Development Success in Open Source Software Projects: Exploring the Impact of Copylefted Licenses

Jorge A. Colazo  
*The University of Western Ontario*, jcolazo@ivey.uwo.ca

Yulin Fang  
*The University of Western Ontario*, yfang@ivey.uwo.ca

Derrick Neufeld  
*The University of Western Ontario*, dneufeld@ivey.uwo.ca

Follow this and additional works at: [http://aisel.aisnet.org/amcis2005](http://aisel.aisnet.org/amcis2005)

**Recommended Citation**

[http://aisel.aisnet.org/amcis2005/432](http://aisel.aisnet.org/amcis2005/432)
Development Success in Open Source Software Projects: Exploring the Impact of Copylefted Licenses

Jorge A. Colazo
Richard Ivey School of Business
The University of Western Ontario
jcolazo@ivey.uwo.ca

Yulin Fang
Richard Ivey School of Business
The University of Western Ontario
yfang@ivey.uwo.ca

Derrick Neufeld
Richard Ivey School of Business
The University of Western Ontario
dneufeld@ivey.uwo.ca

ABSTRACT
Copyleft prevents the source code of open source software (OSS) from being privately appropriated. The ethos of the OSS movement suggests that volunteer developers may particularly value and contribute to copylefted projects. Based on social movement theory, we hypothesized that copylefted OSS projects are more likely than non-copylefted OSS projects to succeed in the development process, in terms of two key indicators: developer membership and developer productivity. We performed an exploratory study using data from 62 relevant OSS projects spanning an average of three years of development time. We found that copylefted projects were associated with higher developer membership and productivity. This is the first study to empirically test the relationship between copylefted licenses and OSS project success. Implications for OSS project initiators as well as future research directions are discussed.

Keywords
Open source software, software licensing, copyleft, software development, membership, productivity, intellectual property.

INTRODUCTION
The process of bringing new software into widespread use occurs under a variety of models. A central component of all of them is the issue of intellectual property ownership – who owns the source code? In that regard, the open source software (OSS) development model presents a stark alternative to the commercial software development process. OSS differentiates from commercial software in that its source code is freely available to developers and users, allowing them to contribute to the code base. OSS also differs from commercial applications in that its source code is freely available to developers and users, allowing them to contribute to the code base. OSS also differs from commercial applications in that the former adopts liberal licensing terms for redistribution and domain of use (DiBona et al., 1999). These particular traits have made the OSS model a thriving alternative to the proprietary development method (Ljungberg, 2000). The OSS environment is characterized by a network of geographically dispersed, mostly volunteer developers who share no formal employment relationship, and who make extensive use of web-based collaboration tools. Perhaps more importantly, OSS applications incorporate feedback from users from the very early stages of their development, and are subject to a frequent release schedule (Raymond, 1999).

Some OSS projects have proved remarkably successful. The Linux operating system reportedly overtook the Mac OS, achieved second place in market share for desktop operating systems and is expected to hold 6% of the world’s market share by 2007 (Macmillan, 2004). Another OSS application, Apache, serves almost 70 per cent of all the web pages sent through the World Wide Web (Netcraft, 2004). A recent survey showed that 60% of the largest companies in North America are planning to implement OSS applications, half of them in mission critical applications (Schadler, 2004).

Despite the success stories, many OSS projects have failed or been abandoned. Krishnamurty (2002) analyzed mature OSS projects and observed that the great majority were single-person efforts that failed to attract other developers. Other projects have become orphaned, without anybody willing to maintain them (Hermann, 2005).

We contribute to the understanding of the factors that influence OSS project success by examining the effects of licensing terms on the development process dimension of OSS success. We propose that the selection of licensing terms is a critical design factor related to OSS project success or failure. In a primarily volunteer-worker setting that lacks salary incentives,
OSS project success must rely on the developers’ desire to participate (Markus et al., 2000). From the perspective of social movement theory, we empirically explore whether copylefted OSS projects are more or less likely to encourage developer attraction and participation. To our knowledge, no study has been conducted yet to empirically assess this question.

The remainder of the paper is organized as follows. First, we review relevant prior research and develop our hypotheses. Next we describe the methodology used, including sampling, measurement and analysis. Finally, we describe the results obtained and discuss limitations of this work, as well as implications of the findings and future research ideas.

PRIOR RESEARCH AND HYPOTHESES

OSS Licensing and Copyleft

Software licenses regulate the scope of use, redistribution, warranty and intellectual property attribution of software products. Basically, a software license can be described as “open source” if: (1) it requires the source code to be made publicly available, and (2) it allows the software to be modified and redistributed, even if the redistribution is under different terms (St. Laurent, 2004).

Richard M. Stallman created the concept of copyleft when he devised the GNU General Public License (GPL) (Stallman and Lessig, 2002). The object of copyleft was to foster developer cooperation by preventing the “hijacking” of the source code by third parties who may want to incorporate it into a proprietary licensed product in order to gain economic benefits or to preclude others from using the program (O’Mahony, 2003). Under the GPL all derivative works from GPL’ed software must also be released, if redistributed, under the very same GPL. For this reason this license is often described as “viral” (Fink, 2003). Stallman later also created the GNU Lesser General Public License (LGPL), which requires only the LGPL’ed part of a derivative to be licensed under the LGPL when redistributed (rather than the whole work). LGPL was primarily intended for software libraries and presumably allowed greater flexibility in the mixing of software licensed under different terms, though Stallman encourages using the GPL if at all possible.

Not all OSS licenses need to be copylefted. For instance, the source may be put in the public domain and later be modified and redistributed under proprietary licenses without copyleft. Examples of non-copylefted licenses are the Berkeley Software Distribution license (BSD), the MIT license and the Apache license.

Extant research on copyleft is scant. For instance, Mustonen (2003) considered software implementation costs to explain whether copylefted products may coexist with proprietary software. Harhoff et al. (2000) used a game-theoretic model to explore the decision to include a license-bound obligation to reveal the source code. Fink (2000) reviewed different licensing schemes from the point of view of a private company wishing to start an OSS project.

Stewart et al. (2005), explored the influence of licensing on popularity and vitality of OSS. Their study considered OSS licenses as divided into “restrictive” and “non-restrictive” (i.e. viral vs. non-viral). They found marginal support for the association between non-viral licenses and software popularity, measured by the number of subscribers in Freshmeat (www.freshmeat.net). They did not find difference between groups in terms of code release frequency, which they named as “project vitality”.

No research has yet been conducted to explore the impact of copyleft on any of the dimensions of project success. Our arguments in this paper are based on the driving idea that copyleft may be thought of not only as a legal device that prevents misappropriation, but also as an ideological beacon that attracts like-minded OSS volunteers to contribute to a project.

Organization of OSS Projects

Previous research has shown that developers working on OSS projects vary in terms of their contribution and administrative rank. For example, in his study of the Apache project, Mockus et al. (2000) reported that approximately 80% of the coding was done by only 15 out of nearly 400 developers. The sub-group of 15 “core” developers was mainly composed of those with administrative privileges, such as the power to veto a patch submission, or to write modifications into the projects concurrent versioning system (CVS). The CVS is a widely adopted centralized code management repository system that stores the contribution of multiple developers, while keeping track of the changes in the code base and identifying conflicts. The core developers input into this repository not only their own contributions, but also those of a larger group of “non-core” developers who primarily test the software, report bugs and sometimes submit patches. This developer structure in the typical OSS development process is helpful in suggesting a number of idiosyncratic indicators of project success, as discussed below.
**Success of the OSS Development Process**

Successful OSS projects are characterized by a continuing process of volunteer developers fixing bugs, adding features and releasing software “often and early” (Raymond, 1999). Given that a large number of OSS projects are abandoned (Ewusi-Mensah, 1997), being able to attract developers on an on-going basis to keep the project sustainable is critical to success (Markus et al., 2000). Among the indicators suggested for OSS development success (Crowston et al., 2003), we can single out two particular items that relate closely to the attractiveness of the project to developers: the number of developers joining in a project and the relative level of the developers’ productivity while they are engaged in the project (i.e., contribution). We will explore the relationship between copyleft license and the project development process by looking into these two success indicators. Anecdotal evidence seems to support a positive relation between the use of copyleft and improved success. For example, St Laurent (2004) argued that copyleft attracts preferentially “serious” OSS software developers, which implicitly would suggest that copylefted projects might attract better, more professionally apt developers and hence be better developed.

**OSS Developer Participation and Social Movements**

The antecedents of developer participation were among the first research topics in OSS, yet there is not complete understanding of the reasons why “thousands of top-notch programmers contribute freely to the provision of a public good (OSS software)” (Lerner and Tirole, 2002). Existing research has identified individual incentives that accrue to volunteer developers. They include software use value, or “scratching a personal itch” (Raymond, 1999), learning and personal enjoyment (von Hippel and von Krogh 2003), solving their own computing problems (Franke and von Hippel, 2003), skill building (Lakhani et al., 2002), peer recognition and reputation (Raymond, 1999), and career advancement opportunities (Lerner and Tirole, 2002).

While we agree that the existing literature has been fruitful in explaining developer participation, we also feel it largely neglects the important and empirically supported role of group identification generally present in socially oriented actions, of which the development of OSS is one instance. Thus, we feel there is a need to address the group-oriented facets of developer participation. We propose in this exploratory study that copyleft may serve as a strategic design choice that motivates group-conscious OSS developers to participate through strengthening their collective identification with the OSS community. We chose social movement theory (Tajfel, 1981) to provide a theoretical base for the exploration of the relationship between copyleft and OSS development success.

Social movements refer to “efforts by large numbers of people, who define themselves and are also often defined by others as a group to solve collectively a problem they feel they have in common, and which is perceived to arise from their relations with other groups” (Tajfel, 1981). The sociology literature has extensively explored the processes by which people voluntarily engage in social movements such as the civil rights movement (Klandermans, 1997; Omoto and Snyder, 1995). A basic sociopsychological process underlying the behavior of social movement participation is the collective identification of the focal group in relation to other groups (Turner et al., 1987).

We consider OSS development to be a social movement (Hertel, 2003; Ljungberg, 2000). For instance, like in other social movements, OSS development is often viewed as embodying a collective voluntary effort to solve a common problem (Raymond, 1999). Similarly, OSS communities, like participants in other social movements, are tied together by strong normative values, such as the belief that people should have autonomy to share and modify software according to their personal needs (Hertel, 2003; Ljungberg, 2000). Additionally, these ethical values are clearly distinguishable from those of other groups (such as those embodied in the proprietary software development environment) and thus form the core of a distinctive ideology, signaling a unique collective social identity. As with other social movements, the collective identity of participants is developed and strengthened through the ideological struggle with its counterpart, in this case proprietary-style development. In fact, before the advent of big commercial software producers, free sharing of source code was a normal practice among programmers (Levy, 1984; Stallman and Lessig, 2002). Furthermore, the sharing of source code was part of the ethical imperative in the hacker community (Himanen, 2001; Levy, 1984). The very concept of OSS was originated in part as a defensive reaction to the appropriation of software publicly available into proprietary software by private companies (Stallman and Lessig, 2002). Hence, we can consider that the collective identity of OSS developers is rooted in an ideological difference in relation to commercial developers.

As a feature of OSS projects, copyleft signals the ideological position that the licensed project stands against private appropriation, and thus helps confirm that the social identity of the project stands along the same line as that of archetypical OSS developers’ ideals. It is fair then to argue that copylefted projects engender in OSS developers a stronger sense of social...
identity than non-copylefted ones. To the extent that volunteer OSS developers prefer projects whose social identities are compatible to their ideals, we propose that copylefted projects will be more likely to attract typical volunteer developers than non-copylefted analogs. Thus we can posit the first hypothesis.

H1: Developer enrollment is higher in copylefted OSS projects than in non-copylefted OSS projects.

Recent studies on social movements have shown that people who have stronger identification with a group exhibit more frequent participation in and higher willingness to contribute to the group’s activities (Simon et al., 1998). We can then suggest that copylefted projects should have a higher level of developer activity. Thus, we expect to see that copylefted projects will reflect that in higher developer productivity. Thus, we hypothesize:

H2: Developer productivity is higher in copylefted OSS projects than in non-copylefted OSS projects.

METHODOLOGY

This section describes the sampling frame definition, data collection, construct description and measurement. The two research hypotheses are then tested using ordinary least squares (OLS) regression.

Sampling

Many OSS projects have failed to draw significant attention (Krishnamurty, 2002) and are practically irrelevant. In contrast, we define the population of interest for this study as all OSS projects that are practically relevant. Our definition of relevancy is related to the project’s ability to attract attention from both the developer and user communities.

The interest in the project showed by the developer community will be straightforwardly operationalized by taking into account for our sampling frame only those projects developed by more than one person.

To select projects that have also attracted the attention of users, we used a slightly more elaborate approach. First, since OSS has historically been closely associated with the Linux operating system (Raymond, 1999), it makes sense to inspect the most popular Linux distributions to detect programs that are being used by a sizeable community. Major Linux distributions select the best software packages from a wide array of available OSS programs of different application types. We selected OSS projects that were included in at least one out of the three most popular Linux distributions as of September 2004: Fedora Core (formerly Red Hat Linux), Mandrake Linux and SuSe Linux. Popularity was determined according to distrowatch, a website specializing in tracking and rating Linux distributions (see http://distrowatch.com). Second, when a project is deemed relevant to users, it is usually adapted from its original development platform to work under some other computer architecture, or “ported” (Crowston et al., 2004). The Fink project (see http://fink.sourceforge.net) maintains a database of OSS projects that have been ported to the Macintosh architecture.

We defined our sample as those projects included in our sampling frame that were hosted in SourceForge (SF) (http://sourceforge.net). SF is the largest web-based hosting service for OSS projects and has been repeatedly used as a major data source for empirical studies (e.g. Krishnamurty, 2002). We obtained 244 projects that fit our sampling frame description. Further reduction of the sample size arose from the availability of complete data, totaling 62 projects. These had an average of twelve quarters of activity as of May 30th 2004. Each quarter for a particular project was taken as one time period during which constructs were evaluated. This approach eventually yielded approximately 700 instances for which the evolution of projects could be registered.

Measurement and Data Collection

Copyleft

A program written in PERL (Practical Extraction and Report Language) was developed to automatically extract data from the projects’ web page in SF. These data included registration date, programming language, application description and license type. For each project, the licenses were classified as copylefted or non-copylefted. Table 1 provides a breakdown of the 62 projects in terms of their license types.
Developers

The number of core developers for each instance was calculated from the CVS log files. We approximated the total number of developers by counting for each instance the number of different people posting messages to the developer's mailing list, further refining the list with a manual screening of retrieved results to eliminate inconsistencies or obviously invalid data (e.g., spam messages).

Productivity

For this exploratory study we decided to rely on a well established empirical metric for software developer productivity; lines of code (LOC) added per developer, per unit of time (Fenton and Neil, 1999).

We used a CVS client to download the source files for each project, obtaining source code snapshots for dates corresponding to quarterly (three-month) periods starting from the project's registration date and until May 30th 2004. Using another PERL script, total LOC for each project, at each of the defined points in time, were collected for each project. The CVS log command provided a history of the modifications committed to files, starting at the registration date. The log files were also extracted to yield the name of the committer, date of commit, name of file modified and LOC added for each commit. We therefore calculated the ratio of LOC added to the number of active core developers for each instance, obtaining our measure for core developer productivity.

Control variables

In addition to the possible effects of copyleft, we included two control variables: (1) size of the project in lines of code (LOC), and (2) time since registration (in quarters). It seemed likely that the developers' enrollment would increase with time and as the project grew bigger and achieved better visibility. We also expected that developer productivity would decrease with time and as the software size increased, because when the project is bigger it is more difficult for developers to understand it and contribute to it.

RESULTS

Since the distribution of number of developers (core and non-core), project size and productivity factor were skewed to the right, a log transformation (base 10) was performed before running the multivariate regression models. The first model had number of developers as the dependent variable, while the second used productivity. Results of the regression models are shown in Table 2. We adopted an OLS model instead of time series techniques because of test power considerations and because of the reduced and unbalanced number of observations for each project. To confirm whether we departed form the assumption of independence of error terms that could arise due to the time dimension involved in the sample, we tested the
regression residuals for dependence using a non-parametric runs test, and it was non-significant at the 5% level (p > 0.05), lending support to the validity of the results. Copyleft was coded as a dummy variable (0 = non-copylefted, 1 = copylefted).

H1 was supported. The number of developers was significantly (p < 0.0001) associated with copyleft, and the positive sign of the coefficient indicates that the copylefted projects attracted more developers. The time and project size controls were also significant (p < 0.01) predictors of number of developers.

H2 was supported. Developer productivity was significantly (p < 0.0001) associated with copylefted projects. Productivity was also inversely associated with the time control variable, possibly showing that developers might lose motivation over time. The positive coefficient for the size control variable was unexpected but may suggest that larger projects make a more intensive use of project management tools or that they are more attractive to developers, possibly because of an increased visibility of the project.

**DISCUSSION, LIMITATIONS AND FUTURE RESEARCH**

This study is to our knowledge the first empirical test of the effect of copyleft on OSS project success. We found that copylefted projects have higher developer membership and core developer productivity than non-copylefted projects. Furthermore, it is also interesting to see that although Source Forge is not thought to be strongly associated with copylefted projects (in contrast to FSF’s Savannah hosting service), more than two-thirds of the projects in our sample are copylefted. This further supports that copylefted projects are more likely to endure the development process and ultimately be successful than non-copylefted ones.

It is interesting to contrast our results with those of Stewart et al. (2005). In that study, using a differently drawn sample, the authors did not find difference between projects developed under viral vis-à-vis non-viral licenses in terms of code release frequency, which they named as “project vitality”, and limited support for the association of non-viral licenses with project popularity, measured by the number of subscribers to the project. The present paper takes a different focus, assuming that the divide between license groups is the presence of copyleft rather than the viral properties. Our classification, in contrast with that of the mentioned paper leads to find significant differences between groups. We may argue that the reason for that is because OSS developers are primarily concerned about the possibility of appropriation of their efforts by third parties, and in an ethically congruent fashion they do not actively seek to “free ride” on source code developed by someone else by “infecting” it with a viral license.

Our study also makes a contribution to practitioners wishing to initiate a OSS project. For them, the choice of licensing terms is a complex decision involving issues such as legal liability, intellectual property dissemination, competitive responses and project visibility. The effect of the license type on project development success is one more dimension that can inform that decision. The effect of copyleft has particular implications to for-profit organizations. Although copyleft seems at first glance to oppose profit making, our findings suggest that proprietary software or hardware makers do not need to rule out the
use of copyleft in their strategies. Companies may choose to release the source for a program under the GPL and invest economic and human resources to build an external community of developers on which to leverage. Other possible avenues through which private software/hardware makers can benefit include creating a copylefted OSS product that is synergistic with a proprietary application (Varner, 2000), or pulling demand for specific hardware by bundling it with a copylefted application (Mustonen, 2003). The findings also strengthen the case for a dual model of licensing wherein commercial customers are offered a type of proprietary license, and the developer community a less restrictive type (Välimäki, 2003).

This study is subject to several limitations. First, an underlying assumption for this exploratory study is that OSS developers are volunteers. This assumption agrees with the historical origin of OSS as well as with most of the OSS literature to date, although empirical evidence has shown also the participation of paid developers (Lakhani et al., 2002). Those projects more likely to include paid developers are company-sponsored projects (e.g. Open Office) which are normally hosted in dedicated sites and hardware (normally under control of the sponsoring company) and not in a public repository such as SF (Stewart et al., 2005). To further examine whether our sample contained projects likely to be managed predominantly by paid developers, we performed a random check of 5 of the projects in the sample (e.g. the home page and history of the projects) and found no evidence that could signal an important influence of paid developers in the project, such as association with a commercial software firm or E-mail addresses with suffixes indicating a commercial company. These considerations led us to assume with some degree of comfort that volunteer developers dominate in our sample, even though the nature of our data did not allow for a more explicit confirmation. Second, we focused only on core developers in terms of productivity. This is supported by the fact that non-core developer contributions are always eventually managed by a core developer, but future work may want to examine the issue of non-core developer productivity as a separate entity. Third, the available data constitutes a fraction of the projects included in our sampling frame and did not allow to assess self-selection bias. Similarly, limiting our analysis to data obtained through SF might introduce a partial view of the universe of OSS projects.

Further work is needed to address the above mentioned limitations, to develop better measures for success dimensions and to cross-verify this study’s findings using different data sources and exploring other sampling frame definitions. It would also be interesting to explore if copyleft impacts other dimensions of the OSS success construct, with the ultimate goal of refining our understanding of this important new software development paradigm.

REFERENCES