AUTHORITY IN PEER PRODUCTION

The Emergence of Governance in the FreeBSD Project

INTRODUCTION

Research interest in software produced and distributed freely over the Internet by voluntary associations of hackers known as free and open source software (FOSS) projects has been constantly increasing during the last ten years. Breaking with many established assumptions about how software ought to be developed, FOSS projects have captivated the attention of organisation theorists: because of the volunteer character of participation in FOSS projects, their administrators 'must resort to other governance mechanisms than those available to firms that pay developers to work' (von Krogh & von Hippel 2006, p. 979) and so in this environment authority 'is persuasive, not legal or technical, and certainly not determinative' (Benkler 2006, p. 105). Of course, to say that authority is not coercive does not mean that authority does not exist: for 'no social system can operate on a continuous basis without support from some mode of authority' (Harrison 1960, p. 233). In view of the fact that FOSS projects have no coercive apparatus through which to mobilise development resources and impose compliance with their rules and norms, how is authority articulated on their development?

LITERATURE REVIEW

Early studies of hacker culture laid emphasis on hackers' disdain for work inside bureaucratic organisations, highlighting their conscious opposition to centralised authority and pointing out that hackers' activities seemed to encapsulate a definite morality – known as the *hacker ethic* – which exalts the joy and autonomy inherent in intrinsically-motivated activities as well as the free sharing of knowledge that lays at the heart of the hacker community (Levy 1984). Interestingly, in spite of the passage of time, the character of hacker morality has not changed, as more recent studies (e.g. Hinanen 2001; Weber 2004) have shown.

The academic discourse that has emerged more recently centred on the organisational characteristics of software projects run by hackers known as Free/ Open Source Software (FOSS) projects. Benkler (2006) coined the term commons-based peer production to underline, on the one hand, the centrality of a regime of common property in enabling FOSS projects and, on the other, the informal and collegial character of participation in them. According to Weber (2004, pp. 159-164), their governance consists in a combination of leadership practices and cultural norms which define who has the legitimate right to redistribute modified versions of the software and make decisions about which contributions are included in public distributions. This type of governance – which Demil and Lecocq (2006) call 'bazaar governance' – represents an institutional framework for work organisation that differs fundamentally from how task coordination is effected in hierarchies, markets and networks, largely on account of the low level of social control in this environment (Demil & Lecocq 2006, p. 1453). Markus (2007) pointed out that the aim of governance in FOSS projects is (a) to incentivise participation and (b) to facilitate task coordination in the development process. de Laat (2007) called attention to the evolutionary character of FOSS governance. While work coordination in small projects is achieved informally through the 'mutual adjustment' of participants, larger projects, due to the coordination costs attendant upon the expansion of scale, require more elaborate means of coordination. Thus, in

contrast to the spontaneous organisation characteristic of small projects, multitudinous groups cannot do without some measure of systematisation of rules and work procedures (de Laat 2007).

In the space of the last ten years, a significant number of research works have explored the governance of large FOSS projects. Shah's (2006) study showed that, as 'heavy-handed control deters participation' (p. 1008), the governance structure affects decisively the number of developers that are attracted to the project. The developers of the FOSS project, which she investigated, emphasised that 'they choose their own tasks and set their own schedules', underlining the role of freedom and creativity in spurring them to participate (p. 1007). According to Garcia and Steinmueller's (2008) case study of Debian, the source of authority in FOSS projects is 'knowledge of purpose and technique acquired and demonstrated through participation in the project' (Garcia & Steinmueller 2008, p. x). Prolific developers are rewarded with reputation for their sustained contribution, thus permitting them to 'exercise authority over the project, and if not its participants, then at least their contributions' (Garcia & Steinmueller 2008, p. 336; see also de Laat 2007, p. 167). O'Mahony and Ferraro's (2007) study of Debian looked at the transformation of its governance system catalysed by conflicts between the project leader and the community of maintainers over what was perceived as a lack of legitimacy of his authority. The resulting 'reform' combined 'elements of democratic and bureaucratic control' (p. 1099) to prevent autocratic rule and nourish a conception of leadership based on consensus-building. In addition to establishing checks upon the project leader's authority, Debian devised a new recruitment process with the aim of ensuring that new recruits possess not only the right skills but also views that are consistent with the socio-political goals of the project (also see Mateos-Garcia & Steinmueller 2008, p. 239). On the same wavelength, O'Neil (2009, chapters 7 and 9) introduced the term 'tribal bureaucracy' to denote, on the one hand, Debian's 'rejection of market economy...in favour of cooperative production' and, on the other, its resolve to demarcate the authority of the project leader by means of a limited form of bureaucracy.

RESEARCH METHODOLOGY

The case studies of Debian by Garcia and Steinmueller (2008), O'Mahony and Ferraro (2007) as well as that of Shah (2006) clearly demonstrate the penetrating insights that a longitudinal study permits by covering a time-span in which the project has grown considerably so that the the relationship between project growth and mode of governance can be examined in a rigorous manner. Dealing with longitudinal data, of course, implies a case study research design, a research strategy commonly used to understand dynamics within single settings (Eisenhardt 1989, p. 534). Thus, our study adopts a case study research design centred on the FreeBSD project, using archives of development activity logs and project communications, questionnaires and observations. We chose FreeBSD because (a) its scale has increased dramatically over time and (b) it is a pure production community: FreeBSD is a collective that not only distributes some product on the Internet (as Debian does) but is also responsible for the overall production of that product.

In this environment, as noted, authority cannot be coercive, for persons in authority are deprived of the means of coercion by which to impose themselves upon the other project members. To be able to exercise any influence over the management of the project, their authority must be perceived as being *legitimate*. The question how authority is legitimated in FOSS projects is therefore crucial. For the purpose of analysing the type of authority that emerges in collaborative enterprises manned by volunteers, we draw upon Max Weber's (1947) classical analytical framework: although about a hundred years have elapsed since its original publication, the emphasis it lays upon voluntary obedience makes it ideally suited to the task at hand. Such an approach also facilitates the comparison of our findings with those of other researchers, as it figures prominently in the literature: studies looking at how authority is articulated in FOSS projects often use the same analytical frame – the same concepts and categories – that Weber pioneered for the study of authority in groups (e.g. Himanen [2001], O'Neil [2009] and O'Mahony

& Ferraro [2007]).

EMPIRICAL SETTING

FreeBSD is a free/open source[1] operating system descended from the Berkeley Software Distribution (BSD), the version of Unix developed at the University of California at Berkeley. The first version of FreeBSD was released in December 1993. Since, FreeBSD has been established as the most popular BSD-descendant with a proven track record in mission-critical deployments,[2] thriving on the contributions of a community of software developers spread the world over. Though development effort is heavily concentrated in North America and Europe (Spinellis 2006) (see Fig. 1 below), FreeBSD development takes place in 34 countries on six continents (Watson 2006) (see Fig. 2 below).

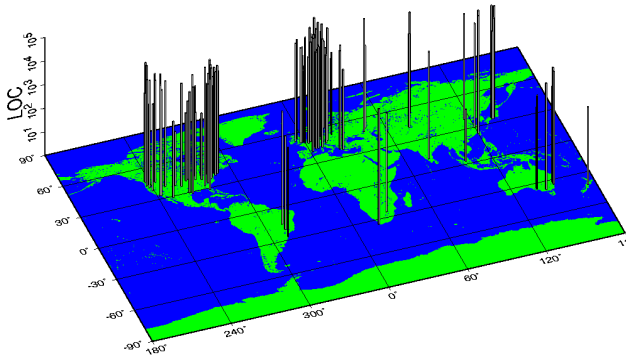

Fig. 1: International development (Source: Spinellis 2006)

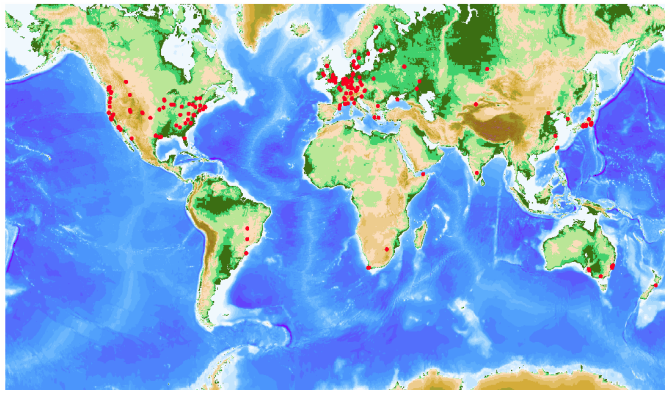

Fig. 2: International development (Source: Spinellis 2006)

The organisational structure of FreeBSD is to large extent inherited from BSD, often credited for codifying a template for what is now known as the open source development model (Leonard 2000). This structure has a *core team* at its centre: a small group of programmers who control access to the codebase, vested with authority to grant or revoke the right to integrate changes into the project's code repository. Spreading out from them are the *committers*, who have the right to check in changes, framed by the wider community of *outside contributors*.

1 FreeBSD is distributed under the terms of the FreeBSD license.
2 See <http://www.bsdstats.org> and BSD Certification Group (2005).

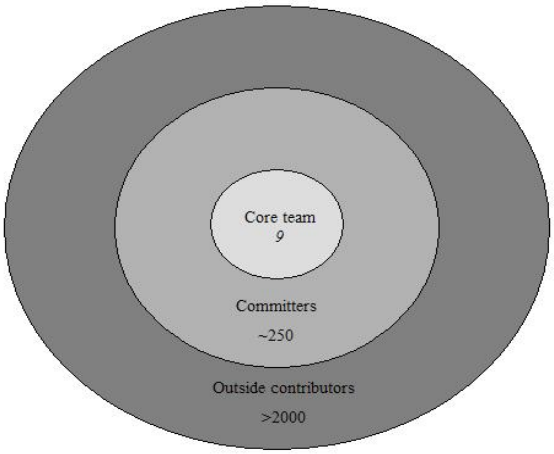

Fig. 3: FreeBSD organisational structure

Outside contributors advance to the ranks of committers when their nomination by an existing committer is approved by the core team, which alone has authority to grant commit privileges. This procedure, as committer Michael Lucas explains, is 'fairly straightforward':

> if you submit enough useful and correct PRs [problem reports], eventually some committer will get sick of taking care of your work and will ask you if you want to be able to commit them yourself. This process serves multiple purposes; after all, the FreeBSD community is made up of people who do the work. For committers, the work consists of creating useful and correct patches. If you don't consistently and regularly create good patches, there's no point in giving you commit access, now is there?...By the time you've submitted several dozen PRs, you'll either work well with the FreeBSD team or everyone will understand that you and the team just can't get along. Direct-commit access is either an obvious next step, or an obviously bad move (Lucas 2002).

New committers are assigned a *mentor*, typically the same person who recommended them for commit privileges. Mentors are responsible for everything their protégés do in the project, including answering their questions, reviewing their changes and familiarising them with FreeBSD's 'rules and conventions'. The period of mentorship, which could last for several months, ends when the mentor 'releases' formally the new committer, feeling that he has proven he can work harmoniously with others in the project (FreeBSD 2011a; Lucas 2002).

Committers focus on either of the three main areas of development at FreeBSD: *src* (kernel and userland), *ports* or *documentation*. Indicatively, a breakdown of the 275 committers who made commits in 2002 (from 31 December 2001 to 31 December 2002) reveals the following division of labour: 201 src committers, 144 ports committers and 41 documentation committers (Saers 2005; see also Watson 2006).[3] Committers' age varies between 17 and 58 years, with a mean age of 32 and median age of 30; the standard deviation is 7.2 years (Watson 2006).

3 The subsequent analysis focuses on *src* committers alone. This analytical choice was made on the grounds that the other two areas of work on FreeBSD (*ports* and *documentation*) pertain less to new code development and more to peripheral, though necessary, activities.

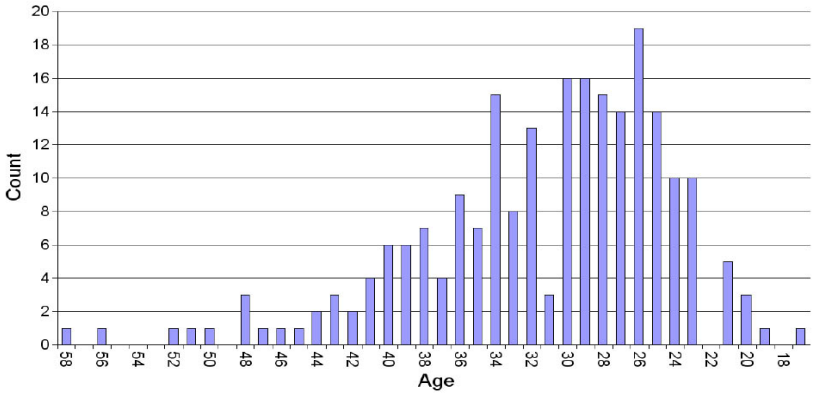

Fig. 4: Age distribution of committers (Source: Watson 2006)

Although FreeBSD is a volunteer organisation and committers receive no remuneration from the project for their contributions, many of them are seasoned professionals working in the IT industry. Therefore, it is not surprising that, for some of them, working on FreeBSD is part of their professional work. In a survey of 72 FreeBSD committers (constituting 35 percent of all committers) conducted in 2000, 21 percent...said that work on their latest contribution had been fully paid for, and another 22 percent partially paid for' (Jørgensen 2005, p. 233). Warner Losh, sitting member of the core team, is one of them. In his opinion, getting paid to work on FreeBSD is not uncommon. As he says: 'my current employer, for example, allows me a certain amount of time each month to work on FreeBSD bugs that impact our ability to deploy a system. These get fed back into the base FreeBSD from time to time. Many other people are in a similar situation' (Losh interviewed in Loli-Queru 2003). For other FreeBSD committers, however, the importance of economic incentives should not be over-emphasised, for, as former core team member Greg Lehey says, 'a lot of people are motivated more than by money to work on FreeBSD. It is their hobby or passion. They find an itch to scratch using FreeBSD and FreeBSD benefits' (Lehey interviewed in Loli-Queru 2003).

Like several other large FOSS projects, FreeBSD has a parallel development structure. There are two simultaneous development processes underway, crystallised in two different branches of the software. The *stable* branch represents the official released version, aimed at a stable and bug-free product. The *current* branch,[4] on the other hand, is experimental: it is where most cutting-edge developments and significant changes (e.g. new features) are first tried out. Fig. 5 below illustrates the development model based on the process by which changes are integrated in the repository.

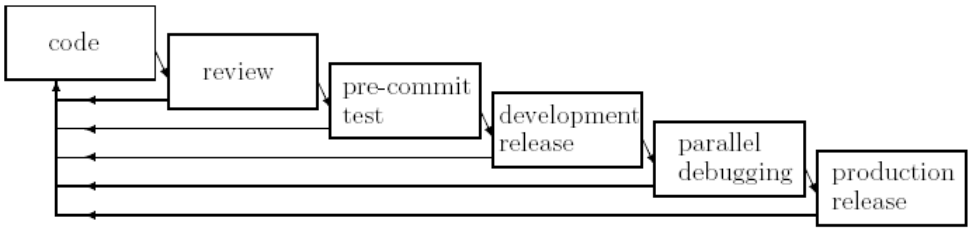

Fig. 5: Change integration process (Source: Jørgensen 2001)

Prior to committing their changes to the repository, committers are expected to ask for community review (FreeBSD 2011d). This practice usually generates a relatively modest amount of feedback,[5] based on which they either have to revisit their code or proceed to testing it on their own systems

4 FreeBSD-Current is also known as *HEAD* or *trunk*.

5 In a survey of 72 FreeBSD committers (constituting 35% of all committers) conducted in 2000, 86% said they received feedback from two or more reviewers (Jørgensen 2001).

(by doing a trial build).[6] Next, they commit the changes to the *current* branch, from which a development release is built and made available for download every few hours. This release is tested and debugged concurrently by many more users and developers who download the software, resulting therefore in significant improvement. Once sufficiently tested and deemed mature enough, the code is merged by the committer in the *stable* branch,[7] from which a production release is made about every four months.[8] The process, despite its incremental character, is recursive: each stage of the process might require of the committer to return to his code for further changes, thereby re-initiating the process. Naturally, as developers work mostly individually,[9] the model is used in parallel by multiple developers (Holck & Jørgensen 2004; Saers 2005).

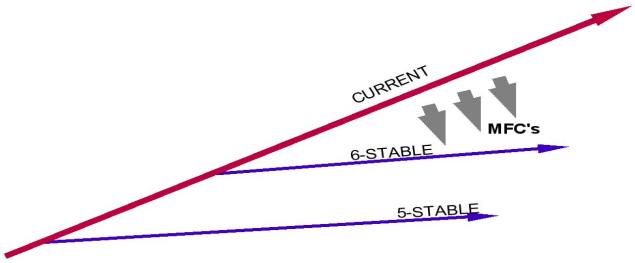

Fig. 6: Branching: stable releases are branched from *Current*; features trickle from *Current* to stable branches as they stabilise (Source: Watson 2006)

Thirty days before the anticipated release date, the repository enters a *code slush*. During this time, only corrective changes (i.e. bug-fixes) can be checked in and they have to be approved by the Release Engineering Team. After the first fifteen days of the code slush, a *release candidate* is released and at the same time the repository enters a *code freeze*, after which point further changes to it become almost impossible. The release candidate is further tested until considered ready by the Release Engineering Team, which then releases it as the official production release (Jørgensen 2001; Stokely 2011; Watson 2006).

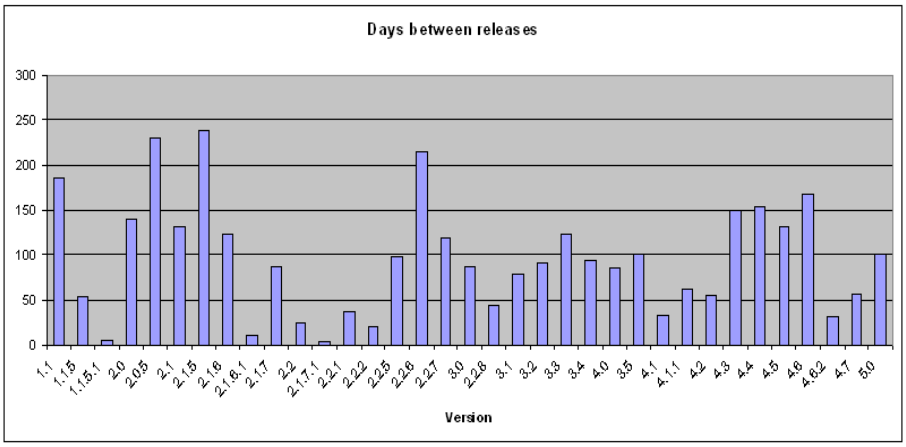

Fig. 7: Days between releases
(Source: FreeBSD Project, http://www.freebsd.org/releases/)

6 Doing a build is an automated process by which (human-readable) source code is compiled to an executable program. If the compilation fails, then the build is said to be broken.

7 The process of merging code from the current branch to the stable branch is known as *Merged From Current* (MFC).

8 The project has been using a schedule with fixed timelines since the start of the 6-CURRENT development branch in 2004.

9 In a survey of 72 FreeBSD committers (constituting 35% of the group of committers) conducted in 2000, '65% said that their last task had been worked on largely by themselves only, with teams consisting of 2 and 3 committers each representing 14%' (Jørgensen 2001).

As can be seen in Fig. 7 above, which shows the time that elapsed between successive FreeBSD releases from the release of version 1 in 1993 until the release of version 5 in 2003, the FreeBSD development process results in a new release being made on average every 96,2 days (with a standard deviation of 62,9 days).

ANALYSIS AND RESULTS: THE EMERGENCE OF GOVERNANCE

Informal governance phase (1993-2000)

FreeBSD evolved for its first seven years (1993-2000) without any formal means of representing its contributors in project governance.[10] During this *informal governance phase*, following the tradition established by BSD, 'those who hacked most became part of the "core group" or "core team"' (Lehey 2002). In 1993 the FreeBSD core team numbered 13 members: the tree founders of the project – Jordan Hubbard, Nate Williams and Rod Grimes – and the most active then-committers. Hubbard served also as the project's president until 1997, which position was 'originally created...to give ISVs and other corporate contacts a more official-sounding person to talk to'. In 1997 he resigned from the position which he also abolished, claiming that it had created 'the illusion of a "super core member"... [and] false expectations of authority' (Hubbard 1997). Growth was continuous throughout this period. Three concurrent phenomena – the growth of peripheral contributors without commit rights to the project, the expansion of the (*src*) committers group from 16 to 138 persons and the growth of the codebase – attest to the dramatic expansion of scale underway.

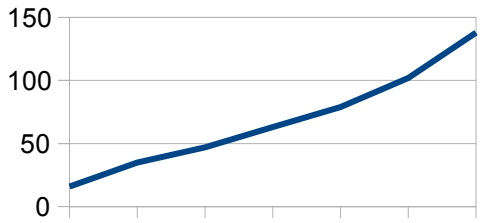

Fig. 8.1: FreeBSD (*src*) committers, *1994-2000*

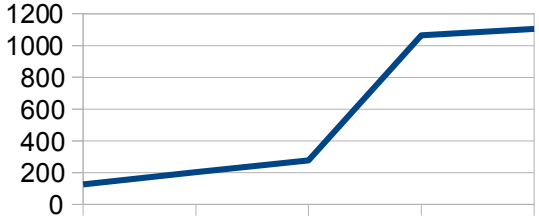

Fig. 8.2: FreeBSD contributors, *1996-2000*

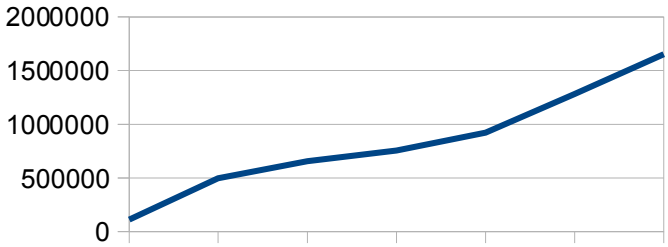

Fig. 8.3: FreeBSD codebase growth (Lines of Code, cumulative), *1994-2000*

This period was however accompanied by a criticism of the way in which the project was

10 We employ the term *governance* to refer to 'the use of institutions, structures of authority and even collaboration to allocate resources and coordinate or control activity' (Bell 2002) in the project. Our employment is akin to that used in international relations, as 'in that context, "governance" is not government, it is typically not authoritative, and in fact it is not about governing in a traditional sense as much as it is about setting parameters for voluntary relationships among autonomous parties' (Weber 2004, p. 172).

governed. Many committers felt that the composition of the core team no longer reflected merit in the project and the core team was censured for abusing its power. In 2000 dissent could no longer be channelled into a manageable form of mediation with the core team. When a prominent committer entered into a confrontation with a core team member, accusing him of trampling on his changes, the situation spiralled out of control, threatening to tear the project apart. In the discussion that ensued on the project mailing lists, Jordan Hubbard outlined a number of possible reforms, including the dismantling of the core team, and called for a vote. The proposal was well received by the base of committers, who elected by vote to adopt an elected core team model, based on the following bylaws[11] that were drafted to regulate core team elections (FreeBSD 2000; Lehey 2002):

- The core consists of nine elected active committers and election is held every two years
- Core members and committers may be ejected by a 2/3 vote of core
- If the size of core falls below 7, an early election is held.
- A petition of 1/3 of active committers can trigger an early election.
- These rules can be changed by a 2/3 majority of committers if at least 50% of active committers cast their vote.

Table 1: Core bylaws (Source: FreeBSD 2000)

Approved by a vote of active committers (passed by 117 yes votes to 5 no votes [Lehey 2002]) on 28 August 2000, these bylaws established criteria of eligibility (all active committers), the size of core team (nine committers), the periodicity of elections (fixed at every two years) and the conditions under which: (a) early elections are held (on the petition of 1/3 of active committers or if size of core falls below 7), (b) a core team member or committer can be expelled from the project (by a 2/3 vote of core) and (c) these bylaws can be modified (FreeBSD 2002). The first core team formed in that way through elections consisted of five former core members (Satoshi Asami, David Greenman, Jordan Hubbard, Doug Rabson, Peter Wemm) plus four new ones (Greg Lehey, Warner Losh, Mike Smith, Robert Watson).

The institutional evolution of FreeBSD is also reflected in a series of documents which the project released with a view to imparting structure to what was until then largely an informal development process. The first version of the *Committer's Guide* (FreeBSD 1999), which laid down guidelines for regulating committers' mode of conduct, was published in 1999 amidst a climate of rising discontent with the project's governance structure. The first version of the *FreeBSD Developers' Handbook* followed in August 2000 – a month before the first core team election – with information geared to new committers about circumnavigating FreeBSD's development model.

In sum, conflicts over the distribution of authority in the project and concerns of a perceived

11 The *core bylaws* do not make up what is normally understood by the term *constitution*: they specify the mode of elections and the duration of the incumbency, but unlike a constitution they make no reference to the principles on which the core team shall be established, the manner in which it shall be organised or the powers it shall have, save for establishing the right of committers to recall the core team by triggering an early election. Some of those questions are dealt with in other documents released by the project. For example, *The FreeBSD Committers' Big List of Rules* clarifies that the authority of the core team is restricted to the task of managing commit privileges: 'In all other aspects of project operation, core is a subset of committers and is bound by the same rules. Just because someone is in core this does not mean that they have special dispensation to step outside any of the lines painted here; core's "special powers" only kick in when it acts as a group, not on an individual basis. As individuals, the core team members are all committers first and core second' (FreeBSD 2011d). On the whole, questions related to the distribution of authority in the project were – and still are – the epicentre of *conflict*: for instance, the reason why decisions are made by consensus does not lie in some formal rule forbidding the core team from making decisions autocratically, but in the vigorous resistance of committers against core team decisions they regard as conflicting with their own will (Lehey 2002).

illegitimacy in its exercise by the core team led to the adoption of an elected core team model in 2000. This institutional restructuring along with the bylaws drafted to regulate elections created a democratic basis of legitimacy for the authority of the core team. Closely related with this reform was the parallel attempt to more elaborately define the scope of development activities, crystallised in the release of the first version of the Committer's Guide in 1999 which elucidated the process through which changes are integrated in the repository and outlined committers' behavioural code.

Democratic governance phase (2000-to date)

The first core team election by vote in September 2000 ushered in the next phase in the institutional evolution of the project, that of *democratic governance*. In 2002 elections were held again as the core team was left with six members following the resignations of Satoshi Asami, Jordan Hubbard and Mike Smith.[12] The new core team had five new members (John Baldwin, Jun Kuriyama, Mark Murray, Wes Peters, Murray Stokely) and four from the previous one formed in 2000 (Greg Lehey, Warner Losh, Robert Watson, Peter Wemm). Of its nine members, only one – Peter Wemm – was part of the original core team. Elections have been held four more times since. The last one in 2010 resulted in the following core team: John Baldwin, Konstantin Belousov, Warner Losh, Pav Lucistnik, Colin Percival, Wilko Bulte, Brooks Davis, Hiroki Sato and Robert Watson. The transition from a self-selected group of veteran committers to an elected one reinforced the already extant tendency toward the systematisation of rules and development procedures.

Indicative of the ongoing formalisation of rules and procedures is that increasingly more of them are being written down as shown by the continuous updates of the FreeBSD Handbook, the Committer's Guide and the Developers' Handbook. More interesting, for the purposes of our analysis, is that this process is closely connected with the exigencies of conflict management. No example illustrates this better than the SMP conflict in February 2002 which erupted over changes made by a committer to the SMP code without the permission of John Baldwin, SMP's most active then-developer. The core team intervened immediately asking him to remove his changes from the repository under the threat of revoking his commit privileges. He complied and asked the core team to resolve the issue. The core team, after a month of discussion and consultation with committers on project mailing lists, decided to delegate authority to John Baldwin to approve or reject changes to the SMP code as he saw fit, and then used the experience to formulate rules for suspending commit rights, thereby creating a standard discipline procedure with set offences and penalties:

1. Committing during code freezes results in a suspension of commit bits for two days.
2. Committing to the security branch without approval results in a suspension of commit privileges for 2 days.
3. Commit wars will result in both parties having their commit bits suspended for 5 days.
4. Impolite or inappropriate behaviour results in suspension of commit bits for 5 days.
5. Any single member of core or appropriate other teams can implement the suspension without the need for a formal vote.
6. Core reserves the right to impose harsher penalties for repeat offenders, including longer suspension terms and the permanent removal of commit privileges. These penalties are subject to a 2/3 majority vote in core.
7. In each case, the suspension will be published on the developers mailing list.

Table 2: Rules for the suspension of commit rights (Source: FreeBSD 2011d; Lehey 2002)

12 See Lehey (2002) for the reasons cited by Hubbard and Smith.

However, in order for the decisions of the core team to be received as legitimate, they must be consistent with the consensus of the opinions of the committers. Characteristically, in June 2002 the core team received another complaint about the same committer. He had again committed changes to an area of the codebase without the approval of the committer who was responsible for it. The core team decided to suspend his commit privileges for five days in accordance with the above disciplinary rules. But 'public reaction was unfavourable': the decision was censured for being politically-motivated, as core elections were underway and the suspended committer was a candidate. Under these circumstances, the core team was forced to reprieve the suspension after two days (Lehey 2002).

The transition to the elected core team model, though it appeased concerns of an illegitimacy in the distribution of authority in the project, did not eradicate conflicts. A case in point is the conflict in 2003 between the core team and Matt Dillon, a prolific committer, which led to the revocation of the latter's commit rights. According to the explanation given by two members of the then-core team, Warner Losh and Greg Lehey, on a popular online discussion forum for hackers, this decision was dictated by social, rather than technological, considerations: Dillon had repeatedly violated FreeBSD's code of conduct: his behaviour clashed with the collective way of doing things (Slashdot 2003). A few months later, Dillon announced his decision to 'fork' FreeBSD – that is, to make a copy of the codebase and start independent development – thus creating an alternative project called DragonFly BSD (Dillon 2003). Dillon, for his part, claimed that he did not launch a new project because of his strained relations with FreeBSD committers, but due to reasons of difference of opinion regarding the technical direction of FreeBSD, stressing its SMP implementation (Biancuzzi 2004). Although Dillon's 'ostracism' illustrates clearly that not all conflicts are amenable to resolution, it also suggests that the freedom to fork a project (which FOSS licenses ensure) mitigates the potential for conflicts. Organisation theorists know well that easy access to the exit option dampens the emergence of conflicts: the potential for conflicts in a group is drastically reduced when members can easily walk out, disengaging themselves from it (Hirschman 1970). Forking is nothing but an extreme example of the exit option: in this way, disputes over the direction of technical change in the project that do not admit of resolution are effectively 'translated' into alternative development lines (see FreeBSD core team interview by Loli-Gueru 2003).

This period – just as that before it – is marked by rapid growth. The massive expansion of scale is illustrated from the increase of (*src*) committers from 138 in 2000 to 209 in 2005 (by 2010 the number of committers had increased to 288 [FreeBSD 2010d]). Although the expansion of scale brought about a significant increase of coordination costs, the increased need of active coordination within the group did not lead to the introduction of direct supervision, meaning an internal hierarchy where contributions are processed upstream through 'gatekeepers'. Rather, it prompted changes in the direction of increased *standardisation*: namely, the standardisation of committers' recruitment process and of outputs through frequent building (Holck & Jørgensen 2003/4; Jørgensen 2007).

A standard argument of organisation theory is that work coordination in a small group may well be informal, based on the 'mutual adjustment' of group members. However, as the group gets larger, it becomes less able to coordinate informally. Thus, control of the work passes into a single individual and direct supervision becomes the chief means of coordination (Mintzberg 1993, p. 7). But FreeBSD, in spite of the dramatic expansion of the committers group, made no attempt to supervise their work process. Rather, it opted to standardise their *skills*[13] by standardising the process through which outside contributors are inducted into the project. The process is as follows: a committer proposes to the core team to grant commit rights to an outside contributor,

13 '*Skills* are *standardized* when the kind of training required to perform the work is specified' (Mintzberg 1993, p. 6).

based on the latter's history of contributions.[14] Usually, the committer who vouches for a new member becomes his mentor, assuming responsibility for everything his protégé does in the project. The mentor is in a sense his supervisor: he is responsible for reviewing and approving his changes prior to being committed to the repository. The mentorship period has no specific duration and ends when the mentor 'releases' officially the new committer. By that time, the new committer is supposed to have developed a strong grasp of project goals and mastered the requisite technical and interpersonal skills (Lucas 2002). Put another way, the standardisation of the recruitment process is designed to harmonise the coexistence of fiercely independent individuals within the committers group by reducing the scope of conflicts related to the integration of changes (Watson 2006). It achieves this by building into the committers-to-be the 'work programs' as well as the bases of coordination. Thus, on the job they appear to be acting autonomously, just as a surgeon and an anaesthesiologist need hardly communicate when they meet in the operating room, knowing through their training exactly what to expect from each other. By virtue of cultivating a *homogeneity of values*,[15] the recruitment process ensures that the conduct of new committers is compatible with the collective way of doing things and so reproduces the structural properties of the FreeBSD social system.

To reduce the need for active coordination, FreeBSD also resorted to standardising *outputs through frequent building* (Holck & Jørgensen 2003/4; Jørgensen 2005, 2007). Doing a 'software build' refers to the process of converting human-readable source code into executable code that can be run on a computer. A successful build therefore implies that a working version of the software can be 'built' from the evolving codebase. Aside from the obvious benefit of testing whether the evolving product is kept in a working state, software companies do frequent builds to facilitate team coordination: 'the key idea is that one large team can work like many small teams if developers synchronize their work through frequent “builds” and periodic “stabilizations” of the product' (Cusumano & Selby 1997, p. 262). FOSS projects are not an exception (Krill 2011). FreeBSD uses three so-called Tinderbox servers that automatically build the most recent version of the software every few hours.[16] The results are posted on the web and on project mailing lists, notifying committers of 'tinderbox failures'. This is focal to the project's use of mailing lists because committers see the effect of the most recent changes and so can pinpoint which change is responsible for breaking the build. Thus, frequent building makes the development process more visible and predictable by allowing committers to monitor their progress in developing new features and stay in sync with the evolving product. As broken builds result in halting further development until the bug responsible for the breakdown is found and fixed, a key rule for committers is to make no changes that cause the build to fail (FreeBSD 2011d). This rule, as Holck and Jørgensen (2003/2004) remark, by specifying a criterion of performance that the work of committers is required to meet, achieves the standardisation of the results of their work. Compliance with the rule reduces the need for active coordination among committers, as 'with outputs standardized, the coordination among tasks is predetermined, as in the book bindery that knows that the pages it receives from one place will fit perfectly into the covers it receives from another' (Mintzberg 1993, p. 6). Similarly, FreeBSD committers coordinate with each other in terms of definite performance standards: they are expected to commit changes that do not break the build; how they do this is their own business.

To more effectively accommodate increased scale, the project proceeded to a series of further measures. First, in 2001 it started using *quarterly status reports* to give contributors an overview

14 This part of the process has been formalised since 2002: the FreeBSD website outlines the exact steps would-be mentors must follow to propose a new committer (FreeBSD 2011c).

15 Organisations which 'generally refuse to legitimate the use of centralized authority...to achieve social control', commonly resort to such a 'selection for homogeneity', as shown by Rothschild-Whitt's (1979, pp. 513-4) classic study of collectivist organisations. This homogeneity is, of course, reinforced by the *self-selection* characteristic of participation in collectivist organisations (Mansbridge 1977, p. 336).

16 The results of the daily build process are accessible online at <http://tinderbox.freebsd.org>. Indicatively, on 21 June 2011, tinderbox machines performed builds of the *-current* version and of six officially released versions of FreeBSD on nine different hardware platforms.

of the various development activities in progress. Second, from 2003 onwards increasingly more development activity migrated from CVS to Perforce and later on to the Subversion revision control environment because of those environments' superior support for parallel development. By 2006 Perforce had replaced CVS as the development site of experimental features, while the Subversion server is where development work on the *src* tree is currently taking place (FreeBSD 2011a; Long 2010; Watson 2006). Third, the project tried to decouple the work of different (groups of) committers by organising the development of important new features as independent sub-projects with their own project manager (Holck & Jørgensen 2003/4, p. 46). In this way, experimental features are developed in a Perforce revision control environment and merged into the main repository only when they are mature enough (Long 2010). Fourth, the project placed a great importance upon developer events, encouraging its contributors to attend them. In fact, one of the activities for which the FreeBSD Foundation was explicitly set up in 2000 is event sponsorship.[17]

In this period roles and responsibilities are increasingly decoupled from individual committers and delegated to teams. In the informal governance phase, to take one example, one person – Satoshi Asami, known as 'Mr Ports' among FreeBSD developers – was responsible for the entire ports collection. In 2001, he was replaced by the Ports Management Team.[18] Similarly, the position of Security Officer expanded into the Security Officer Team in 2002.[19] Whereas FreeBSD machines were administered in the first phase by two or three persons, an admin team was formed for this purpose in the latter phase. In the informal governance phase, public relations were entrusted to one person – the FreeBSD president – who was responsible for interfacing with corporate contacts. Following the abolition of the presidential position in 1997, the task was picked up by the marketing team and, since its founding in 2000, by the FreeBSD Foundation. Every change we have enumerated so far – from the systematisation of rules and procedures to the formation of administrative teams charged with tasks formerly carried out by just one person – attests that there is a contingent relationship between the governance structure and the scale and maturity of a FOSS project (de Laat 2007; O'Mahony & Ferraro 2007, p. 1101; Mateos-Garcia & Steinmueller 2008).

The imperative of autonomy

Although the adoption of the elective principle altered substantially the mode of project governance, it did not affect the mode of work organisation of committers. The process by which changes are integrated in the repository remained the same. Its main feature – the ability of committers to integrate changes directly to the repository – did not change. How is it possible, given the dramatic increase of participating committers over time, that the expansion of scale did not result in a hierarchical structure where changes are processed upstream through a series of gatekeepers? In other words, how is the autonomy committers enjoy accounted for? In the first place, developers come to work in FreeBSD because it offers them substantial control over their work, thus indicating that the governance structure of the development process is an important motivating factor (Jørgensen 2005, p. 122). In a survey of seventy-two FreeBSD committers, more than 80% of them said they were encouraged by the freedom to commit code directly to the repository: 'It is frequently easier to make a change to the code base directly than to explain the change so someone else can do it'; 'I don't feel I am under the whim of a single person' or 'I have submitted fixes to other projects and been ignored. That was no fun at all' (Jørgensen 2005, p. 233; see also Jørgensen 2001, 2007).

Bearing in mind FreeBSD's historical background, the significance its developers attribute to their autonomy is hardly surprising. FreeBSD is descended – via BSD – from Unix, which was

17 According to one of the project founders, developer events contribute to relationship-building and effective conflict management (Hubbard 1998a).

18 The Ports Management Team currently numbers eight members.

19 The Security Officer Team currently numbers eleven members.

developed in a radically informal and anti-bureaucratic fashion. When AT&T's BTL withdrew from the Multics project, whose aim was the development of a multi-user operating system, some BTL employees took it upon themselves to bootstrap their own without any assistance from their employer. So was Unix born. Its development was from the beginning autonomous from BTL, dispensing with its supervision. This, however, served to strengthen the feeling of solidarity among the growing number of users at american universities, turning thus the development of Unix into a truly collaborative enterprise (Pfaffenberger 1996; Raymond 2003; Ritchie 1984; Salus 1994). The subsequent development of BSD at the Berkeley campus of the University of California similarly shunned bureaucratic principles of organisation, pioneering a model which revolved around a group of programmers called committers on account of their power to make changes to the codebase:

> The committers were a group of people we trusted to commit stuff...The notion was that you didn't have all these autocratic controls...we didn't need to tell people not to do that; we didn't have to administratively keep them from doing things they shouldn't be doing. We had set up a culture as well as a structure (McKusick quoted in Leonard [2000]).

In addition to animating the development of Unix and BSD, the principle of autonomy is focal to the model of Internet governance evolved by the hacker community. The prototype of this model is the Internet Engineering Task Force (IETF): formed in 1986, it is the closest thing there is to an institution responsible for the development of Internet standards. Its founding belief, as put forth by David Clark, is as follows:

> We reject kings, presidents and voting. We believe in rough consensus and running code (Clark 1992; Hoffman 2010).

Deciding whether to adopt or reject a standard through *rough consensus* means that while unanimity is not required, 'strongly held objections must be debated until most people are satisfied that these objections are wrong' (Hoffman 2010). In practice, though there is no fixed percentage, most proposals that are accepted have the support of no less than 90% of the working group (Bradner 1999). Similarly in FreeBSD, as FreeBSD committer Joseph Koshy (2010) says, 'formal specifications and design documents are seldom used...Clear and well-written code and well-written change logs are used in their place. FreeBSD development happens by “rough consensus and running code”'.

The analysis of the historical and cultural context in which the development of FreeBSD is embedded brings into sharp focus a broader normative standard with reference to which individual hackers act. It shows that the motive of autonomy attributed to the conduct of FreeBSD developers accords with recognised normative patterns. The freedom they have to commit changes directly to the repository makes sense in terms of accepted norms, as does their imperviousness to taking orders. The role of autonomy as an organising norm explains why the dramatic increase of committers did not lead to the introduction of supervisory hierarchy. The exercise of authority in FOSS projects – as well as its transmutations over time – cannot be understood apart from the influence of the normative standard of autonomy. Under no circumstances is the conduct of bearers of administrative authority – the core team in the case of FreeBSD – allowed to infringe upon developers' autonomy of action, making thus impossible the adoption of organisational configurations which seem to contravene this fundamental principle.

Authority and legitimacy

Max Weber's classic analysis of how authority is legitimised provides a lens through which the historical transformation of FreeBSD's governance structure can be viewed. According to Weber (1947, pp. 124-125), no authority system is stable unless it is based on the belief of those subject

to it in the legitimacy of their subordination. He distinguishes between three types of *legitimate authority*.[20] 1. The first type is that of *legal* (or *legal-rational*) authority. In this case, 'obedience is owed to the legally established impersonal order' (Weber 1947, p. 328) so those subject to legal authority 'owe no personal allegiance to a superordinate and follow his commands only within the restricted sphere in which his jurisdiction is clearly specified' (Giddens 1988, p. 158). Persons in authority occupy a 'position' or 'office', to which they are appointed. Their organisation is hierarchical: 'each lower office is under the control and supervision of a higher one' (Weber 1947, p. 331). 2. *Traditional* authority is based on the sanctity of age-old rules and powers handed down from the past, such as that which is exercised by village elders in small rural communities. 3. *Charismatic* authority, Weber's third type, is that which is recognised by those subject to it as interlaced with the extraordinary abilities of the leader, 'by virtue of which he is...treated as endowed with supernatural, superhuman, or at least specifically exceptional powers or qualities' (Weber 1947, p. 358). To this type belongs the authority exercised, for example, by prophets and religious leaders over their followers or by heroes in war. The claim to legitimacy in charismatic authority is founded upon the belief in the authenticity and uniqueness of the leader's mission, for which he supplies proof through his prodigious feats: hence the prophet has to perform miracles and the war hero triumphant military exploits. The administration of groups subject to charismatic authority is not carried out by 'officials' but by the leader's disciples who share in his charisma. There is no such thing as career or promotion, no salary, no benefice. There is only a 'call', a 'mission' or 'spiritual duty': the leader's administrative staff is summoned to the charismatic mission. There is no hierarchy: the leader merely intervenes when he considers the members of his staff inadequate to the tasks they have been entrusted with. There is no system of formal rules or precedents handed down from the past: 'the genuine prophet, like the genuine military leader and every true leader in this sense, preaches, creates, or demands *new* obligations' (Weber 1947, p. 361).

As a general rule, FOSS projects 'are created with few traditions to guide them and so do not inherit a traditional basis of authority' (O'Mahony & Ferraro 2007, p. 1081). They do not rely upon a legal-rational basis of authority either, as there is no authoritative division of labour. But as authority cannot be validated through tradition or hierarchy, its justification often turns on the charisma of its bearers. The leadership of Unix had, without doubt, a charismatic character during its early development at AT&T. From its inception in 1969 until the mid-70s, the development of Unix is closely connected with the names of Ken Thompson and Dennis Ritchie. In recognition of their important role in the making of Unix, they both have risen to mythical status in hacker folklore. The FOSS literature has the tendency to present them as individuals endowed with extraordinary abilities (e.g. Raymond 2000). The same charismatic qualities are also attributed to their successor Bill Joy, who spearheaded the consequent development of Unix at Berkeley from 1977 until 1982. As one of his Berkeley colleagues describes him: 'He had an infectious enthusiasm about him, where he would just get the people around him to do stuff' (McKusick quoted in Leonard [2000]).

The rule of charisma is however ephemeral. Because of its disdain for the routine and the everyday, it is impossible for charisma to survive unless it undergoes a profound modification. Its 'routinisation' therefore implies the devolution of charismatic authority. It is not hard to discern the occurrence of this transformation in BSD. The project already counted more than five years of development by the time Joy stepped down in 1982. In the wake of his departure, Sam Leffler – Joy's second-in-command – took over the responsibility of completing the release of 4.2BSD. But because 'he was not appointed to Joy's post and felt slighted by this' (Salus 1994), he soon left for Lucasfilm.[21] Following the release of 4.2BSD in August 1983, Leffler was replaced by another member of the team of programmers working on BSD at Berkeley (known since 1980 as the Computer Science Research Group or CSRG for short), Mike Karels, who was joined by Kirk McKusick in December 1984. Under their leadership, the project evolved an organisational

20 *(Herrschaft)*

21 Currently, Leffler is a FreeBSD committer and a member of the FreeBSD Foundation's board of directors.

structure with a core team at the centre and a wider base of committers surrounding it (Leonard 2000). The type of authority relationship that emerges from the routinisation of charisma, according to Weber, is determined in large part by how the 'succession problem' is resolved. In the case of BSD, the successor was not nominated by the predecessor. Nor was he self-selected: in spite of his professed willingness to take on the leader's role, Leffler was not 'appointed' to this position by the CSRG and soon stepped down. On the contrary, the fitness of his substitute for the position, Mike Karels, as well as that of Kirk McKusick, was validated through his designation by the CSRG. The issue of succession was not raised again in BSD. With Karels and McKusick as project coordinators, a two-tier organisational structure began to take shape in which leadership, rather than being vested in a single person, was entrusted to a self-selected group of heavily involved developers. This set the stage for an important reinterpretation of the charismatic principle. Instead of being restricted to the person of the project leader, the 'gift of grace' was extended to a leading cadre of hardcore developers.[22] FreeBSD inherited this conception of charismatic authority from BSD along with its organisational template.

When the FreeBSD project was launched in 1993, the core team included 13 individuals: the last three coordinators of the 'unofficial 386BSD patchkit' plus its most then-active developers. The development of FreeBSD was – and still is – based on a group of programmers who are called committers because of their ability to make changes to the codebase. Committers organised themselves as an *informal meritocracy*: the most active committers were invited by the core team to join its ranks and outside contributors who regularly sent useful patches were offered commit rights. Granting commit rights to an outside contributor amounted to recognition of the technical expertise that his patches demonstrated. In the same way, inviting a committer to join the core team reflected the recognition of his outstanding contribution to FreeBSD and brilliance in coding. Authority was derived from technical competence, acquired and demonstrated through participation in the project.

Although the conception of merit in the project did not change, the criticism of the selection process of the core team as well as of its prerogatives became more virulent over time. Its thrust was, on the one hand, that the core team had degenerated into a gerontocracy of veteran FreeBSD developers which no longer reflected merit in the project and, on the other, that members of the core team abused their power to serve their own ends. When in 2000 a prominent committer announced his intention to quit the project because a core team member was trampling over his work, the criticism of the core team turned to an open conflict that rapidly took on alarming proportions. The intervention of one of the project founders at this point was of decisive importance. He suggested a number of alternative reforms and called on committers to vote. They responded to his call, deciding by vote to adopt an elected core team model. Core bylaws were drafted shortly thereafter to regulate elections.

The transformation of charisma set off by the application of the elective principle to the core team selection was in this case fuelled by the rupture between the group of committers and the core team. The conflict that manifested itself through the growing criticism of the distribution of authority in the project brought about a shift in governance toward an electoral process for the selection of the core team. As a result, the core team, whose legitimacy rested on its members' charisma, then became the core team thanks to the confidence of committers. The introduction of elected core team members entailed a radical alteration in their position: they became the 'servants' of those under their authority. The passage of leadership from a self-selected group to a freely elected one signified that from now on committers were free to elevate to power as well as depose as they pleased. Whereas the recognition of the charisma of the core team was so far perceived by committers as a consequence of its legitimacy, it now began to be considered as its basis. Legitimacy was in this sense democratised.

22 Weber recognised that 'it is possible for any type of authority to be deprived of its monocratic character, which bonds it to a single person, by the principle of collegiality' (Weber 1947, p. 392).

The reconfiguration of the governance system brought about by the transformation of charisma limited the authority of the core team in four important ways. First, the sphere of its authority was circumscribed: the role of the core team was restricted to managing commit privileges and mediating in the event there is a serious disagreement between committers. Second, it was made accountable to the community of committers: the core team is required to defer to their wishes, making only decisions that reflect the consensus of the opinions of committers as manifest on mailing lists. Third, its term of office was specified: new elections would be held every two years. Fourth, project leadership became revocable: the core bylaws invested committers with the power to trigger an early election, thereby recalling the core team. All these traits correspond to the type of governance Weber calls *direct democracy*: the short term of office, the liability to recall, the restricted sphere of jurisdiction, the obligation to render an accounting to the general community of committers as well as submit to it every important question (Weber 1947, pp. 412-3). Direct democracy is characteristic of groups which, in order to preserve their members' autonomy, attempt 'to dispense with leadership altogether' by reducing 'to a minimum the control of some men over others' (Weber 1947, p. 389). In that sense, direct-democratic forms of governance are inherently *anti-authoritarian.*

In FreeBSD, more specifically, the anti-authoritarian transformation of charisma that culminated in the adoption of a direct-democratic mode of governance limited the authority of the core team through the introduction of elements of democratic as well as of legal-rational rule. The principles of consensus-oriented decision making, the limited duration of office and the liability to recall are all institutional safeguards drawing their justification from the sovereignty of the will of committers. The premises for delimiting the authority of the core team by specifying its sphere of jurisdiction are, on the other hand, bureaucratic par excellence. Authority in a bureaucratic organisation is distributed and legitimised only within the particular sphere of the office (Weber 1947, p. 330). The authority of the core team is likewise restricted to a specific field: it can be exercised only in matters touching commit rights and committer disputes (FreeBSD 2011d). The use of hats within the project – that is, of assigning clearly circumscribed areas of responsibility to certain committers – is also indicative of a stripped-down form of bureaucratisation as is the tendency toward the formation of teams that take on the role formerly held by a single committer (e.g. Ports Management and Security Officer teams).

Weber (1947, p. 390) remarked that 'the anti-authoritarian direction of the transformation of charisma normally leads into the path of rationality', as the setting up of an administrative organ that functions reliably invariably involves the systematisation of rules and procedures, fuelling thus the progressive bureaucratisation of the group. Yet the authority of the core team does not belong to the bureaucratic type. If bureaucracy is understood as a 'clearly defined hierarchy of offices' (Weber 1947, p. 333), then core team members are not bureaucratic types. Since there are no officers on the core team, core team members are not integrated in a hierarchical order: they have no superiors who influence their 'promotion' to the core team or supervise their activity. In contrast to bureaucratic organisations which mobilise their members through remunerative incentives, participation in FreeBSD is voluntary and unwaged. Although many committers are professionals in the IT industry,[23] their involvement in FreeBSD cannot be regarded as a career, as conventionally understood. For there is no career advancement in FreeBSD: external contributors can become committers and committers core team members, but that is hardly analogous to moving up in a multi-layered hierarchy of ranks. In fact, the aim of FreeBSD's governance system is to eliminate the division of labour that separates decision making labour (administrative tasks) from executive labour (performance tasks). Not only is the core team, in addition to its managerial duties, expected to be producing code, but more crucially decision making rests on a *consensus* process in which all project members can participate. For decisions to be taken as binding and legitimate, they must carry the consensus of the group behind them. And so to ensure that

23 Indicatively, in a survey of 72 FreeBSD committers (constituting 35% of all committers) conducted in 2000, '43% said an employer had paid for all or part of their time spent on their latest code contribution' (Jørgensen 2001).

committers can participate in the process of formulating problems and negotiating decisions, all issues are discussed on project mailing lists.

What, for Weber, differentiates bureaucracy from other forms of organisation is that it allows for regular control of operations over time. That is what, in his view, makes bureaucracy 'rational'. According to Foucault (1975), whose work deals more extensively with the theme of time-space control, the distinguishing feature of bureaucratic organisation – whether in schools, barracks, factories or hospitals – is that the use of an individual's time and space is constantly monitored and controlled. Every individual is assigned its 'proper place' and has certain duties to perform at any particular moment. This type of administrative authority, Foucault says, connects discipline directly with utility: its goal is to ensure that the use of an individual's time is channelled solely into those activities that the administrators consider useful. By contrast, participation in the development of FreeBSD is not subject to such forms of control. The project does not keep any record of the time committers dedicate to it. Committers participate in their free time, deciding themselves when they will work and for how long. Moreover, their geographical whereabouts are irrelevant: they may work on FreeBSD from the privacy of their homes or from any other place. As seen from the standpoint of time-space control, FreeBSD wholly dispenses with the 'discipline' characteristic of bureaucratic administration: no attempt has ever been made in the project to supervise the individual activities of committers or control with any means the use of their time or space.

The divergence of FreeBSD from the bureaucratic model can also be illustrated from the form of social relationships in the project. While social relations in bureaucratic organisations are based on the formal roles held by their members as laid down by an authoritative division of labour, relationships between FreeBSD developers are far more holistic, affective and personal.[24] For committers, FreeBSD is a *community*; 'a fraternity of peers', so to speak. While bureaucratic organisation separates the 'official' from the 'personal', these two dimensions fuse together in the ideal of community that FreeBSD aspires to (O'Neil 2009, p. 175). In Weberian terms, the orientation of social action in FreeBSD is value-rational: that is, social conduct is based on definite moral values. The actions of individuals are directed to an overriding ideal: being part of the hacker community that coalesces around the development of the FreeBSD operating system (cf. Torvalds 1998). That is not to say that their actions are not informed by pragmatic considerations, chiefly that they want the fruits of their labour to be used by as many people as possible (Hubbard 1998b). But relationships between people in FreeBSD – as is typical of collectively-run volunteer organisations (Rothschild-Whitt 1979, p. 514) – are seen as of value in themselves. Arguably, it is not on account of holding some office that core team members are recognised as figures of authority. Although their opinion may well carry more weight in discussions occurring on project mailing lists than that of other committers, this influence is not the result of their 'powers of office' but rather of the respect and trust given them by committers for their substantial contribution to the project. In collectivist organisations, 'because authority resides in the collectivity as a unit, the exercise of influence depends less on positional opportunities and more on the personal attributes of the individual' (Rothschild-Whitt 1979, p. 524). Prior studies have shown that collectivist organisations find such inequalities of influence 'acceptable in circumstances in which those who exercise power exercise it in the interests of others (usually because their interests are identical with those of others)'(Mansbridge 1977, p. 326). Hence, the reason why committers accept that some of them exert more influence than others is because that influence is seen as compatible with their own interests. Some traces of

24 One may wonder how is it possible that developer relations in FreeBSD are 'personal', given that their interactions occur predominantly in a computer environment. After all, long distance relationships seem rather impoverished, if not shallow, compared to relationships that are based on physical co-presence. It is instructive in this connection to refer to the emphasis Marshall McLuhan (1964) laid on how the diffusion of electronic telecommunications would transform the globe into a 'global village', signalling the return of humanity to a tribal-esque form of sociality. For McLuhan, the effect of telematic technology on social interaction is profound: as its scope is no longer determined by geographical proximity but by affinity, it becomes possible for relations of a more remote kind to be experienced as meaningful and personal.

charismatic authority can still be detected in this type of relationship: the trust of committers in core team members is, to a certain extent, of an emotional type; and the persuasive authority of core team members is to some extent legitimised through the recognition of the authenticity of their technical charisma by committers.

For Weber, the transition from the autocratic selection of the core team to its democratic election by vote signals the end of charismatic rule, as its subjection to norms and rules invariably involves the loss of genuine charismatic authority. Charisma abhors permanent forms of organisation and formal rules. Its claim to legitimacy lies in 'the conception that it is the *duty*' of those subject to charismatic authority to recognise its uniqueness and act accordingly (Weber 1947, pp. 359-60). This conception of authority is no longer representative of FreeBSD. The election of the core team by and amongst committers resulted in changing the basis of its legitimacy. The recognition of charisma is no longer treated by committers as a consequence of the legitimacy of authority but as the basis upon which it rests. While legitimacy formerly rested on the 'duty' of committers to recognise the technical charisma of the core team, it became democratic in the latter period with the application of the elective principle: the authority of the core team was no longer validated by the charisma of its members but by the will of committers. Legitimacy was thus 'democratised'.

The routinisation of charisma in FreeBSD resulted in a direct-democratic governance system in which the distribution of authority is validated by the will of committers. Although that form of governance includes elements of bureaucratic authority, as the authority of the core team is delimited by mechanisms that to some extent reinforce bureaucratic values (such as the functional specificity of authority), its source of legitimacy is fundamentally democratic: it is justified by the imperative to preserve the sovereignty of the committers' will rather than by its adherence to an impersonal hierarchical order. It is important to observe that the transformation of charismatic to democratic authority did not modify the conception of merit in the project, which remains anchored in technical competence, acquired and demonstrated through project participation. What changed markedly however is the conception of leadership: leadership is no longer conceptualised as the informal rule of a self-selected group of heavily involved committers, but as a democratically elected group of committers that is revocable and subject to formal rules.

CONCLUSIONS

We analysed FreeBSD's institutional evolution by distinguishing two phases, based on their corresponding mode of governance. While from 1993 until 2000 FreeBSD had no formal means of representing its contributors in project governance, and leadership consisted in a self-selected group of veteran committers, in 2000 the growing criticism of the distribution of authority in the project brought about a shift toward an elected model, according to which project leadership is exercised by nine persons elected biennially by and amongst committers. Considering the dramatic increase of committers over time, the transformation of the governance system – as well as the systematisation of rules and procedures that runs parallel to it – suggests that a project's governance structure is contingent upon its scale and maturity. The transformation of the governance system, however, did not affect the mode of work organisation of committers in the development process, in spite of the remarkable expansion of scale.

While organisation theory predicts that as a group grows larger it becomes less able to self-organise and so is compelled to turn to supervisory hierarchy as a means of coordination, the expansion of the committers group was not accompanied by changes in that direction. Rather, the project resorted to standardising (a) the recruitment process for new committers and (b) outputs through frequent building. This line of development cannot be understood apart from the influence of the normative standard of individual autonomy of action: it can be accounted for only by bearing into mind that an important reason why hackers are attracted to FreeBSD is the freedom of committers to add changes directly to the repository. The centrality of the autonomy principle elucidates the intervening motivational link between the observed activity – the course of action

FreeBSD took to manage increased scale and achieve work coordination within an expanding group – and its meaning to the actors involved. A basic principle of the hacker ethic is to 'mistrust authority – promote decentralization' (Levy 1984). Hackers espouse the view that the ultimate effect of centralised authority is to strangle the creative potential inherent in self-regulating individuals, thus acting as a check upon their free development. As the activities of hackers are driven by an acute sense of independence, it is not conceivable that they would adopt organisational configurations which contravene their autonomy.

The normative significance of individual autonomy explains why authority in FOSS projects cannot be coercive. Naturally, that is not to say that no authority exists. In FreeBSD it specifically consists in control of the ability to make changes to the codebase. Considering that no authority relationship is stable unless it is recognised by those who submit to it as based on some legitimate order (Weber 1947), we examined how authority is legitimised in FreeBSD, contrasting it with Weber's categories of legitimate authority. We found that legitimacy shifted from the quasi-charismatic authority of a self-selected group of heavily involved committers to the democratic authority of an elected group that is revocable and bound to formal rules.

However, none of Weber's categories captures sufficiently the character of authority in FreeBSD. If authority is defined as a relationship in which an actor obeys a specific command issued by another, as Weber (1947, p. 152) defines it, then FreeBSD is essentially an *organisation without authority*. There is no such thing as giving or following orders in FreeBSD. The administrative organ of the project – the core team – cannot tell committers what to do. When a decision needs to be made, it is made collectively by consensus. If, in the Weberian tradition, we take the basis of authority as the decisive organisational feature, then the mode of organisation of FreeBSD is collectivist, based on direct-democratic procedures of decision making. Seen from the perspective of the division of labour in the project, the mode of organisation of FreeBSD is decentralised and anti-hierarchical: tasks are self-selected by committers as their needs and interests best dictate. The resulting division of labour is spontaneous in the sense that it emerges from the choices of the committers rather than from a central designer. Committers work without supervision, shouldering themselves the ultimate responsibility that the modifications they make to the codebase have been adequately tested and do not clash with the work of other committers. Consequently, FreeBSD illustrates 'a production process that doesn't rely on managers' (Hamel 2007, p. 208). In FreeBSD those who work also manage.

ACKNOWLEDGEMENTS

Based on the author's doctoral research at Delft University of Technology.

REFERENCES

Andrews J. 2008. FreeBSD: Tinderbox Failures. *KernelTrap* (Feb. 28). Accessible at <http://kerneltrap.org/node/595>

Bell S. 2002. *Economic Governance and Institutional Dynamics*. Oxford University Press: Melbourne, Australia

Benkler Y. 2006. The Wealth of Networks: How Social Production Transforms Markets and Freedom. Yale University Press

Biancuzzi F. 2004. 'Behind DragonFlyBSD'. *ONLamp.com*. Accessible at <http://onlamp.com/lpt/a/4991>

Bradner S. 1999. 'The Internet Engineering Task Force'. In Dibona C., Ockman S. (Eds.) *Open Sources: Voices from the Open Source Revolution*. O'Reilly & Associates

BSD Certification Group. 2005. *BSD Usage Survey* (Oct.). Accessible at <http://www.bsdcertification.org/downloads/pr_20051031_usage_survey_en_en.pdf>

Clark D.D. 1992. ‘A cloudy crystal ball: Visions of the future’. Plenary presentation at 24th meeting of the Internet Engineering Task Force (Jul. 13-17), Cambridge, Mass

Cusumano M., Selby R.W. 1997. *Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People*. London: HarperCollinsBusiness

Demil B., Lecocq, X. 2006. Neither Market nor Hierarchy nor Network: The Emergence of Bazaar Governance. *Organization Studies* **27** (10): 1447-1466

Dillon M. 2003. *Announcing DragonFly BSD!*. Message posted to freebsd-current mailing list (Jul. 16). Accessible at <http://lists.freebsd.org/pipermail/freebsd-current/2003-July/006889.html>

Eisenhardt K. 1989. Building Theories from Case Study Research. *The Academy of Management Review* 4 (4)

Foucault M. 1975. *Discipline and Punish: The Birth of the Prison*. New York: Random House

FreeBSD. 2011a. Committer's Guide. Accessible at <http://www.freebsd.org/doc/en_US.ISO8859-1/articles/committers-guide/>

FreeBSD. 2011c. New Account Creation Procedure. Accessible at <http://www.freebsd.org/internal/new-account.html>

FreeBSD. 2011d. The FreeBSD Committers' Big List of Rules. Accessible at <http://www.freebsd.org/doc/en_US.ISO8859-1/articles/committers-guide/rules.html>

FreeBSD. 2001. Quarterly Status Report (Jun.). Accessible at <http://www.freebsd.org/news/status/report-2001-06.html>

FreeBSD. 1999. New Committer Guide. Accessible at <http://docs.freebsd.org/doc/4.0-RELEASE/usr/share/doc/en/articles/committers-guide/>

Garzarelli G., Galoppini R. 2003. *Capability Coordination in Modular Organization: voluntary FS/OSS Production and the Case of Debian GNU/Linux*. Industrial Organization 0312005, EconWPA. Accessible at <http://ideas.repec.org/p/wpa/wuwpio/0312005.html>

Giddens A. 1988. *Capitalism and modern social theory: An analysis of the writings of Marx, Durkheim and Max Weber*. Cambridge University Press: Cambridge

Hamel G. 2007. *The Future of Management*. Harvard Business School Press

Harrison P.M. 1960. Weber's Categories of Authority and Voluntary Associations. *American Sociological Review* **25**(2): 232-237

Himanen P. 2001. *The Hacker Ethic and the Spirit of the Information Age*. Vintage: London

Hirschman A. 1970. *Exit, Voice, and Loyalty: Responses to Decline in Firms, Organizations, and States*. Cambridge, MA: Harvard University Press.

Hoffman P. 2010. *The Tao of IETF: A Novice's Guide to the Internet Engineering Task Force*. Internet Engineering Task Force, at <https://www.ietf.org/tao.html>.

Holck J., Jørgensen N. 2003/2004. Continuous integration and quality assurance: a case study of two open source projects. *Australasian Journal of Information Systems*, Special Issue (2003/2004): 40-53

Hubbard J. 1998a. Editorial: Pulling on one end of the rope. *Freshmeat.* Accessible at <http://freshmeat.net/articles/editorial-pulling-on-one-end-of-the-rope>

Hubbard J. 1998b. What Is FreeBSD? *Performance Computing.* Accessible at <http://bbs.unix-like.org:8080/boards/Server_DOC/M.1006795908.A>

Hubbard J. 1997. *My resignation as president of the FreeBSD Project*. Message posted to freebsd-announce mailing list (Feb. 5). Accessible at <http://lists.freebsd.org/pipermail/freebsd-announce/1997-February/000305.html>

Jørgensen N. 2007. Developer autonomy in the FreeBSD open source project. *Journal of Management and Governance* **11**: 119-128

Jørgensen N. 2005. 'Incremental and decentralized integration in FreeBSD' in J. Feller, B. Fitzgerald, S.A. Hissam & K.R. Lakhani (Eds). *Perspectives on Free and Open Source Software*. MIT Press, pp. 227-244

Koshy J. 2010 Building Products with FreeBSD (version 1.8). *FreeBSD Project*. Accessible at <http://www.freebsd.org/doc/en/articles/building-products/article.html>

Krill P. 2011. 'Why the time is now for continuous integration in app development'. *InfoWorld* (Jul. 7). Accessible at <https://www.infoworld.com/print/165941>

von Krogh, G., von Hippel, E. 2006. The Promise of Research on Open Source Software. *Management Science* **52** (7): 975-983.

de Laat P.B. 2007. Governance of open source software: state of the art. *Journal of Management and Governance* **11**: 165-177

Lehey G. 2003. *The FreeBSD SMP implementation*. Accessible at <http://www.lemis.com/grog/SMPng/Singapore/slides.pdf>

Lehey G. 2002. *Two years in the trenches: The evolution of a software project*. Accessible at <http://www.lemis.com/grog/In-the-trenches.pdf>

Leonard A. 2000. 'BSD Unix: Power to the people, from the code'. *Salon* (May 16). Accessible at <http://archive.salon.com/tech/fsp/2000/05/16/chapter_2_part_one/index.html>

Levy S. 1984. *Hackers: Heroes of the Computer Revolution*. New York: Anchor Press/Doubleday

Loli-Queru E. 2003. 'Focus on FreeBSD: Interview with the Core Team'. *OSNews* (Apr. 28). Accessible at <http://www.osnews.com/story/3415>

Long S. 2010. Perforce in FreeBSD Development. *The FreeBSD Project*. Accessible at <http://www.freebsd.org/doc/en_US.ISO8859-1/articles/p4-primer/article.html>

Lucas M. 2002. 'How to Become a FreeBSD Committer'. *ONLamp.com*, Jan. 31, at <http://onlamp.com/lpt/a/1492>

McLuhan M. 1964. *Understanding Media: The Extensions of Man*. McGraw-Hill: New York

Mansbridge J.T. 1977. Acceptable Inequalities. *British Journal of Political Science* **7** (3): 321-336

Markus L. 2007. The governance of free/open source software projects: monolithic, multidimensional, or configurational? *Journal of Management and Governance* **11**(2): 151-163

Mateos-Garcia J., Steinmueller E. 2008. The institutions of open source software: Examining the Debian community. *Information Economics and Policy* **20**: 333-344

Mintzberg H. 1993. *Structure in Fives: Designing Effective Organizations*. Englewood Cliffs: Prentice-Hall

O'Mahony S., Ferraro F. 2007. The emergence of governance in an open source community. *Academy of Management Journal* **50**(5): 1079-1106

O'Neil M. 2009. *Cyberchiefs: Autonomy and Authority in Online Tribes*. London & New York: Pluto Press

Pfaffenberger B. 1996. "f i want it, it's OK": Usenet and the (Outer) Limits of Free Speech. *The Information Society* **12**(4): 365-386

Raymond E.S. 2003. *The Art of Unix Programming*. Addison-Wesley

Raymond E.S. 2000. *A Brief History of Hackerdom*. Accessible at <http://www.catb.org/~esr/writings/homesteading/hacker-history/>

Ritchie D.M. 1984. The Evolution of the Unix Time-Sharing System. *AT&T Bell Laboratories Technical Journal* **63**(6): 1577-93. Accessible at <http://cm.bell-labs.com/cm/cs/who/dmr/hist.html>

Rothschild-Whitt J. 1979. The Collectivist Organization: An Alternative to Rational-Bureaucratic Models. *American Sociological Review* **44**(4): 509-527

Saers N. 2005. *A project model for the FreeBSD Project.* Accessible at <hhttp://www.freebsd.org/doc/en/books/dev-model/book.html>

Salus P.H. 1994. UNIX at 25. *Byte* **19**: 75-6. Accessible at <http://www.wolldingwacht.de/unix/unix-at-25.html>

Shah S. 2006. Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development. *Management Science* **52**(7), p.1000-1014.

Slashdot. 2003. *FreeBSD Core Developer Thrown Out* (Feb. 3). Accessible at <http://bsd.slashdot.org/story/03/02/03/239238/FreeBSD-Core-Developer-Thrown-Out>

Spinellis D. 2006. Global software development in the FreeBSD project, in P. Kruchten, Y. Hsieh, E. MacGregor, D. Moitra & W. Strigel (Eds) *International Workshop on Global Software Development for the Practitioner* (May), ACM Press, pp. 73–79

Torvalds L. 1998. FM Interview with Linus Torvalds: What motivates free software developers. *First Monday* **3** (3). Accessible at <http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/583/504>

Watson R. 2006. How the FreeBSD Project Works. *Proceedings of EuroBSDCon*, Milan, Italy. Accessible at <http://www.watson.org/~robert/freebsd/2006eurobsdcon/eurobsdcon2006-howfreebsdworks.pdf>

Weber M. 1947. *The Theory of Social and Economic Organization* (translated by A.M. Henderson & T. Parsons). Free Press: Illinois

Weber S. 2004. *The Success of Open Source.* Cambridge: Harvard University Press