Inflow and Retention in OSS Communities with Commercial Involvement: A Case Study of Three Hybrid Projects

MINGHUI ZHOU, Peking University
AUDRIS MOCKUS, University of Tennessee
XIUJUAN MA, National Institute of Network and Information Security
LU ZHANG and HONG MEI, Peking University

Motivation: Open-source projects are often supported by companies, but such involvement often affects the robust contributor inflow needed to sustain the project and sometimes prompts key contributors to leave. To capture user innovation and to maintain quality of software and productivity of teams, these projects need to attract and retain contributors. **Aim:** We want to understand and quantify how inflow and retention are shaped by policies and actions of companies in three application server projects. **Method:** We identified three hybrid projects implementing the same JavaEE specification and used published literature, online materials, and interviews to quantify actions and policies companies used to get involved. We collected project repository data, analyzed affiliation history of project participants, and used generalized linear models and survival analysis to measure contributor inflow and retention. **Results:** We identified coherent groups of policies and actions undertaken by sponsoring companies as three models of community involvement and quantified tradeoffs between the inflow and retention each model provides. We found that full control mechanisms and high intensity of commercial involvement were associated with a decrease of external inflow and with improved retention. However, a shared control mechanism was associated with increased external inflow contemporaneously with the increase of commercial involvement. **Implications:** Inspired by a natural experiment, our methods enabled us to quantify aspects of the balance between community and private interests in open- source software projects and provide clear implications for the structure of future open-source communities.

Categories and Subject Descriptors: D.2.8 [**Software Engineering**]: Metrics

General Terms: Measurement, Human Factors

Additional Key Words and Phrases: Hybrid project, commercial involvement, contributor inflow, contributor retention, extent and intensity of involvement, natural experiment

ACM Reference Format:
Minghui Zhou, Audris Mockus, Xiujuan Ma, Lu Zhang, and Hong Mei. 2016. Inflow and retention in OSS communities with commercial involvement: A case study of three hybrid projects. ACM Trans. Softw. Eng. Methodol. 25, 2, Article 13 (April 2016), 29 pages.
DOI: http://dx.doi.org/10.1145/2876443

This work is supported by the National Basic Research Program of China Grant 2015CB352203 and the National Natural Science Foundation of China Grants 61432001, 61421091, and 91318301.
Authors' addresses: M. Zhou, L. Zhang, and H. Mei, School of Electronics Engineering and Computer Science, Peking University and Key Laboratory of High Confidence Software Technologies, Ministry of Education, Beijing 100871; emails: {zhmh, zhanglu.sei, meih}@pku.edu.cn; A. Mockus, University of Tennessee and Avaya Labs Research, 211 Mt Airy Rd, Basking Ridge, NJ; email: audris@utk.edu; X. Ma, National Institute of Network and Information Security, Beijing, 100031; email: mxj@cert.org.cn.

ACM 1049-331X/2016/04-ART13 $15.00
DOI: http://dx.doi.org/10.1145/2876443

1. INTRODUCTION

Numerous companies participate in and have built business models around open-source software (OSS) projects. The commonly mentioned reasons for such rapid spread of *OSS-Commercial hybrid* projects [Mockus et al. 2002] are the potential to acquire user innovation [von Hippel 2002] and to reduce costs [West and Gallagher 2006].

Being the most frequent contributors of OSS code, software users create innovation in a way that has a significant advantage over the manufacturer-centric innovation development systems [von Hippel 2002]. The manufacturer-centric innovation has been the mainstay of commerce for hundreds of years. The user innovation enables each using entity, whether an individual or a corporation, to develop exactly what it wants rather than being restricted to available marketplace choices or relying on a specific manufacturer to act as its (often very imperfect) agent. Companies are motivated to participate if the innovations enhance profits [Dietmar et al. 2002].

In comparison to software produced without volunteer involvement, volunteer participation in hybrid projects may reduce costs [West and Gallagher 2006; Nagy et al. 2010; Chesbrough 2007]. For example, the innovation-related development costs may be reduced by the greater use of external technology in a firm's own R&D process [Chesbrough 2007].

However, such involvement is fraught with risks: the inflow of new contributors may cease, and key contributors may leave because the commercial control may restrict the community, for example, by regulating the code commit privilege. Thus, contributor inflow is critical for the hybrid projects that aim to leverage user innovation, while contributor retention is critical for projects that aim to reduce costs. Both inflow and retention are fundamental to project success beyond innovation and costs. Inflow is needed to sustain a community that provides a platform for contributors, often volunteers, to participate and collaborate [Ye and Kishida 2003; Zhou and Mockus 2012]. Better retention helps to accumulate expertise and developer fluency (among other benefits) [Mockus 2010; Zhou and Mockus 2010], thus reducing development costs due to improved productivity. Furthermore, departing developers tend to reduce project quality, possibly because of the lost experience [Mockus 2010].

In this study, we aim to find evidence of the commercial involvement's impact on the inflow and retention of participants.[1] We conduct a case study of a natural experiment: a parallel evolution of three products that implement the same JavaEE specification. We employ a data-driven approach that combines analysis of project repository data with an examination of published literature and online documents.

Through our investigation, we discover three distinct combinations of policies and actions employed by these projects to get involved in the OSS community: we refer to these as *Hosting*, *Supporting*, and *Collaborating* models. We find that each model affects the inflow and retention of participants in a unique way, but overall, the increased extent and intensity of commercial involvement may decrease external inflow as well as increase developer retention.

2. BACKGROUND AND RELATED WORK

The nature and performance of OSS development has been subject to numerous investigations. From an economic perspective, the most common topics involve motivation and organization: why do the participants in OSS contribute without material compensation, and how do such apparently unstructured and distributed organizations survive and succeed? Early attempts include understanding the nature of OSS development practice and the reasons for OSS success [Mockus et al. 2000], the study of user

[1]We use "participant" and "contributor" interchangeably in this article.

innovation [von Hippel and von Krogh 2003], and the motivation of participants [Ye and Kishida 2003; von Krogh et al. 2008]. A great effort has been spent on investigating communities, for example, the strategies and processes by which newcomers join [von Krogh et al. 2003; Bird et al. 2007], and the impact of the initial willingness and project environment on newcomers' participation [Zhou and Mockus 2011, 2012].

As commercial participation has become more widespread, the question of how to combine OSS practice with commercial practice has received substantial attention. On the one hand, theoretical models for hybrids were proposed, for example, a framework of hybrid-OSS community within an organization [Sharma et al. 2002] or an innovative software engineering paradigm for large corporations [Dinkelacker et al. 2002]. The idea of improving commercial development by using OSS-style project structure was proposed in Mockus and Herbsleb [2002] and structured as a global sourcing strategy for an organization called opensourcing [Agerfalk and Fitzgerald 2008]. On the other hand, successful hybrid projects were observed to understand how to improve upon existing software development practice. The motivation of commercial participation was extensively studied (see, e.g., Bonaccorsi and Rossi [2006], Crowston et al. [2012], Capek et al. [2005], and Henkel [2006]. The business strategies were analyzed in, e.g., Bonaccorsi et al. [2006], Munga et al. [2009], and Dahlander and Magnusson [2008]), as was volunteer and commercial participation in Wagstrom et al. [2010] and Ma et al. [2013]. An in-depth analysis of commercial involvement in Gnome and Eclipse [Wagstrom et al. 2010] identified two types of commercial involvement. RedHat was an example of a community-focused company building a vibrant Gnome community and monetizing services. The other type was of product-focused firms that rely on product revenues. We build on this work to define the dimensions of commercial involvement (why and how), as described in Section 3.3.

As shown earlier, communities in OSS development have been extensively studied, but the ways commercial participation affects contributor communities and the studies of impact on developer participation are rare. Unlike in prior work, we observe how companies shape the communities and, most importantly, quantify how well they achieve their goals of bringing in new contributors and improving the retention of existing contributors.

Note that the literature uses terms similar to community involvement. *OSS governance* is used to describe the way the OSS communities achieve direction and coordination [Markus 2007]. *Opensourcing* represents the use of the OSS development model as a global sourcing strategy for an organization's software development process [Agerfalk and Fitzgerald 2008]. In this study, we are primarily concerned with the influence the companies exert on the community through policies and actions to achieve their goals; therefore, we use the term *community involvement*.

3. RESEARCH METHOD

3.1. Study Design

We investigate a contemporary phenomenon (how companies shape OSS communities) in depth and within its real-life context with the boundaries between phenomenon and context not being clearly evident [Yin 2009]. This leads to several challenges. First, software development is a knowledge-intensive activity with a large number of potentially confounding factors [Curtis et al. 1986], and this makes it difficult or impossible to discern the impact of commercial involvement. Second, to observe the impact of commercial involvement, it is important to compare the differences between OSS and hybrid projects, and to observe the transition from OSS to hybrid projects over long time frames. Third, there is no easy way to learn companies' intentions motivating

their involvement in the OSS community, and it is even more difficult to examine the effects of such involvement.

To address these challenges, we conducted a case study of a natural experiment that investigated the parallel evolution of several products. Natural experiment is an observational study in which the conditions of interest are determined by nature or by other factors outside the control of the investigators [Dunning 2012]. In general, it includes a comparison of conditions that pave the way for causal inference. We selected three products that were developed from the same specification over a similar period (i.e., parallel evolution) but had a varying nature of commercial involvement practices both among the projects and among epochs within a project. This allowed us to compare the differences among projects and the differences among different epochs of a single project to answer the following questions:

(1) What policies and actions[2] did the companies employ to get involved in the communities?
(2) Did these policies and actions
    (a) increase the inflow of new contributors?
    (b) improve retention of existing contributors?

We employed a data-driven approach that combines data retrieved from project repositories, academic literature, online materials, and interviews as an evidence chain. The chain presented in Online Appendix A can be summarized via following steps:

(1) We selected three projects in the JavaEE application server domain that all encounter changes in the nature or intensity of commercial involvement, as described in Section 3.2.
(2) We used published literature to derive two dimensions of commercial community involvement in hybrid projects, as described in Section 3.3.
(3) We used online documents to locate each project within the dimensions of hybrid space by both formulating and answering seven questions, as described in Section 3.4. As a result, we were able to formulate three commercial involvement models, as reported in Section 4.1.
(4) We collected project repository data and analyzed the affiliation history of project participants, as described in Section 3.5.
(5) We used generalized linear models and survival analysis to quantify how different types of commercial involvement impact contributor inflow and retention and report the results in Section 4.2.
(6) Finally, we used interviews and surveys to validate the results, as described in Section 5.

3.2. Projects and Epochs

Section 3.2.1 explains how we selected the three projects to control for external factors. Section 3.2.2 determines the scope of each project to ensure that an equivalent set of application server functionality in each project is investigated. Section 3.2.3 divides the timeline of each project into epochs with a constant nature and intensity of commercial involvement.

[2]Community involvement policies and actions employed by a company are carefully defined and executed policies and plans pertaining to the building and sustaining of communities of volunteers or other types (e.g., commercial) of participants in a software product or ecosystem.

3.2.1. Projects. The three projects we investigated, JBossAS,[3] Apache Geronimo,[4] and JOnAS,[5] are open-source application servers conforming to the standard specifications of JavaEE. JBossAS and JOnAS started in October 1999, while Geronimo started in August 2003. At the time of study (September 2010), JBossAS was hosted by RedHat and was the most popular open-source application server. Geronimo was heavily supported by IBM. JOnAS was nursed in the OW2 Consortium, which was established by the collaboration of several organizations. Both JBossAS and JOnAS use the LGPL as their open-source license, while Geronimo uses the Apache License.

The same JavaEE specification and similar development periods control for important confounding factors and make it more likely that the observed variation in outcomes would be a result of the particular style of commercial involvement.

We chose to control for three main factors: technology, environment, and project context. To control for technology and environment, we selected three products implementing the same specification over a similar time period. This ensured some level of control over external factors such as application domain and the changes in the technology landscape and world economy.

To control for project context, we compared several epochs of the same project with each epoch representing its unique mixture of company policies and actions. This allowed us to observe the effect of changes in such policies and actions within a single project context. We discovered nine distinct epochs of relatively stable community strategies for the three projects. This setup allowed a comparison of different epochs within a project and a comparison of similar epochs among the projects.

Moreover, some of our investigators have expertise in the technology implemented by the projects. The lead author had several years of experience working on application servers, and the first and third authors have participated in the development of JOnAS with the lead author being on the community council since 2007. This provided us with intimate awareness of the application server technology and its evolution.

Finally, we selected highly visible projects to ensure ample amounts of public data. The online sources of discussions, interview reports, and news articles for these projects provided us a deep and broad understanding of the actions and policies employed by the companies and of the potential relationships between commercial involvement actions and developer inflow and retention.

Studies that control for technology, environment, and project context are quite rare in software engineering as the experiments are considered to be too expensive in industry (a notable exception is the study by Anda et al. [2009]), and software projects tend to implement unique requirements. Our study of natural experiment solves the cost problem and has tremendous benefits. In particular, many of the actions and events that happen in software projects are largely technology driven, and, if we don't control for them, we may get misled by, for example, the impact of a project's governance structure, when the outcome (people joining or leaving) may be primarily driven by technology needs or broader economy. However, the limitations of natural experiments apply and are discussed in Section 7.

3.2.2. Scope. In the history of the JavaEE application server, we identified four significant stages: *Initial stage (J2EE 1.3 specification)*, *J2EE 1.4 specification*, *JavaEE 5 specification*, and *Architecture Refactoring*. The implementation of JavaEE 5 started at the end of 2006. Meanwhile, substantial effort was spent refactoring the growing product to achieve a loose coupling of application server components (see, e.g., You et al.

[3]http://www.jboss.org/jbossas.
[4]http://geronimo.apache.org/.
[5]http://jonas.ow2.org/.

[2009]). Since the last two stages started at approximately the same time, we grouped them into a single *JavaEE 5 and Architecture Refactoring epoch*.

During these epochs, all three projects grew in size organically and by incorporating other projects over time. To isolate the impact of commercial involvement from the confounding factors (e.g., technology landscape), we only considered the common functionality, that is, functionality related to JavaEE specification. In particular, JBossAS has its own Subversion (SVN) repository,[6] while Geronimo is a project hosted by the Apache Software Foundation (ASF) SVN server.[7] JOnAS is hosted as several separate projects with the OW2 Consortium.[8] Each new feature of JOnAS was started as a separate OW2 project. For example, *Easybeans* implements a new feature of the JavaEE 5 specification (i.e., EJB 3.x container) for JOnAS. In order to compare similar functionalities in JOnAS with JBossAS and Geronimo, we followed the suggestions of JOnAS core team members and considered a combination of repositories for *Carol*, *CMI*, *Easybeans*, *JASMINe*, *Jotm*, and *JOnAS* as the JOnAS project considered in this study.

3.2.3. Epochs. Over the decade of development, the three projects experienced significant changes, for example, a change from the open-source stage to a hybrid stage with commercial backing, a transition from one type of company to another, or a change in company objectives. We divided the timeline of each project into relatively homogeneous epochs.

The timeline of JBossAS and Geronimo is divided based on the time when the commercial backing commenced. Both projects started as OSS and received company support a few years later. JBossAS has experienced three changes. In April 2001, JBoss Group LLC was founded. In March 2004, JBoss Inc. was founded, and RedHat acquired JBoss Inc. in April 2006. Geronimo started receiving support from IBM in May 2005.

JOnAS was initiated as a hybrid project, with Bull being its primary backer. We divided its lifetime into three epochs according to the three big changes in technical requirements, as illustrated in Section 3.2.2. We found ample evidence of how the changes in requirements led to changes in strategies of JOnAS. For example, the evolution of the application server specification and architecture led to plans for more collaboration initiated by Bull. Therefore, the changes in JavaEE technology are somewhat similar and possibly related to changes in commercial involvement.

However, the project's behavior cannot change immediately. Therefore, we excluded a brief transition period around these change points to reduce potential noise introduced by transitions and to exclude instability during transitions, both technology changes and commercial involvement. For example, prior to and after an acquisition, some activities like handover of work that is not relevant to normal operations of the project in a stable state might add unwanted noise. We used activity in SVN commit logs and the news on the websites to identify and understand these transitions. As a result of this exercise, we excluded 2 months during the foundation of JBoss Group LLC, 3 months during the foundation of JBoss Inc., and 7 months during the acquisition of JBossAS by RedHat. For Geronimo, we excluded 3 months at the beginning of IBM's participation. For JOnAS, we excluded 3 months at the boundary between the two epochs. While transition periods are not representative of the typical activity for each model and should be excluded, there is an element of subjectivity in deciding exactly when the transition ends, and this brings a threat to the internal validity of our study.

[6]http://svn.jboss.org/repos/jbossas.
[7]http://svn.apache.org/viewvc/geronimo/.
[8]svn://svn.forge.objectweb.org/svnroot/[project name, *e.g., JOnAS, easybeans*].

Table I. Epochs in the Three Projects

JBossAS	Time Span	JEE V	Backing Company	License
JB.E1	10/1999–3/2001	1.3	open source	LGPL
JB.E2	6/2001–2/2004	1.3	JBoss Group LLC	LGPL
JB.E3	6/2004–2/2006	1.4	JBoss Inc.	LGPL
JB.E4	10/2006–9/2010	5	RedHat	LGPL
Geronimo	Time Span	JEE V	Backing Company	License
GE.E1	8/2003–4/2005	1.4	open source	APL
GE.E2	8/2005–9/2010	1.4, 5	IBM	APL
JOnAS	Time Span	JEE V	Backing Company	License
JO.E1	10/1999–3/2004	1.3	Bull	LGPL
JO.E2	6/2004–10/2006	1.4	Bull	LGPL
JO.E3	2/2007–9/2010	5	Bull	LGPL

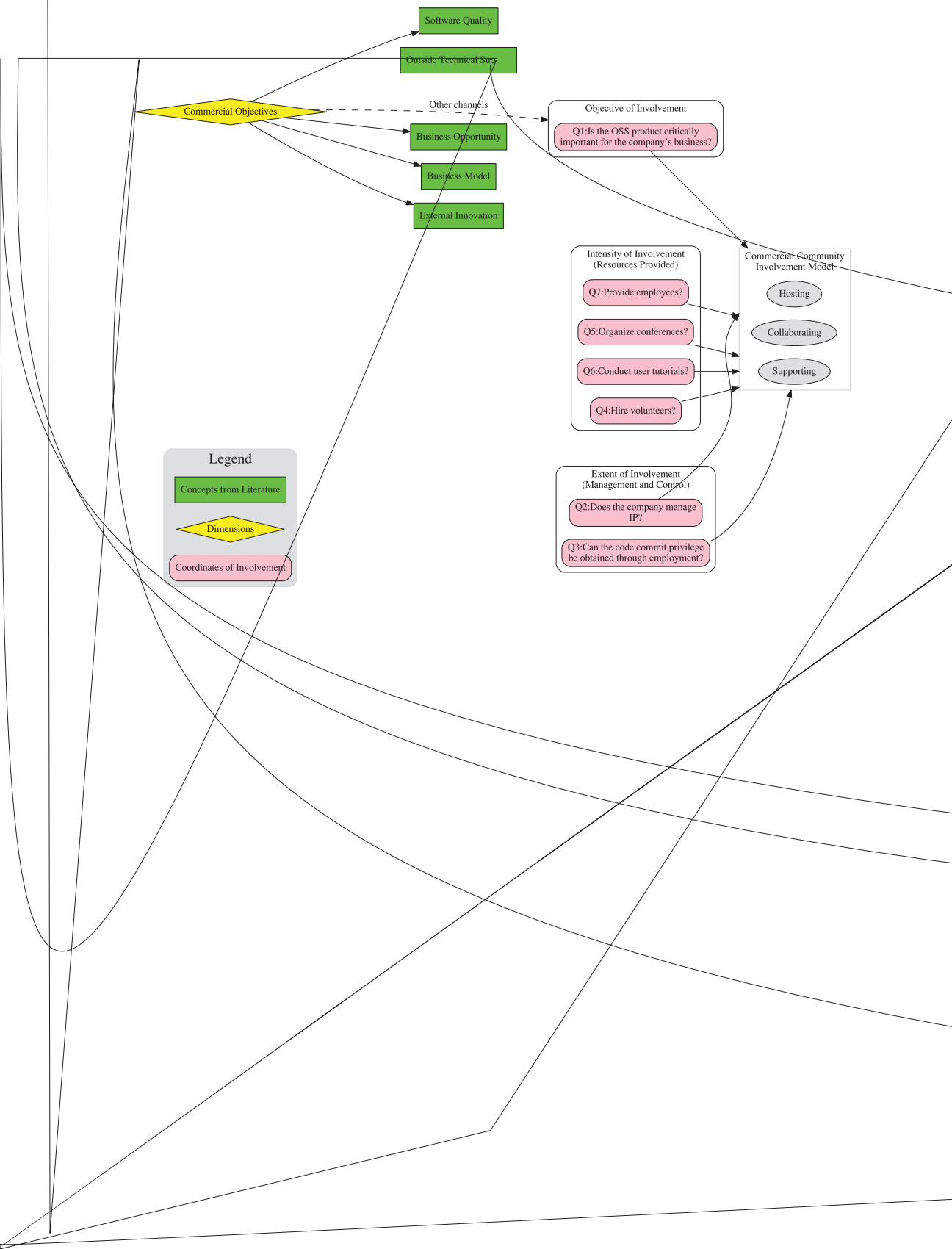
We end up with a total of nine epochs, among which seven epochs belong to hybrid projects and two epochs belong to pure OSS projects. The specific time spans are shown in the second column of Table I (e.g., JB.E1 means $Epoch_1$ of JBossAS).

3.3. Using Literature to Derive Dimensions of Community Involvement

The existing literature provides a number of theories about why and how companies engage in OSS projects. We identified two key dimensions of the involvement: the company objectives that drive the involvement, and the company strategic actions for achieving the objectives. To discriminate among the three projects, we created seven questions that reflect the particular features of each dimension. Figure 1 shows the relationship among concepts from the literature, dimensions, and questions.

3.3.1. "Why" Aspect. What is the primary purpose for a company to get involved in an OSS project, and how well does the purpose align with its business model? A central idea in the literature on open and distributed innovation is that firms can benefit from the creative ideas of individuals outside the company and a good fit between what the firm does and the resources and capabilities available in its external environment [Crowston et al. 2012].

Firms are motivated to be involved with OSS because it allows smaller firms to innovate, because "many eyes" assist them in software development, and because of the quality and reliability of OSS. Not surprisingly, the ideological fight for free software comes at the bottom of the list of concerns [Bonaccorsi and Rossi 2006]. In comparison with individuals, firms are found to focus less on social motivations such as reputation and learning benefits. Similarly, the study on the firm-developed innovations within Linux for embedded devices emphasized the importance of receiving outside technical support as a motivator for revealing code [Henkel 2006]. Meanwhile, how to align OSS participation activity with business goals is a critical concern for commercial organizations. For example, the first major issue for considering an open-source proposal is "business" in IBM, and one of IBM's strategic goals for open source is to use open source as a business tool by keeping the platform open and taking advantage of new business opportunities [Capek et al. 2005]. In particular, IBM planned an OSS strategy in the broad area of what is called middleware, and not in operating systems, because their enterprise customers benefit more directly from middleware functions than from operating system functions. An earlier study highlighted the benefits of a business model that promotes proactive engagement with communities to develop new products and services [Chesbrough and Rosenbloom 2002]. A study of four firms involved with OSS discovered three ways that firms use to connect with OSS communities [Dahlander and Magnusson 2008]: accessing development in the community in order to extend their resource base, aligning their strategy with the work in the community, and assimilating


Software Quality
Outside Technical Supp
Commercial Objectives
Other channels
Objective of Involvement
Q1:Is the OSS product critically important for the company's business?
Business Opportunity
Business Model
External Innovation
Intensity of Involvement (Resources Provided)
Q7:Provide employees?
Q5:Organize conferences?
Q6:Conduct user tutorials?
Q4:Hire volunteers?
Commercial Community Involvement Model
Hosting
Collaborating
Supporting
Legend
Concepts from Literature
Dimensions
Coordinates of Involvement
Extent of Involvement (Management and Control)
Q2:Does the company manage IP?
Q3:Can the code commit privilege be obtained through employment?

property (IP) once it gets involved; for example, adopting licensing practices to clarify ownership is one of the major tactics (see, e.g., Dahlander and Magnusson [2008]) used by firms to connect with OSS communities, and creating a Contributor License Agreement is considered important by IBM (e.g., in Apache) to manage the open-source activities [Capek et al. 2005]. On the other hand, the company may be able to grant code-commit privileges, a decision typically reserved for the community in OSS projects and granted based on substantial and sustained contributions [Sinha et al. 2011]. This leads to two questions:

—Q2: Does the company manage IP?
—Q3: Can code-commit privilege be obtained through employment?

The intensity of involvement is reflected by the amount of resources provided by a company. Devoting manpower to develop, evaluate, and select source code is a tactic adopted by many companies to get involved [Dahlander and Magnusson 2008; Capek et al. 2005]. In fact, commercial firms often provide wider distribution by adding the project to their existing offerings, garner media attention for the project by issuing press releases about the software, and present information about the project and community at trade shows [Wagstrom et al. 2010]. Through creating alliances, firms can also appropriate returns from a larger user base and create lock-ins [Dahlander and Magnusson 2008]. Therefore, this dimension focuses on measuring the various resources that a company can devote to an OSS community. In particular, we identified eight types of resources (see Section 3.4) but present in Figure 1 only the four resources (listed next as four questions) that vary among the investigated hybrid projects:

—Q4: Does the company hire project volunteers?
—Q5: Does the company organize conferences/summits for the project?
—Q6: Does the company conduct user tutorials for the project?
—Q7: Does the company provide employees (and how many)?

3.4. Using Online Documents to Obtain Community Involvement Models

Although we do not have a complete knowledge of the companies' intentions, we could learn about company actions, visions, goals, market share, and values based on the information gathered from company websites, news articles, personal blogs, commit comments, and other documents. Therefore, we conducted an extensive Internet search for the materials relevant to this study:

—We read project and sponsoring company websites to collect the information related to the history of company goals and to their attempts to influence the communities, such as the project history, the wiki pages on community conventions, and the statements of the companies' actions on open source. To reconstruct period-specific information, for example, the websites that have been replaced or had their content changed (such as JBoss Group and JBoss Inc.), we used the Internet Archive[9] that contains snapshots of online documents collected at various points in the past.
—Using Google search, we drew up an initial set of documents by searching for both the project name and the company name. For the convenience of retrieval, we used only the first 100 links of search results. We read through each link and selected the initial set of candidates. We ignored the links that introduce or discuss techniques or are duplicates and reposted content. We retrieved additional pages by following the links in the initial set (e.g., tagged by "see also") and stopped the procedure when no new pages were found. We manually filtered the collected records by removing

[9]http://www.archive.org/web/web.php.

Table II. Combinations of Community Involvement Policies and Actions

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Model
JB.E1	—	—	—	—	—	—	—	Hosting
JB.E2	Y	Y	Y	Y	Y	N	Y(15)	
JB.E3	Y	Y	Y	Y	Y	Y	Y(30)	
JB.E4	Y	Y	Y	Y	Y	Y	Y(150)	
GE.E1	—	—	—	—	—	—	—	Supporting
GE.E2	N	N	N	Y	N	N	Y(35)	
JO.E1	N	Partially	Partially	N	N	N	Y(12)	Collaborating
JO.E1	N	Partially	Partially	N	N	N	Y(13)	
JO.E1	N	Partially	Partially	N	Y	N	Y(15)	

duplicate and irrelevant records. We marked records as irrelevant if they did not contain information related to commercial actions or to contributor participation. This resulted in 70 records for JBossAS (we refer to these as JB1 to JB70), 47 records for Geronimo (we refer to these as GE1 to GE47), and 43 records for JOnAS (we refer to these as JO1 to JO43), including academic papers, web pages on project and company websites, product documents, news articles, blog articles, and interview reports.

We used these sources to locate each project within the dimensions of the hybrid space described in Section 3.3. We did that by both formulating and answering the seven questions. For example, when determining what resources the companies devoted to get involved in the three projects, we started from a resource category retrieved from the literature and looked for a match in the 160 records for each of the three projects. Two of the authors scanned each record (of each project) respectively, marking the relevant resource if there is a match. For example, GE44 says "...IBM's sponsorship, in which a large number of the Geronimo committers were hired by IBM," and we marked it as evidence for *IBM-Hire volunteers*. If there are resources mentioned in the article but not on the category, we mark it with a new label. For example, JB2 describes an event called *xxx's JBoss tutorial* in $Epoch_3$ of JBossAS, and we marked it with *JBI-user tutorial*. We iterated through these steps three times. In cases where the marked resources presented a disagreement, two authors discussed and clarified the contradiction and started again, until reaching the final agreement in the third iteration.

By answering the seven questions in Figure 1 with the procedure just described, we obtained three coherent groups of policies and actions undertaken by sponsoring companies as three models of community involvement discussed in Section 4.1 and shown in Table II. For example, while the companies involved in JBossAS profit directly (Q1), fully control IP (Q2), and manage commit privileges (Q3), neither IBM (in Geronimo) nor Bull (in JOnAS) directly profits (Q1), fully controls IP (Q2), or manages commit privileges (Q3).

3.5. Analyzing Repository Data

We use project repository data to quantify two key variables that may be affected by a company's involvement: the inflow of new contributors and the retention of existing developers. In Section 3.5.1, we retrieve and process historical data from VCSs. In Section 3.5.2, we identify internal and external developers. In Section 3.5.3, we identify different developer roles.

3.5.1. Version Control Systems. The three projects use SVN for version control. We obtained the commit history from SVN repositories. The commit history contained a set

Table III. Number of Commits for the Three Projects

Project	From	To	Count
JBossAS	10/14/1999	9/14/2010	100,969
Geronimo	8/8/2003	9/14/2010	24,637
JOnAS	10/28/1999	9/14/2010	19,288

of tuples consisting of the time, the developer (indicated by the login), the revision number, the comment, and the list of modified files, as shown in Table III.

Some developers used more than one login. For example, JBossAS logins *loubyansky* and *aloubyansky* belong to the same developer according to the list of individual contributors[10] in the copyright licenses used prior to 2006. To map multiple logins to a unique developer ID, we obtained the lists of project committers. The list of Geronimo[11] contained all the logins we extracted from SVN logs. But the list of JBossAS contained only the logins used in the first three epochs. For the logins not in the list, we manually inspected whether the use of similar logins (e.g., *loubyansky, aloubyansky, and alex.loubyansky@jboss.com*) in the commit history overlapped in time and matched them to a full name obtained from public email archives and project-related pages.[12] For JBossAS, we discovered 53 developers who changed their logins at least once. To handle this, we corrected the raw data by replacing their other logins with the last login used. For JOnAS, we used our contacts on the JOnAS team to validate any login changes.

3.5.2. Identifying External Developers. To investigate the inflow and retention of volunteers, we need to identify whether a developer was a volunteer when he or she started contributing code. We used email domains to identify developers of the corresponding commercial companies (internal developers). We consider the remaining developers to be external.

The committers from IBM are listed on the Geronimo website. Others were considered to be external participants.[13] For JBossAS and JOnAS, we used the domain of the contact email addresses in JIRA, mailing list archives, and project websites to derive affiliation. For example, developers with email domains *@Bull.net, @Bull.com, or @ow2.org* were considered to be internal for JOnAS, while the domains *@jboss.org, @jboss.com,* or *@RedHat.com* were internal for JBossAS. There were additional issues; for example, a developer did not use the company's email although he or she was employed there. We verified our derived affiliation through the project mailing list, manually searched for each developer on the web (e.g., LinkedIn), and checked whether the initial developer activity in the issue tracker involved a change to a known internal developer in the "assignee" field. Such action implies that he or she might be a team leader in the same organization as the internal developer, and therefore is an internal participant.

3.5.3. Identifying Roles of External Developers. We take a further step to understand volunteer inflow: we looked into different developer groups (roles). In particular, we separated them into application developers who write code that runs on the application server and infrastructure developers who write the application server itself, because the commercial involvement may have different effects on different types of external

[10]http://www.rocksclusters.org/roll-documentation/camera/5.0/x162.html.

[11]http://geronimo.apache.org/committers.html.

[12]For example, http://community.jboss.org/.

[13]According to Apache's convention, there must be at least three companies supporting one project. So there are also developers from other companies. However, since 35 of 63 are from IBM and only four are from other companies, we regarded the non-IBM developers as external developers for simplicity.

Table IV. Commercial Objectives in the Seven Epochs (See Table I for Epochs)

JBG (JB.E2)	To provide support and services for JBossAS server technology directly from the core developers, with the revenues rewarding the employees.
JBI (JB.E3)	To build professional open-source software and gain profit in the same way as JBG, later adding Subscription to a set of services and tools.
RHT (JB.E4)	To ensure the continuity of an independent open-source application server and integrate it as a division of RedHat. To gain profit through subscription-based services.
IBM (GE.E2)	To support Geronimo and distribute a commercial community version based on it, and to obtain and test innovations transferable to the commercial products, e.g., WebSphere. To gain profit indirectly through the commercial products.
Bull (JO)	General interest: To provide open-source enterprise middleware solutions and support for the customers, and to advertise the open-source strategy for OW2. To gain profit indirectly through other commercial products.
Bull (JO.E1)	Specific Interest (SI) in E1: To develop its early version, through cooperation with several French organizations.
Bull (JO.E2)	SI in E2: To implement and get certified on J2EE 1.4 specification and to be effective at dissemination, helping many third-party developers start building JOnAS-based applications and produce their own products.
Bull (JO.E3)	SI in E3: To implement JavaEE 5 and to develop an innovative version conforming to new specifications based on OSGi and to cooperate with many other organizations to broaden its influence.

developers. JBossAS has the largest of the three communities with a sufficient number of developers in each role for us to be able to observe the differences.

Getting involved in an OSS project often starts from the mailing list [von Krogh et al. 2003]. The mail archives may, therefore, suggest the senders' background (e.g., the domain name in the email address or the signature). Personal details, such as past jobs, could be found on social network websites (e.g., LinkedIn, Facebook). We used these rich information sources to identify the roles of the external developers, as described in Online Appendix B.

4. RESULTS

We report the community involvement models in Section 4.1, quantify new contributor inflow in Section 4.2.1, and measure developer retention in Section 4.2.2.

4.1. Community Involvement Models in the Three Projects

A successful hybrid model should "detail exactly what your organization needs and wants from its community, and how it can achieve those goals," as noted by OpenSUSE Community Manager Joe Brockmeier.[14] This illustrates the two important dimensions of a community involvement model: the company objectives that drive the model and the strategic actions for achieving the objectives.

4.1.1. Commercial Objectives. When a company decides to participate in an OSS project, two concerns must be addressed: do the project and the company share a set of common goals, and is there an opportunity to gain profit? We refer to these as *commercial objectives*. For example, for a vendor of a major Linux distribution, the acquisition of a middleware system (e.g., JBossAS) is closely aligned with the common purpose of providing operating system platforms. It can also bring additional profits by enhancing core business and providing opportunities for support, training, and consulting services. Table IV shows the commercial objectives synthesized from the online records.

From Table IV, we can see that the companies at different epochs have different approaches to benefit from their involvement. JBoss Group LLC (abbr. JBG in the

[14]http://www.cio.com/article/474963/Measuring_Corporate_Contributions_to_an_Open_Source_Project.

Table V. Actions of Community Involvement

	Management and Control		Resources Provided by Companies				
Epoch	IP Management Entity	Way to Be Committer	R1-R4*	Employees	Hire	Conferences	Tutorials
JB.E1	JBoss Org	Contribute	—	—	—	—	—
JB.E2	JBoss Group LLC.	Be employed+	Y	Y (15)	Y	Y	N
JB.E3	JBoss Inc.	Be employed+	Y	Y (30)	Y	Y	Y
JB.E4	RedHat	Be employed+	Y	Y (150)	Y	Y	Y
GE.E1	ASF	Contribute	—	—	—	—	—
GE.E2	ASF	Contribute	Y	Y (35)	Y	—	—
JO.E1	Bull, France Telecom, Lifl, INRIA	Contribute**	Y	Y (12)	—	—	—
JO.E2	the same as above	Contribute**	Y	Y (13)	—	—	—
JO.E3	Adding: SerLi, U of Fortaleza, Peking U.	Contribute**	Y	Y (15)	—	—	Y

*R1: add project to existing offerings; R2: conduct interviews; R3: publish news; R4: issue technical reports/papers.
+People with substantial contributions can also become committers.
**Bull can assign its employees to be JOnAS committers.

table), JBoss Inc (JBI), and RedHat (RHT) make profit directly from the product (e.g., by selling service contracts). IBM is appropriating Geronimo technology for its commercial WebSphere products, and it also uses it as an "upselling"[15] market strategy. Bull is known for selling large-scale systems running on a distribution of Linux and it uses JOnAS to enhance the value of its solutions (side product). Also, Bull uses the JOnAS brand to advertise its open-source strategy. This suggests that JOnAS is not critically important for the business, and therefore, it is more devoted to developing new technologies than to serving customers.

We found that companies create policies and take actions to shape the OSS community in support of their business goals. In the case of JBossAS, these actions supported the primary business objectives of the company, while in Geronimo and JOnAS, they supported the secondary objectives (answers to Q1 in Table II).

4.1.2. Community Involvement Actions. Companies have to consider a number of issues when getting involved in project communities, for example, how to deal with IP and legal issues, how to provide resources, and how their employees should engage the community. We refer to these policies and actions as *community involvement actions*.

We found that the community involvement actions varied significantly in two aspects: the control mechanism (i.e., the extent of involvement) and the amount of resources provided (i.e., the intensity of involvement). This is shown in Table V, where the rows represent epochs, and the columns summarize actions. For example, in the second epoch of JBossAS (JB.E2), JBoss Group LLC managed the IP. It provided full-time employees (15 people) and hired existing project developers. Whoever was employed by JBoss Group LLC or contributed sufficiently to JBossAS could become a committer. JBoss Group LLC also advertised and organized global conferences to promote JBossAS. A dash in this table means that the specific action could not be found in the records of the corresponding epoch. Note that all the companies provided wider distribution by adding projects to existing offerings (R1, does not apply to JBG or JBI), garnered media attention by conducting interviews (R2) and publishing news (R3), and issued technical reports/papers (R4).

(i) Extent of Involvement. The control mechanisms reflected through the management of IP and code commits represent the company's extent of involvement. The management of IP, corresponding to the control the company exerts on the project, can be

[15]Offering high-end commercial WebSphere to low-end Geronimo customers.

In summary, we have the following observations.

OBSERVATION 1. *A company's commercial objective (i.e., the opportunity to gain profit) guides the actions and policies it takes to change (shape) the community. Such involvement varies in two dimensions: the extent of involvement the company adopts over the project, in particular, intellectual property and code-commit privileges, and the intensity of involvement, in particular, employees and other resources the company devotes to the project.*

OBSERVATION 2. *The three models of commercial involvement in the three projects were (in Table II) Hosting: when a company has full control over the project; Supporting: when a company supports a project (e.g., through investing resources), but the project is controlled by another OSS organization; and Collaborating: when a company has a shared control over the project with other organizations.*

4.2. Impact of Community Involvement on Participation

The nine epochs of the three projects demonstrate the varying nature of commercial involvement practices that may affect the growth (decline) of the community of developers contributing code to the project.

Except for the two open-source epochs of JBossAS and Geronimo, epochs belonging to the same project share the involvement model but differ in the intensity of involvement. We therefore can compare the types of community involvement actions among the three projects and the intensities of involvement within the same project.

In particular, we compare each project to other projects to reveal how the community involvement models (i.e., *Hosting*, *Supporting*, *Collaborating*) are associated with the contributor participation.

The changes in technical requirements affected all three projects simultaneously and were likely to affect participation: for example, new requirements may increase participation, while stable requirements may decrease it. Because business interest in JOnAS was primarily driven by changes in technical requirements, it can serve as a contrast to the remaining two projects to control for the potential influence of these changing requirements. Both JBossAS and Geronimo had an open-source epoch, and we measure whether developer participation was affected by the transition to *Hosting* and *Supporting* hybrids.

In summary, we investigate whether the developer inflow and retention are associated with:

—the involvement model by comparing different projects (we use JOnAS as a contrast reflecting the evolution of technology common to all three projects),
—the involvement model by comparing with the two-open source epochs, and
—the intensity of involvement by comparing epochs within a project.

4.2.1. Inflow of External Developers. As mentioned earlier, software users are the most frequent contributors of OSS code and are typically the initial developers of what later become commercially significant new products and processes. As newcomers, contributing users may bring changes and new values to an existing organization [Van Maanen and Schein 1979]. In OSS projects, new contributors may add features and later join the core team [von Hippel and von Krogh 2003]. A thriving volunteer community was imperative for OSS projects.

In hybrid projects, volunteers (i.e., external developers) could also provide valuable contributions that may reduce employment costs and introduce innovations [West and Gallagher 2006; Nagy et al. 2010; Chesbrough 2007]. Therefore, attracting talented volunteers, a goal many hybrid projects would like to achieve, is crucial.

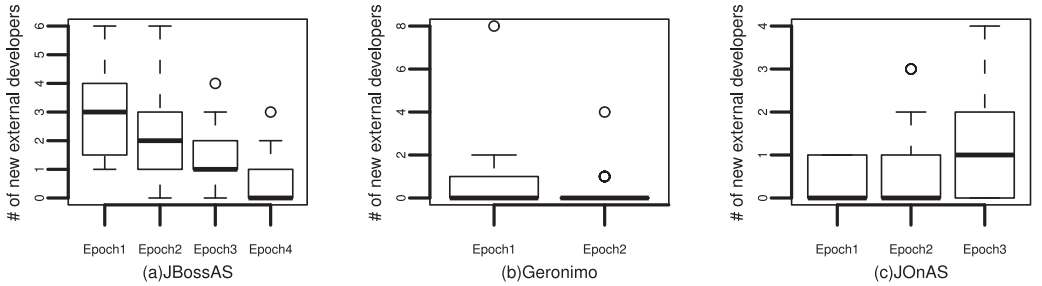


Fig. 2. Inflow of external developers in JBossAS, Geronimo, and JOnAS.

Table VI. Inflow of External Developers (Deviance Explained: 0.36, 0.14, 0.10)

JBossAS	Estimate	Std. Error	*p-value*
(Intercept)	1.07	0.18	<0.0001
epoch2	−0.32	0.22	0.16
epoch3	−0.86	0.28	0.003
epoch4	−2.06	0.35	<0.0001
Geronimo	Estimate	Std. Error	*p-value*
(Intercept)	−0.21	0.4	0.6
epoch2	−1.48	0.68	0.03
JOnAS	Estimate	Std. Error	*p-value*
(Intercept)	−0.99	0.34	0.005
epoch2	0.67	0.42	0.11
epoch3	1.09	0.39	0.007

Figure 2 shows the box-plot[16] of the number of new volunteers per month in JBossAS, Geronimo, and JOnAS within each epoch. For example, during $Epoch_1$ of JBossAS, the number ranges from one to six (i.e., $[1, 6]$) and the median is three. In $Epoch_3$, the range narrows down to $[0, 3]$ with a median of one. To confirm the observed variations via statistical tests, we model the relationship between the number of new volunteers and the epochs that represent the types of involvement models. We use the Generalized Linear Model (GLM) for the overdispersed Poisson distribution (a suitable distribution for the relatively low monthly counts of newcomers):

$$N_{E_{new}} \sim \sum_i Epoch_i, \tag{1}$$

where $N_{E_{new}}$ is the number of new external developers per month, and $Epoch_i$ is an indicator function for each epoch. The estimated coefficients and standard errors are presented in Table VI. The coefficients for $Epoch_{i>1}$ are relative to $Epoch_1$. Positive (negative) coefficients show the increase (decrease) of the influx of volunteers as compared to $Epoch_1$.

Figure 2 and Table VI show that the number of newcomers per month significantly decreases in hybrid epochs ($Epoch_2$ through $Epoch_4$) as compared to open-source epochs ($Epoch_1$) in JBossAS and Geronimo. In JBossAS, the median number of monthly newcomers in the hybrid epochs is lower than in the first epoch, and the negative coefficients for the three epochs ($Epoch_3$ and $Epoch_4$ with $p\text{-}value < 0.01$ and < 0.001) indicate that

[16]The lower and upper boundaries of the rectangle of the box-plot represent the first and the third quartile, respectively, and the thick line is the median. The lowest and upper horizontal lines represent the 1.5 quartiles away from the mean. The points below or above the horizontal lines are suspected outliers.

the change is unlikely to be a random fluctuation. This suggests a decrease of contributor influx. Similarly, in Geronimo, the median number is zero for both epochs (more than half of the months had no newcomers), but the maximum and the third quartiles are lower in the hybrid epoch ($Epoch_2$) than in the open-source epoch ($Epoch_1$). In fact, during $Epoch_1$ of Geronimo, 17 volunteers joined during 21 months, but during the entire $Epoch_2$ of 49 months, only nine external developers joined. Also, the negative coefficient for $Epoch_2$ is statistically significant ($p\text{-}value < 0.05$), indicating a decrease of volunteer inflow in the hybrid epoch.

In summary, we observe *Hosting* and *Supporting* hybrid epochs to be associated with a significant decrease of new external contributors in comparison to the pure open-source stage. It is possible that the decrease was caused by the community involvement actions the companies took. A *Hosting* company aims to make profit directly from and is in full control of the project, and it is likely to enforce its professional policies to ensure control. For example, to provide high-quality software and subscription services, RedHat requires that contributors grant copyright to the company, as described in Section 4.1. This might detract some open-source volunteers, as a vigorous debate suggests,[17] and make it more difficult to become a contributor by adding another step to the process. Even if volunteers want to contribute code, the company may be cautious and not include their code in the product: as one developer complained, "the patches we sent were ignored."[21] In a *Supporting* hybrid, although the company does not have full control of a hosting project, it might still be necessary to provide many full-time developers to keep project development in line with the company's business objectives. This would make it difficult for volunteers to participate, because a large group of developers hired by the supporting companies (especially by one company in this case) may dominate the community. One of the external developers in Geronimo illustrated this concern after IBM got involved: "The result is new potential contributors (non-IBM employees), which will have only limited time to contribute, understand less of what is under the hood, and as a result, are not able to easily get involved."

Furthermore, an intensified commercial involvement may be related to either a decreased or an increased volunteer inflow. For example, for JBossAS, Figure 2 suggests a more rapid decrease of new volunteers with RedHat's involvement in $Epoch_4$. It implies that *Hosting* hybrid epochs with more commercial resources (e.g., developers, marketing) as input, as in $Epoch_4$ of JBossAS, might deter volunteers. Meanwhile, for JOnAS, there is a significantly higher volunteer inflow in $Epoch_3$, when the *Collaborating* activities were intensified. In this epoch, Bull began to adopt an explicit cooperation broadening action (e.g., with Peking University) to build its new-generation application server with OSGi techniques. Many contributors who shared the interest in technical innovation joined during $Epoch_3$ of JOnAS.

While JOnAS had an increasing number of new contributors over time, JBossAS has experienced a notable decrease. This difference is especially striking because both projects have similar epochs punctuated by the change in the same technical requirements. Moreover, the number of new developers in total increased over time, as shown in Figure 3, suggesting an expanding development team in all three projects. This similarity in context makes it more likely that the divergence of developer inflow was caused by the differences in the type of commercial participation. For example, a *Collaborating* hybrid focuses mainly on technical requirements and tends to have looser restrictions on developer participation than a *Hosting* hybrid that focuses mainly on direct profits.

[17]http://blogs.webtide.com/gregw/entry/marc_fleury_leaves_jboss_RedHat, Record redhat13 in SOM[27].

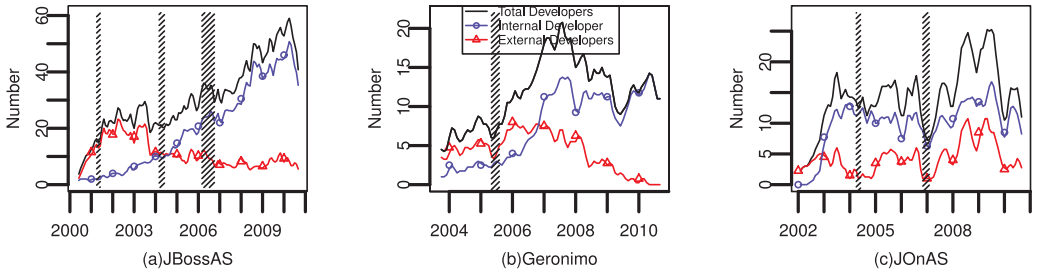


Fig. 3. Inflow of developers in JBossAS, Geronimo, and JOnAS.

Table VII. Inflow of External Developers in JBossAS

Number of External Developers Joining by Epoch			
	App Dvpr In-House+Consulting	Infrastructure Dvpr	Total
epoch1	8 + 5	4	43
epoch2	16 + 4	10	85
epoch3	2 + 5	6	56
epoch4	5 + 3	6	117
Inflow of External App Developers by Epoch (Deviance Explnd: 0.97)			
	Estimate	Std. Error	*p-value*
(Intercept)	−0.08695	0.38958	0.823
epoch	−0.61268	0.15187	5.48e-05

In summary, we have the third observation.

OBSERVATION 3. *As the intensity of commercial involvement increased, JBossAS (Hosting model) and Geronimo (Supporting model) had a decrease of external inflow, while JOnAS (Collaborating model) had an increase of external inflow.*

To understand the reasons for the decreased inflow of external developers, we conducted a further investigation of JBossAS.[18] As described in Section 3.5.3, we divided the external developers into different groups based on the nature of their work: application developers writing code that runs on the application server (for in-house or consulting development) and infrastructure developers writing application server itself. Table VII shows the number of new joiners in each group at different epochs whom we could identify, with the last column showing the total number of new contributors. We fit a GLM model to observe whether the number of new joiners in the application developer role drops over time:

$$UserRole \sim Epoch,$$

where *UserRole* is the indicator function for each new joiner. It is one if the new joiner has an application developer role and zero otherwise. *Epoch* is a number indicating the epoch (from one to four) to represent the increase in the intensity of the commercial involvement.

As Table VII shows, the probability that a new joiner will have an application developer role drops significantly in the later epochs of JBossAS. We also fit another GLM to model the chances that a new joiner will be in the application server developer role. The results show that the probability of a new joiner being in an infrastructure developer role does not change significantly. This suggests that, with a higher intensity of commercial involvement, fewer application developers may participate in the project, but

[18]Geronimo had only two epochs and a very small sample: nine external developers in the second epoch.

the involvement may have less impact on the infrastructure developers. Since software users who contribute to the project are found to bring innovation to the project, the drop in their numbers may indicate a potential threat to project innovation.

This may appear to be counterintuitive because the application developers may appreciate the large and rapidly growing user base of JBossAS and long-term commercial support associated with higher levels of involvement. However, we only investigated the contributions to the application server itself, and the more restrictive licensing terms appear to have had an effect on external contributions from this large base of potential contributors.

4.2.2. Retention of Existing Developers. Experienced developers are necessary for software projects to succeed. Projects that are able to retain developers for longer periods increase the pool of expertise: as the productivity of project novices increases, they gain sufficient experiences to be able to conduct critical project tasks [Shah 2006; Zhou and Mockus 2010]. Thus, once contributors join, it is important to keep them for extended periods of time. Also, software projects may need six or even more novices to replace the productivity of a single experienced developer [Mockus 2009]. Finally, losing an existing developer is detrimental to the project, for example, in the time delays of finding a replacement, in the effort spent on training them, and in the reduced quality due to lost expertise [Mockus 2010]. These and other issues could be avoided with better retention.[19]

There is ample evidence that retention is an important consideration in hybrid projects. For example, the Mozilla community has built a "community manager" dashboard in order to detect (and help retain) the volunteer participants who are about to leave the project.[20]

Measuring retention is complicated as we may not know how long people who have joined may stay with the project and developers may go idle for several periods. For developers whose tenure spans epoch boundaries, it is not clear whether or not the change of epochs causes their departure. In such cases, the time a developer leaves the project is not observed within the relevant epoch, leading to censored observations (a censored observation is a developer who stays across the epoch boundary, and therefore is censored in this epoch). We therefore consider a contributor's tenure as the time from joining until leaving the project or until the epoch boundary, whichever occurs earlier. As a result, we have uncensored observations of tenures for developers who leave within the epoch and censored observations of tenures for the other developers.

Survival analysis techniques are designed to explicitly model the fact that some observations are censored. For this work, we used the "survival" package in R [R Development Core Team 2008]. Our data involves a "right censoring";[21] that is, suppose that a developer joins at time zero and our observation of project state is made at time t, then we only know that the duration of his or her stay (tenure) is of length $tenure > t$ if that developer is still with the project at time t.

Figure 4 shows the survival curves of external developers: the horizontal axis represents the tenure of external developers who join during each epoch (i.e., the number of months into the epoch), and the vertical axis shows the proportion of these newcomers who reach or exceed that tenure. For comparison, Figure 5 shows the survival curves for all developers. As we can see from both figures, the later epochs of JBossAS have a higher proportion of "survivors." This suggests that the company actions in this epoch

[19]http://www.productivityhacks.com/productivity-tips-2/retaining-your-employees-and-increasing-their-productivity/.

[20]http://eaves.ca/2011/04/07/developing-community-management-metrics-and-tools-for-mozilla/.

[21]A survival time is censored if all that is known is that it began or ended within some particular interval of time, and thus the total spell length (from entry time until transition) is not known exactly.

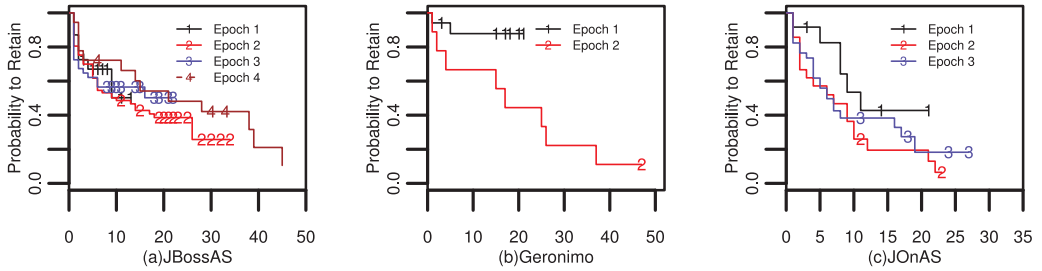


Fig. 4. Survival curves of new external joiners in JBossAS, Geronimo, and JOnAS.

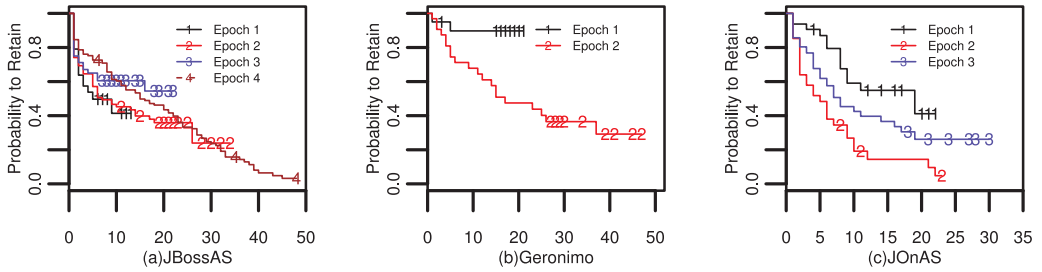


Fig. 5. Survival curves of new joiners in JBossAS, Geronimo, and JOnAS.

may improve the retention of developers, and more intensive commercial involvement of later stages of JBossAS may have had an even more pronounced effect on retention. For example, if we chose a tenure of 12 months, the survival curve for JBossAS external developers (or overall developers) shows that approximately 50% (or 40%) of developers reach that tenure in the pure OSS epoch, which increases to almost 70% (or 60%) in the later epochs. Geronimo appears to exhibit the opposite trend: in the hybrid epoch, retention is reduced for external or overall developers. JOnAS started as a hybrid project and evolved with technology changes similar to JBossAS. However, in a striking contrast to JBossAS, the retention of new joiners decreases in the later epochs.

To verify the observed differences in the retention (survival) curves, we constructed a parametric Cox proportional hazards regression model.[22] In the model, the response is the survival object (including the observed duration of stay within the epoch and the censoring indicator) for each developer, and the epoch indicator is one of the predictors. Such a simple model shows that the observed differences are statistically significant.

To investigate the retention of developers who started in the previous epoch, we used another model. Specifically, in addition to having an observation for each developer in the epoch he or she started (as in the first model), we added another observation if the developer stays until the next epoch. In the added observation, the tenure is counted from the start of the (previous) epoch until the developer leaves or the subsequent (the third) epoch starts (the number of months), whichever occurs earlier. To make sure that these additional observations are recognized and treated differently by the model, we added an extra predictor: the developer's tenure in the previous epoch. For observations representing new joiners in a particular epoch, the predictor is always zero, while for observations representing the same developer in a subsequent epoch, it

[22]Commonly used for censored observations.

Table VIII. Retention of Developers, Deviance Explained: 0.08, 0.12, 0.17

JBossAS	coef	exp(coef)	*p-value*
epoch2	−0.58	0.56	0.02
epoch3	−0.86	0.42	0.003
epoch4	−1.09	0.34	<0.0001
$tenure_{Epoch_{i-1}}$	−0.026	0.97	0.03
Geronimo	coef	exp(coef)	*p-value*
epoch2	1.7	5.7	0.02
$tenure_{Epoch_{i-1}}$	−0.02	0.98	0.2
JOnAS	coef	exp(coef)	*p-value*
epoch2	0.6	1.9	0.03
epoch3	0.5	1.7	0.06
$tenure_{Epoch_{i-1}}$	−0.07	0.9	<0.0001

is equal to their tenure in the prior epoch:

$$Tenure_{epoch_i} \sim Epoch_i + Tenure_{Epoch_{i-1}}. \quad (2)$$

Table VIII shows that JBossAS has a negative coefficient of −0.58 for $Epoch_2$. This means that the dropout rate decreased (retention rate increased) in $Epoch_2$ as compared to $Epoch_1$. The results for JBossAS, Geronimo, and JOnAS are consistent with the survival or retention curves in Figure 5.

Moreover, in JBossAS and JOnAS, the negative coefficients at $Tenure_{Epoch_{i-1}}$ indicate that developers with a longer tenure in the prior epoch have a lower probability of leaving the project ($p\text{-}value < 0.05$). It appears that hybrid projects did a great job of retaining existing contributors. Considering the variances of the retention of new joiners in JOnAS and JBossAS, the improvement in developer retention in JBossAS was not likely caused by technology changes, but rather by changes in commercial participation or other factors. It therefore appears that the variation of developer retention among hybrid epochs is more likely to be attributed to variations in commercial involvement models.

In the JBossAS *Hosting* epochs, the actions and policies employed by the company improve developer retention as compared to the open-source epoch, and more extensive commercial involvement has a more pronounced effect on retention. This observation is consistent with the fact that in *Hosting* models, communities are managed and operated by the backing company. The goal of retaining experienced and productive developers is aligned with company objectives for achieving product and service stability, which is often claimed to be the main value companies add to OSS development. The vice president of RedHat's middleware business confirmed in a news report that RedHat did want to achieve high developer retention.[23]

In Geronimo, retention has not increased partially because IBM frequently reassigned employees supporting Geronimo. For example, one developer *cjblythe* (Christopher Blythe) committed code for Geronimo for 9 months and then returned to his previous assignment in WebSphere development. Although there may be other factors that determine who joins or leaves the project, the fact that 35 out of 63 committers were employed by IBM indicates IBM's influence as the most important factor. In particular, in the open-source epoch of Geronimo, only two out of 20 developers left, while the rest stayed until $Epoch_2$. However, this gives us only two complete observations for the model. Such a sample is too small to draw general conclusions.

[23]http://news.cnet.com/8301-13505_3-9871387-16.html.

As Figure 4 shows, in the JOnAS *Collaborating* hybrid epochs, the variation in community involvement actions does not appear to have a pronounced effect on the retention of developers. However, a small group of contributors with a long tenure in prior epochs appears to have a very low probability of leaving, as explained by $Tenure_{Epoch_{i-1}}$ in Table VIII. The company using the *Collaborating* model focuses mainly on technical innovation and deploys limited resources (in the JOnAS case a small group of paid core developers), while in most cases the remaining developers participate when they have a shared interest.

In summary, we have the following observation.

OBSERVATION 4. *JBossAS (Hosting model) had improved retention, while JOnAS (Collaborating model) had improved retention of the long-term contributors with the remaining contributors having reduced tenure. Geronimo (Supporting model) had reduced tenure.*

5. VALIDATION

We used a variety of ways to validate the following three concerns: (1) whether we identified the external developers correctly, (2) whether the homogeneous periods and the corresponding commercial involvement practices were defined appropriately, and (3) whether the involvement actions employed by the companies affected contributor inflow and retention. The first two validations ensured that our methods accurately reflect the reality. Whether or not the operational data, such as software repository artifacts, accurately reflect reality is a critical concern because such data is not a result of a carefully designed measurement system [Mockus 2014]. Once the facts are established, the analytic construction (i.e., the measures and the model) may proceed.

(i) We first used emails and face-to-face interviews with the project participants, managers, and open-source consortium representatives. We posted requests on the three projects' public mailing lists in the middle of February 2012.[24] We interviewed core members of JOnAS and JBoss during their open-source community days.

In total, we obtained 17 email responses and conducted 11 interviews. First, the responses and interviews support our approaches of using email handles to identify internal and external developers. The JBoss interviewees stated clearly, "If a developer used an email with @redhat, @jboss.com, @jboss.org, they should be employed by RedHat or JBoss, otherwise, they would not have such emails." Some external contributors were later employed by RedHat or JBoss Inc. We counted them as *external* developers at the time they made their first commits. JBoss interviewees agreed with this approach. In JOnAS, one interviewee mentioned that "There are some Bull employees in the community council using '@ow2.org' or '@objectweb.org' email addresses," supporting our approach of using these two kinds of email addresses to identify internal developers. We used the committer list on the Geronimo website to determine IBM developers. Geronimo respondents confirmed it as a valid approach.

Second, all the respondents (except for the JBoss person who responded in the public mailing list but did not answer the questions) agreed with the definitions and the timeframes of the epochs we used to separate the types and intensity of commercial involvement presented in Table V. The JBoss community director confirmed the control mechanisms adopted by RedHat to manage JBoss and, in particular, explained the mechanisms for hiring OSS contributors. Indeed, RedHat and JBoss Inc. were

[24]https://community.jboss.org/thread/194916?_sscc=t, http://mail-archives.apache.org/mod_mbox/geronimo-user/201202.mbox/%3c1328676956.6341.193.camel@maxj07-laptop%3e, http://mail-archive.ow2.org/jonas/2012-02/msg00004.html.

looking for talented and experienced OSS contributors and would compensate them for contributing to JBoss.

IBM's policies with respect to Geronimo were unanimously confirmed by the respondents. There were two types of opinions about IBM's commercial objective for Geronimo. One group stated that Geronimo was a technical solution for IBM's commercial product WebSphere. The other group considered it as a market strategy to expand to the low-end JaveEE application server customers.

The OW2 CTO, who is also a JOnAS developer, carefully checked the goal and participating actions adopted by Bull, as shown in Table IV and V. He agreed with our results but noted that we omitted two organizations that Bull collaborated with in $Epoch_2$. Universidad Politécnica de Madrid and RedHat participated in JOnAS for less than 3 months and were not listed on the official JOnAS website. However, this does not affect the validity of the model we obtained for JOnAS—Bull has a shared control over JOnAS with other organizations.

Third, respondents agreed that the commercial involvement actions may have an impact on contributor inflow and retention. For example, the JBoss director agreed that there is an improved retention in JBossAS. Similarly, the JBoss interviewees explained the reason of long tenure after the RedHat acquirement: "Being paid to do open source. Why would I leave here?"

(ii) We conducted a survey through private email to elicit respondents' opinions about the impact of commercial involvement. To get the most information from a limited number of interviewees, we selected several developers still working on the project (until September 2010) and with the largest number of followers [Mockus 2009]. We then used their names plus the name of the projects they work on to search Google and selected ones who were active in the forum or mailing list. Eventually, we selected two developers from each project (six in total) who were both influential and active, and we sent them questions (an example email is in Online Appendix C). Most of our survey questions were close ended with a few optional open-ended questions for collecting participants' "insights" and "experiences."

The four developers from Geronimo and JOnAS responded. They answered either "very likely" or "likely" when asked whether they perceived the inflow (and retention) change after the company involvement (see Online Appendix C). They also offered insights on why that may be the case. For example, the decreased inflow in Geronimo is because of "Business economics": "If a company like IBM enters a project and plans on 'selling' a product (or support a product), there's reduced incentive for other people/companies to contribute (any long term benefit has been reduced)." Meanwhile, the decreased retention in Geronimo is because "job assignments within a company change. Centers of development may move from one location to another. So, people may join a project (for their company), then be asked to work on something different."

6. DISCUSSION

6.1. Evolution of Commercial Involvement

To observe whether commercial involvement evolves in the three projects, we read recent articles and forum discussions and communicated with core developers at the end of 2014. The three projects have changed over the 2 years since our study was completed. Geronimo has low activity, although their community and committers remain. A perception that IBM decided to focus on the commercial WebSphere application server instead of investing much in Geronimo itself was expressed by our interviewees. As a result, IBM has reassigned their employees to other projects, which reduced the developer tenure in Geronimo.

Bull used to have a small group of employees assigned to work on JOnAS. That is one reason JOnAS had a higher retention rate for long-term developers. However, Bull

stopped investing in JOnAS, with some of the long-standing developers leaving Bull and/or JOnAS for other projects. For example, some JOnAS developers who stayed with Bull were reassigned to work on the Internet of Things, and only maintenance activities are performed for some customers having existing JOnAS deployments.

WildFly 8 is the direct continuation of the JBossAS project. RedHat renamed JBossAS because they want people to know JBoss "is a commercial product with a well-defined SLA and support lifecycle, in contrast to the community project with best effort forum-based support provided by the community (i.e., WildFly)."[25]

In summary, commercial objectives drive commercial involvement, and popular technology is one of the important factors that motivate an objective. JavaEE is currently an established market with a number of proprietary and open-source products. So the interest in application server technology has waned. However, the OSS community may suffer—it may lose the capability of attracting and retaining contributors itself during the intense commercial involvement, so when the company loses its interests and withdraws its resources, the project may end up failing. Fortunately, a new generation of technology and contributors may arise and produce new projects and new communities from the old ones. For example, Geronimo was created in response to "JBoss owned rights to the name and insured that nobody else could make money from supporting JBoss," according to one of our Geronimo respondents. Another Geronimo respondent reported that TomEE, a newer Apache project (the Java Enterprise Edition of Apache Tomcat, i.e., Tomcat + JavaEE = TomEE), which released its first beta version in October 2011, is taking developers from Geronimo.

6.2. Implications for Research

Natural Experiment. The existence of many potential confounding factors makes it extremely difficult to understand software development practice. As shown by this study, a natural experiment can control for technology, environment, and project context, which are often the most important factors that drive software practices. Double-blind controlled experiments are generally not possible in realistic software development settings, and natural experiments provide a way to increase the validity of the results in the software engineering domain.

Two Dimensions of Hybrid Space. The dimensions and coordinates of hybrid space presented in this study appear to reveal the alignment between commercial objectives and employed actions in commercial organizations. Finding the position of additional projects in the hybrid space may reveal new flavors of hybrid development and would likely highlight ways to both extend it and make it more precise and repeatable.

6.3. Recommendations for Practice

Companies that use an OSS product for direct profit or that critically depend on it to conduct their business might consider a *Hosting* model with strong control over the project. This would most likely lead to a more stable and productive development team but would not attract many external contributors because of corporate control on IP and code commits. It is not clear whether the achieved stability would always be worth the price of lost innovation.

The intensity of involvement must be carefully calibrated for the *Supporting* model to avoid putting the project at risk. In our study, both the inflow of external developers and the retention of existing developers were negatively affected in the *Supporting* model. This casts doubts on the extent of innovation from external inflow or gains in productivity and quality from retention that the project under study has captured. It suggests that there may be more effective approaches to attract contributors. For example, projects in the Eclipse ecosystem are encouraged to incorporate contributions

[25]https://developer.jboss.org/blogs/mark.little/2012/10/06/jbossas-renaming.

from developers employed by multiple firms. This is believed to help guard against a single firm leaving the ecosystem and causing a variety of possibly critical projects to falter [Wagstrom 2009]. It is worth noting that Geronimo was dominated by one company (i.e., IBM), and that may differentiate it from other supporting hybrids.

In contrast to the *Hosting* and *Supporting* models, companies that use the *Collaborating* model would share control with other organizations and invest limited resources in the projects. This may help avoid the commercial domination of the community that was observed in the other two models. Therefore, in this model, the interest in technical innovation may determine the coordination mechanisms and task assignment that are similar to open-source development practices. However, its poor record of retaining new contributors may have failed JOnAS, because once the company changed its interests, it moved the long-standing JOnAS developers to other areas, leaving the original with a lack of support.

In summary, to take full advantage of OSS development, it is important to structure community involvement policies and actions into a coherent model, and to understand the ramifications each action may have on contributor inflow and retention.

7. LIMITATIONS

Unlike an experiment, where the subjects and treatments are randomly selected, a natural experiment does neither, reducing internal validity and the ability to interpret the discovered relationships as causal. At the same time, it offers higher external validity as the relationships do exist in real life, not only under tightly controlled experimental conditions.

7.1. Data

Identification of Developers. The same person may have used different logins over time. For example, many JBossAS developers changed their logins after the acquisition by RedHat. This ambiguity might inflate the number of new joiners. To address this issue, we manually inspected the similarity and regularity of all the logins for the three projects. We found 15 of the 414 logins from JBossAS that we could not link to a person's name, such as *nobody123* or *uid48247*. For Geronimo and JOnAS, we successfully identified all the logins. By conducting our analysis both with and without the few logins for which we could not identify a developer, we found that our results were not affected in a noticeable way.

To obtain the inflow of external developers, we did not distinguish whether they participated voluntarily or worked for a cooperating organization. We chose to do so because these developers can make valuable contributions just as well as volunteers. Another reason is the difficulty (or impossibility) of identifying true "volunteers."

Joining and Leaving Dates of Developers. In OSS projects, developers may join or leave the project at will, and, in hybrid projects, "if someone stops commit code, it does not always mean the developer has left the company. Maybe they have become a manager or transferred to other development teams," as one of our JOnAS interviewees commented. Typically, there is no information about the exact date when a developer joins or leaves the project. To approximate these dates, we used the dates of the first and the final commits. If the final commit is recent, it may not represent the date a developer leaves: he or she may commit again in the near future. We therefore excluded developers who joined within 3 months of the date we completed our data collection. The JOnAS interviewees confirmed that among the developers we assumed leaving the project, about 80% really left the project due to retirement, student internship, changing jobs, and so forth. The other 20% who stopped committing code may continue their career in management.

One concern of Geronimo respondents was that the first commit time might have deviation from the joining date of a developer. In Apache, before a developer became

a code committer, he or she should show sufficient sustained contribution, such as submitting patches, answering emails, and so forth. Then, the first commit time might have deviation from the joining dates of developers. The respondents stated that this duration usually took 6 months on average. However, in Geronimo, many IBM employees were project management committee members. The duration might be shorter for IBM employees who apply for commit privilege, as the interviewees indicated. Once a developer becomes a committer, his or her contributions get acknowledged and can exercise voting rights on the development of the project. The first commit date, therefore, could imply the developer truly joining the project.

7.2. Internal Validity

Using Public Materials. One of our respondents mentioned that he did not want to share certain kinds of information publicly, which brings up the question: to what extent do public channels reveal the information we seek? This may limit our approach of looking in public channels.

Division of Epochs. The particular approach of dividing project timelines into epochs might be debatable. For example, we could have divided the whole project lifespan into epochs of equal duration or used the same calendar time to separate all three projects into epochs. The presented approach is mainly based on the time when the companies got involved and the commercial strategies changed according to remarkable technique waves. To exclude noise added by the transition between epochs, we excluded short periods at epoch boundaries.

Low Counts of Developers. To model relatively low monthly counts of newcomers, we used a generalized linear model for the overdispersed Poisson distribution instead of using a linear regression model. In addition, we tried to use these smaller numbers to our advantage by investigating more closely who was joining and whether these specific events were related to the particular involvement model. The small number of developers in Geronimo and JOnAS enabled us to investigate whether they were volunteers when they joined and whether they were hired afterward.

Another issue we encountered in modeling the numbers of developers is autocorrelations. The number of developers during month i is strongly correlated with the number of developers in month $i-1$. This means that we cannot use monthly numbers as observations in a regression model, because (non-time-series) regression models require the observations to be independent. If observations are dependent, the standard errors and p-values would be underestimated. We therefore checked our data for autocorrelations and only reported models where no autocorrelation was found. For example, although the number of developers in each month is strongly autocorrelated, the number of new developers is not.[26]

Confounding Factors. Our findings show an association between the response and predictors, but that association may not be causal. There may be some aspects that we did not measure but that cause both the response and the predictors to behave in the observed pattern.

For example, there are other reasons that new developer inflow may go down. First, when the product becomes stable, there is no need for new features or stability improvements; thus, there is no need for new contributions (and contributors). In this study, the similarity in context makes it more likely that the divergence of developer inflow was caused by the differences in the type of commercial participation. Second, as the development team becomes more professional, it may be more difficult for volunteers to participate—we observed it in Geronimo and JBossAS—and this factor represents

[26]Because developer churn is low from one month to the next, the number of developers in month i is strongly correlated (autocorrelated) with the number of developers in month $i+1$.

one of the defining features of commercial involvement. This effect may be particularly strong in the *Hosting* model that creates a feedback loop: increased retention leads to increased tenure and, in turn, to increased developer experience, thus making it harder for novices to contribute.

7.3. External Validity

The three projects in our investigation are JavaEE application servers. This may limit our findings to a single domain. We made this choice to control for the variations that may be due to differences in application domains.

Also, the models may be defined by the specific companies involved. For example, IBM has a clear agenda for those development teams considering an open-source proposal, including templates that address questions covering every critical area [Capek et al. 2005], that may limit the model to IBM-like companies.

Even so, the way the three projects are operating is not unusual for hybrid projects; in fact, it may be typical. For example, by answering the first three questions (to which we could easily obtain answers for) at the hybrid space of Figure 1, we can say that Oracle hosts MySQL, HP, RedHat, and Rackspace and many other companies support OpenStack together, and IBM collaborates with other companies to run Eclipse. However, to determine the model precisely, the process presented in Section 3.4 is needed.

As pointed out by Yin [2009], "How can you generalize from a single case?" is a frequently heard question for case studies. The answer is not simple, but the short answer is that case studies are generalizable to theoretical propositions and not to populations or universes. The theoretical propositions in this study are the dimensions and coordinates of commercial involvement derived from the literature and documents. In other words, the actual number of models that are discussed in the literature and can be described via these dimensions and coordinates is large, and if the presented values in the dimensions and coordinates for two projects are similar, there could be an overlap.

8. CONCLUSION

Open source has become mainstream, and commercial involvement is a recognized phenomenon, as demonstrated by the popular hybrid projects such as OpenStack, Docker, and Android. In order to understand how commercial involvement impacts participant inflow and retention, we have observed a parallel evolution of three OSS projects from the JavaEE application server domain, with two of the projects transitioning from pure OSS to a hybrid model, and all projects encountering changes in the nature or intensity of commercial involvement. We located three community involvement models and found each model to be associated with developer participation in a unique way: the *Hosting* model was associated with the best contributor retention and with reduced inflow of new contributors. The *Collaborating* model was associated with increased inflow of new contributors and with improved retention of long-term developers. The remaining contributors had reduced tenure associated with this model. The *Supporting* model was associated with decreased inflow and retention as compared to the earlier OSS stage. We believe that our results could help companies better align their goals with the approaches they use to engage OSS development, thereby improving the effectiveness and prevalence of hybrid projects.

In summary, this article makes the following main contributions:

—A natural experiment-inspired method that uses multiple epochs of three projects developing the same technology over the same period of time to control for technology evolution and a number of other external factors. It also uses a set of seven questions to identify and compare community involvement models in the hybrid space.

—Three commercial community involvement models and their hypothesized impact on developer inflow and retention.

We expect that these theoretical constructs will be applied and tested in other projects and in actual hybrid development. To facilitate replications or other types of future work, we the provide data, scripts, and retrieved materials used in this study in supporting online material (SOM).[27]

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

REFERENCES

P. J. Agerfalk and B. Fitzgerald. 2008. Outsourcing to an unknown workforce: Exploring opensourcing as a global sourcing strategy. *MIS Quarterly* 32, 2 (June 2008), 385–409.

B. C. D. Anda, D. I. K. Sjøberg, and A. Mockus. 2009. Variability and reproducibility in software engineering: A study of four companies that developed the same system. *IEEE Transactions on Software Engineering* 35, 3 (May/June 2009).

C. Bird, A. Gourley, P. Devanbu, A. Swaminathan, and G. Hsu. 2007. Open borders? Immigration in open source projects. In *Proceedings of the 4th International Workshop on Mining Software Repositories (MSR'07)*. IEEE Computer Society, Washington, DC, 6.

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

A. Bonaccorsi and C. Rossi. 2006. Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business. *Knowledge, Technology, and Policy* 18, 4 (2006), 40–64.

P. G. Capek, S. P. Frank, S. Gerdt, and D. Shields. 2005. A history of IBM's open-source involvement and strategy. *IBM Systems Journal* 44, 2 (2005), 249–257.

H. Chesbrough. 2007. Why companies should have open business models. *MIT Sloan Management Review* 48, 2 (Winter 2007), 22–28.

H. Chesbrough and R. S. Rosenbloom. 2002. The role of the business model in capturing value from innovation: Evidence from Xerox corporation's technology spin-off companies. *Industrial and Corporate Change* 11, 3 (2002), 529–555.

K. Crowston, K. Wei, J. Howison, and A. Wiggins. 2012. Free/libre open source software development: What we know and what we do not know. *Computing Surveys* 44, 2, Article 7 (Feb. 2012), 35 pages.

B. Curtis, E. M. Soloway, R. E. Brooks, J. B. Black, K. Ehrlich, and H. R. Ramsey. 1986. Software psychology: The need for an interdisciplinary program. *Proc. IEEE* 74, 8 (Aug. 1986), 1092–1106.

L. Dahlander and M. Magnusson. 2008. How do firms make use of open source communities? *Long Range Planning* 41, 6 (2008), 629–649.

H. Dietmar, J. Henkel, and E. von Hippel. 2002. Profiting from voluntary information spillovers: How users benefit from freely revealing their innovations. *MIT Sloan School of Management Working Paper 4125* (May 2002).

J. Dinkelacker, P. K. Garg, R. Miller, and D. Nelson. 2002. Progressive open source. In *Proceedings of the 28th International Conference on Software Engineering (ICSE'02)*.

T. Dunning. October 2012. *Natural Experiments in the Social Sciences: A Design-Based Approach*. Cambridge University Press.

J. Henkel. 2006. Selective revealing in open innovation processes: The case of embedded Linux. *Research Policy* 35 (2006), 953–969.

X.-J. Ma, M.-H. Zhou, and D. Riehle. 2013. How commercial involvement affects open source projects: Three case studies on issue reporting. *Science China Information Sciences* 56, 8 (2013), 1–13.

M. Markus. 2007. The governance of free/open source software projects: Monolithic, multidimensional, or configurational? *Journal of Management and Governance* 11, 2 (2007), 151–163. 10.1007/s10997-007-9021-x.

A. Mockus. 2009. Succession: Measuring transfer of code and developer productivity. In *Proceedings of the 2009 International Conference on Software Engineering*. ACM.

[27]https://www.passion-lab.org/projects/hybrids.html.

A. Mockus. 2010. Organizational volatility and its effects on software defects. In *ACM SIGSOFT/FSE*. 117–126.

A. Mockus. 2014. Engineering big data solutions. In *FOSE, ICSE 2014*.

A. Mockus, R. F. Fielding, and J. Herbsleb. 2000. A case study of open source development: The Apache server. In *Proceedings of the 22nd International Conference on Software Engineering*. 263–272.

A. Mockus, R. T. Fielding, and J. Herbsleb. 2002. Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology* 11, 3 (July 2002), 1–38.

A. Mockus and J. Herbsleb. 2002. Why not improve coordination in distributed software development by stealing good ideas from open source. In *Proceedings of the Workshop on Open Source Software Engineering (ICSE'02)*. Orlando, FL, 35–37.

N. Munga, T. Fogwill, and Q. Williams. 2009. The adoption of open source software in business models: A red hat and IBM case study. In *Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*.

D. Nagy, A. M. Yassin, and A. Bhattacherjee. 2010. Organizational adoption of open source software: Barriers and remedies. *Communications of the ACM* 53, 3 (March 2010), 148–151.

R Development Core Team. 2008. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

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

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

V. S. Sinha, S. Mani, and S. Sinha. 2011. Entering the circle of trust: Developer initiation as committers in open-source projects. In *MSR'11*.

J. Van Maanen and E. Schein. 1979. Towards a theory of organizational socialization. In *Research in Organizational Behavior*, B. M. Staw (Ed.). Vol. 1. JAI Press, Greenwich, CT, 209–264.

E. von Hippel. 2002. Open source projects as horizontal user innovation networks - by and for users. *MIT Sloan School of Management Working Paper 4366-02* (June 2002).

E. von Hippel and G. von Krogh. 2003. Open source software and the private-collective innovation mode: Issues for organization science. *Organization Science* 14, 2 (March/April 2003), 209–223.

G. von Krogh, S. Haefliger, S. Spaeth, and M. Wallin. 2008. Open source software: What we know (and do not know) about motivations to contribute. In *Proceedings of the DRUID Conference 2008, the University of Gothenburg Research Seminar, and the Open and User Innovation Workshop 2008 at Harvard Business School*.

G. von Krogh, S. Spaeth, and K. R. Lakhani. 2003. Community, joining, and specialization in open source software innovation: A case study. *Research Policy* 32, 7 (July 2003), 1217–1241.

M. von Zedtwitz and O. Gassmann. 2002. Market versus technology drive in R&D internationalization: Four different patterns of managing research and development. *Research Policy* 31 (2002), 569–588.

P. Wagstrom, J. Herbsleb, R. Kraut, and A. Mockus. 2010. The impact of commercial organizations on volunteer participation in an online community. In *Proceedings of the Academy of Management Annual Meeting*.

P. A. Wagstrom. 2009. *Vertical Interaction in Open Software Engineering Communities*. PhD Thesis, Carnegie Mellon University, CMU-ISR-09-103 (March 2009).

J. West and S. Gallagher. 2006. *Open Innovation: Researching a New Paradigm*. Oxford University Press.

Y. Ye and K. Kishida. 2003. Toward an understanding of the motivation of open source software developers. In *ICSE'03*. 419–429.

R. K. Yin. 2009. *Case Study Research: Design and Methods*. 4th ed. SAGE Publications.

C. You, M. Zhou, Z. Xiao, and H. Mei. 2009. Towards a well structured and dynamic application server. In *Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference*.

M. Zhou and A. Mockus. 2010. Developer fluency: Achieving true mastery in software projects. In *ACM SIGSOFT/FSE*. 137–146.

M. Zhou and A. Mockus. 2011. Does the initial environment impact the future of developers? In *ICSE'11*. awaii 271–280.

M. Zhou and A. Mockus. 2012. What make long term contributors: Willingness and opportunity in OSS community. In *ICSE'12*. 518–528.

Received June 2014; revised November 2015; accepted December 2015