A Framework for Evaluating Managerial Styles in Open Source Projects

Eugenio Capra[1] Anthony I. Wasserman[2]
1 Department of Electronics and Information, Politecnico di Milano,
Piazza Leonardo da Vinci 32, 20133 Milano, Italy, eugenio.capra@polimi.it
2 Center for Open Source Investigation, Carnegie Mellon West, Mountain
View, CA 94035 USA, tonyw@west.cmu.edu

Abstract This paper presents the Software Project Governance Framework (SPGF) for characterizing management of software projects, based on mechanisms used for communication and collaboration, the organizational structure of projects, and testing and quality assurance procedures. The framework was developed and validated from interviews and surveys with leaders of more than 70 commercial and community-based software projects, including both closed and open source projects.

Keywords Management of OSS development; Working practices; Industry contribution; Community contribution.

1 Introduction

Open source software and accompanying "open" development practices have had a major impact on the software industry. "Open" software development processes involve new managerial styles, governance and social models, working practices and communication techniques (cf. [1], [4], [5], [9], and [10]).

Open source products fall into two major categories, which we term "community" and "commercial". Community Open Source projects are led by a community of developers or stakeholders and are distributed under an approved open source license, e.g., GPL, BSD, or Apache. Companies or institutions may have a significant role in the governance of the project, and may contribute many of the resources needed for the ongoing development of the project, but there are few, if any, limitations on who may participate in the various aspects of the project. Development is done "in the open" so that anyone may have complete, no-cost access to the current state of the project. These projects, such as those sponsored by the Apache Software Foundation, have established policies for granting "commit rights" that allow individuals to modify the code base.

Commercial Open Source projects are led by a company, which has usually developed most or all of the code, and then sells subscriptions and services for the developed product. Commercial Open Source applications are very often distributed with a dual license scheme, one offering unrestricted use of the software (community version) and one intended for commercial use of the software. In some cases, the two

versions of the software differ, with the commercial version including features that are not present in the unrestricted version. In that situation, the commercial version of the software typically includes some closed source code. Also, the license for the community version may not be an "approved", i.e., an OSI-listed, open source license.

These approaches are beginning to blend with traditional closed-source software development. Numerous companies offer both closed and open source products, and also participate in non-commercial open source projects. In many cases, companies have completely different policies for each type of project.

While open source is technically a licensing model, its impact on software development processes goes well beyond licensing. The open source phenomenon has had a global impact on the way organizations and individuals create, distribute, and use software, and has challenged the conventional wisdom of the software engineering and software business communities. For this reason, we have focused on managerial and governance approaches to open source projects with the primary goal of creating a framework to characterize these different managerial styles and to evaluate the "openness" of a software project (as opposed to the software itself). The resulting framework allows potential users of the project to identify well-managed projects and allows potential contributors to see if there is a good opportunity to participate in the project.

Governance theory has been applied to software development in a number of different approaches [6], and is defined as the complex process that is responsible for the control of project scope, progress, and continuous commitment of developers [8]. It would be very difficult to elaborate a framework able to encompass all the possible governance dimensions of a software project. We propose a governance framework that allows one to position a software project along the continuum between "fully open" and "fully closed" approach.

This paper is organized as follows: Section 2 gives a brief overview of the study we conducted to define the framework. Section 3 presents the framework with some quantitative and qualitative metrics to evaluate software projects along several dimensions. Section 4 describes some existing open source projects and discusses how they can be classified according to our framework as an example. Section 5 briefly discusses how the framework was applied to our sample. Finally, Section 6 gives preliminary conclusions and topics for further study.

2 Methodology

Our framework aims at positioning a software project along the continuum between fully open and fully closed governance practices. It was developed through preliminary empirical analysis based on individual face-to-face interviews with 25 project managers of major software projects along the continuum, including

traditional closed source development projects (packaged software and software as a service), Commercial Open Source and Community Open Source projects. Project managers were asked which governance dimensions were most significant to measure the degree of openness of a software project. We identified four fundamental governance dimensions: contribution, project leadership, working practices, and testing. These dimensions were chosen since they were widely cited by the project managers, and since they had the highest ranking of all of the cited dimensions.

Subsequently, we refined and validated our framework through a continuing study of more than 70 software projects. We included projects from SourceForge.org, Apache.org, Tigris and Java.net that met the following criteria:

- Mature status (according to the classification provided by the repositories, when available, or to common sense for major projects);
- At least 2 administrators (committers);
- At least 2 developers or contributors.

We have focused on large and well-known projects, rather than those developed by small teams, because these projects had developed and evolved their managerial and governance approaches. Our goal was to identify the dimensions that best illustrate the continuum between open and closed governance approaches. Our research approach has been informal, aimed at identifying the key dimensions and differentiators among projects of varying age, size, diffusion and domain. Table 1 describes how these parameters vary across the sample.

Variable	**Minimum value**	**Average value**	**Maximum value**
Age [year]	<1	8	30
Size [core developers]	10	100	1,000
Size [kSLOC]	40	1,000	6,000
Diffusion [downloads or users]	2	25,000	200,000

Table 1 – Description of age, size and diffusion of the projects analyzed.

Data was collected through interviews with and surveys of key project personnel, namely community managers, QA managers, VPs of engineering, committers, and project leaders, with follow-up calls made as needed for clarification and consistency.

The interviews focused on the following topics:

- *Governance and organization*: Is the project more similar to a "benevolent dictatorship" or to a democracy? Is it self-organizing or centrally controlled?

What is the role of the internal community versus the external community? How many developers are paid?

- *Work practices and tools*: How is the right to commit code granted? How is code reviewed? How important is automated testing? How many management tasks and non-code-developing tasks are shared in the community?
- *Communication and social culture:* Which tools (cvs, bug tracking, IRC, wiki, etc.) are used? How frequent is face-to-face communication? How open are discussions? How is consensus reached?
- *Comparison between open and traditional projects:* How do closed and open projects differ in management practice? What are the relative advantages of open source development compared with traditional closed development?

The results of these interviews formed the basis for our framework, which we call the Software Project Governance Framework (SPGF). The methodology we adopted to formalize the evidence we gathered is based on three major steps. First, we identified and characterized two hypothetical projects representing the two extremes of the spectrum, i.e. a completely traditional closed software project and a completely open source community-based project (see also [3]). Second, we defined dimensions along which these projects can be evaluated, eventually selecting four dimensions that gave the most accurate picture of governance. Third, we scored each dimension of each project from 1 to 4, where 1 indicates a closed-style approach and 4 signifies an open-style approach. We show the detail of the scoring for each dimension below. Note that neither licensing nor the distribution model are part of the framework.

The SPGF framework provides a qualitative assessment of the degree of openness and, accordingly, scales are ordinal. Assessing governance by means of ratio variables not only is difficult, but may also be misleading [5], [11]. The SPGF framework is intended for comparing projects rather than providing absolute assessments. Moreover, the output of our framework may be employed within quantitative methodology according to the approach discussed by Briand et al. in [1].

We would also note that the interviews covered a wide range of topics, and our dimensions have been extracted as the most important factors to distinguish different approaches to project governance. Some of the interviews ranged beyond the specific issues of the framework and helped us to validate the overall approach.

3 The Framework

This section presents our framework. First, we characterize the properties of a completely closed and a completely open project. Then we describe the four dimensions at the base of the framework and provide a graphical representation methodology.

3.1 A traditional closed source project

A "traditional" software project is led by a company or an organization which strictly controls the development process. The proprietary code is closed and is developed by paid staff, possibly including contractors or outsourced teams. Most projects have a well-defined organizational structure following a development process aimed at producing a high quality product (or service) on a predictable schedule. Members of the team "meet" regularly, and report their progress through their organization's management structure.

Many companies have user groups, advisory boards, forums, and other ways for users to interact with the development team, but the final decisions are all made by the company, which has responsibility for all of the code and documentation. In general, the development team has its own communication mechanism, which is not open to outsiders.

The company does most of its own testing and fixes problems even before releasing a beta version to users. Many make their beta versions available to a broad community of users, providing mechanisms for reporting issues and problems in functionality, performance, installation, usability, stability, and/or security.

3.2 A completely open software project

At the opposite end of the spectrum are the thousands of Community Open Source projects, each with its own community, open to anyone who is interested in the project. The work is done entirely in the open, and is typically hosted in such repositories as SourceForge, Tigris.org, Apache Software Foundation, and Java.net. The software can be acquired and used by anyone, subject to the terms of the project's license agreement.

In an open source project, a project lead (or leadership group) is responsible for overall project management, such as determining when a version of the software is ready (stable version), selecting the license to be used with the software release, and deciding who can have "commit rights" to the code.

Some projects are very informal, without formal organization and governance bodies. Decisions are usually made by informally discussing issues within forums, mailing lists or IRC channels. Some communities may have a voting mechanism for resolving issues.

Project participation is open to all, independent of organizational affiliation. Many projects include both volunteers, who have another job and work on the project in their spare time, as well as people who are paid by companies to work on the project.

Since project participation is often a volunteer activity, the project leadership cannot easily compel someone to work on a specific task or to adhere to a schedule, as is the case in a commercial software project. Participants in these community-

based projects rarely meet in person. Instead, they communicate by mechanisms such as forums, mailing lists, IRC channels, instant messaging systems, wikis, blogs, online shared task lists or similar devices. Each community relies on one or more of these tools according to its tradition and habits.

A Community Open Source project doesn't have formal testing or quality assurance processes, but instead relies upon individual developers to test their own code, and for community members to test the software and post issues (and possibly fixes) using the project's issue tracking system. Well-managed projects respond quickly to posted bugs, relying on individual committers to make any needed changes or enhancements to their code. While commercial projects control the number of releases and offer customer support for those releases, no comparable support mechanism is in place for community-based open source projects.

3.3 Dimensions of the SPGF

Using these typical approaches for project management, we defined the following dimensions along which software projects can be evaluated.

3.3.1 Contributions

This dimension measures the relative amount of voluntary code development. Most Commercial Open Source companies resemble proprietary software companies in their reliance on paid development.

In a community-based open source project, code is usually developed on a voluntary basis. However, contributors may be employed or hired by a company or an organization that wants to lead the project or to accomplish specific tasks (e.g., to implement a new feature or to fix a specific bug).

A significant difference between hired and voluntary developers is that the former have to follow the guidelines and deadlines imposed by their employers, whereas the latter are really free to work according to their will and inclination.

Table 2 provides a quantitative metric for this dimension.

Please note that we use the term *hired developer* rather than *employee* as a way to distinguish volunteers from people who receive regular compensation for their contributions to a software project. Whoever is the employer and whatever the form of contract, a person who is paid to develop an application will behave differently from a person who writes code in his spare time just for personal satisfaction. Some companies pay a nominal "bounty" to individuals for small contributions; we do not consider them to be hired developers.

We used 80% as a threshold since the percentage of code committed by volunteers on commercial projects is typically below 10%. In community-based projects, less than 50% of the code is developed by hired employees.

Value	Description
1	100% of the code is developed by hired developers
2	>80% of the code is developed by hired developers
3	>50% of the code is developed by hired developers
4	Most of the code is developed by volunteers

Table 2 - Evaluation of *contributions* dimension.

3.3.2 Project leadership

This dimension indicates the degree to which the leadership of a project is hierarchical. Commercial Open Source projects are led by a company, which usually defines a roadmap and sets schedules. Companies might also play a significant role in guiding and managing community projects. Some Community Open Source projects are indirectly governed by a predominant company, which defines the roadmap of the project and leverages the community to reach its goals. Communities may be led not only by a company, but also by a foundation or by an independent committee. Some projects are managed by a "benevolent dictator": participation and discussion are fostered, but final decisions are made by the project leader or an entrusted committee. The Linux Kernel project is an example this style of governance. On the other hand, fully open communities often lack a formal organization. Decisions are made by voting or by governance bodies which are directly elected by active contributors. Less formal communities adopt the *lazy consensus* approach, i.e., issues are discussed within forums and mailing lists and decisions are made when nobody has anything more to add.

It is very difficult to provide a quantitative metric to evaluate this dimension. For the cases we analyzed, we developed a qualitative scale, shown in Table 3.

Value	Description
1	Roadmap and development process are led by one company or organization which has a predominant leadership role, makes decisions and sets schedules.
2	Roadmap and development process are led by one company or organization. However, free discussion and participation to the governance of the project is fostered.
3	The community is ruled by some formal rules and principles. Decisions are made mainly by voting or by governance bodies directly elected by contributors.
4	The community completely lacks a formal organization and governance bodies. Decisions are made by informally discussing issues.

Table 3 - Evaluation of *Project leadership* dimension.

3.3.3 Working practices

This dimension indicates the degree to which the working and communication practices of a project are geographically distributed and virtual.

Proprietary software projects and many Commercial Open Source projects rely primarily on a closed community working for a single employer, often in close physical proximity. A Community Open Source project, by contrast, often has a geographically dispersed membership. With little funding to support physical meetings of the project team, these projects rely heavily on collaborative tools. Note that such tools may also be used by those in close proximity to each other.

Table 4 presents a qualitative scale for this dimension.

Value	Description
1	Developers work on the same site, communicate in traditional ways and have regular physical meetings.
2	Most developers work on the same site and have regular physical meetings, with some remote participants
3	The community is dispersed and most developers are remote. Some subsets of developers, however, work at the same location and meet regularly.
4	The community is widely dispersed and all the developers communicate through virtual tools. Physical meetings are totally absent or very rare (1-2 per year).

Table 4 – Evaluation of *working practices* dimension.

3.3.4 Testing

This dimension aims at describing the testing process, as well as the presence and role of a Quality Assurance department (or resources) within the project.

As noted in Section 3.1, commercial software development organizations typically have Quality Assurance departments that define formal test processes and are responsible for the quality of the application. A Quality Assurance department also defines quality standards, including those for contributions submitted for inclusion in the code base.

By contrast, Community Open Source projects rely on their own developers and their user community for testing, with relatively few formal processes or tools. In general, open source projects tend not to have specific QA roles, even though some open source projects have very strict pair reviewing rules that determine when new code or patches can be committed to the code base.

Table 5 provides a qualitative scale for this dimension. Please note that by "internally" we also mean testing done by the committers or the core developers of a project. The word "community" in this context refers to users or casual contributors.

Value	Description
1	All the testing is controlled internally by specific QA roles. New versions of the application are released only after being thoroughly tested.
2	Most testing (>50%) is performed internally before new versions of the application are released. The user community is leveraged as a broader testing platform, for example by releasing beta versions and then collecting feedback and bug notifications.
3	Some testing (<=50%) is performed internally, but most of it is left to the community of users.
4	Testing is completely left to the community of users.

Table 5 – Evaluation of *testing* dimension.

3.3.5 Graphical representation

We use a diamond graph to show where projects fall on the spectrum for each dimension. Figure 1 shows the extreme cases of a traditional closed source and a completely open software projects.

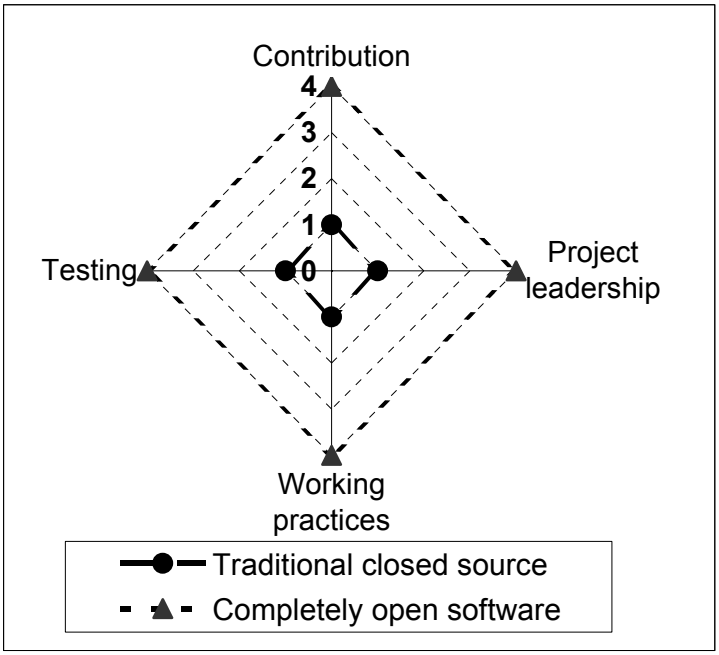

Fig. 1 – Graphical representation of project management dimensions.

4 Case Studies

In this section we provide some examples on how the SPGF may be applied to real projects. We apply the SPGF to three open source projects: OpenOffice.org, MySQL and SugarCRM. We chose these since they are well known applications, and show differences among the dimensions of the framework.

Figure 2 presents a graphical representation of the positioning of these projects. A first glance at the picture shows that OpenOffice.org and MySQL are closer to the completely open source approach, while SugarCRM is closer to closed software projects.

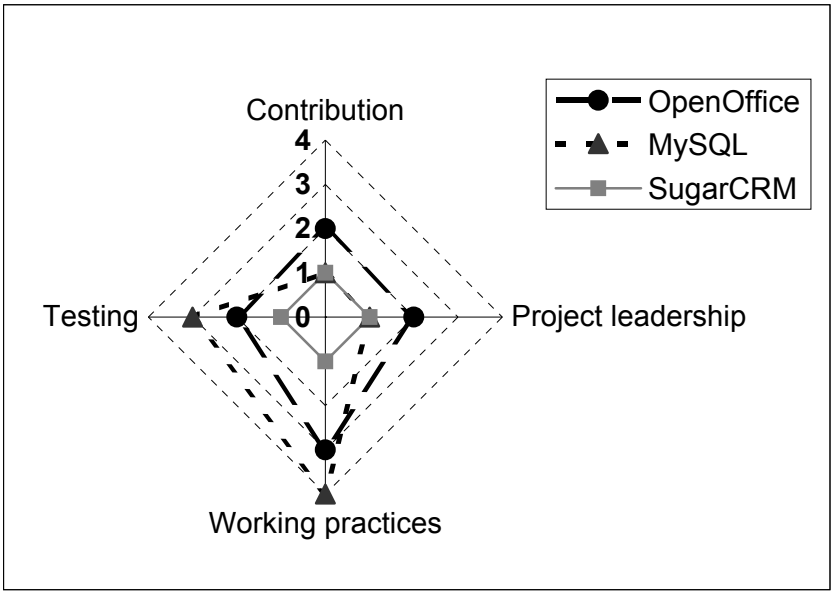

Fig. 2 Graphical representation of the assessment of a project according to the framework.

4.1 OpenOffice

OpenOffice (OpenOffice.org) is a widely used open source office suite with more than 100 million downloads.

OpenOffice is quite a monolithic project. Although everybody can contribute to the project and can earn commit right, Sun Microsystems and IBM have historically contributed almost 90% of the code, paying more than 90 developers for their work. Other companies, such as RedHat and Novell, also contribute to the code. This accounts for the score 2 on *contribution* dimension.

The community has a very structured governance model, based on a Community Council and an Engineering Steering Committee. The Community Council is constituted by members of the community but is deeply influenced by Sun and IBM. The project has a clear and shared roadmap, which probably could not exist without a corporate structure in the background. All these factors lead to score 2 on the *project leadership* dimension.

Communication within the community mainly takes place on mailing lists and on IRC channels. However, most of Sun's developers work in Hamburg and meet daily. As a result, issues are often discussed in person and then conclusions are posted on mailing lists, so that remote community members can be informed. Moreover, occasional cross-corporation meetings are held several times a year. Consequently, *working practices* score 3.

OpenOffice began as StarOffice with Sun, and the QA team that worked on StarOffice now works on OpenOffice. There are currently about 550 QA members with *canconfirm* privilege, i.e. the ability to approve some feature before it is issued. In this particular aspect, OpenOffice is very similar to a traditional software house. Testing is also managed in a very structured way. Specific test suites have been written, integration and system testing are carried out regularly, daily smoke tests, regression testing and code coverage tools are adopted on a regular basis. Every developer is responsible for testing his code, but pair review is applied, too, and the QA team has to confirm the validity of new code. Feedback and bug notifications from users are also accepted and encouraged. This behavior accounts for score 2 on *testing* dimension.

4.2 MySQL

MySQL (www.MySQL.com) is distributed by MySQL, AB (now part of Sun Microsystems). Even though the code is open, it is mainly developed (99%) by employees of the company. The community is invited to submit new code, which is reviewed according to strict and documented internal standards before it is accepted. However, this is quite rare, given the size and complexity of the code base. This accounts for the score 1 on *contribution* dimension.

MySQL (the company) controls governance of the project. The corporate culture is very open to discussion, which is fostered by means of online communication tools, such as blogs, wikis, and forums, but MySQL, as a traditional software house, makes the final decisions. Thus, we assign a score of 1 to *project leadership* dimension.

The real value of the community is mainly to create a broad marketing platform and to provide extensive testing that augments the internal MySQL QA department. Functional tests and cross-platform tests are usually done by the internal development team, then QA tests the alpha versionusing their own scripts. Once the code is released, more than 50% of testing is left to the community, which also

performs most of the integration tests. This combination of internal QA and external testing explains the scores 3 on the *testing* dimension.

Although MySQL is managed as a traditional company, many of its working practices resemble those of community projects. Developers are located in 26 countries around the world, and work from home, meeting only once or twice a year. They mainly communicate through asynchronous tools, such as highly specific internal IRC channels, shared task lists and e-mails, to overcome time zone differences. Telephone conference calls and video chats are also organized, but they are always combined with e-mails or forum posts. This accounts for the score of 4 on the *working practices* dimension.

4.3 SugarCRM

SugarCRM (www.SugarCRM.com) is another Commercial Open Source project. Similarly to MySQL, most of the code is open, but it is developed by internal employees only. The core application is centrally controlled by the company, while the community is involved in the creation of new projects, such as extensions and plug-ins, which are hosted on the SugarForge website. Consequently, the score for *contribution* and *project leadership* dimensions is the same as MySQL.

On the other hand, most of the developers work in the same location and have regular meetings. Forums and mailing lists are used, but by external community members rather than internal developers. VoIP phone conferences are frequent, but this happens even in very traditional closed source projects. Consequently, it scores 1 on the *working practices* dimension.

Most quality assurance and testing is performed by the internal QA department, which is also responsible for bug fixing. This accounts for score 1 on the *testing* dimension.

SugarCRM governance and managerial styles are actually very similar to those of a traditional closed software project, with the only exceptions that most of the code is open and that external people can contribute code.

5 Application of the framework to the sample

After defining the framework, we applied it to our sample of Community Open Source projects. Table 6 shows the distribution of the scoring of the projects in the sample along the four dimensions of the framework.

Dimension x	x<3	3<=x<4	x=4
Contribution	43%	10%	47%
Project Leadership	30%	48%	22%
Working Practice	7%	37%	57%
Testing	33%	44%	15%

Table 6 – Distribution of SPGF scores across the sample.

Most of the communities have some kind of organization and governance bodies, which control new contributions and part of the testing. In particular, the survey showed that approximately 50% of the code of the applications in the sample is developed by hired developers and that physical meeting are held in 35% of projects.

6 Conclusions and Future Work

The Software Project Governance Framework provides a consistent way to analyze projects based on their governance and managerial styles. The central idea behind the framework is that open source has a deep impact on the governance of a software project and, consequently, may impact its quality and costs. The empirical analyses we conducted allowed us to study and embrace a wide range of different software projects. The SPGF provides a structured methodology to analyze managerial and governance models, and to categorize these projects according to the dimensions that are regarded as the most significant by the project leaders. We think that the SPGF may enable a deeper comprehension of software projects and may be useful to a wide range of users.

First, it may be used by researchers to quickly assess and cluster projects. This allows one to select a homogenous sample of projects from a governance point of view before performing further surveys and analysis. We are working on a research project that seeks correlations between the SPGF dimensions and quality of design and development effort of a software project. Second, this framework is valuable to end users seeking information about the structure of various open source projects. For example, a company which is evaluating the adoption of an open source application may want to know and classify the governance approach behind the development of that application. Third, this framework may be used as a reference by developers and project leaders who want to position their products among the different typologies of open source projects and clearly present their managerial style to the public.

In the future, we are planning to further validate and potentially extend this framework. We will expand our sample through additional interviews and surveys, and also seek correlations between these dimensions and project success.

Acknowledgments

We are grateful to the project managers who provided data to us. For the projects identified in this paper, we specifically acknowledge the participation of Louis Suarez-Potts (OpenOffice), Kaj Arnö and Omer BarNir (MySQL), and Jacob Taylor (SugarCRM). We also thank Professor Chiara Francalanci and Francesco Merlo (Politecnico di Milano) for their support and advice.

References

[1] L.C. Briand, K. El Emam, and S. Morasca, "On the application of measurement theory in software engineering", Journal of Empirical Software Engineering, vol. 1, no. 1, pp. 61-88, 1996

[2] B. Fitzgerald, "The Transformation of Open Source Software, *MIS Quarterly,* vol. 30, no. 3, 2006.

[3] K. Fogel, "Producing Open Source Software", O'Reilly, Sebastopol (CA), 2006.

[4] G. Goth, "Open Source Business Models: Ready for Prime Tim", *IEEE Software,* Nov/Dec 2005, pp. 99-100.

[5] M. Griffiths. (2006, Oct. 5). Most software development metrics are misleading and counterproductive [Online]. Agile Journal, Available: http://www.agilejournal.com/content/view/107

[6] L.J. Kirsch, "The management of complex tasks in organizations: controlling the systems development process", *Organization Science,* vol. 7, no. 1, pp. 1-21, 1996.

[7] A. MacCormack, J. Rusnak, and C.Y. Baldwin, "Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code", *Management Science* (forthcoming).

[8] P.S. Renz, *Project governance: implementing corporate governance and business ethics in nonprofit organizations,* Heidelber, Physica-Verl, 2007.

[9] S. Slaughter, J. Roberts, and I. Hann, "Communication Networks in an Open Source Software Project", *Proc. of 2nd Conference on Open Source Systems,* Italy, Jun 2006.

[10] S. Slaughter, J. Roberts, and I. Hann, "Motivations, Participation and Performance in Open Source Software Development", *Management Science*, (forthcoming).

[11] J. Sonnenfeld, "Good governance and the misleading myths of bad metrics", Academy of Management Executives, vol. 18, no. 1, pp. 108-113, 2004.