How Many Penguins Can Hide Under an Umbrella? An Examination of How Lay Conceptions Conceal the Contexts of Free/Open Source Software

Completed Research Paper

Introduction

The Free/Open Source Software (FOSS) development model has, without a doubt, transformed software development practices. At the industry level, FOSS has gained increasing significance to commercial activity (Hyatt & Mickos, 2008). Projects such as Linux, Apache, and Mozilla have proven to be serious competitors for industry leaders (Asay 2013; Phipps 2012). Furthermore, the FOSS production model is steadily gaining ground in the production of information-based and cultural goods (Hahn et al., 2008) while the FOSS model of innovation is diffusing into a variety of other fields and industries such as technical design, pharmaceuticals, or biotechnology (von Krogh and Spaeth, 2007). It has also diffused into a wide array of domains, spawning commons-based peer production initiatives such as Wikipedia or OpenStreetMap.

Since its inception, the FOSS movement has been a fertile terrain for research in a wide array of disciplines and fields of the social sciences. By examining the looming transformation of the information systems field as well as the overall impact on our society caused by the success of FOSS (von Krogh and Spaeth, 2007), IS researchers have generated a bulk of theoretical knowledge through the investigation of a wide array of aspects of the phenomenon and by using a diversity of research approaches and methods (Chengalur-Smith et al., 2010, 2013; Stewart & Gosain, 2006; Singh et al., 2011). For example, the AIS basket of eight journals, considered to be the flagship of our theoretical knowledge, has published 59 articles on FOSS since the beginning of FOSS research in IS (as of 2015). Furthermore, the multi-faceted nature of the FOSS movement combined with the trans-disciplinary nature of information systems (Galliers, 2003) have made the IS field a perfect candidate for engaging into a trans-disciplinary dialog and placing IS research in a position to play a central role in producing a cumulative body of high-quality FOSS research (von Krogh and Spaeth, 2007).

At the turn of the century, skeptics realized that this peer-based mode of software production could outperform the bureaucratic mode of software production, leading to higher quality (Aberdour, 2007) and more reliable software (Raymond, 1999). At the time, the Linux kernel project, as the landmark of the FOSS landscape (Bergquist & Ljungberg, 2001; Gallivan, 2001; Payne, 2002), was perceived by some observers as a magical black box “out of which a coherent and stable system seemingly emerge only by a succession of miracles” (Raymond, 1999).

The opacity of the FOSS movement and the hyped public discourse about the movement has engendered a tendency to underestimate the complexity of the FOSS reality. Besides, the inherent fast-changing nature of FOSS development has also rendered inaccurate or even obsolete some of the results of our cumulative body of FOSS research. Since its beginnings, the FOSS phenomenon has been a ‘moving research target’ for IS researchers. It has now with no doubt transformed into a rather mainstream and diverse form over time (Fitzgerald, 2006). We have come to a point where conducting high-quality FOSS research has become a challenging task as we need to re-examine our body of FOSS research and clearly distinguish the results that hold from those that do not. The urgency of such need can be perceived when realizing that beliefs in FOSS practices that were artefacts of the FOSS reality in the past are increasingly pervading current FOSS research. For instance, despite the widespread belief that the majority of contributors to the Linux kernel project are unpaid volunteers, only 11.8% of kernel development in 2014 was done by unpaid volunteers, the remainder being paid employees of technology organizations (Gold 2015).

The purpose of this paper is to highlight some of the inaccurate or erroneous beliefs that have permeated IS academic scholarship, and to raise awareness of their consequences for how we propose, test, and falsify theories about the FOSS phenomenon. Despite considerable progress in our collective academic understanding of the FOSS movement, our analysis of the IS literature since 2001 suggests that we have to pay greater attention to the variegated contexts in which the FOSS movement operate. By context, we borrow Johns' (2006, p.386) conceptualization, who defines it as "situational opportunities and constraints that affect the occurrence and meaning of organizational behavior as well as functional relationships between variables." In this research, we adopt Gacek and Arief's (2004) viewpoint that considers that a software project can be categorized as 'FOSS' as long as its source code is released under either an Open Source[1] or a Free Software[2] license. Our recommendations for future research include carefully conceptualizing the FOSS projects under study, heeding misspecification and sampling biases, delineating clear boundary conditions of theoretical claims, relying upon empirical data richer than archival datasets of code repositories, and keeping abreast of the unrelenting changes in how the FOSS movement operates. Our contribution is to prime IS researchers to the pitfalls that can impede cumulative advances in generating knowledge about the FOSS movement.

This paper starts by describing the most widespread beliefs about FOSS development in IS research, that have induced conceptual or methodological pitfalls in FOSS research. While doing so, we argue that the notion of 'FOSS project' is an umbrella term which conceals a complex reality. Second, from a careful examination of the current body of FOSS research in the top IS journals since 2001, this paper presents the main theoretical and methodological issues that have arisen in current and past IS research. Finally, we conclude by providing our recommendations for future FOSS research that we hope, will help in the development of a sound and strong body of theoretical knowledge on FOSS development.

The FOSS Umbrella

> *The term "open source" frequently refers to a software development process that relies on the contributions of geographically dispersed developers via the Internet. One basic requirement of an open source project is the availability of its source code, without which the software's development or evolution is difficult, if not impossible. But apart from these characteristics, some confusion exists on what an open source project actually is. (Gacek & Arief, 2004, p.34)*

Eric Raymond's (1999) treatise, "The Cathedral and The Bazaar", still resonates today with many academics and scholars because of its thought-provoking exposition of the FOSS movement. The "great babbling bazaar" metaphor he introduced still underlies lay perceptions of FOSS projects, in which they are considered loosely organized communities of volunteers located around the world, kept together around common values, and working over the Internet on a software project in which inputs and outputs are treated as public goods. In the early 2000s, the growth of this novel mode of software production attracted the curiosity of IS scholars, many of whom took on the task of investigating the inner workings of flagship projects of the FOSS movement, with most of their attention focused on Linux and Apache (Crowston, Wei, Howison, & Wiggins, 2012). By adopting an "inward" view of FOSS, IS scholars attempted to learn lessons from the practices that made FOSS projects effective and successful (Fitzgerald, 2006, p.588). A common conclusion of theirs was that the FOSS movement had moved beyond Raymond's (1999) original metaphor. On the one hand, Crowston et al. (2012) found FOSS projects to be organized and managed in a great diversity of ways. On the other hand, Fitzgerald (2006) observed an ongoing fusion of commons-based and proprietary software production modes. Despite the calls by Fitzgerald (2006) and Crowston et al. (2012) for a more nuanced examination of the various contexts in which FOSS projects operate, it appears that some of their important recommendations have gone unheeded by IS scholars.

In this section we examine some of the most enduring beliefs about FOSS development that tend to resonate in today's research. We begin by questioning the popular conception story grounded in the initiative of a lone developer. Second, we inspect the fragile foundations of the notion that all contributors progress from the periphery to the core a FOSS projects in a linear fashion. Third, we challenge the accuracy of the lay conception that most contributors to FOSS projects are unpaid. Finally,

[1] http://opensource.org/osd

[2] http://www.gnu.org/philosophy/free-sw.en.html

we challenge the commonly held assumption that the openness in terms of intellectual property license transfers into the same openness in terms of project governance, structure, and management.

Belief 1: FOSS project inception: 'Itching the itch' of a lonely developer

"Every good work of software starts by scratching a developer's personal itch." (Raymond, 1999, p.3)

In the Cathedral and the Bazaar, Raymond (1999) speculated on the typical way FOSS projects start. In this scenario, a single developer perceives an opportunity and designs an initial prototype which subsequently attracts a large number of globally distributed developers with different skills and fields of expertise (Fitzgerald, 2006).

Yet, since the very beginnings of the FOSS development movement, the individual recognition of an opportunity has been one of many ways in which FOSS projects can start. More and more projects are not initiated by single developers but rather by organizations that perceive strategic opportunities in opening up their platforms and software development activities (Bonaccorsi, Ginannageli, & Rossi, 2006). For instance, OpenStack is a FOSS cloud computing software platform that was launched in 2010 through the collaboration of NASA and Rackspace Hosting, a vendor specialized into cloud computing services. Other projects may start from the full or partial release of an existing proprietary software artifact. A famous case is the Mozilla Firefox project that started for economic and strategic reasons when in 1998, Netscape Communications released the source code of its proprietary web browser Netscape Navigator to the public. A key motivation behind the decision was to react to the market share drop caused by the growing success of Microsoft's Internet Explorer (Edwards, 1998).

A similar assumption concerns the origin of a FOSS project which can be traced back to a single point in time and space. Yet, the Byzantine origins of the projects OpenOffice and LibreOffice are an exemplar of the complexity that can accompany the birth, life, death, and rebirth of FOSS projects. Star Division owned a proprietary office suite, StarOffice, which StarDivision began offering for free in 1998. Sun Microsystems bought StarDivision in 1999 and released the source code in July 2000, creating the FOSS office suite called OpenOffice.org. In 2009, doubts about Oracle's intentions led to forking the source code of OpenOffice.org in 2010, leading to the creation of LibreOffice, managed ever since by a German non-profit organization called The Document Foundation. Finally, in 2011, in reaction to the feeble market share of Oracle Open Office, Oracle donated the project to the Apache Software Foundation, giving birth to Apache OpenOffice. Such tortuous paths conceal contextual factors of theoretical import which can only be revealed when analyzing FOSS projects' inception and evolution through a biographical lens (e.g. Williams and Pollock, 2012).

Belief 2: The linear and ineluctable career trajectory of contributors

A popular stylized fact about FOSS projects concerns the typical career trajectory followed by contributors, which is said to involve a move from the periphery to the core of a project, associated with an increase in coding and governance responsibilities (Crowston and Howison, 2005). FOSS project structure is often understood in equivalent terms in which each layer represents a distinct role ranging from: passive users, active users, peripheral developers, co-developers, and core developers (within which are included initiator and release coordinator) (Nakakoji et al., 2002; Ye & Kishida, 2003). In this onion-like model, the outer layer has a "virtually unknowable boundary" (Crowston & Howison, 2005, p. 7) consisting of passive users of the software who do not contribute on either the project's discussion lists or forums. While this onion-like model has gained currency in explaining and predicting the progression of coders in a given FOSS community, it also conceals important aspects of how FOSS projects operate.

The first limitation of the model is that a great number of FOSS projects are of modest scale. On the one hand, a FOSS project can consist of a single person uploading a project's code on GitHub or Sourceforge and working alone on it. On the other hand, large projects such as Ubuntu or KDE can have thousands of contributors. The way a handful of programmers work on a software project cannot be transposed to the way a community of thousands of individuals collaborates on a project that has a complex governance structure. Yet, even for projects of considerable scale, the onion-like model hypothesizes that a new contributor, starting as a user, has to work her way through all layers of the onion model in order to reach the core developer level. By studying the concurrent versions system (CVS) used within the GNOME community, Herraiz et al. (2006) found that there was no typical joining pattern for new members, a finding which indicates that the validity of the onion model might be limited to specific contexts.

The second limitation is that once a project reaches a critical threshold of complexity and size, the division of labor goes well beyond bug reporting, patch submission, or simply writing code. When a FOSS project gains the support of an active community of developers, contributors may therefore become involved in one or several roles which go beyond technical activities that shape the software artifact (Gacek & Arief, 2004; Scacchi, 2007). Research efforts have identified a variety of roles that are performed within a FOSS community (Jensen & Scacchi, 2007). In studying role migration within the Mozilla, Apache, and NetBeans communities, Jensen and Scacchi (2007) identified the presence of a wide set of tasks such as quality assurance roles, source code versioning roles (e.g. CVS manager, CVS committer, etc), project planning, usability, licensing, and marketing roles. People can also take responsibility for multiple roles at a time. In a FOSS project, nothing prevents an experienced contributor in taking part in the marketing campaign surrounding the project, be the community manager, be a mentor during the Google Summer of Code programme, as well as handle release management tasks, and all at the same time. As a FOSS project matures, organization-building contributions take on increased importance over technical contributions in one's progression toward the top of its leadership ladder (O'Mahony & Ferraro, 2007). In addition, the structure of roles in FOSS projects has to adapt to the ongoing flow of demands they face. It was shown that the inner core of a FOSS project does not always persist for long periods of time; new generations take the lead over after a certain amount of time (Herraiz et al., 2006).

In sum, the onion-like model assumes a linear and ineluctable career trajectory that omits all the non-code related tasks and assumes that people have single roles with a given project. The static onion-like model is in need of theoretical refinement in order to take into account better dynamic changes to role structures. It should be considered as what it is, a model with hypotheses which demand empirical validation, rather than as a taken-for-granted logic that all FOSS projects follow across contexts.

Belief 3: Contributors are unpaid and work for free

> *"...the open-source movement is about making unpaid contributions to software products. Unpaid!"* (Glass, 2000)

A common lay perception about the FOSS movement is that contributors are unpaid. The romantic idea that FOSS project contributors work for free out of passion or for some ideology shows enduring stickiness. A growing body of evidence shows that this perception is inaccurate. Numerous studies have found that the range of paid contributors vary, on average, from 30 and 40 percent (Hertel et al., 2003; Lakhani & Wolf, 2005; Luthiger & Jungwirth, 2007). At the low end and top end of the range studies have found numbers ranging from 16% (Hars & Ou, 2002) to more than 50% of contributors being directly or indirectly paid (Ghosh et al., 2002). More recent evidence comes from the latest Linux Kernel Development report, which announced that 4 000 developers worked on the latest 3.18 release, of which 80% are paid by their organizations to contribute to the Linux kernel: Red Hat, Novell, Intel, and IBM being the most important sponsors. On a similar note, the GNOME census project gathered data about developers of the GNOME 2.30 release (Neary & David, 2010). It reports about 30% of paid developers in the GNOME project (Red Hat, Novell, Collabora, and Intel being the largest contributors). The amount of contribution by paid developers appears quite substantial, representing 70% of the total number of commits.

This belief has its origins in the ambiguity of the term volunteer, which can have two distinctly orthogonal meanings. The term volunteer is sometimes used to mean someone who works *without payment, for free*. It is also used to mean someone who joins a FOSS project *out of her own volition, that is, someone who does not have an employment relationship with the project she joins*. This confusion was present in Raymond's (1999) treatise, for instance, when he mentioned that "open-source developers are volunteers, self-selected for both interest and ability to contribute to the projects they work on (and this remains generally true even when they are being paid a salary to hack open source.)" (p. 28).

Additional important distinctions concern the *source of payment* and the *type of payment* for paid contributors. Most often, payment does not come from the FOSS project in which they participate, although there are exceptions. Instead, payment comes from the employing organizations which subsidize the person's contribution to the FOSS project. Berdou's (2006) work on the KDE and GNOME projects, identified four assignment categories when organizations pay employees to work on a given FOSS project: free sponsorship, clear mandate, FOSS project' "friendly" jobs, and sub-contracting. Additionally, a wide range of payment types is used to compensate contributors: donations, subscriptions, salary, direct payment for service, and software sales among others (Krishnamurthy, Ou, & Tripathi, 2014).

The questions whether contributors work for free (vs. paid) and out of their own volition (vs. assigned) influence the nature of their motivation to participate in a FOSS project (von Krogh & Spaeth, 2007), and thus has theoretical and practical importance. When contributors are paid in some form, the lack of reported information about the source and the type of payment they receive also impedes theoretical progress, since such arrangements affect the incentives driving contributors' behavior. Such clarifications are left unaddressed by IS researchers when they use the term *volunteer* loosely.

Belief 4: FOSS projects are open communities

A final lay belief about the FOSS movement is that the same openness governing the license of a project's source code is found in the project's practices and structure. This belief also stems from the utopian undertones of the discourse that engulfs the FOSS movement (Kreiss, Finn, & Turner, 2011). According to this belief, anyone can join a FOSS project. In addition, FOSS projects are said to follow egalitarian, democratic, and transparent practices to govern member relationships and make key project decisions. FOSS projects are thus often portrayed by the popular press as the ultimate post-bureaucratic form of organization, a form which liberates workers from arbitrariness, red tape, and alienation.

Yet, what is observed in reality differs significantly from this point of view. Nothing prevents a group of developers working together on a FOSS project to ignore all people that could be interested in contributing. The popular Linux distribution called Gentoo, for instance, relies on a quasi-bureaucratic recruitment process in which recruiters identify and mentor people that are perceived as potentially 'good contributors' for Gentoo. The Debian project is another example in which joining is everything but 'open.' The project relies on a complex and structured sponsorship-based joining process requiring formal mentoring and online tests to be taken. Newcomers must follow a detailed step-by-step process at the end of which they become official contributors. Such practice is a way to bridge the gap between community newcomers and experienced members, and constitute specific instances of rites of integration (Trice & Beyer, 1984).

The governance of FOSS projects has been categorized in three ideal types, which have been called defined, open, and authoritarian (Di Tullio & Staples, 2013). The defined communities have bottom-up goals and decentralized management, but their software development process is tightly controlled by release managers. In contrast, open communities have bottom-up goals and decentralized management, but software development processes are informal and loosely managed. In authoritarian communities, project decisions and role assignment are determined by a core group of project administrators, and the rules governing the software development process are explicit. This later governance ideal type is fairly common; for instance, it accounted for nearly a third of the sample of FOSS projects examined by Di Tullio & Staples (2013).

Authoritarian communities are usually led by a 'benevolent dictator for life' (Ågerfalk & Fitzgerald, 2008; Mockus et al., 2002), who is usually the person that started and 'owns' the project. This person is often the final arbiter of all project-related decisions and may drive the project as undemocratically and idiosyncratically as he or she wants to. A FOSS license guarantees certain 'open' rights that pertain to the project code, but such rights do not apply in any way to how a community has to be managed. For instance, development in the Ubuntu project is led by a private company. The project is managed by a self-appointed benevolent dictator for life but community management and conflict resolution is assured by the Ubuntu Community Council. In contrast, the Debian project is known for its structured and organized governance which approximates the defined community ideal type of Di Tullio & Staples (2013). Debian's 'social contract' describes the core principles that regulate the Debian project and its members, whereas Debian's 'constitution' presents the organizational structure of the project, making explicit how decisions are made and conflicts tackled within the project. A formal election system is in place in order to select the project leader on a yearly basis. Since its creation in mid-1993, 13 different people have been appointed leader of the Debian project.

In addition, lay conceptions of FOSS project governance necessarily assume the use of normative and peer-based controls in lieu of formal controls (Stewart & Gosain, 2006). The above examples of the Debian and Ubuntu projects question such idealized view of the forms control may take in FOSS projects. In fact, Di Tullio & Staples (2013) identified 19 distinct control mechanisms used to govern FOSS projects. O'Mahony & Ferraro (2007) showed that the predicted demise of the bureaucratic forms of governance has not come to pass since FOSS projects exhibit a blend of democratic and bureaucratic forms of governance, a blend which simultaneously confers authority upon the project's leaders and establishes constitutional limits on its use. As a result, FOSS projects are still subject to the expression of power through organizational control mechanisms; the mechanisms have simply morphed into new, less visible forms (Barker, 1993; Kreiss, Finn, & Turner, 2011).

An introspective look within the IS body of FOSS knowledge

In this section, we examine the current body of FOSS research in IS for residues of the beliefs we just discussed. We assess their consequences for the IS academic community's ability to accumulate knowledge about the FOSS movement. To do so, we reviewed the content of FOSS-related papers in the Association for Information Systems (AIS) basket of eight journals.[3] Only the papers that investigated phenomena that were directly related to FOSS development were retained, leading to selecting 39 research outlets out of the 59 that were found to address some FOSS-related issue. The AIS basket of eight is considered the core research outlet of the IS academic community and a salient feature of its identity to neighbor disciplines. Our goal in conducting this selective review was not to provide an exhaustive census of the FOSS literature, which now cuts across many academic communities. Such important service to the IS community has already been done in recent years (e.g., see Aksulu and Wade, 2010; Crowston et al., 2012; Hauge, Ayala, & Conradi, 2010; Nelson, Sen, & Subramaniam, 2006). Instead, the objective of our critical review (Paré, Trudel, Jaana, & Kitsiou, 2015) is more modest in its scope: to highlight the subtle ways in which lay conceptions of FOSS may hinder theoretical progress in our collective understanding of the FOSS movement, and to identify opportunities for future research.

Observation 1: FOSS projects as black-boxes

Construct clarity is essential for theoretical progress (Bacharach, 1989; Suddaby, 2010). Clear constructs provide the foundations of strong theory. When they are parsimoniously defined, they provide researchers with the ability to make distinctions between concepts. Clear constructs also guide researchers in the task of transforming abstract notions into observable indicators. They also allow researchers to establish the boundary conditions of theories, that is, to identify the contexts in which the predictions from a theory may or may not apply. Much of the recurring angst concerning the lack of strong theory in IS research comes from the lack of clarity about some of our discipline's core constructs. For instance, the IT artifacts studied in our research has been repeatedly reported as lacking in conceptual refinement and clarity (Orlikowski & Iacono, 2001). As we have shown in the first part of the paper, FOSS projects have multiple manifestations that span diverse contexts. We thus first examined how IS researchers have defined and conceptualized "FOSS projects" in their studies.

We observed that FOSS research published in information systems journals tended to black-box the construct of "FOSS project." Our review revealed that in 34 articles out of 39, 'FOSS projects' were treated as an umbrella entity that encompassed all ways such projects can be organized and structured. In these studies, FOSS projects were defined in a coarse manner or, in some instances, left undefined. Less than 50% of the studies (18 studies) provided a definition of what was considered a FOSS project. A majority of studies defined a FOSS project following a combination of the criteria that it is: (1) any kind of software project producing software under a FOSS license, and/or (2) any kind of software project following a FOSS development methodology. We found a typical instance of such definition in Sojer & Henkel (2009, p. 870): "Strictly speaking, software is OSS if it comes under an open source license [...] Since much OSS

[3] We used keyword searches for "FOSS" and its variants ("open source," "FLOSS," "free software," for instance) in the abstracts and full-text of all issues published between 2001 and 2013. We identified 39 articles: 32 articles are empirical while 7 are conceptual. The Appendix presents the list of papers examined.

is developed by informal collaboration in public OSS projects [...] the term "OSS" is often also understood to imply that the software has been developed in the OSS fashion."

Despite the critical differences between projects claiming allegiance to either the free software movement or the open source movement (Chou & He, 2011; Scacchi et al., 2006), we found only 5 studies that clearly made a distinction between both project types in specifying which type they were investigating. This lack of construct clarity limits any generalization of findings about human behavior in FOSS project, because the free software and the open source movements are governed by clashing beliefs, values, and practices (Stewart & Gosain, 2006; von Krogh et al., 2012).

Some exceptions to the blanket labeling of FOSS projects were found nonetheless. Three studies narrowed the scope of their investigation to FOSS project subtypes (community-based FOSS projects, FOSS projects involving cross-project coordination, internal FOSS projects). Two other studies focused on very specific entities: heterogeneously-licensed software systems.

Interestingly, 14 studies studied FOSS projects as an instance of a broader, more abstract type of collective form of human organization. These studies drew upon a broad range of theoretical perspectives to establish their conceptualization of FOSS projects. They considered FOSS projects as instance of virtual organizations (3 studies), open innovation communities (2), virtual teams (2), knowledge firms (1), global distributed collectives (1), online collaborative networks (1), communities of practice (1), online communities (1), peer production communities (1), and organized volunteering forms (1). While this multiplication of labels has the obvious benefit of relating what happens in FOSS projects to other forms of human organization, it also complicates a researcher's task of establishing the boundaries of the claims she makes.

Overall, the lack of clear definitions of the type of FOSS projects studied ultimately has important consequences for theoretical progress and research relevance. It makes it harder to build upon each other's theoretical contributions, since the object of study is ambiguous. In other words, it is sometimes fair to ask if we are actually studying the same phenomenon. It also makes it difficult to translate our findings into consumable research for our stakeholders, who may brush off our findings judging that they don't concern them.

Observation 2: Selective sampling biased towards forges

One of the goals of scientific inquiry is to produce theories that generalize to populations (Tsang & Williams, 2012). An omniscient researcher equipped with profuse resources would have the luxury to test a theory in all empirical settings in which a theory is hypothesized to hold. Unfortunately, we have to rely upon the fallback strategy of identifying a small and selective sample most of times. Consequently, we use an apparatus of inferential statistics to make probabilistic statements about the extent to which findings from our samples can be transposed to larger populations (Aguinis & Vandenberg, 2014). This practice is not without pitfalls, and some criticism is occasionally expressed regarding the tendency of IS researchers to over-claim the generalizability of their results (Seddon & Lyytinen, 2008).

There are two basic ways in which selective sampling can introduce bias in research findings (Denrell & Kovács, 2008). The first is selective unit sampling, which is akin to sampling on the dependent variable. In this scenario, the researcher samples only a limited number of units from the sample, and these units exhibit a systematic trait which has theoretical import. For instance, a study of FOSS projects would exhibit unit selection bias if only FOSS projects that grew over a certain size, measured in terms of contributors or volume of activity, were included in the study from a given sampling frame.

The second way bias can be introduced is by selecting unrepresentative settings. This problem occurs when the researcher relies upon sampling frames that are unrepresentative of the population. Even if all units contained in the sampling frame are given an equal chance of getting selected, bias is still introduced since the frame from which they are drawn does not approximate, in some systematic fashion, the attributes of the population. Note that while the source of bias is distinct, both ways of conducting selective sampling can simultaneously bias a study's findings. It is this second type of bias, created by unrepresentative sampling frames, which we observed in our review of FOSS studies published in information systems journals.

Since the beginnings of IS FOSS research, web-based code repositories, also called 'forges' in FOSS vernacular, have provided attractive sampling frames for FOSS researchers. Such code repositories provide free access to large amounts of data and give the possibility to mine the software project artifacts (Howison & Crowston, 2004). We observed that out of the 32 empirical papers about FOSS development from the AIS basket of eight journals, 18 relied on data from the Sourceforge platform. Sourceforge is a web-based source code repository, one of the earliest to host the code and technical artifacts of FOSS projects. Given the vast number of platforms on which FOSS projects can be hosted nowadays, this was a surprising finding, for a number of reasons.

In a study that questioned the quality of data gathered on Sourceforge projects, Rainer and Gale (2005) studied a sample of 50,000 projects and found that not even 1 percent were active. Average activity was found to be extremely low in terms of discussion (Krishnamurthy, 2002), bug reports, and feature requests (Rainer & Gale, 2005). A proportion of forge projects were found to be either student projects or the outcome of some developer experimentation (Wen et al., 2013). Other research concluded that the average number of contributors per project was between one and two individuals (Krishnamurthy, 2002; Rainer & Gale, 2005). Rainer and Gale (2005) concluded that a vast majority of projects on the forge are 'impulse projects' that did not raise the interest developers and users. If care is not taken to weed out such projects, researchers will have a high chance of including projects that are not actual FOSS projects in their datasets, biasing their study's findings.

There has also been criticism about the extent to which forge projects, more particularly Sourceforge ones, are truly open source, from a strict definitional perspective. In its early launch, Sourceforge was considered to be a showcase for FOSS but criticism was raised shortly after realizing in the early 2000s that the Sourceforge platform was running on proprietary software. FOSS advocates started recommending projects to move away from the hosting site. This tendency amplified as of 2012 when Dice Holdings acquired Sourceforge in 2012, requesting projects to be downloaded using a closed source installer that prompts users to install other proprietary software, a decision at odds with the libertarian beliefs held by some FOSS participants.

Moreover, Sourceforge is only one among the many forges where a concentration of FOSS development takes place. Popular alternatives include GitHub, Bitbucket, Launchpad, Assembla, Codeplex, and GNU Savannah. In a study providing a classification of FOSS forges, Squire and Williams (2012) reported 25 distinct forges with a number of hosted projects ranging from several hundred (e.g. Objectweb) to over ten million (GitHub). It thus appears that our accumulated body of knowledge is biased toward the narrow context of projects that are hosted on the Sourceforge platform, leaving a wide range of other representative contexts of FOSS projects unexplored.

Observation 3: Reliance on code repositories for measurement

Our third observation is that published FOSS research in information systems journal has heavily relied upon the data contained in code repository as indicators of the constructs they were investigating. In comparison to many other domains of IS research, FOSS research benefits from an environment where all software artifacts and all activity logs are freely accessible (downloadable) online. In an academic world in which researchers are often directly or indirectly pressurized to produce research outlets, the availability of FOSS project data can be seen as an attractive research opportunity. Although this exclusive reliance on archival data could be considered a form of selective sampling bias by some, it is instead a source of bias introduced by the choice of measurement instrument.

By relying upon such digital traces of activity one makes at least two assumptions. First, one assumes that all theoretically relevant activities can be captured by code repository platforms. Second, one also assumes that all contributors to a FOSS projects are active on the platform. Assimilating the population of FOSS project contributors to the developers who contribute code could be appropriate for projects of lesser size. However, this is a lot less true in larger projects. In such communities, FOSS project leaders have to daily coordinate a lot of non-code related activities such as advocating, marketing, translations, and newcomer welcoming and mentoring. These non-code related activities often influence the dependent variables of interests to IS researchers (e.g. project success, project sustainability).

It is sometimes legitimate to narrow down the scope of contributors to developers if one is studying a phenomenon that occurs within the software engineering domain. If indicators for all the constructs that are considered for a given study can be logically derived from the code repository dataset, then the use of such archival data does not introduce bias. Nonetheless, there are many contexts in which the two abovementioned assumptions lie on shaky grounds. For instance, when researchers attempt to shed some light on more project-encompassing issues such as project success or sustainability, the assumption that all relevant activities and contributors are captured by the code repository is untenable. Large projects, more specifically, have a large pool of non-developers (in addition to users) without whom they could not survive and sustain.

Furthermore, FOSS project success has been often measured by the total number of software downloads for a given period of time (e.g. Daniel et al., 2013; Subramaniam, Sen, & Nelson, 2009). A threat to the internal validity of models of FOSS project success lies in the omitted variables that stem from the activities of non-developers. Few studies have considered controlling for the activities of non-developers, who can increase the likelihood of project success by spreading the word about a project through online, advocating, talking at FOSS conferences, networking at FOSS events… Such activities raise the visibility of the FOSS project and therefore could have generated a significant number of downloads, irrespective of the software development process followed. For instance, the spread of the Linux operating system is inseparable from the contributions provided by an enormous number of non-developers: advocates, designers, marketers, translators, as well as the involvement of commercial organizations whose business model depend on the success of the project (Fitzgerald, 2006). It indicates the existence of a vast unexplored research territory: finding ways to understand the FOSS reality by encompassing all code and non-code related contributions from contributors, people and organizations-alike.

Observation 4: Context matters

Considering the uniqueness of every single project and their dynamic nature, the systematic understanding of the socio-historical context (using Johns' (2006) conceptualization of the notion of context) surrounding each investigated FOSS project is crucial. In FOSS projects, there are many aspects of context that have theoretical relevance, many of which we have discussed in the first part of this paper. Situational opportunities and constraints may reside at an upper level of analysis, in events that occur in the path-dependent history of the project, or in the dependence relationships of a project with its external stakeholders for instance. Attention to context is important, because it thwarts threats to internal and external validity. Context may explain why a given theoretical relationship does or does not hold, as well as the form it may take. Taking into account context also makes it easier to establish the boundaries of a given theoretical claim.

From the papers we reviewed, we observed that contextual factors were not often given the attention they deserved. For instance, we found a great variation in terms of the ratio of volunteer/paid contributors, ranging from zero to 100% volunteer contributors per project. Yet, it was surprising to identify eleven studies, all having been published between 2006 and 2013, which did not clarify what the nature of the "volunteering" tasks involved for the projects they considered. Another lack of attention to context involved the assumption that FOSS projects can encompass for instance, two students hosting their computer science project on Sourceforge (usually for convenience reasons), ten fervent free software defenders working on creating a FOSS substitute to a famous proprietary web browser, and a large project in which hundreds of professional developers are paid by their company to contribute.

Another source of bias introduced by a lack of attention to context involves left-censorship, or the assumption of "zero-time." This assumption considers that the history of projects that occurred before the start of a given's study does not matter. Yet, a large number of projects that started more than two decades ago and which life has been punctuated by a series of events and releases (often by the tens or hundreds). The Linux kernel is obviously a good exemplar with its 0.01 release dating from 1991 with about then thousand lines of code and the latest 3.10 release (in 2013) with nearly 16 million lines of code. Larry Wall released the first version of the Perl programming language back in 1987 while Guido van Rossum announced the first release of Python in 1991. The Linux distribution Debian started its existence in 1993, the GIMP graphics editor was launched in 1996, and the Sourceforge-hosted project phpMyAdmin was initially released in 1998. The life of each project must be taken into consideration when studying a given project as for instance, each software release has its own context and history.

Each event that happened during the life of a FOSS project has an impact on the development processes and also the human layer that inherently surround every single project. More importantly, there is a significant impact of such project changes on the social practices that surround the software development processes. The Linux kernel project moved from a monolithic architecture in which every change is controlled and validated by the project leader (Linus Torvalds) to a more modular form where control is exerted by the project leader only within a central component (the core of the Linux kernel project). The impact of such management and governance shift led an important leap forward in terms of lines of code being written, number of functions being added, average cyclomatic complexity (Caprio et al., 2001), and shortened development time (De Goyeneche & De Sousa, 1999). The functioning and structure of the community that surrounded the project at that time was impacted with the addition of an additional layer in the project division of labor. 'Module maintainers', having full control over implementation and design details within a module, then appeared. This is just one example of the many events that punctuate the life of FOSS projects.

In short, the often long and tormented life of FOSS projects makes it critical for FOSS researchers to bear in mind that FOSS 'project' IT artifacts are not static but rather living and shifting entities. They shall always be seen as 'embedded in some time, place, discourse, and community' (Orlikowski & Iacono, 2001, p.131).

Recommendations for future FOSS research and conclusion

We now turn our attention to the repairs and future lines of inquiry that IS researchers can adopt in light of the issues that we observed. All our recommendations have the overarching goal of increasing awareness of how lay conceptions (beliefs) of how the FOSS movement operates can slip into our studies, and if left unchecked, can lead to some conceptual or methodological biases. We believe the following recommendations can help derive an updated and contemporary view of the FOSS phenomenon.

A first opportunity lies in expanding the sources of data on which FOSS research relies. This should allow FOSS research to tap into a larger range of contexts, and thus, avoid the problems associated with unrepresentative sample frames (Observations 2 and 4). One obvious candidate for such expansion is to replicate existing findings on the GitHub platform. GitHub was launched in 2008 and was built around Git, a decentralized version control system created by Linus Torvalds. Announced as "the next big social network" by Forbes, GitHub provides social networking functionalities (such as feeds, followers, wikis, social network graphs) in addition to allowing programmers to collaborate around repositories of code that can be downloaded, forked and/or committed back in all possible formats without restrictions. GitHub has experienced lightning growth since its creation. In 2010, GitHub hosted over a million repositories, and now hosts over 11 million projects involving more than 5 million contributors (as of 2014). In comparison, SourceForge has about 3 million registered users (as of May 2013). Even though GitHub is not such a new phenomenon, it is striking to realize that the flagship of our FOSS theoretical knowledge does not contain a single research article using data from GitHub. From a theoretical viewpoint, the evolution of FOSS development into 'social programming' practices is a major change and deserves the attention from researchers.

Taking a step further into our reflection about the fast-changing nature of the FOSS reality and the important methodological constraints this exerts when conducting FOSS research, this paper calls for new and alternative research methods. For instance, *netnography* (Kozinets, 2015) is a well-documented and acknowledged approach for conducting ethnography research on the Internet developed by Robert Kozinets since the late nineties. It is a qualitative, interpretive research methodology aiming at studying social media by adapting traditional ethnographic techniques. It appears as a potential candidate for providing insightful results when studying the new turn taken by FOSS development: social programming.

Second, researchers should strive to acquire contextual knowledge about the FOSS projects from which data is collected (Observation 4). This will in turn allow to identify potential cross-level effects that can impact the phenomenon being studied. Replication can also play an important role in the development of a sound and solid body of FOSS knowledge, allowing to tackle Observation 1, boundary conditions being too often not clearly defined in FOSS research. Indeed, the development of a body of knowledge within mis-specified theoretical boundaries is a serious threat to sound knowledge generation as it is a direct obstacle for replication. On the contrary, FOSS research shall be a ground where replication shall be encouraged due to the inherent theoretical variability among all FOSS development initiatives.

Investigating a phenomenon within a small and confined domain and gradually extending the validity of the results through replication is a much sounder approach rather than over-generalizing the results of a study to a broad domain without any theoretical justification or empirical evidence.

For instance, a researcher ignoring the shift from a monolithic to a modular architecture in the Linux kernel while studying developer participation at that precise point in time may identify individual-level causal factors explaining the overall participation increase whereas such factors may simply be direct consequences of the project management change. Since 1994, the Linux project implemented a parallel release structure based on the principle that in even-numbered releases would experiment new features, while odd-numbered versions would be the stable releases (Moon & Sproull, 2000). Developers' motivations, attitudes, and behavior obviously differ between a project aiming at testing a whole range of new features and one where stability is the main objective.

A third opportunity resides in relying upon additional and complementary sources of data, which would provide indicators that are not related to coding and technical activities per se (Observation 2). In an age of big data, this recommendation sounds like it's going against the dominant winds, but it does not have to. A richer attention to the social life of FOSS projects can still be found online, it's just that FOSS researcher would have to look in non-traditional places. For instance, the software development-related question-and-answer website StackOverflow has gained increasing popularity among FOSS projects to the point that projects have started to migrate their developer support forums on the site (Squire, 2014). StackOverflow uses a particularly handy tagging system in which anyone can ask and reply to questions, the best answers being voted up and rising to the top of the list. From a research standpoint, a large amount of valuable and insightful data is being generated but not captured while studying projects through the data pool provided by code repositories. The multiplication of such platforms upon which FOSS activities take place raises important issues at the methodological level, especially in terms of construct validity (Observation 1). How should we best assess project-level or individual-level activity-related constructs such as participation, knowledge-sharing, or performance?

For most FOSS-related research questions, studying FOSS projects by depriving them from the social layer that surrounds each and every single of them, seriously threatens our understanding of the FOSS reality (Observation 1). Furthermore, FOSS project repositories are data goldmines for researchers but the widespread tendency to use such sources of data in an exclusive manner, tends to transform the job of FOSS researchers into the one of archeologists rather than social scientists. Archeologists study past human life, cultures, and social practices based on the only sources of data that are accessible to them: artifacts, constructions, and any other remains. From a scientific viewpoint, the validity of the drawn conclusions is never above criticism because of the inherent challenges of the discipline: data is inaccurate (due to the passage of time) and largely incomplete (there is no possibility for direct observation). FOSS project repositories contain important data to understand their functioning, but the social life of FOSS projects goes way beyond the data captured on the version control system, the source code repository, the list of bug reports, and the mailing list logs. Nothing prevents project contributors from interacting physically, via email, chat, or video conferencing tools. A number of projects organize events and conferences where contributors can talk to each other, exchange their viewpoints, work collaboratively, and discuss about the future directions of a project.

For instance, the Akademy world summit organized by the KDE project, gathers more than 400 attendees which profile covers a wide range of skills and roles: advocates, developers, artists, translators, and users. The OpenStack Foundation, a FOSS cloud operating system, organizes large conferences throughout the world. The Drupal community has its own DrupalCon. Perl has its own Yet Another Perl Conferences (YAPCs). There is an opportunity for IS researchers to rely upon social media websites (YouTube, Flickr, MeetUp), which are used to report on all kinds of FOSS projects and social gatherings (Scacchi, 2010).

Furthermore, FOSS development also regularly happens in the physical world. The idea of getting together for a session of collaborative coding is not new and originates from the hacker culture. An exponentially growing number of events are organized to generate intense collaborative project development (so-called hack days, hackfests, or codefests), regrouping developers but also graphic designers, interface designers, translators, and project managers. Large projects such as Ubuntu have set up LoCo teams (around 140 as of 2015) that gather Ubuntu users and contributors in common geographical locations and who meet up regularly. By stereotyping FOSS projects as communities of developers loosely collaborating on a FOSS-licensed software project via an online project platform, we disregard the massive amount of information that is not captured on platforms and also neglect the myriad of non-code related tasks and roles without which a project could not be what it is. FOSS project forges or other dedicated hosting platforms shall not be seen as the only available remains of a long extinguished project but rather as one among the many sources of data that can be used when investigating a FOSS-related phenomenon.

To significantly mitigate the conceptual and methodological pitfalls highlighted in all four observations, it is important to point out that the shift from an archeologist stance to embracing the complexity of the living FOSS reality, will happen only if researchers engage into active collaborations with FOSS projects and practitioners. By doing so, researchers will gain contextual knowledge about the IT artifacts under investigation which will mitigate some of the theoretical and/or methodological pitfalls previously addressed in this paper. This approach could also be aligned with the claim to bridge the qualitative–quantitative divide in IS research through the use of mixed-methods (Venkatesh, Brown, & Bala, 2013). Such engagement also provides the opportunity to collect primary data that capture some of the contextual information as well as the activities and social practices that cannot be observed by scanning FOSS project development artifacts and repositories. Given the peculiar norms of the FOSS movement, researchers should expect some sort of moral contract with FOSS projects, aligned with the FOSS beliefs and mechanisms usually characterized by gift economy principles. In other words, a good practice in this context would be of sharing datasets under FOSS licenses (such as the ODC Open Database License, or ODbL), or communicating the results and implications to project leaders but also on online resources associated with projects (blogs, discussion lists, a project's main website…).

Eventually, these steps shall also help IS researchers to keep on top of the constant technological and social evolution of FOSS development. The world of FOSS development evolves at a fast pace. Acknowledging the importance of context when studying FOSS also means that FOSS researchers have to make a constant effort at engaging with projects. By not getting immersed into the fast changing FOSS reality (by getting in constant contact with projects), FOSS research increases the risk of becoming irrelevant by not being able to follow the pace, rendering the body of knowledge rapidly obsolete.

Overall, by heeding the recommendations suggested in this paper, we believe that the IS field will strengthen its academic legitimacy through the "salience of the issues studied, the production of strong results, and the maintenance of disciplinary plasticity" (Lyytinen & King, 2004). The richness of the FOSS world makes its study fascinating and never ending. Besides, FOSS has permeated our society to such an extent that FOSS research implications go far beyond the realm of software development. We hope that our modest nudge derived from our observations may trigger a broader reflection about how to improve and catalyze our theoretical progress about the FOSS movement. Indeed, revisiting our understanding of the FOSS reality could in turn impact the significance of our contribution to neighboring disciplines interested in FOSS-derived phenomena, such as peer production, open innovation, and crowdsourcing. In conclusion, we believe that the multi-faceted nature of the FOSS phenomenon combined with the trans-disciplinary nature of our field (Galliers, 2003) have the potential to place the IS field at the center of an ongoing trans-disciplinary dialog aiming at producing a cumulative and encompassing body of high-quality FOSS research (von Krogh and Spaeth, 2007). As a result, we hope that that our reflection can help bridge the gap with surrounding disciplines as we believe that our results and recommendations still hold, to some varying degree, to the majority of fields of the social sciences that have been investigating FOSS-related phenomena. It is only through a consensual understanding about how FOSS development operates that academics will be able to fully seize the overall complexity but fascinating nature of the FOSS world.

As a parting thought, our recommendations highlight the need to pay extra caution in theorizing the FOSS IT artifacts under investigation when conducting research. The multifaceted and ever-changing nature of FOSS IT artifacts strongly echoes Orlikowski and Iacono's (2001) warning about the tendency to take IT artifacts for granted whereas they are dynamic and interconnected entities that are entwined into complex social practices, and embedded into some time, place, and community. Past FOSS research has already identified some reference discipline theories that could help shed some light on the FOSS reality (Niederman et al., 2006). We do not believe that a single theoretical lens can allow researchers to capture all the underlying nuances of FOSS. Rather, it is the combination of the various theoretical lights provided by all the disciplines investigating FOSS-related phenomena that will illuminate our understanding of

FOSS. To achieve such ultimate goal, again, trans-disciplinary exchanges are key. We sincerely hope this paper will trigger such dialog.

Appendix

Legend:

Applicability : Types/categories of FOSS projects for which the findings/implications of the paper apply, according to the authors.

Theorization: Theoretical foundations/perspective of the paper. An X corresponds to a weak theoretical grounding with no solid link to theoretical foundations

Reference	**Investigated phenomenon**	**Theoretical background**	**Artifact/ Applicability**	**Theorization**	**Type**	**Sample**
Alspaugh et al. (2010)	Software licensing heterogeneously-licensed systems	German and Hassan (2009) 's licensing model, Hohfeld (1913)'s jural relations	heterogeneously-licensed software systems(FOSS + proprietary)	X	empirical (analysis of software licenses)	licences from Firefox, Gnome Evolution, AbiWord
Aksulu and Wade (2010)	Development of of a taxonomy for FOSS research	Systems theory	FOSS projects	X	theoretical (literature review)	N/A
Amrit and van Hillegersberg (2010)	Core-periphery movements in FOSS projects	Socio-technical patterns (Alexander)	FOSS projects	X	empirical (quantitative)	archival data from 9 Sourceforge projects.
Bach and Caroll (2010)	Dynamics of FOSS project user experience design practices	Activity awareness	FOSS projects	X	empirical (qualitative)	archival data and interviews of Firefox and OpenOffice.org
Bergquist and Ljungberg (2001)	Gift giving mechanisms in FOSS projects	gift giving culture	FOSS projects	X	conceptual	archival data and various online sources
Chengalur-Smith et al. (2010)	FOSS project sustainability	organizational ecology	FOSS projects	virtual organizations	empirical (quantitative)	2,772 SourceForge projects (active projects with more than 5 artifacts - bugs, feature requests, patches, etc.)
Chou and He (2011)	FOSS project effectiveness	social capital theory	community-based FOSS projects	X	empirical (quantitative)	160 OSS members from five Taiwanese communities
Chua and Yeow (2011)	Cross-project coordination practices in FOSS projects	Ordering Systems Lens	FOSS projects involving cross-project coordination	X	empirical (qualitative)	posts from 3 cross-project cases within an open-source computer game development community: Jagged Alliance 2
Colazo and Fang (2010)	Effect of temporal dispersion on FOSS team performance	coordination theory	FOSS projects	global virtual teams	empirical (quantitative)	276 SourceForge projects being developed by six or more core team members
Cornford et al. (2010)	structuring and organizing mechanisms in FOSS project collectives	Science and Technology Studies and Actor Network Theory	FOSS projects	X	empirical (qualitative)	Linux Kernel Mailing List in the 1995 to 2003 period.
Daniel et al. (2013)	effect of diversity on FOSS project success	Harrison and Klein (2007)	FOSS projects	global distributed collectives, virtual teams	empirical (quantitative)	archival data from 357 projects hosted on SourceForge
Fand and Neufeld (2009)	FOSS project socialization	legitimate peripheral participation	FOSS project	communities of practice	empirical (qualitative)	archival data from 1 Sourceforge project: phpMyadmin
Feller et al. (2008)	Business exchanges in open source service networks	Benkler	open source service network	peer production communities	empirical (qualitative + quantitative)	10 interviews, 4 workshops, 71 employees (working on Apache projects)
Fitzgerald (2006)	The transformation of FOSS development	X	FOSS projects		conceptual	N/A
Gallivan (2001)	Trust and control mechanisms in FOSS projects	theory of the 'McDonaldization' of society	FOSS projects	virtual organizations	empirical (qualitative)	content analysis of case studies: Linux Kernel, Apache, Fetchmail, Jun, GNU/linux, Perl, Mozilla
Hahn et al. (2008)	Emergence of new FOSS project teams	social network theory	FOSS projects	online collaborative	empirical (quantitative)	archival data from 2,349 Sourceforge projects

Reference	Investigated phenomenon	Theoretical background	Artifact/ Applicab ility	Theorization	Type	Sample
				networks		
Hahn et al. (2013)	Sustained participation in FOSS projects	signaling theory, job matching theory, labor economics	FOSS projects	forms of organized volunteering	empirical (quantitative)	archival data from contributors to Apache projects.
Jorgensen and Jørgensen (2001)	FOSS development methodology	x	FOSS projects	x	empirical (qualitative + quantitative)	interviews: 8 developers from FreeBSD project survey: 72 developers from FreeBSD project
Ke and Zhang (2010)	FOSS project developer motivations and satisfaction	self-determination theory	FOSS projects	x	empirical (quantitative)	230 developers from Sourceforge projects, as well as MySql, OpenOffice and "other projects".
Koch and Schneider (2002)	Effort, coordination and cooperation in FOSS projects	x	FOSS projects	x	empirical (quantitative)	archival data from CVS repository of the GNOME project
Ljungberg (2000)	organization of FOSS projects	x	FOSS projects	virtual organizations/ knowledge firms	conceptual	N/A
Mehra and Mookerjee (2012)	Human capital and FOSS project participation	optimal control theory	FOSS projects	x	empirical (mathematical model development)	N/A
Peng et al. (2013)	Network ties and FOSS project success	social network theory	FOSS projects	x	empirical (quantitative)	archival data from 1228 Sourceforge projects
Sen et al. (2008)	FOSS project license choice	motivational theory, attitudinal theory	FOSS projects	x	empirical (quantitative)	196 developers from Sourceforge projects
Setia et al. (2012)	Peripheral participation in FOSS projects		FOSS projects	x	empirical (quantitative)	archival data from 147 Sourceforge projects
Sharma et al. (2002)	implementation of FOSS practices in software firms	organizational theory	FOSS projects	x	theoretical	N/A
Singh and Tan (2010)	Developer heterogeneity / Formation of comunication networks in FOSS projects	non-cooperative game theory	FOSS projects	x	empirical (quantitative)	archival data from 186 Sourceforge projects
Singh et al. (2011)	Learning dynamics in FOSS projects	learning curve literature	FOSS projects	x	empirical (mathematical model development)	archival data from 251 developers working on 25 Sourceforge projects
Singh et al. (2011)	network effects on FOSS project success	x	FOSS projects	x	empirical (quantitative)	2,378 Sourceforge projects
Sojer and Henkel (2009)	Code reuse in FOSS projects	Theory of Planned Behavior	FOSS projects	open innovation communities	empirical (quantitative)	686 developers from Sourceforge projects
Stewart and Gosain (2006)	Impact of ideology on FOSS project effectiveness	x	FOSS projects	x	empirical (quantitative)	67 administrators of Sourceforge projects
Stewart et al. (2006)	Impact of license restrictiveness and organizational sponsorship FOSS project success	x	FOSS projects	x	empirical (quantitative)	archival data from 138 FreshMeat projects
Torkar et al. (2011)	implementation of FOSS practices in software firms	Avison and Fitzgerald (1995) frameworks for software development methodologies	FOSS projects	x	empirical (qualitative)	Documentation, interviews, survey from the Linux Kernel, the FreeBSD operating system, and the JBoss application server
Vitharana et al. (2010)	Internal FOSS project development and code reuse	x	Internal FOSS projects	x	empirical (qualitative)	interview and observational data from internal open source program at IBM
Vlas and Robison (2012)	Design and validation of an automated natural language requirements classifier for FOSS projects	Requirements engineering theory, McCall's quality model	FOSS projects	x	empirical (design)	archival data from 30 Sourceforge projects
von Krogh and Spaeth (2007)	FOSS research as a means to promote a transdisciplinary research dialog	x	FOSS projects	x	conceptual	N/A

Reference	**Investigated phenomenon**	**Theoretical background**	**Artifact/ Applicab ility**	**Theorization**	**Type**	**Sample**
von Krogh et al. (2012)	FOSS project developer motivations	social philosophy of Alasdair MacIntyre	FOSS projects	x	conceptual (review and research framework development)	40 research articles
Wen et al. (2013)	Impact of intellectual property rights on FOSS project success	x	N/A	x	empirical (quantitative)	904 Sourceforge projects and 2,311 Sourceforge projects over 17 months for FireStar/DataTern v. Red Hat.
Zhang et al. (2013)	Continued participation in FOSS projects	x	x	x	empirical (quantitative)	archival data (discussion forums) from 312 Sourceforge projects

References

Aberdour, M. 2007. “Achieving quality in open source software,” *IEEE Software* (24:1), pp. 58–65.

Ågerfalk, P., and Fitzgerald, B. 2008. “Outsourcing to an Unknown Workforce: Exploring Opensourcing as a Global Sourcing Strategy,” *MIS Quarterly* (32:2), pp. 385.

Aguinis, H., and Vandenberg, R.J. 2014. “An Ounce of Prevention Is Worth a Pound of Cure: Improving Research Quality Before Data Collection,” *Annual Review of Organizational Psychology and Organizational Behavior*. (1), pp.569-595.

Alspaugh, T. A., Scacchi, W., & Asuncion, H. U. 2010. “Software licenses in context: The challenge of heterogeneously-licensed systems,” *Journal of the Association for Information Systems* (11:11), pp. 730-755.

Amrit, C., & van Hillegersberg, J. 2010. “Exploring the impact of socio-technical core-periphery structures in open source software development.” *Journal of Information Technology* (25:2), pp. 216-229.

Asay, M. 2013. “Does open source’s rise spell the end of traditional software vendors?,” *ReadWrite*. October 3 (http://readwrite.com/2013/10/02/does-open-source-rise-spell-the-end-of-traditional-software-vendors).

Bach, P. M., & Carroll, J. M. 2010. “Characterizing the dynamics of open user experience design: The cases of Firefox and OpenOffice.org.” *Journal of the Association for Information Systems* (11:12), pp. 902-925.

Bacharach, S. B. 1989. “Organizational theories: Some criteria for evaluation.” *Academy of management review* (14:4), pp. 496-515.

Barker, J. R. 1993. “Tightening the iron cage: Concertive control in self-managing teams,” *Administrative Science Quarterly* (38:3), pp. 408-437.

Berdou, E. 2006. “Insiders and outsiders: paid contributors and the dynamics of cooperation in community led F/OS projects,” in E. Damiani, B. Fitzgerald, W. Scacchi, M. Scotto, & G. Succi (Eds.), *Open Source Systems - IFIP International Federation for Information Processing* (Vol. 203, pp. 201–208). Boston: Springer

Bergquist, M., and Ljungberg, J. 2001. “The power of gifts: organizing social relationships in open source communities,” *Information Systems Journal* (11:4).

Bonaccorsi, Andrea, Silvia Giannangeli, and Cristina Rossi. 2006. "Entry strategies under competing standards: Hybrid business models in the open source software industry." *Management Science.* (52:7), pp. 1085-1098.

Caprio, F., Casazza, G., Penta, M. Di, and Villano, U. 2001. “Measuring and Predicting the Linux Kernel Evolution,” in *Proceedings of 7th International Workshop on Empirical Studies of Software Maintenance, 9 Nov. 2001, Florence (IT)*, pp. 77–83.

Chengalur-Smith, I., Sidorova, A., and Daniel, S. L. 2010. “Sustainability of free/libre open source projects: A longitudinal study,” *Journal of the Association for Information Systems* (11:11), pp. 657–683.

Chou, S. W., and He, M. Y. 2011. “The factors that affect the performance of open source software development: The perspective of social capital and expertise integration,” *Information Systems Journal* (21:2), pp. 195–219.

Chua, C.E.H., and Yeow, A.Y.K. 2010. "Artifacts, actors, and interactions in the cross-project coordination practices of open-source communities." *Journal of the Association for Information Systems* (11:12), pp. 838-867.

Colazo, J. A., and Fang, Y. 2010. “Following the sun: Temporal dispersion and performance in open source software project teams,” *Journal of the Association for Information Systems* (11:11), pp. 684-707.

Cornford, T., Shaikh, M., & Ciborra, C. 2010. “Hierarchy, laboratory and collective: Unveiling Linux as innovation, machination and constitution,” *Journal of the Association for Information Systems* (11:12), pp. 809-837.

Crowston, K., and Howison, J. 2005. “The social structure of free and open source software development,” *First Monday* (10:2), pp. 1–100.

Crowston, K., Wei, K., Howison, J., and Wiggins, A. 2012. "Free/Libre open-source software development: What we know and what we do not know." *ACM Computing Surveys* (44:2, article 7), pp. 1-35.

Daft, R. L. 1983. “Learning the craft of organizational research,” *Academy of Management Review* (8), pp. 539–546.

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

De Goyeneche, J., and De Sousa, E. 1999. “Loadable kernel modules,” *IEEE Software* (16:1), pp. 65–71.

Denrell, J., and Kovács, B. 2008. “Selective sampling of empirical settings in organizational studies.” *Administrative Science Quarterly* (53:1), pp. 109-144.

Di Tullio, D., & Staples, D. S. 2013. “The Governance and Control of Open Source Software Projects.” *Journal of Management Information Systems*, (30:3), pp. 49-80.

Edwards, J. 1998. “The changing face of freeware,” *IEEE Computer* (31:10), pp. 11–13.

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

Feller, J., Finnegan, P., Fitzgerald, B., & Hayes, J. 2008. “From peer production to productization: A study of socially enabled business exchanges in open source service networks,” *Information Systems Research* (19:4), pp. 475-493.

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

Fligstein, N. 1990. The transformation of corporate control. Cambridge, MA: Harvard University Press.

Gacek, C., and Arief, B. 2004. “The many meanings of open source,” *IEEE Software* (21:1), pp. 34–40.

Galliers, R. 2003. “Change as crisis or growth? Toward a trans-disciplinary view of information systems as a field of study: A response to Benbasat and Zmud’s call for returning to the IT,” *Journal of the Association for Information Systems* (4:6), pp. 337–351.

Gallivan, M. J. 2001. “Striking a balance between trust and control in a virtual organization: a content analysis of open source software case studies,” *Information Systems Journal* (11:4), pp. 277–304.

Ghosh, R. A., Glott, R., Krieger, B., and Robles, G. 2002. “Free/Libre and Open Source Software: Survey and Study”, *Report of the FLOSS Workshop on Advancing the Research Agenda on Free/Open Source Software, European Commission.*

Gold, J. 2015. “Torvards: ‘People who start writing kernel code get hired really quickly.” *Network World*. February 18 (http://www.networkworld.com/article/2885168/linux/torvalds-people-who-start-writing-kernel-code-get-hired-really-quickly.html).

Glass, R. L. 2000. “The Sociology of Open Source,” *IEEE Software* (17:3), pp. 104-105.

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.

Hars, A., and Ou, S. 2002. “Working for free? Motivations for participating in open-source projects,” *International Journal of Electronic Commerce* (6:3), pp. 25–39.

Hauge, Ø., Ayala, C., & Conradi, R. 2010. « Adoption of open source software in software-intensive organizations–A systematic literature review,” *Information and Software Technology* (52:11), pp. 1133-1154.

Herraiz, I., Robles, G., Amor, J. J., Romera, T., and González Barahona, J. M. 2006. “The processes of joining in global distributed software projects,” In *Proceedings of the 2006 International Workshop on Global Software Development for the practitioner* - (pp. 27–33). New York, New York, USA: ACM Press.

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), pp. 1159–1177.

Howison, J., and Crowston, K. 2004. “The perils and pitfalls of mining SourceForge” in *26th International Conference on Software Engineering*.

Hyatt, J., and Mickos, M. 2008. “The oh-so-practical magic of open-source innovation,” *MIT Sloan Management Review* (50:1), pp. 14–19.

Jensen, C., and Scacchi, W. 2007. “Role migration and advancement processes in OSSD projects: A comparative case study,” In *29th International Conference on Software Engineering* (ICSE’07), pp. 364–374.

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

Jørgensen, N. 2001. “Putting it all in the trunk: incremental software development in the FreeBSD open source project,” *Information Systems Journal* (11:4), pp. 321-336.

Ke, W., & Zhang, P. 2010. “The effects of extrinsic motivations and satisfaction in open source software development,” *Journal of the Association for Information Systems*, (11:12), pp. 784-808.

Koch, S., & Schneider, G. 2002. "Effort, co-operation and co-ordination in an open source software project: GNOME," *Information Systems Journal* (12:1), pp. 27-42.

Kozinets, R. V. 2015. Netnography: Redefined. Thousand Oaks, CA: SAGE Publications.

Kreiss, D., Finn, M., and Turner, F. 2011. "The limits of peer production: Some reminders from Max Weber for the network society." *New Media & Society* (13:2), pp. 243-259.

Krishnamurthy, S. 2002. "Cave or community? An empirical examination of 100 mature open source projects," *First Monday* (7:6).

Krishnamurthy, S., Ou, S., & Tripathi, A. K. (2014). "Acceptance of monetary rewards in open source software development," *Research Policy* (43:4), pp. 632-644.

Lakhani, K., and Wolf, R. G. 2005. "Why hackers do what they do: Understanding motivation and effort in free/open source software projects," In J. Feller, B. Fitzgerald, S. A. Hissam, and K. R. Lakhani (Eds.), Perspectives on Free and Open Source Software. Boston, MA: MIT Press.

Ljungberg, J. 2000. "Open source movements as a model for organizing," *European Journal of Information Systems* (9:4), pp. 208-216.

Luthiger, B., and Jungwirth, C. 2007. "Pervasive fun," *First Monday* (12:1).

Lyytinen, K., and King, J. L. 2004. "Nothing At The Center? Academic Legitimacy in the Information Systems Field," *Journal of the Association for Information Systems* (5:6), pp. 220–246.

Mehra, A., & Mookerjee, V. 2012. "Human capital development for programmers using open source software," *MIS quarterly* (36:1), pp. 107-122.

Mockus, A., Fielding, R., 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.

Moon, J. Y., and Sproull, L. 2000. "Essence of distributed work: The case of the Linux kernel," *First Monday* (5:11).

Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K., and Ye, Y. 2002. "Evolution patterns of open-source software systems and communities," In *Proceedings of the International Workshop on Principles of Software Evolution* (pp. 76–85): ACM Press.

Neary, D. and David, V. 2010. "The GNOME Census: Who writes GNOME?" presented at *The 11th GNOME Users and Developers Conference* 2010, The Hague (Netherlands), July 24th -30th

Nelson, M., Sen, R., and Subramaniam, C. 2006. "Understanding open source software: A research classification framework," *Communications of the Association for Information Systems* (17:12).

Niederman, F., Davis, A., Greiner, M. E., and York, P. T. 2006. "Research Agenda for Studying Open Source II: View through the Lens of Referent Discipline Theories," *Communications of the Association for Information Systems* (19:8), pp. 2–46.

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

Orlikowski, W. J., and Iacono, C. S. 2001. "Research Commentary: Desperately Seeking the IT in IT Research - A Call to Theorizing the IT Artifact," *Information Systems Research* (12:2), pp. 121–134.

Paré, G., Trudel, M.-C., Jaana, M., and Kitsiou, S. 2015. "Synthesizing information systems knowledge: A typology of literature reviews," *Information & Management* (52:2), pp. 183-199.

Payne, C. 2002. "On the security of open source software," *Information Systems Journal* (12:1), pp. 61–78.

Peng, G., Wan, Y., & Woodlock, P. 2013. "Network ties and the success of open source software development," *The Journal of Strategic Information Systems* (22:4), pp. 269-281.

Phipps, S. 2012. "How Microsoft was forced to open Office," *InfoWorld*. August 17 (http://www.infoworld.com/article/2618153/open-source-software/how-microsoft-was-forced-to-open-office.html).

Rainer, A., and Gale, S. 2005. "Evaluating the Quality and Quantity of Data on Open Source Software Projects," *In M. Scotto and G. Succi (Eds.), Proceedings of the 1st International Conference on Open Source Software.*

Raymond, E. 1999. "Linux and Open - Source Success," *IEEE Software* (16:1), pp. 85–89.

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

Scacchi, W. 2007. "Free/open source software development: Recent research results and methods," *Advances in Computers* (69), pp. 243–295.

Scacchi, W. 2010. "Collaboration Practices and Affordances in Free / Open Source Software Development," *Collaborative Software Engineering*, 307–327.

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

Seddon, P., and Lyytinen, K. 2008. "Panel on generalisability in information systems theory," *In 19th Australasian Conference on Information Systems (ACIS)*. Christchurch, New Zealand.

Sen, R., Subramaniam, C., & Nelson, M. L. 2008. "Determinants of the choice of open source software license," *Journal of Management Information Systems* (25:3), pp. 207-240.

Setia, P., Rajagopalan, B., Sambamurthy, V., and Calantone, R. 2012. "How peripheral developers contribute to open-source software development," *Information Systems Research* (23:1), pp. 144–163.

Sharma, S., Sugumaran, V., & Rajagopalan, B. 2002. "A framework for creating hybrid-open source software communities," *Information Systems Journal* (12:1), pp. 7-25.

Singh, P. V., and Tan, Y. 2010. "Developer heterogeneity and formation of communication networks in open source software projects," *Journal of Management Information Systems* (27:3), pp. 179-210.

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

Singh, P. V., Tan, Y., and Youn, N. 2011. "A hidden Markov model of developer learning dynamics in open source software projects," *Information Systems Research* (22:4), pp. 790-807.

Sojer, M., and Henkel, J. 2009. "Code Reuse in Open Source Software Development: Quantitative Evidence, Drivers, and Impediments," *Journal of the Association for Information Systems* (11:12), pp. 868–901.

Squire, M. 2014. "Forge++: The Changing Landscape of FLOSS Development," In *Proceedings of the 47h Hawaii International Conference on System Sciences*. Big island, HI, USA: IEEE.

Squire, M., and Williams, D. 2012. "Describing the Software Forge Ecosystem," in 45th Hawaii *International Conference on System Sciences*, pp. 3416–3425

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

Straub, D. W. 1989. "Validating instruments in MIS research," *MIS Quarterly* (13:2), pp. 147.

Subramaniam, C., Sen, R., and Nelson, M. L. 2009. "Determinants of open source software project success: A longitudinal study," *Decision Support Systems* (46:2), pp. 576–585.

Suddaby, R. 2010. "Editor's comments: Construct clarity in theories of management and organization." *Academy of Management Review* (35:3), pp. 346-357.

Torkar, R., Minoves, P., & Garrigós, J. 2011. "Adopting free/libre/open source software practices, techniques and methods for industrial use," *Journal of the Association for Information Systems* (12:1), pp. 88-122.

Trice, H., and Beyer, J. 1984. "Studying organizational cultures through rites and ceremonials," *Academy of Management Review* (9:4), pp. 653–669.

Tsang, E.W.K., and Williams, J.N. 2012. "Generalization and induction: Misconceptions, clarifications, and a classification of induction," *MIS Quarterly* (36:3), pp. 729-748.

Venkatesh, V., Brown, S., and Bala, H. 2013. "Bridging the qualitative quantitative divide: Guidelines for conducting mixed methods research in information systems," *MIS Quarterly* (37:1), pp. 1–34.

Vitharana, P., King, J., & Chapman, H. S. 2010. "Impact of internal open source development on reuse: Participatory reuse in action," *Journal of Management Information Systems* (27:2), pp. 277-304.

Vlas, R. E., & Robinson, W. N. 2012. "Two rule-based natural language strategies for requirements discovery and classification in open source software development projects," *Journal of Management Information Systems* (28:4), pp. 11-38.

Von Hippel, E., and von Krogh, G. 2003. "Open source software and the "private-collective" innovation model: Issues for organization science," *Organization Science* (14:2), pp. 209.

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

Von Krogh, G., and Spaeth, S. 2007. "The open source software phenomenon: Characteristics that promote research," *Journal of Strategic Information Systems* (16:3), pp. 236–253.

Wen, W., Forman, C., and Graham, S. J. H. 2013. “Research Note — The Impact of Intellectual Property Rights Enforcement on Open Source Software Project Success,” *Information Systems Research* (24:4), pp. 1131–1146.

Williams, Robin, and Neil Pollock. 2012. "Research Commentary-Moving Beyond the Single Site Implementation Study: How (and Why) We Should Study the Biography of Packaged Enterprise Solutions." *Information Systems Research* (23:1), pp. 1-22.

Ye, T., and Kishida, K. 2003. “Toward an understanding of the motivation of open source software developers,” In *Proceedings of the 25th International Conference on Software Engineering* (pp. 419–429). 3-10 May, Portland, Oregon, USA.

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