St. Dev.	Min	Pctl(25)	Median	Pctl(75)	Max
Blackbox Reuse	0.003	0.057	0	0	0	0	1
Coaffiliation	0.001	0.033	0	0	0	0	1
License Similarity	0.473	0.499	0	0	0	1	1
Description Similarity	0.016	0.072	−0.300	0.000	0.000	0.000	1.000
Age (Reused)	2.034	1.123	0.008	1.252	1.890	2.597	7.299
Size (Reused)	2.807	1.339	1.099	1.946	2.398	3.091	11.701
Activity (Reused)	1.365	1.644	0.000	0.000	0.693	2.639	8.418
Indegree (Reused)	0.245	0.699	0.000	0.000	0.000	0.000	6.751

Table 5. Descriptive statistics for the H1a sub-sample (DV: Blackbox Reuse)

	Blackbox Reuse	Coaffiliation	License Similarity	Description Similarity	Age (Reused)	Size (Reused)	Activity (Reused)	Indegree (Reused)
Blackbox Reuse	1							
Coaffiliation	0.172	1						
License Similarity	0.011	0.011	1					
Description Similarity	0.036	0.024	0.020	1				
Age (Reused)	0.063	0.010	0.009	0.009	1			
Size (Reused)	0.056	0.016	0.010	0.019	0.173	1		
Activity (Reused)	0.083	0.038	0	0.018	−0.101	0.259	1	
Indegree (Reused)	0.258	0.048	0.015	0.018	0.362	0.217	0.292	1

Table 6. Pearson correlations for the H1a sub-sample (DV: Blackbox Reuse)

Statistic	Mean	St. Dev.	Min	Pctl(25)	Median	Pctl(75)	Max
Whitebox Reuse	0.001	0.035	0	0	0	0	1
Coaffiliation	0.001	0.032	0	0	0	0	1
License Similarity	0.473	0.499	0	0	0	1	1
Description Similarity	0.016	0.073	−0.200	0.000	0.000	0.000	1.000
Age (Reused)	2.031	1.120	0.005	1.252	1.890	2.595	7.299
Size (Reused)	2.803	1.335	1.099	1.946	2.398	3.091	11.701
Activity (Reused)	1.356	1.638	0.000	0.000	0.693	2.565	8.418
Indegree (Reused)	0.237	0.674	0.000	0.000	0.000	0.000	6.751

Table 7. Descriptive statistics for the H1b sub-sample (DV: Whitebox Reuse)

	Whitebox Reuse	Coaffiliation	License Similarity	Description Similarity	Age (Reused)	Size (Reused)	Activity (Reused)	Indegree (Reused)
Whitebox Reuse	1							
Coaffiliation	0.222	1						
License Similarity	0.013	0.015	1					
Description Similarity	0.118	0.040	0.021	1				
Age (Reused)	−0.015	0.007	0.009	0.005	1			
Size (Reused)	0.025	0.015	0.008	0.021	0.172	1		
Activity (Reused)	0.029	0.036	0	0.018	−0.109	0.254	1	
Indegree (Reused)	0.021	0.036	0.012	0.010	0.354	0.211	0.283	1

Table 8. Pearson correlations for the H1b sub-sample (DV: Whitebox Reuse)

Statistic	Mean	St. Dev.	Min	Pctl(25)	Median	Pctl(75)	Max
Coaffiliation (binary)	0.001	0.034	0	0	0	0	1
Coaffiliation (Weighted)	0.003	0.108	0.000	0.000	0.000	0.000	8.287
License Similarity	0.473	0.499	0	0	0	1	1
Description Similarity	0.016	0.072	−0.200	0.000	0.000	0.000	1.000
Age (Reused)	2.033	1.122	0.005	1.249	1.890	2.597	7.299
Size (Reused)	2.809	1.339	1.099	1.946	2.398	3.091	11.701
Activity (Reused)	1.367	1.644	0.000	0.000	0.693	2.639	8.418
Indegree (Reused)	0.245	0.698	0.000	0.000	0.000	0.000	6.751

Table 9. Descriptive statistics for the H2 sub-sample (DV: Reuse Type)

	Coaffiliation (binary)	Coaffiliation (Weighted)	License Similarity	Description Similarity	Age (Reused)	Size (Reused)	Activity (Reused)	Indegree (Reused)
Coaffiliation (binary)	1							
Coaffiliation (Weighted)	0.911	1						
License Similarity	0.014	0.014	1					
Description Similarity	0.034	0.036	0.020	1				
Age (Reused)	0.007	0.003	0.011	0.006	1			
Size (Reused)	0.017	0.016	0.009	0.021	0.174	1		
Activity (Reused)	0.041	0.040	−0.003	0.021	−0.102	0.257	1	
Indegree (Reused)	0.049	0.041	0.015	0.017	0.361	0.218	0.292	1

Table 10. Pearson correlations for the H2 sub-sample (DV: Reuse Type)

Statistic	Mean	St. Dev.	Min	Pctl(25)	Median	Pctl(75)	Max
Coaffiliation (binary)	0.004	0.064	0	0	0	0	1
Blackbox Reuse	0.0002	0.013	0	0	0	0	1
Whitebox Reuse	0.00003	0.006	0	0	0	0	1
License Similarity	0.474	0.499	0	0	0	1	1
Description Similarity	0.016	0.072	−0.300	0.000	0.000	0.000	1.000
Age (Destination)	0.863	0.679	0.003	0.318	0.715	1.252	7.299
Size (Destination)	2.633	1.276	1.099	1.872	2.251	2.773	11.701
Activity (Destination)	1.961	1.573	0.000	0.000	2.079	3.135	8.418
Contributors (Destination)	0.253	0.554	0.000	0.000	0.000	0.000	5.252

Table 11. Descriptive statistics for the H4 sub-sample (DV: Coaffiliation)

	Coaffiliation (binary)	Coaffiliation (Weighted)	License Similraity	Description Similarity	Age (Reused)	Size (Reused)	Activity (Reused)	Indegree (Reused)
Coaffiliation (binary)	1							
Coaffiliation (Weighted)	0.911	1						
License Similraity	0.014	0.014	1					
Description Similarity	0.034	0.036	0.020	1				
Age (Reused)	0.007	0.003	0.011	0.006	1			
Size (Reused)	0.017	0.016	0.009	0.021	0.174	1		
Activity (Reused)	0.041	0.040	−0.003	0.021	−0.102	0.257	1	
Indegree (Reused)	0.049	0.041	0.015	0.017	0.361	0.218	0.292	1

Table 12. Pearson correlations for the H4 sub-sample (DV: Coaffiliatione)

- The *weighting* method corrects for the case control design by attributing weights to the 0 and 1 cases and then performing a weighted logistic regression. If the 1 subscript denotes observations for which the dependent variable is 1, and the 0 subscript denotes observations for which the dependent variable is 0, then the the vector of weights w_i is obtained by:

$$\begin{aligned} w_1 &= \frac{\tau}{\bar{y}} \\ w_0 &= \frac{(1-\tau)}{(1-\bar{y})} \\ w_i &= w_1 Y_i + w_0(1-Y_i) \end{aligned}$$

This corrects the biases in the estimates of the the coefficients β due to finite sample or rare events bias. In addition, relogit corrects the other quantities of interest, such as the predicted probabilities (Choirat et al., 2017).

G. King and Zeng (2001b) recommend the weighting method over the prior correction method when the finite sample constraints do not apply and there is a chance that the model is misspecified. The novelty of some of the measures that I juxtapose, means that model misspecification is a possibility in this study. Therefore, given that I can raise the size of the samples to the required level to achieve estimate stability when using the weighting method, I opted for this latter.

Hypotheses 2a and 2b propose a comparison between the relative effect size of a covariate (i.e., coaffiliation) on two different outcome variables (i.e., blackbox and whitebox reuse). This requires estimating the two corresponding regressions in a single system that allows covariance between the two outcome variables, either using Zellner's (1962) seemingly unrelated regressions method (SUR) or through structural equation modelling (SEM). But a binomial logistic SEM is equivalent to a multinomial logistic regression when the separation between the outcome variables is complete. Given the minimal overlap — of only two cases — between the blackbox and whitebox reuse instances, I randomly assign the two cases to one of the two types of reuse and I run the estimation as a conventional multinomial logit. Unfortunately there is no widely accepted method to extend the correction and penalization performed by rare events logistic models to multinomial

logit or logistic SEM, therefore I will suffice to running the standard estimation algorithms in this case.

Results

The results are, overall, supportive of the picture of reuse and collaboration practices in open source communities offered in the theory development section. Since the hypotheses cover two different sets of dependent variables, knowledge reuse and coaffiliation, I have divided the results section into two subsections.

Knowledge Reuse. Hypotheses 1a to 3b explore the drivers of knowledge reuse between the open source projects. The results of the corresponding analyses are presented in tables 13, 14 and 15.

Hypotheses 1a and 1b suggested that developer coaffiliations between projects would have a positive effect on the subsequent creation of reuse ties between the projects. The results firmly support these core hypotheses.

The unadjusted logit estimates indicate that two projects coaffiliated through developers are 78 times more likely to create function calls to each other (Table 13 — Model 3). All else equal, this translates into a rise in the predicted probability of functional reuse from 0.07% to 5.4% where coaffiliation ties are present. The odds are estimated at 11 times for the adjusted relogit model (Table 13 — Model 4). Given that the relogit method includes prior correction in order to take into account the rarity of the outcome event, it provides much more conservative probability estimates, which is 0.0025% for non-coaffiliated project dyads versus 0.0281% for coaffiliated dyads.

Similarly, the unadjusted odds are put at 141 times for the reuse of code through cloning the instructions and algorithms subsequent to coaffiliation (Table 14 — Model 3). That points to an increase in the risk of establishing code clone ties from 0.03% to 4.51% for the coaffiliated project dyads. The adjusted relogit estimates put the odds of clone reuse further to coaffiliation at 78 times (Table 14 — Model 4). This latter figure translates into a predicted probability of 0.0004% for cloning in absence of coaffiliation, as compared to 0.0318% for cloning where the dyad is coaffiliated.

The above estimates also offer a hint about the results for Hypothesis 2a, as the relationship

between coaffiliation and whitebox reuse seems to be much stronger than that of coaffiliation and blackbox reuse. In order to make sure that the difference between the coefficient estimates for coaffiliation in the two models is significant, I recoded the types of reuse as a multinomial variable and entered it in a multinomial logistic regression (See Table 15). Model 3, just like the previous regressions, includes coaffiliation only as a binary variable. The effect of the coaffiliation variable on whitebox reuse is, as expected, noticeably larger than on blackbox reuse: The coefficients are 4.78 for whitebox versus 4.26 for blackbox. The coefficients are comparable to the uncorrected logit estimates in tables 13 and 14 (Model 3). A Wald test of the equality of the coefficients firmly rejects the null hypothesis, and thus the difference of the two coefficients is significant ($\chi^2 = 13.87$, $p = 0.0002$).

On a side-note, the coefficient estimates for the effect of description similarity on creation of reuse ties are significantly different for the two types of reuse — 6.269 for whitebox reuse versus 3.736 for blackbox reuse. Description similarity seems to be an important predictor of type of reuse beside coaffiliation. This provides additional support for H2a by confirming the local search argument put forward to justify the differential effect of coaffiliation ties on the two types of reuse. Indeed, project dyads with whitebox reuse ties seem to be semantically closer to each other as compared to the dyads with blackbox reuse ties.

Hypothesis 2b provides that despite the stronger effect of the existence of coaffiliations on whitebox reuse, stronger coaffiliations between projects must favour blackbox reuse. I put this proposition to test with Model 4 by entering both a binomial and a continuous (weighted) version of the coaffiliation variable in the regression so that we can delineate the effect of *existence* of coaffiliation from the effect of its *strength*. Adding the weighted coaffiliation variable to the equation renders the coefficients of the binary variable non-significant in the model with the control variables (the two variables are 0.94 correlated). But the Wald test shows that the difference between the coefficients of the binary coaffiliation remains highly significant in Model 4, thus the effect on whitebox reuse is still stronger ($\chi^2 = 8.98$, $p = 0.0027$). The difference of the coefficients for the weighted coaffiliation, a proxy for the strength of the tie, is significant at 0.01 level in the same model ($\chi^2 = 7.52$, $p = 0.0061$), denoting a stronger effect on blackbox reuse.

Although at the surface level the results seem to fully support Hypothesis 2b, the high degree of collinearity between the two coaffiliation variables puts the reliability of the results in doubt. In order to obtain additional assurance, I ran the same regression, keeping only the weighted coaffiliation variable and the subset of the data with coaffiliation links. The results still show a slightly larger coefficient for blackbox reuse, but the difference in the coefficients is far from significant (χ^2 = 0.1263, p = 0.7223). The mean of the coaffiliation weight for the project dyads with whitebox reuse ties is not significantly different from that of the dyads with blackbox reuse ties (t = 1.1788, p = 0.2394). Therefore, I conclude that the apparent support for H2b must be taken with caution.

As far as Hypotheses 3a and 3b are concerned, the relogit corrected estimates for the effect of license similarity on reuse are significant for both types of reuse (See tables 13 and 14 — Model 4). The effect is both stronger and more significant for whitebox reuse, what corresponds to the expectation, since the license restrictions mainly apply to the modification and the creation of derivative work, while the case of co-deployment or combined distribution is rarely targeted by the licenses. It is of note that technically every case of whitebox reuse is also a case of creating derivative work, while the same cannot be said for blackbox reuse.

Coaffiliation. Hypotheses 4a to 6 explore the drivers of developer coaffiliation between the open source projects. The results of the corresponding analyses are included in tables 16, 17 and 18. The content of table 18 is enough to follow the analysis below, but for the sake of completeness I have included the similar estimations in independent regressions for the two types of reuse in tables 16 and 17.

Hypothesis 4a and 4b suggested that the existence of knowledge reuse ties between projects, whether of blackbox or whitebox type, would have positive effect on the subsequent creation of coaffiliations between the projects. Hypothesis 5 suggested that this effect would be stronger for blackbox reuse ties. The results support the two core hypotheses unambiguously, while lending only mixed support to H5.

All else equal, the relogit adjusted estimates indicate a project that draws on the functionalities

	Dependent variable:				
	Blackbox Reuse				
	logistic			*rare events logistic*	
	(1)	(2)	(3)	(4)	Odds
Coaffiliation (binary)	4.993***	4.800***	4.356***	2.417***	11.214
	(0.071)	(0.074)	(0.099)	(0.183)	(2.048)
License Similarity		0.313***	0.144***	0.117*	1.124
		(0.037)	(0.041)	(0.050)	(0.056)
Description Similarity		3.180***	3.391***	3.384***	29.484
		(0.123)	(0.146)	(0.190)	(5.592)
Project Age			−0.165***	−0.147***	0.864
			(0.015)	(0.021)	(0.018)
Project Size			−0.068***	−0.083***	0.921
			(0.016)	(0.022)	(0.020)
Project Activity			0.143***	0.148***	1.159
			(0.012)	(0.014)	(0.017)
Project Indegree			1.220***	1.136***	3.115
			(0.014)	(0.017)	(0.052)
Constant	−5.823***	−6.027***	−7.220***	−10.592***	0.00003
	(0.019)	(0.028)	(0.041)	(0.040)	(0.00000)
Observations	948,705	948,705	948,705	948,705	948,705
Log Likelihood	−19,709.340	−19,436.440	−12,139.180	−87.250	−87.250
Akaike Inf. Crit.	39,422.680	38,880.870	24,294.370	190.500	190.500
Note:					*p<0.05; **p<0.01; ***p<0.001
					All variables are mean-centered.
					All node-level variables pertain to the reused project.
					Relogit uses robust standard errors.

Table 13. Logit and relogit regressions with blackbox reuse as dependent variable (H1a & H3a)

	Dependent variable:				
	Whitebox Reuse				
	logistic			*rare events logistic*	
	(1)	(2)	(3)	(4)	Odds
Coaffiliation (binary)	5.802***	5.409***	4.947***	4.354***	77.782
	(0.104)	(0.127)	(0.140)	(0.252)	(19.634)
License Similarity		0.531***	0.555***	0.518***	1.678
		(0.082)	(0.083)	(0.088)	(0.148)
Description Similarity		6.413***	6.382***	6.085***	439.243
		(0.133)	(0.138)	(0.208)	(91.277)
Project Age			−0.687***	−0.761***	0.467
			(0.048)	(0.087)	(0.040)
Project Size			0.282***	0.297***	1.346
			(0.021)	(0.021)	(0.028)
Project Activity			0.140***	0.110**	1.116
			(0.024)	(0.034)	(0.038)
Project Indegree			0.431***	0.441***	1.554
			(0.044)	(0.067)	(0.104)
Constant	−6.898***	−7.569***	−8.000***	−12.408***	0.00000
	(0.041)	(0.070)	(0.080)	(0.097)	(0.00000)
Observations	586,538	586,538	586,538	586,538	586,538
Log Likelihood	−4,998.480	−4,286.706	−3,995.876	−8.239	−8.239
Akaike Inf. Crit.	10,000.960	8,581.412	8,007.752	32.479	32.479
Note:					*p<0.05; **p<0.01; ***p<0.001
					All variables are mean-centered.
					All node-level variables pertain to the reused project.
					Relogit uses robust standard errors.

Table 14. Logit and relogit regressions with whitebox reuse as dependent variable (H1b & H3b)

	Dependent variable:							
	Whitebox Reuse	Blackbox Reuse	Whitebox Reuse	Blackbox Reuse	Whitebox Reuse	Blackbox Reuse	Whitebox Reuse	Blackbox Reuse
	(1)		(2)		(3)		(4)	
Coaffiliation (binary)	5.812***	4.997***	2.894***	2.426***	4.781***	4.265***	0.884*	−0.305
	(0.099)	(0.070)	(0.303)	(0.220)	(0.125)	(0.096)	(0.391)	(0.265)
Coaffiliation (weighted)			0.940***	0.848***			1.241***	1.511***
			(0.081)	(0.064)			(0.105)	(0.078)
License Similarity					0.573***	0.144***	0.552***	0.134**
					(0.082)	(0.041)	(0.082)	(0.041)
Description Similarity					6.199***	3.747***	6.269***	3.736***
					(0.131)	(0.137)	(0.130)	(0.139)
Project Age					−0.575***	−0.160***	−0.568***	−0.156***
					(0.044)	(0.015)	(0.044)	(0.015)
Project Size					0.279***	−0.072***	0.277***	−0.075***
					(0.021)	(0.016)	(0.021)	(0.016)
Project Activity					0.135***	0.138***	0.125***	0.132***
					(0.023)	(0.012)	(0.023)	(0.012)
Project Indegree					0.483***	1.207***	0.513***	1.224***
					(0.041)	(0.014)	(0.041)	(0.014)
Constant	−7.521***	−5.967***	−7.518***	−5.964***	−8.558***	−7.347***	−8.551***	−7.370***
	(0.041)	(0.019)	(0.041)	(0.019)	(0.077)	(0.041)	(0.077)	(0.041)
Observations	1,095,893	1,095,893	1,095,893	1,095,893	1,095,893	1,095,893	1,095,893	1,095,893
Log Likelihood	-25,640.033	-25,640.033	-25,492.100	-25,492.100	-17,128.946	-17,128.946	-16,883.746	-16,883.746
Akaike Inf. Crit.	51,288.070	51,288.070	50,996.200	50,996.200	34,289.890	34,289.890	33,803.490	33,803.490

Note: *p<0.05; **p<0.01; ***p<0.001
All variables are mean-centered.
All node-level variables pertain to the reused project.

Table 15. Multinomial logistic regressions with reuse type as output variable (H2a & H2b)

of another is 40 times more likely to subsequently share a developer with the same project, compared to 70 times according to the unadjusted logit estimates (See Table 18 — Model 4). This corresponds to a predicted probability of 0.33% for coaffiliation between project dyads with functional reuse ties versus 0.008% for project dyads with no prior reuse ties. The unadjusted probability figures are 8.62% and 0.13% respectively. The odds stand at 17 (49 unadjusted) for two projects that share code. Having prior clone reuse ties brings up the predicted probability of coaffiliation to 0.14% from the baseline of 0.008%.

Comparing the size of the coefficients for the two types of reuse confirms the claim put forward in Hypothesis 5; that blackbox reuse must have a stronger effect on coaffiliation than whitebox reuse (Table 18 — Model 4). But in order to verify the significance of this difference, a Wald test of the equality of estimates is due. The result of the test can not reject the null hypothesis, which is equality of the two coefficients ($\chi^2 = 2.03$, $p = 0.1543$). This is mainly due to the wide confidence intervals for the estimates of the very sparse whitebox reuse variable.

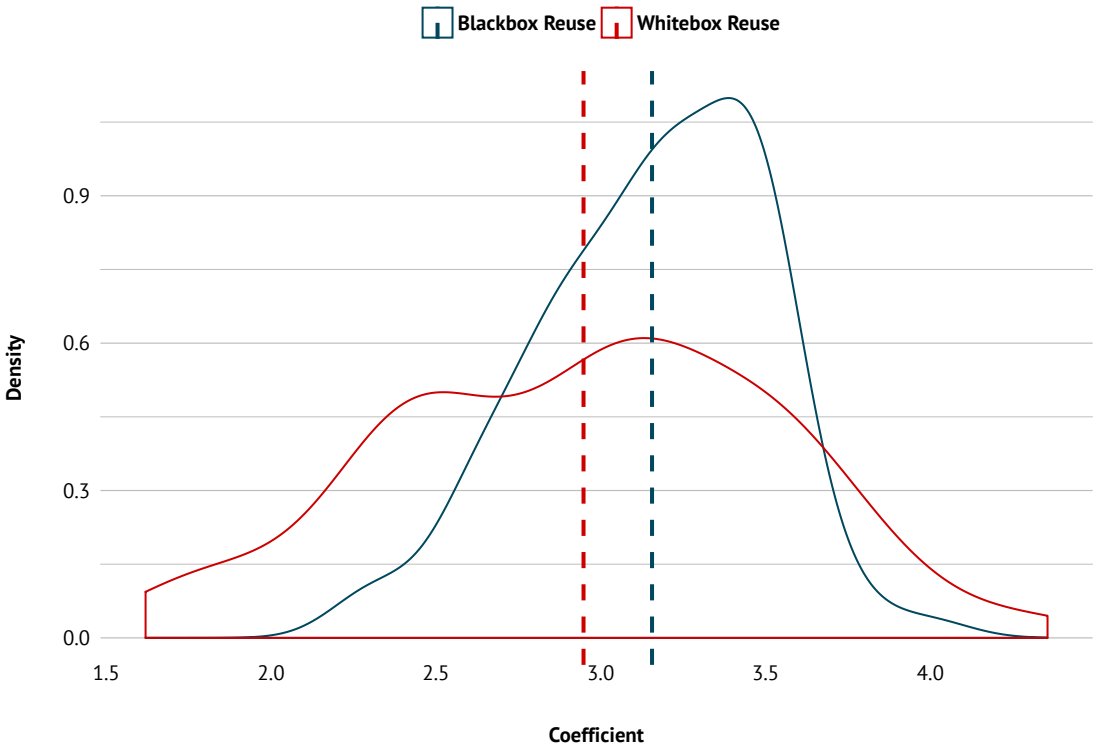


Figure 13. Comparative density plot for β coefficients

But given that acquiring new samples for this test comes at no cost, there is no need to remain confined within the limitations of this specific sample. Therefore, I drew 100 samples with similar specifications from the data, put them through the same regressions, and recorded the corrected relogit coefficient estimates for the two variables of interest — whitebox reuse and blackbox reuse. Welch's unequal variance t-test indicates that the mean of coefficient estimates for blackbox reuse ($\bar{x} = 3.16, s = 0.34$) is larger than that of whitebox reuse ($\bar{x} = 2.95, s = 0.59$), rejecting the null hypothesis of equality at a high level of significance ($t = 3.04$, $p = 0.0014$). The difference between the two means indicates that the odds of blackbox reuse preceding coaffiliation is 1.2 times that of whitebox reuse (See Figure 13).

Finally, Hypothesis 6 posits that projects with similar licenses are more likely to share developers. The variable *License Similarity* is significant in this regression, indicating that the projects with similar licenses are 1.6 times more likely to have developers in common (See Table 18 — Model 4). The results, therefore, support the final hypothesis.

	Dependent variable:				
	Coaffiliation (binary)				
	logistic			*rare events logistic*	
	(1)	(2)	(3)	(4)	Odds
Blackbox Reuse	4.654***	4.453***	4.378***	3.817***	45.453
	(0.083)	(0.085)	(0.102)	(0.170)	(7.733)
License Similarity		0.671***	0.480***	0.485***	1.624
		(0.016)	(0.017)	(0.018)	(0.029)
Description Similarity		2.349***	2.106***	2.045***	7.729
		(0.062)	(0.072)	(0.079)	(0.610)
Project Age			0.757***	0.838***	2.312
			(0.009)	(0.012)	(0.027)
Project Size			0.011	0.003	1.003
			(0.006)	(0.006)	(0.006)
Project Activity			0.203***	0.225***	1.252
			(0.006)	(0.006)	(0.008)
Project Contributors			0.938***	0.886***	2.427
			(0.009)	(0.009)	(0.023)
Constant	−5.498***	−5.892***	−6.616***	−9.418***	0.0001
	(0.008)	(0.013)	(0.015)	(0.016)	(0.00000)
Observations	4,239,665	4,239,665	4,239,665	4,239,665	4,239,665
Log Likelihood	−112,837.700	−111,345.800	−83,432.320	−1,064.139	−1,064.139
Akaike Inf. Crit.	225,679.500	222,699.600	166,880.600	2,144.277	2,144.277
Note:					*p<0.05; **p<0.01; ***p<0.001
					All variables are mean-centered.
					All node-level variables pertain to the destination project.
					Relogit uses robust standard errors.

Table 16. Logit and relogit regressions with coaffiliation as dependent variable (H4a & H6)

	Dependent variable:				
	Coaffiliation (binary)				
	logistic			*rare events logistic*	
	(1)	(2)	(3)	(4)	Odds
Whitebox Reuse	5.517***	5.013***	4.522***	3.414***	30.398
	(0.165)	(0.175)	(0.215)	(0.553)	(16.799)
License Similarity		0.671***	0.481***	0.491***	1.634
		(0.016)	(0.017)	(0.018)	(0.029)
Description Similarity		2.352***	2.112***	2.051***	7.779
		(0.062)	(0.072)	(0.079)	(0.617)
Project Age			0.757***	0.838***	2.312
			(0.009)	(0.012)	(0.027)
Project Size			0.011	0.002	1.002
			(0.006)	(0.006)	(0.006)
Project Activity			0.205***	0.225***	1.252
			(0.006)	(0.006)	(0.008)
Project Contributors			0.935***	0.886***	2.426
			(0.009)	(0.009)	(0.023)
Constant	−5.490***	−5.885***	−6.607***	−9.413***	0.0001
	(0.008)	(0.013)	(0.015)	(0.016)	(0.00000)
Observations	4,239,665	4,239,665	4,239,665	4,239,665	4,239,665
Log Likelihood	−113,257.300	−111,762.700	−83,812.360	−1,065.976	−1,065.976
Akaike Inf. Crit.	226,518.600	223,533.400	167,640.700	2,147.952	2,147.952
Note:					*p<0.05; **p<0.01; ***p<0.001
					All variables are mean-centered.
					All node-level variables pertain to the destination project.
					Relogit uses robust standard errors.

Table 17. Logit and relogit regressions with coaffiliation as dependent variable (H4b & H6)

	Dependent variable:				
	Coaffiliation (binary)				
	logistic			*rare events logistic*	
	(1)	(2)	(3)	(4)	Odds
Blackbox Reuse	4.533***	4.341***	4.255***	3.701***	40.473
	(0.087)	(0.089)	(0.106)	(0.176)	(7.128)
Whitebox Reuse	5.135***	4.578***	3.900***	2.834***	17.017
	(0.182)	(0.195)	(0.237)	(0.550)	(9.353)
License Similarity		0.668***	0.478***	0.485***	1.625
		(0.016)	(0.017)	(0.018)	(0.029)
Description Similarity		2.292***	2.059***	2.027***	7.594
		(0.063)	(0.073)	(0.079)	(0.602)
Project Age			0.756***	0.835***	2.305
			(0.009)	(0.012)	(0.027)
Project Size			0.009	0.002	1.002
			(0.006)	(0.006)	(0.006)
Project Activity			0.204***	0.224***	1.251
			(0.006)	(0.006)	(0.008)
Project Contributors			0.938***	0.888***	2.429
			(0.009)	(0.009)	(0.023)
Constant	−5.500***	−5.892***	−6.616***	−9.417***	0.0001
	(0.008)	(0.013)	(0.015)	(0.016)	(0.00000)
Observations	4,239,665	4,239,665	4,239,665	4,239,665	4,239,665
Log Likelihood	−112,614.600	−111,162.800	−83,321.630	−1,065.358	−1,065.358
Akaike Inf. Crit.	225,235.300	222,335.500	166,661.300	2,148.717	2,148.717
Note:					*p<0.05; **p<0.01; ***p<0.001
					All variables are mean-centered.
					All node-level variables pertain to the destination project.
					Relogit uses robust standard errors.

Table 18. Logit and relogit regressions with coaffiliation as dependent variable (H5)

Discussion

This thesis was motivated by two specific gaps in the literature. First, the extant body of literature on knowledge perspective has predominantly examined the effect of social capital on the creation of intellectual capital, with limited attention to the reverse effect of the development of the stock of intellectual capital on social capital. Second, the large majority of studies on knowledge networks have focused on formal organizations. To the best of my knowledge, no theory-driven study has yet looked into the antecedents and the outcomes of knowledge reuse in open innovation communities.

In addressing these gaps, I examined the interplay between the mobility of developers across projects, or project-developer coaffiliation, and knowledge reuse practices of projects in the Ruby community. The results are generally supportive of the hypotheses. I discuss the findings of the study further in the following section. The findings addressing the first gap have been organized under the section "An Interactive View of Knowledge in Networks". Those addressing the second gap are presented under the section "Communities as Spaces for Knowledge Collaboration". Finally, in the section entitled "Communities as Adaptive Networks", I seek to link the findings to another body of research that has recently shown interest in the qualities of social networks.

An Interactive View of Knowledge in Networks

Since the seminal work by Jaffe et al. (1993), researchers have used patent data to track knowledge flows and study how they are affected by social and geographical distance. Two sorts of relationship between social ties and knowledge reuse can be envisaged: (1) Existing social ties may facilitate subsequent reuse of the knowledge spawned across those specific ties, as compared to the otherwise readily available knowledge stocks; (2) On the other hand, reuse of knowledge artefacts originated elsewhere may signal the potential or prepare the context for mutually beneficial social ties, and eventually lead to direct contact between originating and reusing entities..

The extant theories lead us to expect to observe a co-evolutionary relationship between social ties and knowledge reuse whereby each is sustained by the other. Nahapiet and Ghoshal (1998), for instance, recognize that intellectual capital may as well facilitate the creation of social capital, and that

the evolution of these two forms of capital may lead to organizational advantage, notably facilitating the creation of economic capital. Recent simulations have also demonstrated the possibility that the interactions between knowledge exchange and social exchange follow co-evolutionary dynamics (Luo, Du, Liu, Xuan, & Wang, 2015). Yet, the vast majority of the extant literature on knowledge networks has given primacy to social ties, designating them as one of the various determinants of knowledge reuse and recombination (Phelps et al., 2012).

This thesis brings strong evidence that, at least in a context where the reuse possibilities are not curtailed by restrictive intellectual property terms, there is a two-sided relationship between the social network of collaboration and the network of knowledge reuse. The effects persist after controlling for several important variables, including one marking the similarity of licenses. This suggests a dialectical relationship between the two forms of capital, a virtuous generative cycle in which social capital and intellectual capital feed into each other. The findings associate the relational facet of social capital with new possibilities for creation of intellectual capital through reuse, as well as establishing reuse of intellectual capital as a factor affecting subsequent creation of social capital.

The results also include evidence showing that compatibility in the terms of licenses is a significant predictor of reuse — that is, restrictive measures in intellectual property protection that may lead to incompatibilities in terms of reuse bear a negative effect on reuse. This corroborates the findings of the recent empirical research on copyright restrictions (Nagaraj, 2017). The results also show that, apart from restricting reuse, differences in licensing terms compartmentalize social spheres along legal fault lines and weaken collaboration ties in-between spheres corroborating previous findings by (J. Singh, 2005).

It is a well-established fact and an assumption of many sociological and economic theories that better information circulation leads to the emergence of the trust needed for collaborative activities (Fisman & Khanna, 1999). What the combination of the results above additionally shows is that barriers to knowledge transfer can affect collaboration through two different mechanisms: (1) Directly, and through the carry-over effect identified by J. Singh (2005), given that individuals tend to attach themselves to social and organizational regimes they already know about; (2) Indirectly,

and by reducing the chances of knowledge reuse.

Another implication of these findings is that restrictive intellectual property protection terms can potentially limit or outright hinder the generative cycle of interaction between the two forms of capital. The knowledge perspective has epitomized knowledge as the most important and strategic factor of production (Brown & Duguid, 2001; Nahapiet & Ghoshal, 1998; Nonaka, 1994; Spender, 1996). Any social system interested in its own welfare must be interested in promoting knowledge creation and knowledge collaboration processes. Thus, it is safe to assume that a social system would want to enhance the generative interaction between social capital and intellectual capital, and allow the mutual reinforcement mechanism run its course on its own inertia. The role of intellectual property protection regimes is, therefore, to regulate and balance the conversion rate between the two types of capital when needed — for instance in case of excess reuse as compared to the degree of contribution. This reopens an old debate about the origins of intellectual property protection by raising questions about the restrictions imposed by these regimes.

Specific to the context of software development is the possibility of measuring reuse in two distinct ways, representing shallow and deep knowledge reuse. The two measures of knowledge reuse used in this thesis were initially included as much for theoretical as redundancy reasons. While both measures broadly point towards the same concept, they measure slightly different reuse behaviours that by extension may have partially different antecedents and outcomes. The difference in the size of the effects of coaffiliation on the two types of reuse testifies the importance of relying on more than one measure of knowledge reuse in research.

For instance, one minor but interesting finding of this study is that the above conversion rate is significantly different for the two types of reuse. Strong collaboration ties in the form of developer coaffiliation across projects were found to be more strongly related to whitebox code reuse, which constitutes a deeper integration of code and requires a better mastery of the reuse target. The results partially mirror the previous findings of the literature on knowledge networks (e.g., Hansen, 1999). Yielding a similar result as the previous studies validates once more the comparability of the concepts across contexts and also credibility to the chosen proxy measures (i.e whitebox and blackbox code

reuse).

Conversely, the results also suggest that blackbox reuse is more likely to lead to future social ties, as it effectively entangles the future development of the reuser and the reused. In other words, the shallower type of knowledge reuse more significantly drives the collaborative ties between projects. The dual nature of code, harbouring both functionality and expression, is of note here. The reusers may be primarily motivated by the functionality of the reused code, while they graduate as potential collaborators as they deepen their understanding of the reused code to the point of contributing to it. The generative cycle of reuse and collaboration, thus, would not be as readily attainable without the possibility of shallow reuse.

Communities as Spaces for Knowledge Collaboration

A second objective of this thesis was to shed light on the knowledge collaboration and reuse practices in open innovation communities. There is no shortage of editorials in research outlets calling for the study of this new organizational fabric, including the knowledge flows and the dynamics of collaboration as they occur in this context (Faraj et al., 2011; Faraj et al., 2016; Zammuto, Griffith, Majchrzak, Dougherty, & Faraj, 2007). Open intellectual property regime, communal normative control, and the tendency to self-assign tasks and organically assemble with related others have been pointed out as factors that can potentially shape the way knowledge collaboration unfolds in the communities (Benkler, 2002; Faraj et al., 2016; Lerner & Tirole, 2002).

In response to those calls, one set of this study's hypotheses sought to confirm in the community context some well-known patterns of reuse and collaboration that the previous studies in knowledge networks have shown to hold in or in-between formal organizations. Another set of hypotheses tested new and potentially context-specific propositions.

The first finding of this thesis with regard to the reuse practices in the communities points to a similarity with the reuse practices previously studied in other contexts. Social ties have often been characterized as conduits for knowledge, minimizing the frictions and the costs associated with knowledge acquisition and the subsequent reuse. This indicates that the localization of knowledge,

the condition in which knowledge remains constrained by social distance and embedded within social networks, continues to hold in communities.

Open innovation communities, however, operate based on open access principles and do not impose explicit charges for knowledge transfer. The localization of knowledge in open communities, thus, occurs despite the absence of the explicit knowledge acquisition and reuse costs. This can be explained by the fact that only a part of the reuse costs are explicit and expressed in monetary terms, while many implicit search and processing costs associated with knowledge acquisition and reuse remain in place in open contexts. Search costs may even be exacerbated in an open context where content generation and diffusion outpace the community's capacity to structure and quality-control the content, leading to a sustainable state of information overload. Therefore, relational social capital remains an important factor for mitigating such costs.

The results also show that the effect of social ties on reuse is significant, over and above the effect of license-related restrictions. In other terms, there are factors beyond context-related restrictions that lead to localization of knowledge. For all the freedom of action that the open licenses provide to the potential reusers, they don't seem to completely detach the invisible string between the creators and their work, nor do they undermine the centrality of the creators in the reuse process. Creators, as well as their close ties in the network of collaboration, remain the ones who resort the most to the reuse of the created knowledge artefacts. That is testimony to the laboriousness of the knowledge transfer and reuse process even in absence of explicit barriers, downplaying the importance of legal mechanisms in protecting intellectual capital.

What sets the social conditions of reuse in open innovation communities apart, I argue, is the way reuse and collaboration processes reciprocate and feed into each other in a manner that may not be readily possible in other contexts. That is, in this context, not only knowledge networks are socially embedded, but the social networks are as well structured by the network of knowledge reuse. This is a possibility hinted at in simulation studies of knowledge networks (Luo et al., 2015), pending demonstration in empirical research. The evidence provided in this thesis is clear. It hints at a generative cycle and an interactive relationship between social and intellectual capital in open

innovation communities, potentially generating the drive towards cumulative innovation. The findings demonstrate that, in open innovation communities, not only those who reuse contribute, but also those who contribute reuse.

In his seminal work on user innovation, von Hippel (1994) suggests that the users are best placed to innovate as their knowledge of their problems is situated and sticky, and thus costly to transfer. This is the type of knowledge that Garud (1997) has called know-what, or the knowledge of what is worth pursuing, often an exchange token between the creators and the users. On the same line, Benkler (2002) has argued that markets and hierarchies suffer from a type of information loss in decision processes that the open innovation communities mitigate by allowing individuals to self-select into tasks based on their own understanding of their expertise and the impact of their participation.

The main observation of this study, that there is a narrow alignment of the acts of collective creation and reuse, pushes that line of argument one step further towards an empirical validation of the processes underlying this view. Such an alignment has the potential to minimize the cost of knowledge transfer between the creators and the reusers. Given the high degree of permeability of the boundaries of open projects as compared to the other varieties of collaborative structures (e.g., formal organizations), the reciprocal relationship between collaboration and reuse may be a characteristic feature of open communities, or at least one that is primarily visible in this context.

Communities as Adaptive Networks

Knowledge networks can ultimately be characterized as social networks, as there cannot be any knowledge network — or any knowledge for that matter — in the absence of human actors. But knowledge networks and collaboration networks each represent the coordinates, as well as the movements, of a different type of socially-acquired resource. This study makes a genuine and original contribution to the literature on knowledge networks through its main outcome, demonstrating that the relationship between two layers of a multiplex social network (i.e., collaboration and reuse), as well as the relationship between the stock of capital that each carries, are reciprocal and not

unidirectional. Yet, this finding can not be considered unexpected or surprising.

Several other studies have demonstrated the interactive nature of the relationships between different layers of multiplex social networks (e.g., Gould, 1991; Lazega, 2001; Padgett & Ansell, 1993). Putnam (1993) observes that social capital tends to be self-reinforcing, cumulative, and often transferable from one context to the other. I argue that the study of the relationship between the network of social relations and the network of knowledge exchange or reuse can immensely benefit from pursuing an interactive view. The advantage of adopting the interactive view as the baseline is by no means merely methodological, but rather one that gives access to novel ways of theorizing the phenomenon.

The self-organizing nature of online communities, their dynamic resource allocation and their generative responses to innate tensions are sources of questions for the scholars (Faraj et al., 2011). One plausible explanation for the organizational continuity and the social sustenance of community forms is the deliberate interventions of human agency in the form of organized action, governance practices and leadership (e.g., as shown in Faraj et al., 2015; O'Mahony & Ferraro, 2007). The outcomes of this study point at a second plausible path for explaining the almost serendipitous success of the online communities in allocation of resources and organization of activities, through an adaptive framework.

Adaptive or co-evolutionary networks are networks that exhibit a mutual interaction between their local state dynamics and their network-level topological changes (Gross & Blasius, 2008), often leading to a highly robust global self-organization (Bornholdt & Röhl, 2003; Bornholdt & Rohlf, 2000). Viewing social networks as adaptive networks is a relatively new research direction in social network studies. The findings of this thesis provide anecdotal evidence about the adaptive nature of social networks in open innovation communities. Empirically demonstrating the reciprocal relationship between the network of knowledge reuse and the social network of collaboration is a modest first step towards establishing online communities as adaptive social networks. For instance, different conversion rates between these two networks would potentially lead to different global organizations in the online community. Slowing down one of the two sides of this mutual relationship

would break the generative cycle between them by under-supplying one resource or the other. Such situations would come across as what has been called ebb and flow of resources (Faraj et al., 2011) in communities, when observed from a bird's eye view.

Implications

This thesis started with the promise of studying knowledge collaboration and reuse in open innovation communities, and used a variety of methods imported from computer science and political science to test two groups of hypotheses about the relationship between networks of reuse and collaboration. The results demonstrated strong support for all of the core hypotheses and most of the peripheral hypotheses, while three of them were only weekly supported. The findings establish that the relationship between knowledge reuse and collaboration is reciprocal — that is, the reuse ties between projects are highly likely to bring about new collaboration ties in form of developer-project coaffiliations, and the existence of project coaffiliation ties is likely to lead to reuse ties between the focal projects.

In the following sections I first describe the contributions and the limitations of this study. I proceed to briefly discuss the generalizability of the findings. Finally, I outline some possible future directions in this line of research.

Contributions

Methodological Contributions. This study made novel use of several measures and analysis tools, what can be considered a methodological contribution to the field.

The extant research on knowledge transfer and reuse has mainly focused on self-reported measures (e.g., advice networks), as well as patent citation data, to trace knowledge flows. Self-reported data are by nature prone to bias, and the patent citation data has often been criticized for its flaws (Alcácer & Gittelman, 2006).The limitations of these measures, specially the limitations in their context of application, show the importance of adding new measures for knowledge reuse to the toolbox.

The research on knowledge exchange and diffusion in online communities has often used logs of the email conversations, as well as newsgroup and forum posts, as evidence for knowledge flows. While communication records constitute valid measures for knowledge exchange, they are not particularly adapted for measuring reuse.

The use of whitebox and blackbox code reuse indicators as measures of knowledge reuse adopted in this study constitutes a first successful implementation of its kind. The blackbox reuse measure relies on function call information from each software project's metadata. The whitebox reuse measure, on the other hand, was calculated using a purpose-built clone detection tool based on the SpamSum hashing algorithm. The tool is composed of more than ten thousand lines of code and will be made public in the future.

The study also makes use of the commit history of software projects to identify project affiliations of the community members. This is a more accurate measure than the often-used project metadata, as it indicates the periods in which a developer has been truly active on a project, rather than counting on the veracity of official membership records.

There are two main advantages associated with the above measures. First, they make possible the study of knowledge reuse in software projects in general, and in online communities focused on software development in particular. Second, they all refer to the work outcomes and the real work-related activities of the actors, rather than relying on self-reported measures or communication records. This is ideal because knowledge reuse is best measured in the act of reusing, not in what leads up to it.

Theoretical Contributions. To the best of my knowledge, no study has yet tried to jointly examine social networks and knowledge networks in the online community context. While various studies have covered one or the other, this is likely the first study to consider both at the same time. Moreover, and regardless of the context, no study has been able to establish the reciprocal effect of the two networks on each other. Considered separately, most of the evidence supports the effect of social networks on knowledge networks; only a few of studies have tried to demonstrate the effect of the knowledge networks on social networks, and none has focused on the context of online communities.

The main theoretical contribution of this study is to show the co-evolutionary nature of the relationship between intellectual capital and relational social capital in open innovation communities. More precisely, it demonstrated the reciprocal effect of collaboration ties and knowledge reuse ties on

each other within the context of open innovation communities. This can be considered a contribution both to the literature on knowledge networks and the literature on open innovation communities.

Moreover, this study contributes to the literature on open innovation communities by showing the differential effect of the collaboration ties on the two types of reuse and the differential effect of the two types of reuse on the collaboration ties. Additionally, the study shows that the software license differences between two focal projects can directly and negatively affect the likelihood of collaboration and knowledge reuse between projects. These findings shed light on the social and work processes under the community forms.

Limitations

The main limitation of this study is that it is uniquely focused on testing and validating the probable outcomes of the mechanisms it investigates, but not the mechanisms per se.

The other limitations of this study are of methodological order:

1. The current license similarity measure is simply a boolean flag indicating whether the license and the license group used by a project in a dyad is the same as that of the other project in the dyad. It would be more accurate to construct this measure by comparing the actual provisions of the licenses.

2. Despite the sizeable dataset used in this study, the networks used to generate the network-based measures remain extremely sparse. This reduces the stability of the regression analyses even when using the appropriate penalized models, and therefore increases the size of the standard errors, preventing a refined analysis of the coefficient estimates and the effect sizes. This alone explains why the results from Hypotheses 2b are not significant enough. It would be desirable to either find larger samples, or to devise methods that prevent the loss of the incomplete data records, such as triangulating the data from different sources.

3. Finally, for lack of a better choice, a simple multinomial logic was used to test H5. However, in such a sparse dataset a penalized multinomial logistic model or a zero-inflated multinomial

model would be the method of choice. Although there have been some efforts to implement such models (e.g., B. E. Bagozzi, 2015), by the time of this writing there was no publicly available implementation to use.

Future Research

One important factor that has contributed to the long-held one-sided view of the interplay between social networks and knowledge networks is the availability of data. Many studies of knowledge networks have relied on sampling and data gathering based on their dependent variable (Rosenkopf & Almeida, 2003, e.g., all patent-holding firms in the semi-conductor industry founded in a ten-year period). Therefore, switching the dependent variable would essentially cost them a second round of data gathering. As such, the reverse relationship between knowledge networks and social networks remains largely unexplored.

One way to remove the necessity for re-sampling based on research questions is to obtain a whole-network sample. A whole-network sample consists of complete network and node information on a reasonably self-contained bounded population. Despite obvious merits of a whole-network sampling strategy, including the possibility to distinguish between local versus whole-network influences, relatively few studies of knowledge networks have relied on whole-network samples (Phelps et al., 2012).

In order to better tackle the type of question posed in this study, the future research on knowledge networks needs to move towards whole-network samples, specially given the wide availability of digital trace data. This is the only way the main finding of the current study can be replicated in other contexts. Having evidence from more than one context will allow for comparisons between the effect sizes across contexts, with important policy repercussions.

For all the cautionary tales about the perils of collective action and tragedy of the commons occurring in the community forms, it does not seem near-enough attention has been paid to the contrary issues of information loss and the tragedy of anti-commons in markets and hierarchical organizations (Kogut & Metiu, 2001). More research is needed to find out whether the same generative cycle between knowledge reuse and collaboration occurs in contexts other than communities,

and under which conditions. But at this stage one can already conclude that the contexts in which such interactive social processes can not take on, not only impede the innovation processes, but also possibly hinder effective collaboration.

Finally, the literature on online communities and open innovation communities, on the other hand, needs to consider more carefully the possibility of researching adaptive social processes in the context of online communities. This study only found anecdotal evidence of such processes at work. Deeper investigation of the adaptive processes will not only require whole-network samples, but also multiplex network data. Special attention must be paid to the relations between the different layers and the occurrence of certain local dynamics in conjunction with global equilibriums to unveil the adaptive processes.

Bibliography

About The Licenses - Creative Commons. (2013). Version 4. Creative Commons. Retrieved from https://creativecommons.org/licenses

Abrahamson, E. & Rosenkopf, L. (1997). Social network effects on the extent of innovation diffusion: A computer simulation. *Organization Science*, *8*(3), 289–309.

Acedo, F. J., Barroso, C., Casanueva, C., & Galán, J. L. (2006). Co-Authorship in management and organizational studies: An empirical and network analysis*. *Journal of Management Studies*, *43*(5), 957–983.

Adler, P. S. & Kwon, S.-W. (2002). Social capital: Prospects for a new concept. *Academy of Management Review*, *27*(1), 17–40.

Adler, P. S., Kwon, S.-W., & Heckscher, C. (2008). Perspective—Professional work: The emergence of collaborative community. *Organization Science*, *19*(2), 359–376.

Akgün, A. E., Byrne, J., Keskin, H., Lynn, G. S., & Imamoglu, S. Z. (2005). Knowledge networks in new product development projects: A transactive memory perspective. *Information & Management*, *42*(8), 1105–1120.

Albert, R. & Barabási, A.-L. (2001). Statistical mechanics of complex networks. *Reviews of Modern Physics*, *74*(1), 47–97.

Alcácer, J. & Gittelman, M. (2006). Patent Citations as a Measure of Knowledge Flows: The Influence of Examiner Citations. *The Review of Economics and Statistics*, *88*(4), 774–779.

Allman, E. (2012). Managing technical debt. *Communications of the ACM*, *55*(5), 50–55.

Almeida, P. & Kogut, B. (1999). Localization of knowledge and the mobility of engineers in regional networks. *Management Science*, *45*(7), 905–917.

Aral, S. (2016). The future of weak ties. *American Journal of Sociology*, *121*(6), 1931–1939.

Aral, S. & Van Alstyne, M. (2011). The Diversity-Bandwidth trade-off. *American Journal of Sociology*, *117*(1), 90–171.

Argote, L. & Ingram, P. (2000). Knowledge transfer: A basis for competitive advantage in firms. *Organizational behavior and human decision processes*, *82*(1), 150–169.

Argyris, C. (1976). Single-Loop and Double-Loop models in research on decision making. *Administrative Science Quarterly*, *21*(3), 363–375.

Arrow, K. J. (1962). Economic welfare and the allocation of resources for invention. In *The rate and direction of inventive activity: Economic and social factors* (pp. 609–626). Princeton University Press.

Atkinson, A. B. (1970). On the Measurement of Inequality. *Journal of economic theory*, *2*(3), 244–263.

Austin, J. R. (2003). Transactive memory in organizational groups: The effects of content, consensus, specialization, and accuracy on group performance. *The Journal of Applied Psychology*, *88*(5), 866–878.

Bagozzi, B. E. (2015). The baseline-inflated multinomial logit model for international relations research. *Conflict Management and Peace Science*, *33*(2), 174–197.

Bagozzi, R. P. & Dholakia, U. M. (2006). Open source software user communities: A study of participation in linux user groups. *Management Science*, *52*(7), 1099–1115.

Baldwin, C. Y. & Clark, K. B. (2006). The architecture of participation: Does code architecture mitigate free riding in the open source development model? *Management Science*, *52*(7), 1116–1127.

Barabási, A.-L. & Albert, R. (1999). Emergence of Scaling in Random Networks. *Science*, *286*(5439), 509–512.

Barabási, A.-L., Jeong, H., Néda, Z., Ravasz, E., Schubert, A., & Vicsek, T. (2002). Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications*, *311*(3-4), 590–614.

Baxter, I. D., Yahin, A., Moura, L., Sant'Anna, M., & Bier, L. (1998). Clone detection using abstract syntax trees. In *Proceedings of International Conference on Software Maintenance* (pp. 368–377).

Bayer, R. & McCreight, E. M. (1972). Organization and maintenance of large ordered indexes. *Acta Informatica*, *1*(3), 173–189.

Bechky, B. A. (2003). Sharing meaning across occupational communities: The transformation of understanding on a production floor. *Organization Science*, *14*(3), 312–330.

Benbya, H. & Belbaly, N. (2010). Understanding developers' motives in open source projects: A multi-theoretical framework. *Communications of the AIS*, *2010*(27), 586–610.

Benkler, Y. (2002). Coase's Penguin, or, Linux and The Nature of the Firm. *The Yale Law Journal*, *112*(3), 369–446.

Bernstein v. US Dept. of Justice. (1999). Court of Appeals, 9th Circuit.

Blau, P. M. (1964). *Exchange and power in social life*. Wiley.

Boisot, M. H. (1998). *Knowledge assets: Securing competitive advantage in the information economy*. Oxford University Press.

Boissevain, J. (1974). *Friends of friends: Networks, manipulators and coalitions*. Basil Blackwell.

Bonaccorsi, A., Giannangeli, S., & Rossi, C. (2006). Entry strategies under competing standards: Hybrid business models in the open source software industry. *Management Science*, *52*(7), 1085–1098.

Borgatti, S. P. & Halgin, D. S. (2011). Analyzing affiliation networks. In J. Scott & P. J. Carrington (Eds.), *The SAGE handbook of social network analysis* (pp. 417–433). SAGE Publications.

Bornholdt, S. & Röhl, T. (2003). Self-organized critical neural networks. *Physical Review E*, *67*(6), 066118.

Bornholdt, S. & Rohlf, T. (2000). Topological Evolution of Dynamical Networks: Global Criticality from Local Dynamics. *Physical Review Letters*, *84*(26), 6114–6117.

Boudreau, K. J. (2010). Open platform strategies and innovation: Granting access vs. devolving control. *Management Science*, *56*(10), 1849–1872.

Boudreau, K. J. (2011). Let a thousand flowers bloom? an early look at large numbers of software app developers and patterns of innovation. *Organization Science*.

Bourdieu, P. (1986a). The Forms of Capital. In J. G. Richardson (Ed.), *Handbook of Theory and Research for the Sociology of Education* (Chap. 9). Greenwood Press.

Bourdieu, P. (1986b). The forms of capital. In J. G. Richardson (Ed.), *Handbook of theory and research for the sociology of education* (pp. 241–258). Greenwood Press.

Bouty, I. (2000). Interpersonal and interaction influences on informal resource exchanges between R&D researchers across organizational boundaries. *Academy of Management Journal*, *43*, 50–65.

Bradford, R. B. (2008). An Empirical Study of Required Dimensionality for Large-scale Latent Semantic Indexing Applications. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management* (pp. 153–162).

Bresman, H. (2010). External learning activities and team performance: A multimethod field study. *Organization Science*, *21*(1), 81–96.

Brown, J. S. & Duguid, P. (1991). Organizational learning and Communities-of-Practice: Toward a unified view of working, learning, and innovation. *Organization Science*, *2*(1), 40–57.

Brown, J. S. & Duguid, P. (2000). *The social life of information*. Harvard Business School Press.

Brown, J. S. & Duguid, P. (2001). Knowledge and organization: A Social-Practice perspective. *Organization Science*, *12*(2), 198–213.

Burt, R. S. (1987). Social contagion and innovation: Cohesion versus structural equivalence. *American Journal of Sociology*, *92*(6), 1287–1335.

Burt, R. S. (1992). *Structural holes: The social structure of competition*. Harvard University Press.

Burt, R. S. (2000). The network structure of social capital. *Research in Organizational Behavior*, *22*, 345–423.

Burt, R. S. (2001). Structural Holes vs. Network Closure as Social Capital. In Nan Lin, Karen S. Cook, Ronald S. Burt (Ed.), *Social Capital: Theory and Research* (pp. 31–55). Transaction Publishers.

Butler, B. S. (2001). Membership size, communication activity, and sustainability: A Resource-Based model of online social structures. *Information Systems Research*, *12*(4), 346–362.

Cappelli, P. & Sherer, P. D. (1991). The missing role of context in OB - the need for a meso-level approach. In B. M. Staw & L. L. Cummings (Eds.), *Research in organizational behavior* (Vol. 13, pp. 55–110). JAI Press.

Carlile, P. R. (2004). Transferring, translating, and transforming: An integrative framework for managing knowledge across boundaries. *Organization Science*, *15*(5), 555–568.

Centola, D. & Macy, M. (2007). Complex contagions and the weakness of long ties. *American Journal of Sociology*, *113*, 702–734.

Chesbrough, H. W. (2006). *Open innovation: The new imperative for creating and profiting from technology*. Harvard Business School Press.

Chesbrough, H. W. & Crowther, A. K. (2006). Beyond high tech: Early adopters of open innovation in other industries. *R&D Management*, *36*(3), 229–236.

Choirat, C., Gandrud, C., Honaker, J., Imai, K., King, G., & Lau, O. (2017). Rare Events Logistic. Accessed: 2017-8-3. Retrieved from http://docs.zeligproject.org/articles/zelig_relogit.html

Cohen, W. M. & Levinthal, D. A. (1990). Absorptive capacity: A new perspective on learning and innovation. *Administrative Science Quarterly*, *35*(1), 128–152.

Coleman, J. S. (1988). Social capital in the creation of human capital. *American Journal of Sociology*, *94*, 95–120.

Coleman, J. S., Katz, E., & Menzel, H. (1957). The diffusion of an innovation among physicians. *Sociometry*, *20*(4), 253–270.

Constant, D., Sproull, L. S., & Kiesler, S. (1996). The kindness of strangers: The usefulness of electronic weak ties for technical advice. *Organization Science*, *7*(2), 119–135.

Cross, R., Sproull, L. S., Constant, D., & Kiesler, S. (2004). More Than an Answer: Information Relationships for Actionable Knowledge. *Organization Science*, *15*(4), 446–462.

Dahlander, L. & O'Mahony, S. C. (2011). Progressing to the center: Coordinating project work. *Organization Science*, *22*(4), 961–979.

Das, T. K. & Teng, B.-S. (1998). Between Trust and Control: Developing Confidence in Partner Cooperation in Alliances. *Academy of Management Review*, *23*(3), 491–512.

Dasgupta, P. & David, P. M. (1994). Toward a new economics of science. *Research Policy*, *23*(5), 487–521.

Deerwester, S., Scott, D., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of The American Society For Information Science*, *41*(6), 391–407.

Demil, B. & Lecocq, X. (2006). Neither market nor hierarchy nor network: The emergence of bazaar governance. *Organization Studies*, *27*(10), 1447–1466.

Drucker, P. F. (1969). *The age of discontinuity: Guidelines to our changing economy*. Harper & Row.

Economides, N. & Katsamakas, E. (2006). Two-Sided competition of proprietary vs. open source technology platforms and the implications for the software industry. *Management Science*, *52*(7), 1057–1071.

Evangelopoulos, N., Zhang, X., & Prybutok, V. R. (2010). Latent semantic analysis: Five methodological recommendations. *European Journal of Information Systems*, *21*(1), 70–86.

Eveland, J. D. & Bikson, T. K. (1987). Evolving electronic communication networks: An empirical assessment. *Office Technology and People*, *3*(2), 103–128.

Fairtlough, G. (1994). *Creative compartments: A design for future organization*. Adamantine Press.

Faraj, S. & Azad, B. (2012). The materiality of technology: An affordance perspective. In P. M. Leonardi, B. A. Nardi, & J. Kallinikos (Eds.), *Materiality and organizing* (pp. 237–258). Oxford University Press.

Faraj, S., Jarvenpaa, S. L., & Majchrzak, A. (2011). Knowledge collaboration in online communities. *Organization Science*, *22*(5), 1224–1239.

Faraj, S. & Johnson, S. L. (2011). Network exchange patterns in online communities. *Organization Science*, *22*(6), 1464–1480.

Faraj, S., Kudaravalli, S., & Wasko, M. M. (2015). Leading collaboration in online communities. *MIS Quarterly*, *39*(2), 393–412.

Faraj, S., von Krogh, G., Monteiro, E., & Lakhani, K. R. (2016). Online Community as Space for Knowledge Flows. *Information Systems Research*.

Feldman, K. A. & Newcomb, T. M. (1969). *The impact of college on students*. Transaction Publishers.

Finholt, T. A., Sproull, L., & Kiesler, S. (2002). Outsiders on the Inside: Sharing Know-How Across Space and Time. In P. J. Hinds & S. Kiesler (Eds.), *Distributed Work* (Chap. 14, pp. 335–356). MIT Press.

Finholt, T. & Sproull, L. S. (1990). Electronic groups at work. *Organization Science*, *1*(1), 41–64.

Fisman, R. & Khanna, T. (1999). Is trust a historical residue? Information flows and trust levels. *Journal of economic behavior & organization*, *38*(1), 79–92.

Flath, C. M., Friesike, S., Wirth, M., & Thiesse, F. (2017). Copy, transform, combine: exploring the remix as a form of innovation. *Journal of Information Technology Impact*, 1–20.

Fleming, L. (2001). Recombinant uncertainty in technological search. *Management Science*, *47*(1), 117–132.

Fleming, L. & Waguespack, D. M. (2007). Brokerage, boundary spanning, and leadership in open innovation communities. *Organization Science*, *18*(2), 165–180.

Ford, P. (2015). What is code? *Bloomberg Businessweek*. Retrieved March 15, 2016, from http://www.bloomberg.com/graphics/2015-paul-ford-what-is-code

Fosfuri, A., Giarratana, M. S., & Luzzi, A. (2008). The penguin has entered the building: The commercialization of open source software products. *Organization Science*, *19*(2), 292–305.

Fowler, M. & Beck, K. (1999). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional.

Frakes, W. & Terry, C. (1996). Software reuse: Metrics and models. *ACM Computing Surveys*, *28*(2), 415–435.

Franke, N. & Shah, S. K. (2003). How communities support innovative activities: An exploration of assistance and sharing among end-users. *Research Policy*, *32*(1), 157–178.

Gallini, N. & Scotchmer, S. (2002). Intellectual property: When is it the best incentive system? In Jaffe, Adam B and Lerner, Josh and Stern, Scott (Ed.), *Innovation Policy and the Economy* (Chap. 2, Vol. 2, pp. 51–78). MIT Press.

Garud, R. (1997). On the distinction between know-how, know-what, and know-why. *Advances in strategic management*, *14*, 81–102.

Gayoso Martínez, V., Hernández Álvarez, F., & Hernández Encinas, L. (2014). State of the art in similarity preserving hashing functions. In *Proceedings of the international conference on security and management (SAM)*.

GNU General Public License. (2007). Version 3. Free Software Foundation. Retrieved from http://www.gnu.org/licenses/gpl.html

Gould, R. V. (1991). Multiple Networks and Mobilization in the Paris Commune, 1871. *American Sociological Review*, *56*(6), 716–729.

Granovetter, M. S. (1973). The strength of weak ties. *American Journal of Sociology*, *78*(6), 1360–1380.

Granovetter, M. S. (1985). Economic action and social structure: The problem of embeddedness. *American Journal of Sociology*, *91*(3), 481–510.

Grant, R. M. (1996). Toward a Knowledge-Based theory of the firm. *Strategic Management Journal*, *17*, 109–122.

Grewal, R., Lilien, G. L., & Mallapragada, G. (2006). Location, location, location: How network embeddedness affects project success in open source systems. *Management Science*, *52*(7), 1043–1056.

Gross, T. & Blasius, B. (2008). Adaptive coevolutionary networks: A review. *Journal of the Royal Society, Interface / the Royal Society*, *5*(20), 259–271.

Haefliger, S., von Krogh, G., & Spaeth, S. (2008). Code reuse in open source software. *Management Science*, *54*(1), 180–193.

Hahn, J., Moon, J. Y., & 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.

Hansen, M. T. (1999). The Search-Transfer problem: The role of weak ties in sharing knowledge across organization subunits. *Administrative Science Quarterly*, *44*(1), 82–111.

Hansen, M. T. (2002). Knowledge networks: Explaining effective knowledge sharing in multiunit companies. *Organization Science*, *13*(3), 232–248.

Hansen, M. T., Mors, M. L., & Løvås, B. (2005). Knowledge sharing in organizations: Multiple networks, multiple phases. *Academy of Management Journal*, *48*(5), 776–793.

Hars, A. & Ou, S. (2002). Working for free? motivations for participating in Open-Source projects. *International Journal of Electronic Commerce*, *6*(3), 25–39.

Heinemann, L., Deissenboeck, F., Gleirscher, M., Hummel, B., & Irlbeck, M. (2011). On the extent and nature of software reuse in open source java projects. In *Top productivity through software reuse* (pp. 207–222). Springer.

Hertel, G., Niedner, S., & Herrmann, S. (2003). Motivation of software developers in open source projects: An internet-based survey of contributors to the linux kernel. *Research Policy*, *32*(7), 1159–1177.

Hollingshead, A. B. (2000). Perceptions of Expertise and Transactive Memory in Work Relationships. *Group processes & intergroup relations: GPIR*, *3*(3), 257–267.

Hollingshead, A. B., Fulk, J., & Monge, P. (2002). Fostering Intranet Knowledge Sharing: An Integration of Transactive Memory and Public Goods Approaches. In P. J. Hinds & S. Kiesler (Eds.), *Distributed Work* (Chap. 14, pp. 335–356). MIT Press.

Inkpen, A. C. & Tsang, E. W. K. (2005). Social capital, networks, and knowledge transfer. *Academy of Management Review*, *30*(1), 146–165.

Jaffe, A. B., Trajtenberg, M., & Henderson, R. (1993). Geographic Localization of Knowledge Spillovers as Evidenced by Patent Citations. *The Quarterly Journal of Economics*, *108*(3), 577–598.

Johns, G. (2006). The essential impact of context on organizational behavior. *Academy of Management Review*, *31*(2), 386–408.

Junger v. Daley. (2000). Court of Appeals, 6th Circuit.

Kaplan, C. (1998). Is software like a can opener or a recipe? *The New York Times*. Retrieved May 25, 2016, from http://www.nytimes.com/library/tech/98/07/cyber/cyberlaw/17law.html

King, G. & Zeng, L. (2001a). Explaining Rare Events in International Relations. *International organization*, *55*(03), 693–715.

King, G. & Zeng, L. (2001b). Logistic Regression in Rare Events Data. *Political Analysis*, *9*(2), 137–163.

Kitch, E. W. (1977). Nature and function of the patent system. *The Journal of law & economics*, *20*(2), 265–290.

Kogut, B. & Metiu, A. (2001). Open-Source Software Development and Distributed Innovation. *Oxford Review of Economic Policy*, *17*(2), 248–264.

Kogut, B. & Zander, U. (1992). Knowledge of the Firm, Combinative Capabilities, and the Replication of Technology. *Organization Science*, *3*(3), 383–397.

Kornblum, J. (2006). Identifying almost identical files using context triggered piecewise hashing. *Digital Investigation*, *3*(Supplement), 91–97.

Kyriakou, H., Nickerson, J. V., & Sabnis, G. (2017). Knowledge Reuse for Customization: Metamodels in an Open Design Community for 3D Printing. *MIS Quarterly*, *41*(1), 315–322.

Lakhani, K. R. & von Hippel, E. (2003). How open source software works: "free" user-to-user assistance. *Research Policy*, *32*(6), 923–943.

Lakhani, K. R. & Wolf, R. G. (2005). Why hackers do what they do: Understanding motivation and effort in Free/Open source software projects. In *Perspectives on free and open source software* (pp. 3–22). MIT Press.

Latour, B. (1999). *Pandora's hope: Essays on the reality of science studies*. Harvard University Press.

Laursen, K. & Salter, A. (2006). Open for innovation: The role of openness in explaining innovation performance among U.K. manufacturing firms. *Strategic Management Journal*, *27*(2), 131–150.

Lave, J. & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. Cambridge University Press.

Lazega, E. (2001). *The Collegial Phenomenon: The Social Mechanisms of Cooperation Among Peers in a Corporate Law Partnership*. Oxford University Press.

Lemley, M. A. (2004). Ex-ante versus ex-post justifications for intellectual property. *The University of Chicago Law Review*, *71*, 129–149.

Lerner, J. & Tirole, J. (2002). Some simple economics of open source. *The Journal of Industrial Economics*, *50*(2), 197–234.

Lessig, L. (2001). *The future of ideas: The fate of the commons in a connected world*. Vintage Books.

Lessig, L. (2004). *Free culture: The nature and future of creativity*. Penguin.

Lessig, L. (2006). *Code: Version 2.0*. Self-published.

Lessig, L. (2008). *Remix: Making art and commerce thrive in the hybrid economy*. Penguin Press.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, *10*(8), 707–710.

Lewis, K. (2004). Knowledge and Performance in Knowledge-Worker Teams: A Longitudinal Study of Transactive Memory Systems. *Management science*, *50*(11), 1519–1533.

Libby, R., Trotman, K. T., & Zimmer, I. (1987). Member Variation, Recognition of Expertise, and Group Performance. *The Journal of Applied Psychology*, *72*(1), 81–87.

Littlepage, G. E. & Silbiger, H. (1992). Recognition of expertise in decision-making groups: Effects of group size and participation patterns. *Small Group Research*, *23*(3), 344–355.

Luo, S., Du, Y., Liu, P., Xuan, Z., & Wang, Y. (2015). A study on coevolutionary dynamics of knowledge diffusion and social network structure. *Expert Systems With Applications*, *42*(7), 36q19–3633.

MacCormack, A., Rusnak, J., & Baldwin, C. Y. (2006). Exploring the structure of complex software designs: An empirical study of open source and proprietary code. *Management Science*, *52*(7), 1015–1030.

Mackenzie, A. (2006). *Cutting code: Software and sociality*. Peter Lang.

Malmberg, A. & Maskell, P. (2002). The Elusive Concept of Localization Economies: Towards a Knowledge-Based Theory of Spatial Clustering. *Environment & planning A*, *34*(3), 429–449.

Marsden, P. V. (2009). Recent developments in network measurement. In P. J. Carrington, J. Scott, & S. Wasserman (Eds.), *Models and methods in social network analysis* (pp. 8–30). Cambridge University Press.

Marsden, P. V. & Campbell, K. E. (1984). Measuring Tie Strength. *Social forces; a scientific medium of social study and interpretation*, *63*(2), 482–501.

Merton, R. K. (1968a). *Social theory and social structure*. Free Press.

Merton, R. K. (1968b). The matthew effect in science. *Science*, *159*(3810), 56–63.

Misztal, B. (1996). *Trust in modern society*. Polity Press.

Mockus, A. (2007). Large-Scale code reuse in open source software. In *First international workshop on emerging trends in FLOSS research and development* (pp. 7–7).

Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and mozilla. *ACM Transactions on Software Engineering and Methodology*, *11*(3), 309–346.

Moran, P. & Ghoshal, S. (1999). Markets, firms, and the process of economic development. *Academy of Management Review*, *24*(3), 390–412.

Mowery, D. C., Oxley, J. E., & Silverman, B. S. (1996). Strategic alliances and interfirm knowledge transfer. *Strategic Management Journal*, *17*(S2), 77–91.

Murray, F. & O'Mahony, S. C. (2007). Exploring the foundations of cumulative innovation: Implications for organization science. *Organization Science*, *18*(6), 1006–1021.

Nagaraj, A. (2017). Does Copyright Affect Reuse? Evidence from Google Books and Wikipedia. *Management Science*.

Nahapiet, J. & Ghoshal, S. (1998). Social capital, intellectual capital, and the organizational advantage. *Academy of Management Review*, *23*(2), 242–266.

Navarro, G. (2001). A Guided Tour to Approximate String Matching. *ACM Computing Surveys*, *33*(1), 31–88.

Nelson, R. R. & Winter, S. G. (1982). *An evolutionary theory of economic change*. Belknap Press of Harvard University Press.

Newcomb, T. M. (1961). *The acquaintance process*. Holt, Rinehart & Winston.

Nickerson, J. A. & Zenger, T. R. (2004). A Knowledge-Based theory of the Firm—The Problem-Solving perspective. *Organization Science*, *15*(6), 617–632.

Nonaka, I. (1994). A dynamic theory of organizational knowledge creation. *Organization Science*, *5*(1), 14–37.

O'Mahony, S. C. (2003). Guarding the commons: How community managed software projects protect their work. *Research Policy*, *32*(7), 1179–1198.

O'Mahony, S. C. & Ferraro, F. (2007). The emergence of governance in an open source community. *Academy of Management Journal*, *50*(5), 1079–1106.

Ombredanne, P., Yang, J., Balusa, R., et al. (2017). ScanCode toolkit. Retrieved from https://github.com/nexB/scancode-toolkit

Orlikowski, W. J. (2002). Knowing in practice: Enacting a collective capability in distributed organizing. *Organization Science*, *13*(3), 249–273.

Padgett, J. F. & Ansell, C. K. (1993). Robust Action and the Rise of the Medici, 1400-1434. *American Journal of Sociology*, *98*(6), 1259–1319.

Peddibhotla, N. B. & Subramani, M. R. (2007). Contributing to public document repositories: A critical mass theory perspective. *Organization Studies*, *28*(3), 327–346.

Phelps, C., Heidl, R., & Wadhwa, A. (2012). Knowledge, networks, and knowledge networks: A review and research agenda. *Journal of Management*, *38*(4), 1115–1166.

Phene, A. & Tallman, S. (2014). Knowledge spillovers and alliance formation. *Journal of Management Studies*, *51*(7), 1058–1090.

Pickering, J. M. & King, J. L. (1995). Hardwiring weak ties: Interorganizational Computer-Mediated communication, occupational communities, and organizational change. *Organization Science*, *6*(4), 479–486.

Polanyi, M. (1967). *The tacit dimension*. London, Routledge & K. Paul.

Prieto-Diaz, R. (1993). Status report: Software reusability. *IEEE Software*, *10*(3), 61–66.

Putnam, R. D. (1993). The prosperous community: Social capital and public life. *The American Prospect*, *13*(13), 35–42.

Putnam, R. D. (2001). *Bowling alone: The collapse and revival of american community*. Simon and Schuster.

Ravichandran, T. & Rothenberger, M. A. (2003). Software reuse strategies and component markets. *Communications of The ACM*, *46*(8), 109–114.

Raymond, E. S. (2001). *The cathedral & the bazaar: Musings on linux and open source by an accidental revolutionary*. O'Reilly Media.

Reagans, R. & McEvily, B. (2003). Network structure and knowledge transfer: The effects of cohesion and range. *Administrative Science Quarterly*, *48*(2), 240.

Řehůřek, R. & Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks* (pp. 45–50).

Rogers, E. M. (1962). *Diffusion of innovations*. Free Press of Glencoe.

Rosenkopf, L. & Almeida, P. (2003). Overcoming local search through alliances and mobility. *Management Science*, *49*(6), 751–766.

Rosenkopf, L., Metiu, A., & George, V. P. (2001). From the bottom up? technical committee activity and alliance formation. *Administrative Science Quarterly*, *46*(4), 748–772.

Rosenkopf, L. & Nerkar, A. (2001). Beyond local search: Boundary-spanning, exploration, and impact in the optical disk industry. *Strategic Management Journal*, *22*(4), 287–306.

Rosenkopf, L. & Padula, G. (2008). Investigating the microstructure of network evolution: Alliance formation in the mobile communications industry. *Organization Science*, *19*(5), 669–687.

Roy, C. K., Cordy, J. R., & Koschke, R. (2009). Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. *Science of Computer Programming*, *74*(7), 470–495.

Schumpeter, J. A. (1934). *The theory of economic development: An inquiry into profits, capital, credit, interest, and the business cycle*. Transaction publishers.

Shah, S. K. (2006). Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science*, *52*(7), 1000–1014.

Simon, H. A. (1991). Bounded rationality and organizational learning. *Organization Science*, *2*(1), 125–134.

Singh, J. (2005). Collaborative networks as determinants of knowledge diffusion patterns. *Management Science*, *51*(5), 756–770.

Singh, J. & Fleming, L. (2010). Lone inventors as sources of breakthroughs: Myth or reality? *Management Science*, *56*(1), 41–56.

Singh, P. V. & Phelps, C. (2013). Networks, social influence, and the choice among competing innovations: Insights from open source software licenses. *Information Systems Research*, *24*(3), 539–560.

Singh, P. V., Tan, Y., & Mookerjee, V. M. (2011). Network effects: The influence of structural capital on open source project success. *MIS Quarterly*, *35*(4), 813–829.

Sojer, M. & Henkel, J. (2010). Code reuse in open source software development: Quantitative evidence, drivers, and impediments. *Journal of The Association for Information Systems*, *11*(12), 868–901.

Spender, J.-C. (1996). Making knowledge the basis of a dynamic theory of the firm. *Strategic Management Journal*, *17*(S2), 45–62.

Sproull, L. S. & Arriaga, M. (2007). Online communities. In H. Bidgoli (Ed.), *Handbook of computer networks* (pp. 898–914). John Wiley & Sons.

Stallman, R. (1999). The GNU Operating System and the Free Software Movement. In C. DiBona, S. Ockman, & M. Stone (Eds.), *Open sources: voices from the open source revolution* (pp. 31–38). O'Reilly Media.

Star, S. L. & Griesemer, J. R. (1989). Institutional ecology, 'translations' and boundary objects: Amateurs and professionals in berkeley's museum of vertebrate zoology, 1907-39. *Social Studies of Science*, *19*(3), 387–420.

Stewart, K. J. & Gosain, S. (2006). The impact of ideology on effectiveness in open source software development teams. *MIS Quarterly*, *30*(2), 291–314.

Stuart, T. E. (1998). Network Positions and Propensities to Collaborate: An Investigation of Strategic Alliance Formation in a High-Technology Industry. *Administrative Science Quarterly*, *43*(3), 668.

Stuart, T. E. & Podolny, J. M. (1996). Local search and the evolution of technological capabilities. *Strategic Management Journal*, *17*(S1), 21–38.

Susarla, A., Oh, J.-H., & Tan, Y. (2012). Social networks and the diffusion of User-Generated content: Evidence from YouTube. *Information Systems Research*, *23*(1), 23–41.

Sutcliffe, A. & Sutcliffe, A. G. (2002). *The domain theory: Patterns for knowledge and software reuse*. CRC Press.

Taylor, H., Dillon, S., & Van Wingen, M. (2010). Focus and diversity in information systems research: Meeting the dual demands of a healthy applied discipline. *MIS Quarterly*, *34*(4), 647–667.

TIOBE Software. (2016). TIOBE Index. Retrieved March 25, 2016, from http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Tracz, W. (1995). *Confessions of a used program salesman: Institutionalizing software reuse*. Addison-Wesley.

Tsai, W. (2001). Knowledge transfer in intraorganizational networks: Effects of network position and absorptive capacity on business unit innovation and performance. *Academy of Management Journal*, *44*(5), 996–1004.

Tsai, W. & Ghoshal, S. (1998). Social capital and value creation: The role of intrafirm networks. *Academy of Management Journal*, *41*(4), 464–476.

Tufano, M., Palomba, F., Bavota, G., Oliveto, R., Penta, M. D., Lucia, A. D., & Poshyvanyk, D. (2015). When and Why Your Code Starts to Smell Bad. In *37th IEEE International Conference on Software Engineering* (Vol. 1, pp. 403–414).

Uzzi, B. (1997). Social structure and competition in interfirm networks: The paradox of embeddedness. *Administrative Science Quarterly*, *42*(1), 35–67.

Uzzi, B. & Lancaster, R. (2003). Relational embeddedness and learning: The case of bank loan managers and their clients. *Management Science*, *49*(4), 383–399.

Uzzi, B. & Spiro, J. (2005). Collaboration and creativity: The small world problem. *American Journal of Sociology*, *111*(2), 447–504.

Von Hippel, E. (2001). Innovation by User Communities: Learning from Open-Source Software. *MIT Sloan Management Review*, *42*(4), 82–86.

von Hippel, E. (1988). *The sources of innovation*. Oxford University Press.

von Hippel, E. (1994). "Sticky Information" and the Locus of Problem Solving: Implications for Innovation. *Management Science*, *40*(4), 429–439.

von Hippel, E. (1998). Economics of product development by users: The impact of "sticky" local information. *Management Science*, *44*(5), 629–644.

von Hippel, E. (2005). *Democratizing innovation*. MIT Press.

von Krogh, G., Haefliger, S., Spaeth, S., & Wallin, M. W. (2012). Carrots and rainbows: Motivation and social practice in open source software development. *MIS Quarterly*, *36*(2), 649–676.

von Krogh, G. & von Hippel, E. (2006). The promise of research on open source software. *Management Science*, *52*(7), 975–983.

Wang, C., Rodan, S., Fruin, M., & Xu, X. (2014). Knowledge networks, collaboration networks, and exploratory innovation. *Academy of Management Journal*, *57*(2), 484–514.

Wasko, M. 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), 155–173.

Wasko, M. M. & Faraj, S. (2005). Why should I share? examining social capital and knowledge contribution in electronic networks of practice. *MIS Quarterly*, *29*(1), 35–57.

Wasserman, S. & Faust, K. (1994). *Social network analysis: Methods and applications*. Structural Analysis in the Social Sciences. Cambridge University Press.

Wegner, D. M., Erber, R., & Raymond, P. (1991). Transactive memory in close relationships. *Journal of Personality and Social Psychology*, *61*(6), 923–929.

Wenger, E. (2000). Communities of practice and social learning systems. *Organization*, *7*(2), 225–246.

West, J. (2003). How open is open enough?: Melding proprietary and open source platform strategies. *Research Policy*, *32*(7), 1259–1285.

West, J. & Gallagher, S. (2006). Challenges of open innovation: The paradox of firm investment in open-source software. *R&D Management*, *36*(3), 319–331.

Williams, S. (2011). *Free as in Freedom: Richard Stallman's Crusade for Free Software*. O'Reilly Media.

Yang, H., Phelps, C., & Steensma, H. K. (2010). Learning from What Others Have Learned from You: The Effects of Knowledge Spillovers on Originating Firms. *Academy of Management Journal*, *53*(2), 371–389.

Zammuto, R. F., Griffith, T. L., Majchrzak, A., Dougherty, D. J., & Faraj, S. (2007). Information Technology and the Changing Fabric of Organization. *Organization Science*, *18*(5), 749–762.

Zander, U. & Kogut, B. (1995). Knowledge and the speed of the transfer and imitation of organizational capabilities: An empirical test. *Organization Science*, *6*(1), 76–92.

Zellner, A. (1962). An Efficient Method of Estimating Seemingly Unrelated Regressions and Tests for Aggregation Bias. *Journal of the American Statistical Association*, *57*(298), 348–368.

Appendix A

Glossary of Open Source and Ruby-related Terms

Gem A reusable software bundle for Ruby.

Package Each release (version) of a gem is a package.

RubyGems Ruby's sophisticated package/dependency manager as well as the community's reusable software repository hosting Ruby gems. Virtually all gems are hosted on RubyGems and are directly accessible through RubyGems' command line program accompanying every Ruby distribution: `gem install [gem name]`.

GitHub A source code repository with very close ties to Ruby community, offering a hosted version of Git version control system. Apart from a handful of exceptions all gems with a declared public source code repository are hosted on GitHub.

Git A distributed version control system initially designed by Linux kernel developers to tackle the challenges of large scale software development efforts. Git allows developers to work on *forks* of each others' code and provides facilities for reintegration after forking. It also allows several *branches* of a code to be developed concurrently. Git keeps a complete log of code authorship and commit activity.

Fork A fork is a new copy of the entire code base and development history of a code repository. Git requires developers to keep a fork of the projects they work on, with the objective of synchronizing the forks at an ulterior stage. But forks can also be used to make a separate development thread and part ways with the original project.

Commit A commit is the unit of code contribution to source code repositories. In a distributed development environment like GitHub developers always work on their local forks and commit their own repository. It is only after having finished at least one unit of work (e.g. completing a feature, patching a bug, etc.) that they make a *pull request* to the original repository so that their work can be integrated in the main code base.

Pull Request It's a facility on GitHub that allows the contributor to a fork requests the administrator of the upstream project to integrate the downstream contributions to the fork into the original code base.

Appendix B

Overview of the Gathered Network Data

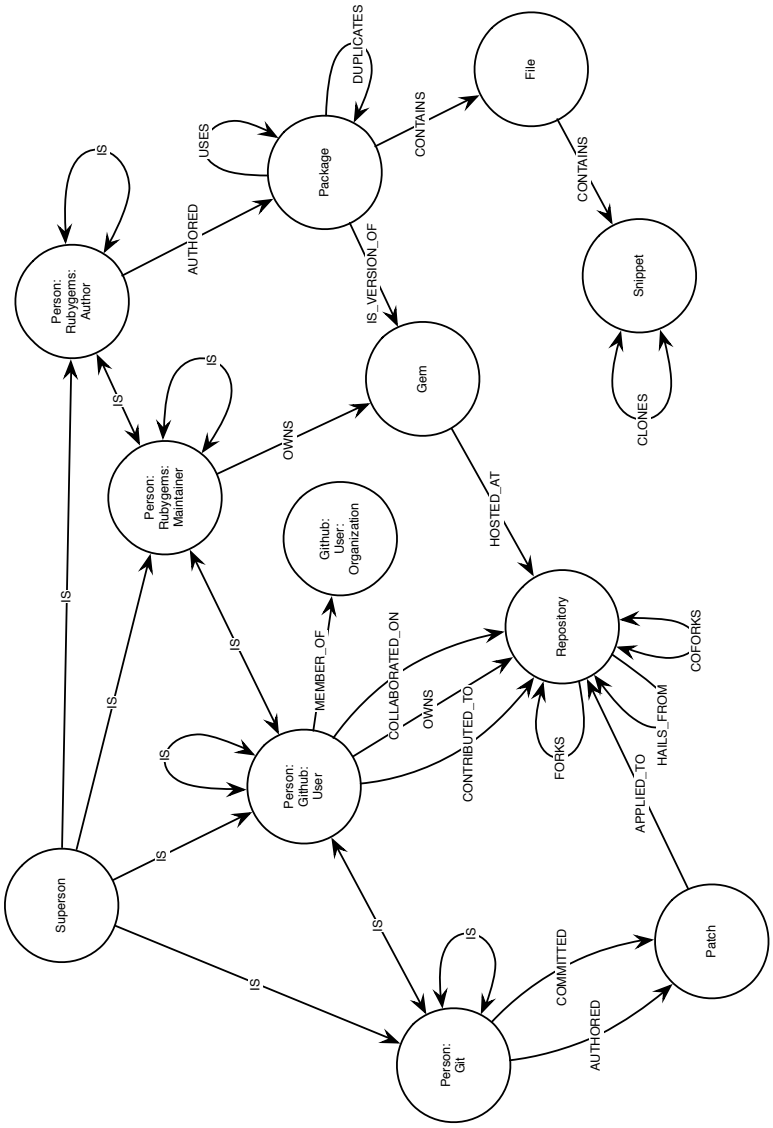


Figure B1. The diagram of the network data gathered for the thesis