Transition of governance in a mature open software source community: Evidence from the Debian case

Bert M. Sadowski, Gaby Sadowski-Rasters, Geert Duysters

Abstract

As open source software (OSS) communities mature, they have to introduce a variety of governance mechanisms to manage the participation of their members and to coordinate the launch of new releases. The Debian community introduced new mechanisms of informal administrative control based on a constitution, elected leaders, and used interactive communication channels. We show that these control mechanisms were introduced as a response to emerging innovative opportunities due to the usage of source packages and to the need to build a responsive organization within the Debian OSS community.

1. Introduction

There is a continuing fascination with open source software (OSS) communities, not only due to an explosion in the number of volunteers participating in these communities but also due to the unique forms of producing and distributing OSS. The growth of OSS communities, however, in combination with increased demand for OSS software, has created mounting pressure on these communities to adapt their organizational forms to these changes. As a result, OSS communities have adopted a variety of new ways of producing and distributing OSS (Lee and Cole, 2003; Lerner and Tirole, 2002; van Krogh et al., 2003).

In OSS communities, the creation of new knowledge requires, on the one hand, a set of organizational rules and structures that allow critical evaluation of existing knowledge, innovation, and rapid elimination of error (Kogut, 2000). On the other hand, there is a growing need to reduce the time available for the introduction of new releases, while at the same time maintaining high quality (Michlmayr, 2004, 2007). Innovation and the need to develop new software features should therefore not be at the expense of reducing the quality of existing (and new) software packages. In fact, OSS communities need to create a situation in which they simultaneously balance exploration strategies, aimed at innovating and business development, with exploitation strategies that are primarily directed at making the most of existing competences. Tushman and O'Reilly (1996) compared this balancing act to the god Janus who had two sets of eyes. One pair was for looking at what lies behind and one pair for focusing on what lies ahead. Due to this balancing act, the organizational forms to coordinate and govern collaborative work have to be flexible and should be able to adapt easily to heterogeneous learning conditions (March, 1991).

The Debian OSS community fits this general picture, with the number of developers increasing from a sheer total of 60 in 1996 to over 9000 in 2005, and with the number of source packages rising from 250 in 1995 to 10,869 by February, 2006. During this ten-year period, the growth of Debian was accompanied by a search for different governance forms after the ori-
Debian.

oped within the Debian OSS community and explains the particular form of organization of production and distribution of

1999).

emerged, ranging from quasi-hierarchical (Linux) to (somewhat) centralized (Apache) approaches (Demil and Lecocq,

effective production than do other modes of governance. As a result, a number of mixed forms of bazaar governance have

for the increased need for administrative (informal as well as formal) control mechanisms. It provides fewer incentives for

structural complexity of OSS communities, however, the bazaar mode of governance is not sufficiently efficient to account

for the increased need for administrative (informal as well as formal) control mechanisms. It provides fewer incentives for

exploration and exploitation (March, 1991).

An important challenge for OSS communities has been to deal with two faces of learning within their communities:

exploration and exploitation (March, 1991). The distinction between exploration and exploitation originated in the work of

Holland (1975) and was further developed by March (1991). Exploitation is generally associated with the refinement

and extension of existing technologies and leads to further improvements in the existing knowledge base. This in turn

strengthens a firm's existing business and current revenue streams, thereby enhancing its competitiveness in the short term

(Vanherverbeke et al., 2007). Exploitation is often characterized as a process of routinisation, which facilitates the extension of

both the existing knowledge base and the competences of firms, without changing the specific nature of activities (March,

1991). Exploration, on the other hand, involves experimentation with new alternatives and the exploration of a new (tech-

nological) field (March, 1991). Exploration is generally associated with technology and knowledge that reflects a firm's fu-

ture business and revenue streams. Generating such know-how requires, in particular, an emphasis on the acquisition of

novel insights. More recent literature (Benner and Tushman, 2003) suggests that exploration and exploitation are not mutu-

ally exclusive, but can be operated at the same time.

It is often considered to be a particularly difficult task for organizations to find the delicate balance between exploration

and exploitation. To deal with these two faces of learning and with an increased need for coordination within OSS commu-

nities, Demil and Lecocq (1999) have shown that the bazaar structure, i.e., a "great babbling bazaar of different agendas and

approaches" (Raymond, 2001), can serve as a new emerging mode of governance within an OSS community (Demil and

Lecocq, 1999; Raymond, 2001). Even under conditions of very high uncertainty, the bazaar mode of governance assures coor-

dination based on reputation effects that are induced by the community phenomenon. In the face of increasing technical and

structural complexity of OSS communities, however, the bazaar mode of governance is not sufficiently efficient to account

for the increased need for administrative (informal as well as formal) control mechanisms. It provides fewer incentives for

effective production than do other modes of governance. As a result, a number of mixed forms of bazaar governance have

emerged, ranging from quasi-hierarchical (Linux) to (somewhat) centralized (Apache) approaches (Demil and Lecocq,

1999). As we will show below, a unique mixed approach of bazaar governance, based on informal authority, has been de-

veloped within the Debian OSS community and explains the particular form of organization of production and distribution of

Debian.

2 For an overview of these different modes of governance of OSS communities see Markus, Manville and Agres (2000).
3. Governance mechanisms in transition: The Debian OSS community

3.1. Characterizing the Debian OSS community

As noted above, the Debian OSS community has experienced a rapid growth since its establishment in 1993 by Ian Murdock, currently involving more than 9000 volunteer package maintainers. In line with other OSS communities, Debian software is developed free of charge. However, the software development work of the community is concentrated not on producing code, but on integrating code into a coherent system. In this respect, Debian is more in line with Red Hat, SUSE and Mandriva than with Linux, Apache and Mozilla (Bauer and Pizka, 2003; González-Barahona et al., 2004; Narduzzo and Rossi, 2003). Within Debian OSS, two separate code structures (trees) that are running in parallel can be identified: a stable branch and an unstable branch (that includes an experimental branch). The stable branch consists of the basic software package. The unstable branch, in particular the experimental branch, serves as a test bed for new features of (public) releases of Debian. The coordination of software development across these two branches has been a continuous challenge for the Debian OSS community.

Software development is based on characteristics like tacit knowledge and informal communication which fosters exploration. At the same time, software development requires explicit knowledge and traditional formal communication to allow for exploitation. The organizational challenge has been to create flexibility needed for new development, but at the same time to guarantee the rigor necessary to deliver high quality software packages (Lee et al., 2006). To respond to this challenge, coordination mechanisms are required that allow for successful innovation activities and simultaneously enable an improvement of existing business processes (Tushman and O'Reilly (1996)).

Within the Debian community, the task structure has been focused around a “core” which consists of the Debian project leader (DPL), and developers as well as a “periphery” of maintainers. As the “core” has been responsible for the production of new code, the maintainers on the “periphery” have dealt with the integration of these codes for particular applications. These two different activities are mainly related to exploration at the “core” and exploitation at the “periphery”. This structure differs from other OSS communities like Linux (Lee and Cole, 2003) as it is sometimes difficult to draw a line between the developers and the maintainers. In Debian’s case distinct units each have their own role in exploitation and exploration but can be integrated at a different (informal) level. Examining the specifics of the code structure used by the Debian OSS and the evolving task structure is essential to understanding the development of the different informal governance forms within the Debian OSS community.3

In terms of methodology we must emphasize that the ‘community’ phenomenon was central to our analysis of the history of the Debian OSS community. We extensively used the feedback from the community to improve the results of this study (see Rasters, 2004, pp. 42–44). We closely followed the development of other OSS communities (such as Apache, Linux or Pearl) and other OSS communities developing packaged software distributions (Red Hat, SuSe). Our aim was not only to better understand the specifics of open software programming and distribution (e.g., Kraut and Streeter, 1995) but also to characterize general as well as specific factors driving the growth of OSS communities. For this purpose, we extensively examined websites of these OSS communities and subscribed to different mailing lists such as Floss discussion threads or diverse Linux mailing lists (for more information on the methodology see Appendix 1).

3.2. Organizational growth and the emergence of informal forms of governance

3.2.1. The project initiation stage: Ian Murdock’s vision

In this stage, OSS projects commence because one or more people realize that there is a computing-related problem or challenge left unfilled, and for one or more reasons, they decide to take it on (Godfrey and Tu, 2000). Here the “itching problem” described by E. Raymond comes into play: “every good work of software starts by scratching a developer’s personal itch.” (Raymond, 1998). At this stage, many important OSS projects have mainly been driven by the efforts of a single developer.4 Leadership based on ‘real’ informal authority has been critical at this stage for attracting programmers who are willing to follow the developer’s ‘recommendations,’ which were rooted in a particular ‘vision’ (agenda for work, milestones, etc.) of a new emerging OSS project (Lerner and Tirole, 2002).

The Debian project was started by Ian Murdock from scratch after he became dissatisfied with the SLS (Softlanding Linux System) release. In 1993, when Murdock decided to start an Open Source distribution that would always be free, he founded a group of like-minded people to work with him. The stated goal was to create a complete operating system that would be ‘commercial grade’ but not, in itself, commercial. Ian Murdock posted his intentions to the Usenet in August of 1993 and immediately found outside interest, including that of the Free Software Foundation, the creators of much of the core software of all Linux-based systems. Murdock posted his announcement in an attempt to reach out for a small group of motivated individuals who had ideas for the project. The question of freedom was important to Murdock. Debian started as a small, tightly-knit group of free software programmers, and gradually grew to become a large, well-organized community of devel-

---

3 It furthermore is important to know that the Debian OSS community has not been influenced by strategies of sponsoring companies. Other OSS communities are (still) operating in other market segments (like Ubuntu in the desktop market and in the individual user segment) or specific markets (like Mandriva) and do not have (yet) an extensive support organization as Debian (or Red Hat or Novell) already provide.

4 We thank one of the reviewers for this important point on OSS evolution.
opers and users (Varghese, 2003). The foundation for the parallel code structure was already laid down during this period, leading to (public) releases of Debian and the (rudimentary) package system called dpkg.

3.2.2. The “Going Open” stage: Debian’s “New Maintainer” process

In order to enter the going open stage, OSS communities face certain challenges such as achieving project and product credibility, developing adequate communication mechanisms, and creating effective recruitment strategies, as well as the development of appropriate forms of governance. To achieve credibility the project needs to obtain support from a number of enthusiastic “core developers,” to show some “plausible promise,” to attract interest from programmers due to its innovativeness, to have some importance while allowing a (future) large number of developers to participate, and to demonstrate that the right amount of the problem has already been solved before the project becomes “open” (Schweik and Semenov, 2003).

In order to develop appropriate communication channels, different internet-based forms of communication are exploited, ranging from “free form” discussions (e.g., mailing lists, IRC channels) to strongly structured discussions (e.g., bug tracking systems or trouble ticketing at helpdesks), to knowledge based discussions (e.g., wiki platform). To create effective recruitment strategies, the initiator has to choose a platform for announcing the project that has the potential of reaching as many readers as possible.

3.3. Institutionalizing the vision: the Debian social contract

When “going open” the Debian OSS community was facing similar challenges when Ian Murdock felt that Debian software was ready to be shared. On September 2nd 1993 Murdock officially announced the Debian project.

Murdock decided to follow the Open Source Developers licensing principles; he made the decision to follow the GNU and to receive a general public license (GPL). In contrast to other OSS communities providing their software (also) commercially, Debian (built on GNU/Linux) has been a strong supporter of free software. Since many different licenses were used for the Debian software, Murdock thought it was necessary to develop a set of guidelines, the Debian free software guidelines (DFSG), to provide a reasonable definition of what constitutes free software. Only software that complied with the DFSG was allowed in the main distribution of Debian. As the DFSG was a first step in defining “free” software, the second step was to “institutionalize” the moral norms of the growing Debian OSS community. This was done by creating the Debian social contract (DSC), which included the DFSG. Initially designed as a set of commitments that Debian developers agreed to obey, DSC was adopted by the free software community as the basis of the open source definition.

3.4. Establishing the initial Debian production and distribution structure

The public release of Debian 0.91 in January 1994 gave a first glimpse of the Debian philosophy. By this time, about a dozen people were involved in software development, though Ian Murdock was still largely packaging and integrating the releases himself. After this first public release of Debian, attention was turned toward developing the package system called dpkg. A rudimentary dpkg existed in Debian 0.91, but at that time this was mostly used for manipulating packages once they were installed, rather than as a general packaging utility. By summer 1994, early versions of dpkg were becoming usable and other people besides Murdock began joining the packaging and integration process by following guidelines that explained how to construct packages that were modular and integrated into the system without causing problems. By fall 1994, Murdock was overloaded, now coordinating the efforts of dozens of people in addition to his own development work. He transferred responsibility of the package system to Ian Jackson, who proceeded to make many valuable enhancements, and shaped it into the current system.

After months of hard work and organization, the Debian Project finally made its first distributed release in March 1995: Debian 0.93 Release 5. Debian 0.92 had never been released, and Release 1 through Release 4 of Debian 0.93 had been development releases made throughout the fall and winter of 1994. These development releases were intended to experiment and to further improve on public releases, as they were used as a learning device. This “dual” system of working with “development” as well as “public” releases was later used for devising a package life cycle in which new releases were characterized as ‘unstable’ or ‘stable’ (Krafft, 2005). Table 1 provides an overview of Debian releases and major events during this second phase.

As can be seen in Table 1, the steady growth in the number of packages since 1995 has been accompanied by the increase in the number of developers in the Debian community. By 1995, the Debian Project, as it had by then become known, had grown to include over sixty people. In the summer of 1995, Ian Murdock transferred responsibility of the base system, the core set of Debian packages, to Bruce Perens, giving Murdock time to devote to the management of the growing Debian Project. Murdock left the Debian Project in March 1996 and Bruce Perens assumed the leadership role, guiding the Project through its first release (called “Buzz” or Debian 1.1) in June 1996. In 1997, during Perens’ leadership period, the Debian developers ratified the Debian Social Contract, which included the Debian free software guidelines (DFSG) and provided the open source definition for the Debian community. As the DFSG provided guidelines on what constitutes free software in the Debian context, new members had to agree with the Debian Social Contract and the DFSG in order to join the Debian OSS community.
3.5. Informal authority within the Debian OSS community

Ian Jackson, the successor of Bruce Perens, was the first elected Debian project leader (DPL). He had major influence on formalizing activities within the growing community that lead to the Debian Constitution, which was approved in 1998 by a voting procedure. As shown in Fig. 1, the Debian Constitution was a first attempt to define different roles (e.g., the DPL, the

![Diagram of Debian Constitution](image)

Table 1
New releases and important events in the Debian History (1993 – March 2008)

<table>
<thead>
<tr>
<th>Timeline</th>
<th>Release</th>
<th>Package System dpkg</th>
<th>Packages</th>
<th>Developers</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall-Winter 1993</td>
<td>Several internal releases</td>
<td>Rudimentary dpkg</td>
<td>Small</td>
<td></td>
<td>Founder Ian Murdock</td>
</tr>
<tr>
<td>January 1994</td>
<td>Public release of Debian 0.91.</td>
<td>Usable early versions of dpkg</td>
<td></td>
<td></td>
<td>Ian Murdock still largely packages and integrates the releases himself Rudimentary packing system are used for manipulating packages</td>
</tr>
<tr>
<td>Summer 1994</td>
<td>Responsibility over dpkg (I. Jackson)</td>
<td></td>
<td></td>
<td></td>
<td>With early versions of dpkg and guidelines explaining how to construct packages other people besides Ian Murdock join packaging and integration</td>
</tr>
<tr>
<td>Fall 1994</td>
<td>First distributed release (Debian 0.93 Release 5)</td>
<td>250</td>
<td>60</td>
<td>It now is called The Debian Project</td>
<td></td>
</tr>
<tr>
<td>1995</td>
<td>Responsibility over base system (Perens)</td>
<td></td>
<td></td>
<td></td>
<td>Ian Murdock transfers responsibility of base system (core set of Debian packages) to Bruce Perens, he still is responsible for Debian management</td>
</tr>
<tr>
<td>June 1996</td>
<td>1.1 (Buzz)</td>
<td>474</td>
<td>90</td>
<td></td>
<td>Ian Murdock leaves the Debian Project in March 1996; Bruce Perens assumes leadership role</td>
</tr>
<tr>
<td>End 1996</td>
<td>1.2 (Rex)</td>
<td>848</td>
<td>120</td>
<td></td>
<td>Debian Social Contract including Debian Free Software Guidelines (DFSG) and Open Source Definition</td>
</tr>
<tr>
<td>1997</td>
<td>1.3 (Bo)</td>
<td>974</td>
<td>200</td>
<td></td>
<td>Debian Constitution ratified by vote (constitution includes election methods, leadership debate), first elected leader Ian Jackson</td>
</tr>
<tr>
<td>1998</td>
<td>2.0 (Hamm)</td>
<td>1500</td>
<td>400</td>
<td></td>
<td>Freeze on accepting new maintainers. Constitution of the New Maintainer process</td>
</tr>
<tr>
<td>1999</td>
<td>2.1 (Slink)</td>
<td>2250</td>
<td>410</td>
<td></td>
<td>Leadership elections within a new format, Discussion about a Debian Project Leader (DPL) team</td>
</tr>
<tr>
<td>2000</td>
<td>2.2 (Potato)</td>
<td>3900</td>
<td>450</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2002</td>
<td>3.0 (Woody)</td>
<td>9000</td>
<td>1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td>3.1 (Sarge)</td>
<td>10869</td>
<td>&gt;9000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>February 2008</td>
<td>4.0r3 (Etch)</td>
<td>15626</td>
<td></td>
<td></td>
<td>Source: (Lameter, 2002) and own information.</td>
</tr>
</tbody>
</table>

Fig. 1. The Debian constitution, source: (Ronneburg, 2006), reproduced with permission from creative commons.
Technical Committee, and Developers) in a form of hierarchy within the Debian community (Garzarelli and Galoppini, 2003). The role of the coordinator was assumed by the DPL. Jackson helped to define the project’s vision, lent authority to developers, and made any decision that required urgent action. The leader also represented Debian the project to the outside world (e.g., by attending conferences and giving talks). All Debian developers could vote to elect the project leader. Still, the developers, who are at the bottom of this hierarchy, could override any decision taken by the project leader or the technical committee. Furthermore, the constitution did not impose any obligation on anyone to work continuously on the Debian project; in fact, a contributor could leave the project at any time or resign from his or her position or duty by a simple announcement.

3.6. Guaranteeing quality of new releases: the new maintainer process

During the period 1996–1999 three more stable releases were provided by Debian developers and maintainers. Within the Debian community, a task structure had developed in which certain developers (including the DPL) contributed to new releases even if they were sometimes not directly linked to a particular package. In addition, so-called maintainers were taking on existing open software packages and were creating a ready-to-install Debian package (Robles, 2005). This division of tasks within Debian was decisive in dealing with challenges of an ambidextrous production and distribution structure at Debian that allowed for exploration at the “core” as well as exploitation at the “periphery.” As the Debian OSS project could rely from the beginning on a modular architecture based on GNU/Linux OS, an increasing number of programmers were able to participate in the project and to develop new releases, even if important design decisions, such as on the package management systems, had to be solved at the “core.”

In 1999 Debian entered the phase in which the community became really concerned about the quality of maintainers who were joining the project. There was even a hold on accepting new maintainers. A crisis occurred when the Debian community no longer felt that it could adequately protect its boundaries and closed its doors to potential new members. As the acting DPL at that time, Wichert Akkerman observed: “I have to acknowledge that Debian has reached the point where it has grown too much and cannot continue as before. At the moment we already have chaos all over with no proper leadership. Only very few people are taking care of general management tasks. Remember this is an association of more than 500 people. There is still no proper management. Guess what would have happened if it were a company (…)”.5

Under his leadership the new maintainer process was constituted and membership criteria were decided upon. The Debian new maintainer process was designed as a series of required proceedings to become a Debian developer or maintainer. It consisted of a registration process of new maintainers (NM) that was handled by the NM-Committee, which is a body of people who control the new maintainer process. The NM-Committee was composed of the front desk, the application managers, and the developer accounts managers. The front desk officers received new application requests and passed them to appropriate application managers. The application manager was a Debian developer who was assigned to an Applicant in order to monitor their progress through the application process. Developer accounts managers (DAMs) managed user accounts on Debian machines, and finalized the details of membership by assigning accounts to new developers. The DAMs were delegates appointed by the DPL (see Fig. 1).

The decision to institutionalize the New Maintainer Process has been a way of keeping Debian open, but at the same time, a way to manage its boundaries. The New Maintainer process defined a mechanism for managing membership that allowed evaluation of whether or not new members’ skills, goals, and ideology were in line with that of the community (O’Mahony and Ferrar, 2004). From 1999 onwards there were three other Debian releases, but there was a gap of three years between the 3.0 release in 2002 and the last Sarge release in 2005 (see Table 1).

4. The growth stage

As Schweik and Semenov (2003) observe, open source projects can grow at this stage based on new membership. They may also remain stable, relying on the same number of participants as in the going open stage, or they might gradually decline due to a lack of participant interest (Schweik and Semenov, 2003). As has been shown in Table 1, the Debian project grew rapidly after its initiation from only a few developers into a large community.

4.1. Communication channels within the Debian OSS community

During this growth phase, the community found ways to cope with its expansion, mainly by streamlining and coordinating communication. By providing for reciprocity and reputation, communication processes were streamlined and coordinated by using, in particular, the various Debian mailing systems. Within the Debian project, mailing lists fulfilled three different functions (Lanzara and Morner, 2003): First, they were used as virtual construction sites to continuously create, update, modify and repair software constructs; second, they were used as a sort of electronic crossroads to exchange information and problems as well as discuss solutions; and third, they were used as a form of weblog that recorded the history of the Debian OSS community. The mailing lists allowed unrestricted access to discussions, they allowed knowledge circulation

---

5 Based on a face-to-face interview with Akkermann in June 2001, held in Delft, the Netherlands.
and they have been a means to structure the communication within the Debian community. At the same time the mailing lists allowed dissemination activities of the Debian project to take place quasi-automatically, because documentation of built software products or solutions can circulate throughout the web almost instantaneously. The dissemination process has been linked to the development activity, and has been embedded in the Internet-based information and communication structure. As a result of these new functions, mailing lists were considered as a new mechanism of governance within the Debian OSS community (Lanzara and Morner, 2003, p. 37).

4.2. The call for more team-based leadership approaches

A continuous challenge of managing the Debian OSS community has been the slow release cycle of Debian. The Debian community has always been proud of the fact that it will not release buggy software, and that it will release only when the software has been stable. Within the Debian OSS community, the Debian project leaders developed their own leadership style to deal with the problems of slow release management and with the growth of the community as a whole. As Table 2 shows, the Debian project has been headed since 1993 by a number of leaders experimenting with very different leadership styles. At the beginning, when there were only a few people involved in the Debian project, strong leadership based on technical contributions was accepted (Fleming and Waguespack, 2005). These contributions included processes of modularizing, i.e., critical programming activities that were aimed at decomposing complex modular architecture of Debian OSS (Lanzara and Morner, 2003). Compared to other OSS projects like Apache or Linux, Debian has been unusual, as the community experienced frequent leadership changes to deal with its increasing structural complexity (O’Mahony and Ferrar, 2004). This was the point in 1996 when leadership elections were arranged by the project leader secretary. The ways in which elections were organized also changed over time, starting from simple plain text mission statements, then moving towards personal election platforms and finally ending up with election debates on IRC channels.

Ian Jackson led the Debian project from January 1998 until December 1998. This was the point in time when the election of project leaders began. The enormous growth of the community prohibited informal ways of transferring leadership. Jackson tried, together with the community, to “fit the governance structure” to the size of the community and to the feelings of freedom that lived in the community. He had a major influence on how the Debian project become structured with respect to writing the constitution, election methods and the description of leadership models. During this period, leadership within the Debian OSS community was increasingly based on organization building and to a lesser extent on technical contributions to the OSS (O’Mahony and Ferrar, 2004).

In 2000, the leadership debate and a speech of the opponents were introduced in the election. After the debate, a log of the debate was posted, so voters could read everything at their own pace. In the leadership elections of the year 2005 a major difference with previous leadership elections emerged. The year 2005 was a very interesting one in the evolution of the Debian community. The Debian GNU/Linux version 3.1 codenamed “Sarge” was released after nearly three years of continuous development. Within the Debian community, criticism increasingly mounted about the slow release management cycle of the project. Within the leadership elections, the slow release management and the growth of the user community were considered as “hot” items among candidates running for election even if this issue had already been intensely discussed in previous elections. Interestingly, the candidates running for election this time presented new solutions to these critical issues. They suggested a whole new approach towards leading the Debian project. The election platforms of two running candidates, Brandon Robinson and Andreas Schulde, suggested forming a small formal team of Debian developers aimed at supporting the project leader. This team, nicknamed “Project Scud,”6 was organized in the last few weeks of 2004. It was “a team of concerned Debian Developers who have resolved to take some new approaches to resolve long-standing problems within the project.” According to Scud members, having a DPL team would allow them to distribute the workload, and to avoid burnouts and problems related to real-world unavailability of individual developers.

The Scud team identified small teams (up to seven people) as probably the single most important unit for the Debian project to grow in a healthy way. If the team would function well it could solve more problems than could individual developers. The team should be able to provide a smooth entry point for new developers to gain proficiency and develop skills. Furthermore, teams should be the place where developers can get to know each other quickly and well (due to the small number of people in the group). Another advantage proposed by the Scud team was that people could form a knowledge pool when cooperating on package maintenance, infrastructural or organizational tasks, and it was less likely that such a pool would get lost compared to the knowledge and skills lost if a single developer were departing. This change would make Debian more resilient against unmaintained packages or head hunters. As these teams could grow and divide, they were considered as self-organizing and would provide for very good scalability in numerical growth.

While the members of the Scud team have been enthusiastic about their new ideas, there has been some controversy within the Debian community about Project Scud, which has also been referred to as a self-appointed group of advisors to the DPL. The Scud proposal has been a source of some concern, especially about how it would integrate within the Debian constitution and the existing organizational structure. There have been great concerns from Debian developers about at-
tempts to formalize the Scud team. The Debian community, however, adopted a wait-and-see attitude. To disappointment of Wolffenaar, the DPL team did not work as expected.7

5. Summary and discussion

For OSS communities, the “going open” stage has generally been considered as critical in deciding whether or not these communities will face further growth, maturity or decline. To facilitate the adaptation of an OSS community during this stage, a wide variety of task structures with different degrees of formalized technical as well as administrative structures have emerged. As the evolution of different task structures has been rooted in heterogeneous processes of learning, the formalization of the technical and administrative structures has been driven by the needs within the OSS community to explore and exploit knowledge. Therefore, the evolution of different governance forms has to be considered in the context of these task structures as well as technical and administrative structures. Traditionally, organizational theorists assumed that exploration and exploitation were two mutually exclusive processes. More recently it has been argued that most successful organizations are those that have an ambidextrous nature in the sense that they have been able to jointly exploit existing competencies and to explore new innovations through a dedicated structure. So far there is little evidence of typical ambidextrous organizations in the literature. Debian, however, seems to be a case in point.

In exploring different stages in the development of OSS communities, this paper has linked the evolution of different informal governance forms within Debian to the parallel code structure and the task structure within this community. For the separate code structures running in parallel within Debian (i.e., a stable and a more experimental version of Debian software) integration has proven to be vital. It provided the basis for a “core” around the Debian project leader and developers as well as a “periphery” of maintainers. As the conceptions of authority shifted over time within the Debian

---

7 “First, because the team had no official status and the chosen DPL did not give the team the priority it deserved. Robinson liked the idea, but was not an enthusiastic proponent of the team approach. He lacked the leadership skills to lead the team in an effective manner. There have been Scud meetings, and to a certain extent they were useful, but it was not so that the Scud fulfilled DPL functions. These functions still were carried out by the project leader himself.”
OSS community, this shift was rooted in a shared basis of authority among members and limited by democratic mechanisms like the Debian Social Contract and different communication channels. The Contract allowed experimentation with authority without endangering the growth of the Debian OSS community. In this context, the New Maintainer Process represents an important step to further institutionalize this structure and protect the boundaries of the Debian OSS community.

The emergence of an elected leader in conjunction with a project leadership team and the establishment of an ambidextrous organizational structure provides new evidence for the need to search for novel and alternative forms of governance of OSS communities. In the face of growing structural and technical complexity, they provide a solution to the dilemma of OSS communities during the “going open” stage of their development.

Acknowledgements

We have to thank Ray Dassen and Jeroen van Wolffenaar of Debian for their continuous support and their valuable inputs in this paper. All remaining errors, of course, are ours.

Appendix. 1: Methodology

In order to characterize governance mechanisms during transition of OSS communities, we examined the history of the Debian OSS community based on data triangulation. That allowed us to characterize the different perspectives of stakeholders within the Debian community like Debian project leaders, maintainers or developers. It also enabled us to get an understanding of specifics of the Debian community compared to other OSS communities.

To examine the development of the Debian OSS community in more detail, a wide variety of data sources were consulted (see Table 3): We primarily used internal documents related to the contents and context of different Debian projects. We complemented the analysis with semi-structured interviews (both face-to-face and by telephone) with key individuals (DPL leaders, maintainers, developers) during the period 2002–2005.9

Similar to Dafermos (2001), we used semi-structured interviews as they provide more detailed information of greater value than straightforward question and answer sessions, especially when the research is explorative (Dafermos, 2001). These semi-structured interviews were also useful in engaging in a continuous conversation with the interviewees. The face-to-face interviews were taped and transcribed verbatim. As a check, the interviews were sent to the interviewees for comments. The interviews that were undertaken by telephone were written down as accurately as possible. Again, the transcripts were sent to interviewees in order to check their accurateness.

Furthermore, we attended several Debian conferences and were “lurking around” on the Debian mailing lists, websites, IRC channels, etc. Interviews were used to gain further insights into the Debian community. In addition, articles on Slashdot.org, members’ biographical writings and diaries, previous interviews with key members and descriptions of the community written by other researchers and key people were extensively utilized. As a result we were able to follow the Debian project in great detail with respect to its history as well as its ongoing development and activities.

References


---

9 The identities of the respondents are available upon request from the authors.