REUSING OR REINVENTING THE WHEEL: THE SEARCH-TRANSFER ISSUE IN OPEN SOURCE COMMUNITIES

Research-in-Progress

Mahmood Shafeie Zargar
McGill University
Montreal, QC, Canada
mahmood.shafeiezargar@mail.mcgill.ca

Abstract

Despite the raising awareness about the importance of open innovation communities in knowledge economies, empirical evidence about the structural determinants of knowledge reuse in these communities is lacking. In order to address this gap, the current study sets out to investigate the network-level determinants of knowledge reuse in open source projects. I suggest tracking code reuse across open source projects as a feasible and accessible proxy measure for knowledge reuse. I argue that in spite of favorable conditions, search and processing costs associated with knowledge reuse remain high enough to localize reuse behavior of open source developers. I hypothesize that network-level proximity of projects within the social network of open source community is a significant determinant of code reuse. A report on the progress of the empirical section of the research project along with some confirmatory preliminary results has been included.

Keywords: Knowledge-based view, Knowledge sharing, Code reuse, Intellectual property rights, Open source software, OSS/FLOSS, Social networks, Code cloning

Introduction

Knowledge, as a fundamental resource in modern economies, has been subject of extensive debates in management literature. Contemporary theories of the firm have embraced the knowledge perspective, placing the capacity to create, transfer and transform knowledge resources as the cornerstone of firm's competitive advantage (Argote and Ingram 2000; Boisot 1998; Conner and Prahalad 1996). The knowledge perspective, however, has played only a tangential role in the efforts to theorize informal organizing. Ironically, in knowledge economies informal organizations have served as a new home for innovation, where knowledge-intensive work can thrive under loosely structured governance mechanisms that stand apart from the better known *markets* and *formal organizations* (Demil and Lecocq 2006; Hippel and Krogh 2006). Examples include open sourcing and crowd sourcing initiatives that have proved themselves as viable alternatives to their formal counterparts (Lakhani and Hippel 2003).

The open initiatives in general, and the open source projects in particular, have triggered a sweeping wave of academic investigations during the last ten years. The spotlight of the research has mainly been on the supply side of knowledge production, shedding light on an array of social, psychological, economic, hedonistic and altruistic motivations that undergird the participation of individuals in open source projects (Benbya and Belbaly 2010; Hars and Ou 2002; Hertel et al. 2003; Hippel and Krogh 2006; Lakhani and Wolf 2001; Lerner and Tirole 2002; Mockus et al. 2002; Shah 2006) as well as the structural determinants that shape the degree and mode of participation of the community in projects (Baldwin and Clark 2006; Dahlander and O'Mahony 2010; O'Mahony and Ferraro 2007; Singh et al. 2011). This narrow focus, though, has been to the expense of other research questions that are not necessarily less impactful or intriguing than the question of participation.

For instance, there has been little effort to obtain empirical evidence regarding the determinants of innovative capacity, such as knowledge transfer, reuse and recombination, in open source communities (Haefliger et al. 2008). Given that the intellectual property regimes governing open initiatives are much more permissive than their proprietary counterparts in terms of knowledge disclosure, access and reuse (Murray and O'Mahony 2007), it is likely that the knowledge reuse behavior and its associated consequences in this context do not correspond to those observed under proprietary regimes. This begs the question whether "different" in this case means more favorable to innovation in a knowledge-based economy. The general perception with regard to the impact of the open source on innovation has been divided. Microsoft officials have gone as far as declaring it "un-american" (Leonard 2001), as it can erode the economic incentives of the innovators to, while the open source activists have found it the only means of bypassing the obsolete laws that hinder the free flow of knowledge across society (Stallman 1985).

But does the codified nature of the project source codes, combined with open access licenses and today's sophisticated code and content search engines lead to a flat world where knowledge flows seamlessly, and the impact of social and structural factors on the likelihood of projects drawing on each other's development is negligible? Or does knowledge reuse remains fragmented across the community, albeit for different structural factors that drive knowledge reuse across open source projects?

This paper studies the network-level determinants of source code reuse, as a proxy for knowledge reuse, in open source projects. Despite the seemingly frictionless intellectual property regimes governing open source projects and a technological context that promotes access and disclosure, I expect to observe a significant localization of code reuse due to persisting information search and processing costs. I suggest that the structure of the affiliation network that connects the projects and the individual developers correlates with the structure of the technical network of code reuse among projects. In other words, I claim that the pattern of code reuse follows the social network connections. Such a finding signifies to what extent knowledge about open source codes is embedded within the social network surrounding them and may provide an explanation for the possible benefits that accrue to individual developers despite the fact that the intellectual property regimes governing open source projects do not guarantee pecuniary incentives for the developers. The current paper also proposes methods to test this thesis and reports on the progress made in operationalizing the empirical test along with preliminary results.

Knowledge Reuse in Open Source Communities

The common understanding about knowledge, as a non-rival and non-excludable public good, is that unless it is protected through a cultural or legal framework, its appropriability for the originator is doubtful (Arrow 1959; Murray and O'Mahony 2007; Olson 1965). As a response, the proprietary intellectual property regimes rely on legal mechanisms such as patens and copyright to strike a balance between the degree of disclosure needed to broadcast the capabilities of an innovation and the protection required to direct the flow of rewards towards the property right owners (Arrow 1959; Murray 1988). In contrast to these macro-level concerns about the unbounded imitability of knowledge, the experiences of organizational actors as well as our personal experiences seem to indicate that knowledge transfer and reuse processes may not be as streamlined as they appear.

The extant research has identified at least five categories of determinants for knowledge transfer and reuse in and across organizations. (1) *Knowledge Characteristics* – Knowledge is directly transferrable only to the extent that it can be codified (Polanyi 1958). Consequently, certain types of knowledge that are more amenable to codification are more prone to be subject to direct transfer (Argyris and Schön 1978; Nonaka 1994). Apart from codifiability, other knowledge characteristics that are determinants of the degree of transferability include teachability, complexity and system dependence (Zander and Kogut 1995). (2) *Source Characteristics* – The actor at the source of knowledge can affect knowledge transfer to the extent that he or she is willing to facilitate knowledge transfer by transforming knowledge into suitable format (Nonaka 1994). Thus, the incentives and interests of the source can impact knowledge transfer (Haefliger et al. 2008; Hansen 2002). (3) *Recipient Characteristics* – Knowledge transfer cannot be accomplished unless if the recipient has the *absorptive capacity* to decode, make sense of, evaluate, assimilate, and exploit the knowledge. Absorptive capacity is related to prior relevant knowledge of the recipient (Cohen and Levinthal 1990). (4) *Context Characteristics* – The intellectual property regime governing the knowledge transfer setting can facilitate transfer by securing knowledge *disclosure* and *access*, as well as providing *reward mechanisms* for the property right owners (Murray and O'Mahony 2007). It is notable that all the above knowledge transfer determinants ultimately influence diffusion of knowledge by facilitating knowledge *disclosure* and *access*, either on the offer side or on the demand side. (5) *Social Network Patterns* – Since individuals themselves are vehicles of knowledge, the social movements of individuals as well as the social structure in which they are embedded is instrumental in knowledge diffusion (Abrahamson and Rosenkopf 1997; Susarla et al. 2012). Social network patterns may not directly impact knowledge diffusion and access costs, but are instead instrumental in mitigating the search costs associated with knowledge disclosure, and the transfer costs associated with knowledge access at a local level (Hansen 1999; 2002; Rosenkopf and Nerkar 2001; Rosenkopf and Padula 2008).

With regard to the above determinants, there are at least three arguments that contribute to the general perception that the open source communities provide a frictionless environment for knowledge diffusion and reuse. (1) The knowledge content of open source projects seems to be highly codified, either in form of software code or in form of communication logs from discussion forums and mailing lists. The knowledge characteristics are, thus, favorable to diffusion. (2) Cumulative intellectual property regimes, such as those governing open source-licensed software code or creative commons licenses, prioritize sharing over protection and encourage provision of knowledge as a public good rather than restricting the rights for the knowledge produced (Murray and O'Mahony 2007). These regimes provide unrestricted access to knowledge for reuse and recombination as long as the open source license requirements are not violated (O'Mahony 2007). (3) Cumulative intellectual property regimes also promote full disclosure of knowledge. This latter, combined with sophisticated search engines capable of scanning through software code (e.g. code.ohloh.net) as well as text data should further enhance disclosure and discoverability, and possibly minimize search costs for those developers who seek to draw on knowledge from other open source projects. Thus, context characteristics of open source initiatives can also be identified as largely favorable to knowledge reuse. All these indicate that search and transfer costs in open source communities should be minimal and knowledge reuse should be pervasive.

If code reuse is any indication of the larger pattern of knowledge reuse, the empirical evidence shows that code reuse occurs as extensively in open source software projects as it does in proprietary software projects (Haefliger et al. 2008). The bulk of this reuse occurs in form of API calls, while less recurrently reuse is achieved through cloning snippets or entire files of code (Heinemann et al. 2011). This is despite the fact that the percentage of source files that are used and included in more than one project exceeds 50% all the code base in open source communities (Mockus 2007). But how this strong pattern of reuse is organized in open source communities and what drives it are valid research questions that the above

findings warrant the interest. More particularly, in absence of regular legal, organizational and technical barriers, should we expect near-zero knowledge search and transfer costs? And given that social network patterns have only a cost-mitigating effect, should we expect a random distribution of knowledge reuse across open source communities, unaffected by social network patterns? Sojer and Henkel's (2010) survey of SourceForge developers suggests otherwise. The study provides that the developers with larger social networks and those with experience in a greater number of projects tend to reuse more. The authors attribute this tendency to lower local search costs when looking for reusable artifacts. Yet, the study lacks network-level evidence, and does not answer the question of where the socially active developers pick the code they reuse. It is this significant gap in the literature that the current study sets out to address.

Indeed, it is possible to argue that the favorable conditions under which open source communities operate can only partially eliminate the costs related to dissemination and reuse of knowledge. Knowledge reuse patterns in open source communities may be influenced by a number of factors that are endemic to these communities. In other words, the general perception that open source communities provide a unique platform for the innovative individuals to attain novel solutions by freely recombining their ideas with the knowledge available from the community is probably sensible. Breaking free from the excessively restrictive legal barriers imposed by proprietary regimes has certainly fostered creativity and innovation in many areas. Yet, there can be forces at work in open communities that constrict knowledge flows in ways idiosyncratic to these communities. Such forces may explain the existence of for-profit open source software, as they point to the reasons for which knowledge, and thus knowledge exploitation, may remain local despite open access.

For instance one can argue that while open source project source codes are in themselves codified knowledge, and that most of the communications regarding the projects are also codified in text format and widely accessibly, the cost of processing this codified knowledge can be prohibitive for a potential reuser. This is aggravated by the fact that many open source projects suffer from a paucity of structured technical, architectural and user documentation (DiBona et al. 1999 p. 37). Furthermore, since under cumulative intellectual regimes there is little or no provision for appropriating the returns on the knowledge produced, it is likely that the originators do not find enough incentives to transform their production into reusable form, specially given that making a piece of software code reusable generally costs more than the initial software development costs (Tracz 1995). Again in such a case the potential reuser will be confronted with exorbitant processing costs in order to evaluate and integrate knowledge (Murray and O'Mahony 2007).

The promise of free access in open source communities also comes with its proper limitations. Permissiveness of open software licenses does not prevent them from containing viral clauses that renders them incompatible with other licenses that co-exist inside the community (O'Mahony 2007). Singh and Phelps (2009) have shown that software project license choice is itself socially induced. Such legal barriers will not prevent the potential reusers to be inspired by the codebase or the architecture of a homologous project while developing their own project, but may impose prohibitive processing costs on the potential reuser. All the above obstacles may look minimal as compared to the restrictions posed by corporate and proprietary license obligations, but they still generate non-zero costs of knowledge reuse, particularly in form of processing costs.

In the same vein, the promise of full disclosure engraved in open source software licenses does not necessarily translate into zero search costs for the reuser, even though today's powerful search engines go a long way in finding and extracting relevant information. Search engines provide undifferentiated information along with broad quality indicators at best (Lynch and Ariely 2000), which are of limited interest to the potential reuser who aims to distinguish useful from useless (Haefliger et al. 2008).

The costs associated with knowledge search and transfer render knowledge "sticky", localizing it geographically, socially and with regards to the knowledge domain (Hippel 1994; Rosenkopf and Nerkar 2001). Since geography rarely matters in the virtual world of open source (Ducheneaut 2005) and the knowledge domains correlate with social network structures (Barabási 2002), I suggest that the issue of knowledge localization in open source communities should manifest itself mainly through social networks. Random-access information search and analysis solution are effective only to a limited extent in attenuating the localization effects of knowledge asymmetry by reducing partially the search costs. Issues such as trust and reputation that are of a more sensitive nature on the internet (Brynjolfsson and Smith 2000) and are, by definition, embedded within social networks (Burt 1987; 2000; Coleman 1988) can

only augment these localization effects. In other words, since social network patterns such as proximity or centrality can mitigate for certain search and transfer costs that I deem lasting, I expect to see a correlation between social network structure and the patterns of knowledge reuse.

Code Reuse as a Proxy for Knowledge Reuse

Social network patterns as determinants of knowledge reuse among firms or organizational units have fueled an entire stream of research in organization studies. Measuring knowledge reuse through patent citation statistics, prior research has found a relationship between mobility of innovators and the reuse of intellectual property they have authored. It is commonplace for the firms to overcome their knowledge search and transfer limitations by recruiting the author of an innovation they covet, rather than trying to acquire the capacity to reuse that innovation relying on their existing workforce (Almeida and Kogut 1999; Rosenkopf and Almeida 2003). Worker mobility, thus, can mitigate a number of knowledge search and transfer costs and barriers. Furthermore, the mobility of individuals across the organizational and social boundaries leads to diffusion of knowledge from one social sphere to the other. Boundary-spanners play a critical role in knowledge transfer and innovative reuse. For instance, Rosenkopf and Nerkar (2001) have demonstrated that the knowledge generated in a firm yields more impactful recombinations when moved across firm boundaries by boundary-spanners than when it remains within the confines of the firm boundaries for domestic use. The traceability of the patents as well as the reliability of the existing data on patent citations have allowed innovation scholars to study organizational issues on knowledge transfer and innovation diffusion that no other dataset would allow.

I argue that tracking the patterns of code reuse among open source projects offers an equally useful and reliable proxy for knowledge reuse in open source communities and I set out to investigate the relation between social network patterns and code reuse patterns in open source projects. I do not consider each instant of code reuse as an equally important instance of knowledge reuse, but instead a traceable sign of a knowledge transaction between the two projects. This is a deliberate choice and I am fully aware of the reductionist nature of such an assumption. Proxy variables are often of reductionist nature.

Code reuse can occur under two distinct forms (Heinemann 2011). A software project may draw on the knowledge embedded in another project by developing functional dependency over that project's source code. This is normally achieved by creating within-code references (API calls) to the functional interfaces provided by the other project for the specific purpose of reuse. Reusing another code through API calls can be considered an instance of *"blackbox"* reuse, as the inner-workings of the reused project need not be transparent for the reuser as long as the interface is well-documented and it provide consistent responses to the calls. Reuse in such a case is a matter of mastering API interfaces, even though having a more comprehensive knowledge of the internals of the reused project won't hurt. In other cases, including when a project has not been designed with reusability in mind or does not provide the appropriate functional interfaces, the reusers may be forced to adopt a piece of the original code, modify it to their linking, and integrate it within their own code. I will refer to this type of code reuse as *"glassbox"* reuse in order to reflect the effort needed on the side of potential reusers to discover the right code content within the codebase of another project, understand its inner-workings and finally integrate it into their own code. Both types of code reuse are interesting indicators of knowledge dissemination and reuse among open source projects. By juxtaposing the data from code reuse and that of the affiliation network of open source developers and projects I propose to investigate the network-level knowledge reuse localization effects.

Open source developers work under high resource restraints and their decision to reuse code is mainly a means of cutting on costs and development times in order to dedicate more resources to what is more important or interesting to them in the project (Haefliger et al. 2008). It follows that the perceived cost of reuse for the developers should be lower than that of rewriting the code (Banker et al. 1993). The issue of reuse versus rewrite in software projects can be compared to the issue of make or buy in firms whereby the firm has to strike a balance between the costs of internalization versus outsourcing of an operation. The literature on outsourcing assumes that organizations, as rational actors, are aware of the costs inherent to market transactions as compared to costs of developing a home-brewed solution (Walker and Weber 1984; Williamson 1975). A parallel can be drawn here with the open source developers, as rational actors who try to strike a balance between the costs associated with code reuse and code rewrite in order to use their limited resources efficiently, avoiding reinventing the wheel to the extent possible. The mentioned code search and transfer costs will unavoidably enter the developers' reuse vs. rewrite cost-

benefit analysis. I suggest that since network proximity can mitigate a number of costs associated with knowledge reuse (i.e. search and processing), developers affiliated with one project will show a tendency to reuse the code of socially proximate projects more regularly. This includes the projects they are or they have been affiliated with, or the projects they get to know through their connections.

Considering the two categories of costs associated with code reuse, for a variety of reasons network proximity may correlate with diminished costs of code search and processing for reuse. Network ties, regardless of their strength can be a resource for non-redundant information (Burt 2009 p. 25), and by this virtue they can help developers discover the existence of a code or project. Moreover, closely tied individuals are more likely to allocate resources to help each other and are more likely to develop a trust relationship (Coleman 1988), that is admittedly an important issue for a developer who is trying to find and chose a software package or code snippet to build on (Haefliger et al. 2008). These, as well, can reduce the perceived costs of code reuse. Finally, seen as an affiliation network (i.e. A two-mode or bipartite network. See the methods section for further details.), co-affiliations of individuals with different projects (i.e. strong ties between projects) provides a golden opportunity for the members of each project to get to know and draw on the other project's code. Thus, the existence of any tie between two projects is likely to reduce the search or processing costs, and consequently likely to induce code reuse.

Hypothesis 1) Projects reusing a focal project are more likely to have social ties to the project than not

In a social network the actors maintain a tie by exchanging either tangible or intangible resources such as information, goods and services, or financial means. The strength of a tie varies depending on a number of factors. Granovetter (1973) distinguishes between strong and weak ties and asserts that tie strength depends on the amount of time, the emotional intensity, the intimacy, and the reciprocal services associated with the relationship. Strong ties are characterized by a sense of special relationship, an interest in frequent interactions, and a sense of mutuality of the relationship (Uzzi 1997). Since maintaining strong ties demand a higher level of investment, their number tends to be limited by the available resources, while the weaker network ties can be more numerous and diverse (Burt 2000). Both strong and weak ties are important for efficient knowledge exchange in the networks, nonetheless they play different roles. Strong ties are useful in instilling a sense of social closure and maintaining trust, 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). The homogeneity and the high level of vested interests among strongly tied entities makes them better suited to transfer of complex and system-dependent knowledge, as these ensure both the willingness and the ability to go through an exchange process that is likely to require deliberate effort (Hansen 1999).

I propose that two strongly-tied open source projects that have developers in common are more likely to draw on each other's code in a glassbox manner, as strong ties are more likely to reduce the high processing costs typical to glassbox reuse. Glassbox reuse requires awareness of the content of a code, that is not easily advertisable nor readily comprehensible. A thorough understanding of the code and the system that englobes it can be more resource-consuming than rewriting the code, and trust in a code to the point of modifying one's own code to accommodate it is difficult to achive. Strong ties between two projects, defined as developer co-affiliations (common developers), provides both the trust and the in-depth understanding that it takes to resort to glassbox reuse.

Hypothesis 2) The strong ties are more strongly related to glassbox reuse than to blackbox reuse

The strength of weak ties, though, is in resolving search issues. Weak ties require lower investment to maintain and by that virtue they are likely to be more numerous and more diverse. They are also more likely to lead to contacts holding non-redundant knowledge as compared to strong ties (Burt 2000; Granovetter 1985). On the other hand weak ties are not well-suited for exchanging complex, system-dependent or implicit knowledge. Despite the strictly codified nature of software code, the developers often have some implicit knowledge about the code that is key understanding and integrating the code with reasonable effort.

In the context of open source communities a weak tie between two projects can be defined as two projects not having developers in common but having developers who know each other through other projects. Being connected indirectly to a variety of different projects can help the development team discover with more ease the opportunities for reuse in those projects. Blackbox reuse typically needs less investment in

the reused solution as it can be accomplished with merely a good understanding of the API interface and requires less structural adaptation to the grafted code. I propose that the projects that are weekly tied to a focal project are more likely to be reused by that project in a blackbox manner.

Hypothesis 3) The weak ties are more strongly related to blackbox reuse than to glassbox reuse

Finally, if all the above holds, then the projects that are located close to many other projects should be more likely to be subject to code reuse. In other words, the projects that are highly embedded in the social network of open source community should be more intensely reused. Network embeddedness may refer to different concepts: the degree of entrenchment in the network of relationships, the extent to which an entity connects to the other entities in the network, or the degree of connectivity to the entities that are highly entrenched in the network (Grewal et al. 2006). Any of these definitions can justify the argument that a network position that puts a project socially closer to a larger number of other projects and improve its connectedness within the social network, is likely to increase the degree of reuse of that project. The network embeddedness of an open source project has been shown to correlate with a variety of other measures, such a the commercial success (Grewal et al. 2006) and the community success (Singh et al. 2011) of open source projects.

Hypothesis 4) Network embeddedness of a focal project is positively related to its reuse

The main focus of this study is on the structural factors that drive knowledge reuse, and as such it relies heavily on network-level measures, or edge data. Yet, network-level determinants constitute only one of the five categories of drivers of knowledge reuse identified in the literature and covered in the literature review section. Network characteristics can be successfully combined and complemented with node-level, which is, individual-level characteristics (Kilduff and Krackhardt 1994). Experience of developers (Sojer and Henkel, 2010), age and maturity of projects (Haefliger et al. 2008), code structure (Baldwin and Clark 2006; MacCormack et al. 2006), software license (O'Mahony 2003) along with many other factors such as the category and the popular success of a project are among the node-level characteristics that can potentially shape patterns of reuse. While the effect of many of these characteristics can be captured through network measures, I intend to use the others as control variables to tease out purely structural drivers of knowledge reuse from those induced by individual-level characteristics.

Methods

Sampling, Data Structure and Measures

Sampling: Preliminary trials demonstrated that a random network of open source projects, even if adjusted for the purpose of this study, provides unacceptably sparse network data. Thus I reached the decision to focus on a rather self-contained population of projects, a community that has low interdependence with other communities. I identified such a population of projects in the community around Ruby programming language. Ruby programs and libraries are predominantly distributed in packages called “Gems”, which are centrally hosted on RubyGems.org. The large majority of gem source codes are hosted on GitHub. This high level of centralization facilitates data gathering, and access to the full population rather than a sample. Currently I have a complete local copy of all versions of all gem source codes and the project metadata available from RubyGems.org. This comprises more than one hundred gigabytes of compressed source code, and the complete metadata of the 61,956 gems available in the central repository. Work is in progress to integrate GitHub social network and incremental code evolution data. I consider each gem a project in this context.

Affiliation Network: I have currently constructed an undirected bi-partite affiliation network of interconnected individuals and projects in which a link between two projects is identified when an individual has publishing rights over those projects and a link between two individuals is identified when both have publishing rights over at least one common project. Work is in progress to include social network data from GitHub, that contains information regarding all the committers, rather than only those who have publishing rights. Along with the edge data, some project and developer characteristics data such as project age, the release date and the popularity of each version, the publishing developers of each version, and the activity span of the developers have also been included. Integration of GitHub data will provide more fine-grained node and edge characteristics.

Reuse Networks: Project reuse data, too, can be seen as a network of interrelations among projects. This technical network is a directed network wherein project A reusing code from project B will be shown as an arc from A towards B. Each of the two types of reuse, blackbox and glassbox, will constitute a distinctive network. The data for blackbox reuse network (API calls) has already been gathered from project metadata, which indicates the version of development time and run time libraries that each version of a project uses. I am currently testing several software packages in order to construct the glassbox reuse network using code cloning detection algorithms (e.g. Black Duck, CCFinder, Protecode, Simian, Atomiq, Conqat, etc..). The objective is to choose a software package that is capable of identifying type-1 to type-3 clones (from direct copies to copies with extensive changes) in very large code bases of Ruby language (Schwarz et al. 2012).

Embeddedness: Measures of node centrality such as closeness, betweenness, and node degree (Borgatti and Halgin 2011) can be used to operationalize this dependent variable.

Strong Ties: The strong ties will be measured as the existence or if possible the count of co-affiliations between projects within the bi-partite affiliation network.

Weak Ties: The weak ties will be measured as the existence or if possible the count of indirect links between projects within the network resulted from reducing the original bi-partite affiliation network into a single-mode social network by suppressing developer nodes.

Blackbox Reuse: Refers to the existence or the count of functional dependency links between two projects in the technical network of blackbox code reuses.

Glassbox Reuse: Refers to the existence or the count of code snippet duplication links between two projects in the technical network of glassbox code reuses.

Controls: Information regarding many additional developer and project characteristics (e.g. project modularity, size, maturity, age, etc... as well as developer seniority, activity level, etc...) is being gathered to be included as control variables.

Analysis

The main goal of this project can be described as finding coincidences between the relations in the technical network of code reuse and the relations in the affiliation network of open source projects. For the time being these networks contain exclusively static data, and allow for simple correlational analysis. Work is in progress to include time-relevant data in order to construct dynamic networks allowing causal rather than of correlational tests. Since I can safely assume that I have access to the real population data, very simple statistic models will suffice to demonstrate the veracity of the hypotheses.

The current stage of data gathering allows me to partially test Hypothesis 1, which states that reuse links between projects generally tend to coincide with social links between projects. I test this using the cross-sectional data on direct blackbox links (API dependencies) and on direct social links of high strength (projects having administrators in common). Hypothesis 1 can be formalized in the following way:

$$P(Rxy|Sxy = 1) > P(Rxy|Sxy = 0)$$

Where P is probability, R denotes a link between projects x and y in the reuse network and S denotes a link between projects x and y in the social network. This probability can also be interpreted as a comparison of the density of the reuse network where social links exist and where they are missing. The density of a network is defined as the number of actual links over the number of potential links, and for the reuse network it amounts to 0.03331 for the socially linked projects which is by levels of magnitude larger than the same measure for the socially unlinked project which amounts to 0.0000913. Since these measures are calculated based on the real population data, any difference is statistically significant. The overall density of the reuse network is 0.000098, which is predictably situated between the two number. Additionally from the 340 cases of mutual dependency between projects, 196 of them coincided with developer co-affiliations in projects (0.57%), which confirms furthermore how knowledge flow is correlated with the mobility of the developers between the projects. Since the partial nature of the current

data only makes these findings more conservative, I conclude that the preliminary results do support Hypothesis 1.

The next steps in the analysis will comprise of constituting the complete social network and the two complete reuse networks using GitHub data and extracting the required network measure for Hypotheses 2, 3, and 4 using Pajek and UCINET. Using the dynamic network analysis methods the link established between the social network and the reuse networks can be verified for causal links, as the time data will make clear the order of evolution of the networks.

Discussion & Conclusion

Knowledge reuse and recombination are the main drivers of technological innovation and economic value added in modern economies (Schumpeter 1934; 1942). Determining the mechanisms underlying knowledge reuse is particularly relevant when assessing the innovative capacities of governance structures under specific economic regimes (Murray and O'Mahony 2007). As the new technologies make novel forms of organizing possible, it is vital to reiterate the question of how knowledge reuse is achieved under these hitherto unseen circumstances.

This paper looks into the issue of knowledge reuse, a well-researched topic in organization studies, and brings it to a context where there is a paucity of empirical evidence regarding the determinants of knowledge reuse. Previous empirical studies have documented the non-negligible role of network location and relations in both technical and commercial success of open source communities (Grewal et al. 2006; Hahn et al. 2008; Singh et al. 2011). There is little empirical evidence, though, that can testify the impact of social networks on knowledge flows in such communities. Deconstructing the myth of frictionless knowledge disclosure, access and reuse in open source communities; I suggest that the persisting knowledge search and processing costs are strong enough to localize knowledge and knowledge reuse.

The potential contributions of the current study are threefold. From an empirical standpoint, it contributes to research on knowledge transfer and reuse by identifying the determinants of knowledge reuse in a context where legal, technical and organizational barriers to knowledge reuse are virtually inexistent. With regard to theory, this study expands our understanding of the notion of knowledge stickiness (Hippel 1994), to demonstrate how knowledge is reused where it is produced, and how in absence of artificial barriers those who create knowledge still stand to benefit he most from the knowledge they have created. This can partially explain the involvement of commercial actors in open source projects, as it demonstrates how innovators are still best-placed to exploit their own innovations

By this I do not intend to prescribe open and cumulative intellectual property regimes over the proprietary regimes, but rather point out one of the basic mechanisms that makes open source partnerships viable for the firms. Given the increasing involvement of firms in open source projects and their investments in building communities and attracting developers (West and Gallagher 2006), this study can also provide the practitioners with first hand evidence of the network-level determinants of innovation and knowledge diffusion in these communities. More particularly the firms that pursue a razors-and-blades strategy, offering their core software or software libraries in open source and hoping for returns on value-added components and services, will be most interested in knowing how to lure the community towards building upon their software and libraries. The relevant findings of this study suggest that such firms may want to push their developer community to hold a plethora of indirect connections to as many open source projects as they can.

With regards to research methods, this study promotes the use of code reuse as a proxy for measuring knowledge reuse and transfer in open source communities. I argue that learning to extract and analyze such data can significantly change the type of questions we are able ask and answer about open source phenomenon, in the same way the public patent data has instigated a whole stream of research in the field of technology and innovation management.

References

Abrahamson, E., and Rosenkopf, L. 1997. “Social Network Effects on the Extent of Innovation Diffusion: A Computer Simulation,” *Organization Science* (8:3), pp. 289–309.

Almeida, P., and Kogut, B. 1999. "Localization of Knowledge and the Mobility of Engineers in Regional Networks," *Management Science* (45:7), pp. 905–917.

Leonard, A. 2001, February 15. "Life, liberty and the pursuit of free software," *salon.com*.

Argote, L., and Ingram, P. 2000. "Knowledge Transfer: A Basis for Competitive Advantage in Firms," *Organizational Behavior and Human Decision Processes* (82:1), pp. 150–169.

Argyris, C., and Schön, D. A. 1978. *Organizational learning*, Reading, Mass.: Addison-Wesley Pub. Co.

Arrow, K. 1959. *Economic welfare and the allocation of resources for invention*, (Rand Corporation,)Santa Monica CA: Rand Corporation.

Baldwin, C. Y., and Clark, K. B. 2006. "The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model?," *Management Science* (52:7), pp. 1116–1127.

Banker, R. D., Kauffman, R. J., and Zweig, D. 1993. "Repository evaluation of software reuse," *Software Engineering, IEEE Transactions on* (19:4), pp. 379–389.

Barabási, A. L. 2002. *Linked: The New Science of Networks*, *"A" Plume book*Perseus Pub.

Benbya, H., and Belbaly, N. 2010. "Understanding Developers' Motives in Open Source Projects: A Multi-Theoretical Framework.," *Communications of AIS* (2010:27)Association for Information Systems, pp. 586–610.

Boisot, M. H. 1998. *Knowledge Assets : Securing Competitive Advantage in the Information Economy: Securing Competitive Advantage in the Information Economy*, Oxford University Press.

Borgatti, S. P., and Halgin, D. S. 2011. "Analyzing Affiliation Network," In *The SAGE Handbook of Social Network Analysis,* SAGE Publications.

Brynjolfsson, E., and Smith, M. D. 2000. "Frictionless Commerce? A Comparison of Internet and Conventional Retailers," *Management Science* (46:4), pp. 563–585.

Burt, R. S. 1987. "Social Contagion and Innovation: Cohesion Versus Structural Equivalence," *The American Journal of Sociology* (92:6), pp. 1287–1335.

Burt, R. S. 2000. "The network structure of social capital," *Research in organizational behavior* (22), pp. 345–423.

Burt, R. S. 2009. *Structural Holes: The Social Structure of Competition*, Harvard University Press.

Cohen, W. M., and Levinthal, D. A. 1990. "Absorptive Capacity: A New Perspective on Learning and Innovation," *Administrative Science Quarterly* (35:1), pp. 128–152.

Coleman, J. S. 1988. "Social Capital in the Creation of Human Capital," *The American Journal of Sociology* (94), pp. S95–S120.

Conner, K. R., and Prahalad, C. K. 1996. "A Resource-Based Theory of the Firm: Knowledge Versus Opportunism," *Organization Science* (7:5), pp. 477–501.

Dahlander, L., and O'Mahony, S. 2010. "Progressing to the Center: Coordinating Project Work," *Organization Science* (22:4), pp. 961–979.

Demil, B., and Lecocq, X. 2006. "Neither Market nor Hierarchy nor Network: The Emergence of Bazaar

Governance," *Organization Studies* (27:10), pp. 1447–1466.

DiBona, C., Ockman, S., and Stone, M. 1999. *Open Sources: Voices from the Open Source Revolution*, Sebastopol, CA: O'Reilly & Associates, Inc.

Ducheneaut, N. 2005. "Socialization in an Open Source Software Community: A Socio-Technical Analysis," *Computer Supported Cooperative Work (CSCW)* (14:4), pp. 323–368.

Granovetter, M. 1985. "Economic Action and Social Structure: The Problem of Embeddedness," *The American Journal of Sociology* (91:3), pp. 481–510.

Granovetter, M. S. 1973. "The Strength of Weak Ties," *The American Journal of Sociology* (78:6), pp. 1360–1380.

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

Haefliger, S., Krogh, von, G., and Spaeth, S. 2008. "Code reuse in open source software," *Management Science* (54:1), pp. 180.

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

Hansen, M. T. 1999. "The search-transfer problem: The role of weak ties in sharing knowledge across organization subunits," *Administrative Science Quarterly*, pp. 82–111.

Hansen, M. T. 2002. "Knowledge Networks: Explaining Effective Knowledge Sharing in Multiunit Companies," *Organization Science* (13:3), pp. 232–248.

Hars, A., and Ou, S. 2002. "Working for Free? Motivations for Participating in Open-Source Projects," *Int. J. Electron. Commerce* (6:3), pp. 25–39.

Heinemann, L., Deissenboeck, F., Gleirscher, M., Hummel, B., and Irlbeck, M. 2011. "On the extent and nature of software reuse in open source Java projects," in *Top Productivity through Software Reuse*, Springer, pp. 207–222.

Hertel, G., Niedner, S., and 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), pp. 1159–1177.

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

Hippel, von, E., and Krogh, von, G. 2006. "Free revealing and the private-collective model for innovation incentives," *R&D Management* (36:3), pp. 295–306.

Kilduff, M., and Krackhardt, D. 1994. "Bringing the Individual Back in: A Structural Analysis of the Internal Market for Reputation in Organizations," *The Academy of Management Journal* (37:1), pp. 87–108.

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

Lakhani, K. R., and Wolf, R. 2001. "Does Free Software Mean Free Labor? Characteristics of Participants

in Open Source Communities."

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

Lynch, J., Jr, and Ariely, D. 2000. "Wine Online: Search Costs Affect Competition on Price, Quality, and Distribution," *Marketing Science* (19:1), pp. 83–103.

MacCormack, A., Rusnak, J., and Baldwin, C. Y. 2006. "Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code," *Management Science* (52:7), pp. 1015–1030.

Mockus, A. 2007. "Large-Scale Code Reuse in Open Source Software," Presented at *the Emerging Trends in FLOSS Research and Development, 2007*, pp. 7–7.

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

Murray, A. I. 1988. "A Contingency View of Porter's "Generic Strategies"," *The Academy of Management Review* (13:3), pp. 390–400.

Murray, F., and O'Mahony, S. C. 2007. "Exploring the Foundations of Cumulative Innovation: Implications for Organization Science," *Organization Science* (18:6), pp. 1006–1021.

Nonaka, I. 1994. "A Dynamic Theory of Organizational Knowledge Creation," *Organization Science* (5:1), pp. 14–37.

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

O'Mahony, S. C. 2007. "The governance of open source initiatives: what does it mean to be community managed?," *Journal of Management and Governance* (11:2), pp. 139–150.

O'Mahony, S. C., and Ferraro, F. 2007. "The Emergence of Governance in an Open Source Community," *The Academy of Management Journal* (50:5), pp. 1079–1106.

Olson, M. 1965. *The logic of collective action: public goods and the theory of groups*, Cambridge Mass.: Harvard University Press.

Polanyi, M. 1958. *Personal knowledge; towards a post-critical philosophy*, Chicago: University of Chicago Press.

Rosenkopf, L., and Almeida, P. 2003. "Overcoming Local Search through Alliances and Mobility," *Management Science* (49:6), pp. 751–766.

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

Rosenkopf, L., and Padula, G. 2008. "Investigating the Microstructure of Network Evolution: Alliance Formation in the Mobile Communications Industry," *Organization Science* (19:5), pp. 669–687.

Schumpeter, J. A. 1934. *The Theory of Economic Development*, Cambridge, MA: Harvard University Press.

Schumpeter, J. A. 1942. *Capitalism, Socialism, and Democracy*, New York, NY: Harper & Brothers.

Schwarz, N., Lungu, M., and Robbes, R. 2012. "On how often code is cloned across repositories," Presented at the *Proceedings of the 2012 International Conference on Software Engineering, Zurich, Switzerland*, pp. 1289–1292.

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

Singh, P. V., and Phelps, C. C. 2009. "Determinants of Open Source Software License Choice: A Social Influence Perspective," *SSRN eLibrary*.

Singh, P. V., Tan, Y., and Mookerjee, V. 2011. "Network Effects: The Influence Of Structural Capital On Open Source Project Success.," MIS Quarterly (35:4), pp. 813–A7.

Sojer, M., and Henkel, J. 2010. "Code Reuse in Open Source Software Development: Quantitative Evidence, Drivers, and Impediments.," *Journal of the Association for Information Systems* (11:12), pp. 868–901.

Stallman, R. 1985. "The GNU Manifesto," *gnu.org*.

Susarla, A., Oh, J.-H., and Tan, Y. 2012. "Social Networks and the Diffusion of User-Generated Content: Evidence from YouTube," *Information Systems Research* (23:1), pp. 23–41.

Tracz, W. 1995. *Confessions of a used program salesman: institutionalizing software reuse*, Addison-Wesley Pub. Co.

Uzzi, B. 1997. "Social Structure and Competition in Interfirm Networks: The Paradox of Embeddedness," *Administrative Science Quarterly* (42:1), pp. 35–67.

Walker, G., and Weber, D. 1984. "A Transaction Cost Approach to Make-or-Buy Decisions," *Administrative Science Quarterly* (29:3)Sage Publications, Inc. on behalf of the Johnson Graduate School of Management, Cornell University, pp. 373–391.

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

Williamson, O. E. 1975. *Markets and hierarchies : analysis and antitrust implications : a study in the economics of internal organization*, New York: Free press.

Zander, U., and Kogut, B. 1995. "Knowledge and the Speed of the Transfer and Imitation of Organizational Capabilities: An Empirical Test," *Organization Science* (6:1), pp. 76–92.