CHAPTER FIVE

TWO ETHICAL MOMENTS IN DEBIAN

F/OSS projects take place on the Internet. Varying in size from a couple of isolated developers to a network of nearly one thousand, they are notable sites where programmers coordinate and produce high-quality software over long distances. A growing body of social science literature has addressed questions of developer motivation (Raymond 1999, Weber 2004), project structures and changing implications for software development (O'Mahoney 2002), open source legality (Heffan 1997, McGowan 2001), their non-economic incentive mechanisms and the influence of cultural norms (Kollock 1997, Ghosh 1998, Weber 2004, Himanen 2001), utilitarian and rational choice incentive structures (Gallaway and Kinnear 2004, Lancashire 2001, Von Hippel and von Krogh 2003), and the broader socio-political implications of F/OSS production (Lessig 1999; Kelty 2008; Benkler 2002; Weber 2004; Coleman 2004).

While some of these studies tangentially discuss ethical questions (e.g., conflict resolution within F/OSS projects), rarely do they address how developers commit themselves to an ethical vision *through*, rather than prior, to their participation in a F/OSS project. Much of the F/OSS literature is heavily focused on the question of motivation or incentive mechanisms and often fails to account for the plasticity of human motivations and ethical perceptions.

Many of these authors acknowledge the importance of shared norms (Weber 2004, O'Mahoney 2002) and usually address this by referring to or quoting the famous passage in Steven Levy's *Hackers* where he defines the tenets of the hacker ethic (1984). In a general sense, these principles still powerfully capture the spirit of hacker ethical commitments. Nonetheless, by falling back on Levy, what goes missing is a detailed demonstration of how these precepts take actual form and how they change over time. Most important, the literature has tended to ignore how hacker valuations, motivations, and commitments are transformed by the lived experiences that unfold in F/OSS projects and institutions that are mediated through project charters and organizational procedures.[1]

After presenting my argument that F/OSS projects are sites for a series of important ethical transformations, the rest of the chapter demonstrates how this is so using the example of the Debian project. In this chapter I draw heavily on the work of the legal theorist Robert Cover (1992) who examines the ways in which "jurisgenesis," the production and stabilization of inhabited normative meanings, requires *ongoing* and sometimes conflicting interpretation of codified textual norms. "Some small and others private, others immense and public" (1992: 99), these continual acts of reinterpretation and commitment establish what he calls a *nomos:*

> We inhabit a *nomos* - a normative universe. We constantly create and maintain a world of right and wrong, of lawful and unlawful... No set of legal institutions or prescriptions exists apart from the narratives that locate it and give it meaning. For every constitution there is an epic, for each decalogue a scripture. Once understood in the context of the

1 A prolific literature in the sociology and anthropology of science fruitfully dissects how vocational and professional identities and ethical commitments are established during periods of training (Good 1994), practicing a vocation (Luhrmann 2002, Rabinow 1996, Gusterson 1996) and are sustained by the coded language of professions that obviate certain ethical concerns (Cohn 1989). Although I am not drawing on any one of these authors, all of them have pushed me to think about how ethical commitments are forged by a range of micropractices, many of then narrative-based, and as they unfold institutionally.

> narratives that give it meaning, law becomes not merely a system of rules to be observed, but a world in which we live (1992: 95).

As Debian has organizationally matured, it has also concurrently matured along legal and ethical lines, codifying key principles in two related documents: the Social Contract, which lists a set of promises to the F/OSS community; and the Debian Free Software Guidelines, which clarify the legal meaning of freedom for the project. Developers continually draw on these texts to craft a dense ethical practice that sustains itself primarily via ongoing acts of narrative interpretation.

While the idea of *nomos* provides a useful framework for understanding how ethical stances are internalized, I specify its meaning by distinguishing among a repertoire of everyday micropractices that I group under two distinct (and contrasting) ethical moments: enculturation and punctuated crisis. While in practice these two moments exist in a far more complicated intermixture and copresence, here they are separated for the sake of clarity and analytical value. Each one tells us a slightly different story about how people use narrative to adopt values, animate them, and transform them over time.

By "ethical enculturation," I refer to a process of relatively conflict-free socialization. Among developers, this includes learning the tacit and explicit knowledge (such as technical, moral, or procedural knowledge) needed to effectively interact with other members of a project, as well as acquiring trust, learning appropriate social behavior, and establishing "best practices." Although ethical enculturation is ongoing and distributed, the most pertinent instance of it in the Debian project is the New Maintainer Process—the procedure of mentorship and testing through which prospective developers apply for and gain membership in Debian. Fulfilling the mandates of the NMP is not a matter of a few days of paper shuffling. It can take months of hard work: a

prospective developer has to find a sponsor and advocate, learn the complicated workings of Debian policy and technical infrastructure, successfully package a piece of software that satisfies a set of technical standards, and meet at least one other Debian developer in person for identity verification. This period of mentorship, pedagogy, and testing ensures that developers enter with a common denominator of technical, legal, and philosophical knowledge, and thus enter as trusted members of the collective.

The other ethical moment I investigate is crisis. As the number of developers in the Debian project has grown from one dozen to nearly one thousand, punctuated crises routinely emerge around particularly contested issues: matters of project transparency, internal and external communication, size, openness, the nature of authority within the project, the role of non-free packages, and the licensing of Debian. Many of these crises have an acute phase (usually spurred by a provocative action or statement) in which debate erupts on several media all at once: mailing lists, IRC conversation, and blog entries. While the debate during these periods can be congenial, measured, rational, and sometimes peppered with jokes, its tone can also be passionate and uncharitable, sometimes downright vicious.

During these moments, we find that while developers may share a common ethical ground, they often disagree about the implementation of its principles. Though the content of these debates certainly matters (and will be discussed to some extent), my primary focus is on the productive affective stance that these crises induce. I argue these are moments of assessment, in which people turn their attentive, ethical being toward an unfolding situation and engage in very difficult questions. In this mode, passions are animated and values are challenged and sometimes reformulated. Although these debates sometimes result in project stasis, demoralization, or even people leaving the project, they *can* produce a heightened and productive ethical orientation among developers. Crises can be evaluated as moments of ethical

production in terms of not only their functional outcomes but also their ability to *move* people to reflexively articulate their ideals—an important condition of possibility for further action. Such dialogical and conflicted debate reflects the active engagement of participants who renew and sometimes alter their ethical commitments. Thus, crisis can be vital to establishing and reestablishing the importance of normative precepts.

The main purpose of this chapter is to explicate how different instances of ethical labor define the cohesive yet non-unitary moral commitments that developers hold towards Debian and its philosophy of freedom. In order to do so, however, a brief introduction to Debian's history and structure will be necessary.

Thus, much of what is described in this the first half of the chapter is Debian's historical transition from an informal group (organized largely around charismatic leadership, personal relationships, and ad-hoc decision making) to a larger and more routinized institution, like the bureaucracies described by Max Weber. Most F/OSS projects in their infancy, including Debian, operated without formal procedures of governance and instead were guided by the technical judgments of a small group of participants. This informal technocracy is captured in a famous pronouncement by a hacker pioneer who helped develop the early protocols of the Internet: “We reject: kings, presidents and voting. We believe in: rough consensus and running code” (Dave Clark, MIT). Even though an ideal of “rough consensus” still exists in Debian today, Debian developers have had to demarcate membership criteria, explicitly lay out boundaries of project authority, and implement an advanced form of egalitarian voting in order to successfully expand the project without it disintegrating.

While charisma, ad-hoc actions, and personal relations still matter in the project today, these have been supplemented by other modes of formal governance. Through Debian's tremendous growth, developers have cobbled a hybrid organizational structure that integrates

three different modes of governance—democratic majoritarian rule, a guild-like meritocracy, and an ad-hoc process of rough consensus. It is thus unsurprising that many of Debian's crises result from the conflict between these three models. What is interesting is that responses to these crises are often used to clarify the purposes and limits of each form of organization.

Democratic voting reveals Debian's populist face; it is an acknowledgment that each developer is a valuable contributor to the project who deserves an equal say in its future. However, democracy and especially voting is often viewed as ineffective and improper method for resolving technical matters, because the mediocrity of the majority can overrule the “right” technical decisions.

For this reason, the normative ideal for technical decision making is a process of open-ended technical argumentation, where the process of debate, conducted over mailing lists, bug reports, and IRC channels, ideally clarifies the right solution and leads to enough “rough consensus” to proceed. In this mode, everyone is treated as an equal subject with the potential to convince others of the merit of a given technical solution, regardless of their status on the project. This approach represents a strong liberal affirmation of the importance of speech and debate in determining a non-partisan and technical resolution to collective problems (Habermas 1981 **Check year James. This refers to the Structural Transformation of...**).

However, this system of equal opportunity is also the basis for the hierarchies that inhere in meritocracies. Individual developers who over time come to demonstrate their technical worth through a combination of ability and dedication eventually obtain the status of a trusted technical guardian. A meritocratic system thus quite naturally arises that vests power in roles, such as delegates and various technical “masters” who hold power for unspecified periods of time. As the very names of their roles suggest, guardians are not unlike the guild masters of times past who held the trust and respect of other guild members because their wisdom, experience, and

mastery of the craft.

If democratic rule is sometimes treated with overt suspicion and dislike, there is a far more subtle fear concerning the importance of meritocracy and the meritocrats it produces; namely, the fear of corruption. Specifically, there is discomfort in the idea that the technical guardians could (as they are vested to do) exercise their authority without consulting the project as a whole and thus foreclose precisely the neutral, technical debate that allowed them to gain their authority in the first place. Debian developers express an incipient discomfort with the idea that delegates hold the power to hinder the egalitarian process of technical debate and consensus, even if they endowed with such power. This anxiety is voiced indirectly through the humorous acronym TINC, “There Is No Cabal.” It is manifested more critically when developers in positions of authority are seen to violate what I call “meritocratic trust”—the expectation that entrusted guardians act in good technical faith and not for personal interest.

Thus, the development of new ways of building trust between developers and the emergence of new organizational procedures have been central to the organizational growth of Debian and to balancing its modes of governance. The importance of building and verifying trust has been a recurrent theme in science and technology studies. Whether expressed through the trustworthiness of gentlemanly character, as was the case in seventeenth-century British scientific enterprise (Shapin 1994), or the transformation of books into transparent and vetted objects of truthful knowledge (Johns 1998), trust has been essential to bringing coherence, order, and stability to emergent social institutions, objects, and technical practices. Within F/OSS projects issues of trust are no less pressing (Kelty 2005 Trust Among the Algorithms" in CODE: Collaborative Ownership in the Digital Economy ed. Rishab Ayer Ghosh MIT Press 2005.), especially since they are being formalized for the first time in Debian (as elsewhere). Questions of whom and what to trust—whether delegates, voting procedures, a piece of code, a license, or a

guideline—are central to the repertoire of ethical practices that are the primary focus of this chapter.

In part one of the chapter, I provide a descriptive introduction to the Debian project history, social organization, modes of governance, and charters. This overview will help illuminate part two, where I more closely examine two facets of ethical labor in Debian. Each description of a moment begins with a series of theoretical issues and moves on to an ethnographic demonstration of how various narrative micropractices engender a common moral space.

Part One: Debian and its Social Organization

Brief History and its Social Organization

Debian is a project, made up of just over a thousand volunteers at the time of this writing, that creates and distributes a Linux-based operating system composed of thousands of individual software applications. As is the case with most mid- to large-size projects (i.e., those with over a couple of hundred developers), Debian is an extraordinarily complex institution that has undergone considerable changes over the course of its seventeen year history. In its nascence, Debian was run on an informal basis; it had fewer than two dozen developers who communicated primarily through a single e-mail list. Excitement, passion, and experimentation drove the project's early development. In order to accommodate technical and human growth, however, significant changes in policy, procedures, and structure occurred between 1997 and 1999. Now Debian boasts a complex hybrid political system; a developer IRC channel; a formalized membership entry procedure (NM); and a set of charters that includes a Constitution,

a Social Contract, and the Debian Free Software Guidelines (DFSG). Debian has produced detailed policy and technical manuals; controls development, testing, and mirroring machines located around the world; and manages bug tracking and collaborative software. It also produces a weekly newsletter, hosts a group blog, and organizes an annual conference.

The bulk of the volunteer work for Debian has always consisted of *software packing*—the systematic compartmentalization, customization, and standardization of existing software into one system. Taken together, these packages constitute the Debian Linux distribution. Along with package maintainers, teams of Debian developers (DDs) support infrastructure and develop Debian-specific software while others write documentation and others translate them into various languages. Each DD has at least one piece of software (and usually several) packages that she maintains. In local lingo, Debian is a "distribution," the unit of software is a "package," and developers are often referred to as "package maintainers."

Much of the work on Debian thus occurs in an independent, parallel, distributed fashion through informal collaboration IRC channels or on mailing lists where developers ask for and receive help.[2] Some collaboration is mediated by "bug reports." Written by Debian developers or users, these are filed on a publicly viewable Bug Tracking System (BTS). Incoming reports can identify a technical problem with a piece of software along with crucial details; they sometimes even provide the solution in the form of code to be incorporated as a "patch."

While a maintainer holds no legal ownership of the software he packages, Debian's norms of civility dictate that, within the boundaries of the Debian project, it is his responsibility alone. The assumption that maintainers enjoy near absolute control over their software package means that alterations should not occur without their explicit permission. However, if

2 One can maintain a software package that one does not actually code. The person who is the developer for the software is called "the upstream author." In many cases the upstream author and maintainer are the same, although there is no term to make this differentiation.

modifications are needed to eliminate a release-critical bug or to fix a security problem, there is a socially accepted protocol for doing so: the Non-Maintainer Upload (NMU). This mechanism is designed to allow a non-maintainer to upload a package in order to fix critical bugs or to compensate for a busy or missing maintainer. While many developers appreciate contributions provided by an NMU, seeing it as a handy mechanism to encourage co-participation, others find the protocol annoying or even downright obnoxious insofar as it allows inexperienced developers to insert shoddy solutions into their software. Other developers see it in a slightly different light, as a means of exposing poor work. As one developer on IRC told me half-jokingly, an NMU reveals “our laundry for public inspection.”

If much of Debian's packaging work occurs through individualized but parallel efforts, work assumes a more collaborative and populist tone during the period before the release of a new version of Debian: bug squashing parties are held more frequently and developers work round-the-clock on an IRC channel to resolve what have been identified as “release critical (RC) bugs” in the bug tracking software (BTS). During this period, which can last anywhere from a few months to over a year, Debian's release managers sanction NMU's with much less stringent criteria, and they are correspondingly more frequent.[3]

Among the hacker and F/OSS publics, Debian's fame rests on four developments, two technical and two social. Technically, it is one of the best-equipped distributions, offering more than 20,000 individual pieces of software, all with DFSG approved licenses. Relatedly, Debian can currently run on eleven hardware architectures—more than any other Linux distribution.

---

3 There is a more complicated etiquette surrounding how to proceed with an NMU that I am not able to address here (for reasons of space). There are policy guidelines that explain best practices and from time to time, the release manager can wave his magic wand (that actually is hard to wave as I will discuss shortly) to sanction NMU's for the sake of focusing attention to “release critical bugs” that must be fixed before a release can happen. For a telling example of the delicacy of this encouragement, see Anthony Towns' (the previous release manager), email entitled “It's Hunting Season” which opens with ... “or, _How to NMU and Get Away With It_. A LaMontiana Jones adventure.” http://lists.debian.org/debian-devel-announce/2002/01/msg00014.html .

This is one of the reasons Debian has earned the title of the “Universal OS.” On the social side, Debian holds the distinction of having more individual members than any F/OSS project. It is also known for its strong commitment to the ethical principles of free software, as elaborated in two key documents—the Social Contract and the Debian Free Software Guidelines. These documents figure prominently in the New Maintainer process I discuss in part 2. They are also the foundational texts that orient the project's sense of identity. It is via engagement with these documents that developers forge ethical commitments to the project and to free software more generally. Now let's take a closer look at the Debian charters and the governing structures that have emerged in the last decade.

Social Charters and Three Modes of Governance

In 1993, inspired by the Linux kernel project, Ian Murdock founded the Debian project because he was dissatisfied with existing distributions. He attracted a group of volunteers and they began to design a package management system that could integrate the contributions of *every single developer* who chose to maintain the package. I emphasize this point because it marked an important shift in the history of the Unix collaborative ethical temperament, one that explicitly honored transparency, accessibility, and openness for the purpose of facilitating and encouraging participation. It represented a new historical chapter in how hackers conceived of and implemented meritocracy.

One longtime DD, who came of age as a hacker well before the Linux era, powerfully captured the spirit of this shift in a short comment he made during a Debian history round-table discussion at their annual conference. He described how collaboration on Unix proceeded, before the Linux era, at Berkeley:

> There was a process by which you wrote some code and submitted in the 'I am not worthy, but 'I-hope-that-this-will-be-of-use-to-you supplication-

> mode' to Berkeley and if they kinda looked at it and thought, oh this is cool, then it would make it in and if they said, interesting idea, but there is a better way to do that they might write a different implementation of it.

While the Berkeley Unix gurus accepted contributions from those who were not already participating on the project, it was difficult to pierce the inner circle of authority and become an actual member of the team. This, from the point of view of the developers participating in the round-table, produced an unacceptable form of project participation, characterized by a degraded elitism that failed to equalize the terrain on which developers could prove their worth. As I argued earlier, the F/OSS hacker system of meritocracy compels individuals to release the fruits of their labor in order to constantly equalize the conditions for production, so that others can also engage in the life-long project of technical self-cultivation within a community of peers. When Linus Torvalds and Ian Murdock developed their own projects (the Linux kernel and Debian respectively), they did things differently than the earlier cadre of Unix hackers by fostering a more egalitarian environment of openness and transparency. Participation was encouraged and recognition was given where it was due, allowing developers to prove their worth through sheer talent. Accepting more contributions was also, of course, seen to improve and encourage technical efficiency.

Ian Murdock, who did the majority of the early work on Debian, acted as the project's leader. Debian in this sense was not unusual. Many of the early free software projects worked and still work along this logic of informal leadership by extension of a work-ethic charisma (O'Neil 2009 *Cyberchiefs*). Authority is established through the amount of sheer work one puts in the project; participants in return offer their loyalty to it. But what crucially distinguished Debian from earlier collaborative efforts was that everyone who technically contributed to the

project received membership. By 1995 there was a software package system in place that harnessed the power of individuality to produce a distribution that far exceeded the contributions of any single person.

In 1996 Debian grew in size to 120 developers and released a version of the operating system that contained about 800 packages. Around this time, Ian Murdock passed on the reins of leadership to Bruce Perens who, along with a handful of other developers, was responsible for the bulk of the project's technical work. Perens, already famous in the geek public sphere for both his passionate commitment to the principles of free software and his desire to make free software more visible in the business sector, had a marked impact on the organization of Debian, in ways recalled as both positive and negative.[4] Perhaps most important for our purposes, Perens helped coordinate the drafting of the Social Contract and Debian Free Software Guidelines, which were first suggested by a Texan, Debian Developer Ean Schuessler.

The genesis of the Social Contracts is worth briefly recounting, for it reveals the strong sense of responsibility and accountability to a larger commonweal of users and developers that has characterized the project since its inception (when Ian Murdock first articulated these values in the Debian Manifesto). Ean Schuessler came up with the idea for the Contract after a conversation at a conference with Bob Young, the co-founder of a then-emergent commercial Linux distribution, Red Hat. Ean suggested that Red Hat might want to guarantee in writing that as they grew larger they would always provide GPLed software. Young replied that “that would be the kiss of death,” implying that such a guarantee made to the users of free software could prove disastrous to his business. Ean (who was himself a business owner) was both amused and disturbed by Young’s answer, and with other developers at the conference he decided that it

4 He was considered too much of a micro-manager but was respected for giving so much time and dedication, especially in guiding the project through the creation of the SC and DFSG.

would behoove Debian to provide such a guarantee in writing.

If immediate inspiration for the Social Contract was a conversation that brought to light two divergent interpretations of accountability to a wider community of technical users, the time was quite ripe in Debian for the contracts' acceptance. As the project grew larger, many felt that the group had outgrown the Debian Manifesto. In particular many developers felt it was important to clarify their position on free software, for there was a small group of developers who were aggressively advocating that Debian should distribute non-free software or risk losing users to the other distributions that did so. Thus when the SC was proposed, it seemed like an ideal opportunity to clarify the project's goals to both outsiders and newcomers who were joining at an accelerating rate.

Led by Bruce Perens, who wrote large chunks of the document, developers produced a statement of intent that helped to define Debian's unique role within a larger field of production. A crisp and short document, the Social Contract makes four promises and gives one qualification:

> "Social Contract" with the Free Software Community
>
> **Debian Will Remain 100% Free Software**
>
> We promise to keep the Debian GNU/Linux Distribution entirely free software. As there are many definitions of free software, we include the guidelines we use to determine if software is "*free*" below. We will support our users who develop and run non-free software on Debian, but we will never make the system depend on an item of non-free software.
>
> **We Will Give Back to the Free Software Community**
>
> When we write new components of the Debian system, we will license them as free software. We will make the best system we can, so that free

software will be widely distributed and used. We will feed back bug-fixes, improvements, user requests, etc. to the "*upstream*" authors of software included in our system.

We Won't Hide Problems

We will keep our entire bug-report database open for public view at all times. Reports that users file on-line will immediately become visible to others.

Our Priorities are Our Users and Free Software

We will be guided by the needs of our users and the free-software community.

We will place their interests first in our priorities. We will support the needs of our users for operation in many different kinds of computing environment. We won't object to commercial software that is intended to run on Debian systems, and we'll allow others to create value-added distributions containing both Debian and commercial software, without any fee from us. To support these goals, we will provide an integrated system of high-quality, 100% free software, with no legal restrictions that would prevent these kinds of use.

Programs That Don't Meet Our Free-Software Standards

We acknowledge that some of our users require the use of programs that don't conform to the Debian Free Software Guidelines. We have created "contrib" and "non-free" areas in our FTP archive for this software. The software in these directories is not part of the Debian system, although it has been configured for use with Debian. We encourage CD manufacturers

> to read the licenses of software packages in these directories and determine if they can distribute that software on their CDs. Thus, although non-free software isn't a part of Debian, we support its use, and we provide infrastructure (such as our bug-tracking system and mailing lists) for non-free software packages.

This charter is a strong statement of intent concerning Debian's role, commitments, and goals, declared notably beyond the Debian project to the users of this distribution. It elevates the virtues of transparency and accountability and seeks to foster a commonweal that upholds the production of free software and the pragmatic needs of users. Although the charter affirms a well-defined moral commitment to free software and a community of users, it also formulates, in its last provision, the pragmatic limits to such “ideological” adherence, sanctioning to a limited degree the use of non-free software. In part, this decision reflected the state of free software at the time the SC was written, as well as an existing desire to ground Debian's shared moral commitments within technical pragmatism. At the time the charter was drafted, there were a number of important software applications, like browsers and word processors that simply had no robust free software equivalent. For example, while Netscape existed, and was free as in beer, it was not free as in speech; the source was unavailable for use, modification, and circulation.

Following the creation of free software equivalents to these programs over the years, Debian has routinely debated dropping its support of non-free programs; this has even led to a “General Resolution,” (resolutions voted by the entire project) to eliminate non-free. At the time of this writing, the most recent resolution in March 2004 re-affirmed Debian’s commitment to this provision, though given the voluminous debate it generated, it is an issue that I imagine will be revisited again in the near future. Drawing the line between pragmatism and utility, on one

hand, and ideological purity, on the other, is a task that Debian developers are constantly struggling with, as we will see in part two of this chapter.

Drafted primarily to clarify the meaning of free, the Debian Free Software Guidelines document is the legal corollary to the Social Contract. For a license to meet the standard of "free," it must meet the following criteria:

> **1. Free Redistribution**
>
> The license of a Debian component may not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license may not require a royalty or other fee for such sale.
>
> **2. Source Code**
>
> The program must include source code, and must allow distribution in source code as well as compiled form.
>
> **3. Derived Works**
>
> The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.
>
> **4. Integrity of The Author's Source Code**
>
> The license may restrict source-code from being distributed in modified form **_only_** if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software. (*This is a*

compromise. The Debian group encourages all authors not to restrict any files, source or binary, from being modified.)

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to Debian

The rights attached to the program must not depend on the program's being part of a Debian system. If the program is extracted from Debian and used or distributed without Debian but otherwise within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the Debian system.

9. License Must Not Contaminate Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be free software.

The DFSG both generalizes and specifies the GPL's four freedoms (access, use, modification, and distribution). It generalizes them so that the DFSG can act as a pragmatic standard to determine the relative “freeness of a license” or as the baseline to create a new license. At the same time, it specifies the meanings of freedom, largely by including an explicit language of non-discrimination, one of the document’s most salient themes. The DFSG has been assiduously excavated for discussion and debate on Debian-legal and other mailing lists to help developers decide whether a piece of software they want to package and maintain has a DFSG license, or what changes to an existing license need to be made to make it DFSG free (to be explored in the next chapter).

It is important to highlight that these texts explicate a set of moral precepts—transparency, openness, accountability, and non-discrimination—that are used to establish correct procedures for technological production, licensing, and social organization. Openness and accountability are something these developers expect from code (made possible by licensing) as well as project governance.

Along with these two seminal documents, Debian also has an explicit “Constitution” that was drafted after a failed first election[5] and in an effort to prevent the type of authoritarian leadership some developers identified with Bruce Perens. The Debian Constitution outlines in great detail Debian's organizational structure, which includes non-elected and elected roles and responsibilities. Contained within this document is a representation of Debian's overall system of governance—a combination of majoritarian democracy, meritocracy, and ad hoc consensus.

Debian’s democratic commitments are manifest in its voting protocols. Using a version of

5 Since, at the time, the procedures were not well established or clear, they ran into procedural problems so one leader basically dropped out of the race, leaving the other runner as the de facto DPL.

the Condorcet method (which guards against simple majority rule by means of a complicated ranking system), the project now votes every year for the Debian Project Leader (DPL), and any developer can propose a General Resolution relating to technical, policy, or procedural matters for a project-wide vote. These two provisions demonstrate the populist commitment to give all developers a voice and acknowledge that regardless of their level or quality of contribution, all developers, once accepted into the project, deserve some decision-making influence. This said, the DPL has assumed a decidedly non-technical function, and proposing General Resolutions related to any technical matter is fastidiously discouraged. Strictly technical problems are not seen as appropriate objects for democratic voting.

For example, in 2004 when one developer proposed a technically-based GR (calling for the support of a new architecture), on the mailing lists this suggestion was ripped to shreds and was thus effectively halted by many contributors, including some of the most respected and visible DDs. The response below conveys the distrust of political inclusion within the technical arena that many developers hold and will consistently give voice to:

> I won't even consider this proposal until you or someone else explains to me why we should use the voting system to decide an issue like this... If recent experience has shown us anything, it's that votes HURT Debian. Please don't take us further down this path.

Voting, in other words, blocks the methodology of open debate, the proper and most popular means by which technology should be re-visioned and improved.

If the DPL has a non-technical role, then what is he empowered to do? Most developers agree that the DPL acts as a public spokesperson at conferences and other events. Within the

Debian community, the DPL acts to coordinate and facilitate discussion, perhaps opening blocked pathways of communication or aiding in conflict resolution. The DPL's most significant power lies in his ability to assign or legitimate non-elected official roles in the form of delegates and teams. Typically these are composed of the technical guardians, who hold respect because of their superior talents and dedication to the project. These teams and delegates perform much of the Debian-wide work, such as administering mailing lists, accepting new members, running votes, and maintaining and integrating new software into the master archive. There is general faith that the DPL either legitimates teams already in existence because of the work they do or assigns roles to people already doing the work.

If it is incumbent upon developers to make decisions, there are nonetheless groups of developers who are empowered with special authority to make certain kinds of decisions, usually by virtue of holding non-elected posts as individual delegates or within teams or committees.[6] While the DPL can assign a DD as a delegate, and in theory is empowered to revoke any existing position, this action has never been taken within the last five years and possibly ever. This is significant. While in theory the DPL or a GR can revoke the position of a technical guardian, in practice this would never happen. Guardianships are vetted positions and there is a strong pressure to let them remain in their position so long as they are doing work and desire to remain there.[7]

These positions are technical in nature. Current teams include the release manager and team, the listmasters, the webmasters, the debian-admin team, the new-maintainer team, the security team, and the policy team. These teams coordinate to work on larger-scale

6 In addition to these more formalized positions, decisions made by informal ad hoc groups permeate the entire organization.

7 This is a complicated topic but it is worth stating that one of the reasons that the DPL is discouraged from changing positions is because it smacks too much like politics (i.e. brining your own cronies at the expense of those already doing a good enough job).

infrastructural or organizational structures and procedures. Important among these are the ftpmasters who existed before there was a DPL assigning teams; they review by hand all new submitted pieces of software packages for technical and licensing glitches, integrate them into the "master archive."[8]

It is said that the people who hold these non-elected positions do so because they initially undertook the work necessary to accomplish the tasks of the position. For example, some ftpmasters hold their position because they coded the software used to handle package uploads and verification or the package repository software. Power, in other words, is said to closely follow the heels of personal initiative and its close cousins, quality technical production and personal dedication to the project.

Even if most developers prefer meritocracy to democracy—in fact nearly every developer interviewed stated with pride that Debian is meritocratic—this form of power is nonetheless shrouded in some level of distrust. Positions of authority, like the ftpmasters, undeniably represent a form of centralized and lifelong authority, potentially subject to corruption or—just as dangerous—knowledge specialization and hoarding. Hackers generally tend to honor decentralization and the distinct power of the individual to trump authority, so *any* centralized authority is bound to act like a lightning rod for reflection and debate.

However there is a more specific reason for this distrust. To fully appreciate the cultural texture of this controversy over authority, we must revisit the argument laid out earlier. Debian developers operate within a social imaginary rooted in a Millian conception of individualism that requires them to cultivate their skills, improve technology, and prove their worth to other hackers within their elite fraternity. Figures of central authority, such as team members and delegates,

8 ftp refers to file transfer protocol and used to be the main method by which developers uploaded pieces of software to the repository. It is no longer the case that this is the only method used but the name ftpmaster stuck.

represent a potential threat to the conditions for this perpetual process of technical self-fashioning. As Wendy Donner argues in her discussion of Mill's model of self-development, those who gain authority because of merit nonetheless “can only act as guides to others,” never as “authorities.” If they attempt to impose “judgments of value on others,” this “paradoxically undermines that claim to development.” (1991: 152). Everyone, not a select order of people, must be able to exercise their capacity for thought, discrimination, and critical intervention and at all times.

The anxiety that power could *potentially* corrupt those who enjoy privileges and block conditions for public self-development (by making choices) emerges from time to time, although in a less coherent and sustained fashion than the critiques of Debian’s democratic elements. It is far more common to joke about the existence of what is called “the cabal,” usually stated as a denial “There is *NO Cabal*” (TINC). Long before Debian existed this was a running joke on Usenet where a similar discomfort over the potential for corruption by the meritocratic leaders played out (Pfaffenberger 1996 **James this is in the Annual Review piece)**.[9]

In Debian joking enjoys a wide purview, and playful joking about the cabal is littered everywhere. For example, the evening before the 2005 DPL winner was to be announced, a group of developers, including a DPL candidate and the release manager, casually slipped into discussion jokes about the cabal, unprompted by anything except the announcement by the project secretary that there were 24 hours to go until the DPL voting period was over:

<Markel> less than 24 hours to go

<Crawlspace> cue sinister music

9 See Usenet Cabal FAQ. http://www.subgenius.com/bigfist/hallscience/computers/X0012_Internet_History.html

> <mickmac> I expect it all to be a conspiracy
>
> <mickmac> The cabal will already have chosen their candidate
>
> <JabberWalkie> mickmac: Nah, there's still time; got your last-minute bribes ready? ;)
>
> <mickmac> JabberWalkie: Well, uh, no.
>
> <mickmac> Damnit.
>
> <JabberWalkie> mickmac: Better luck next year. :)
>
> <mickmac> I can offer you beer if you come to Debconf?
>
> * vapor-b shakes his fist at the cabal

Developers use cabal joking to express chronic anxieties about the general corruptibility of meritocracy. Most of the time, these jokes are used playfully. They work like a safety valve to diffuse tension and are used as a creative and oblique reminder to those in positions of authority that their intentions must be transparent for them to receive the continued trust of the developer population.

At other times, developers couch discussions of the cabal as accusations, seeking more trustworthy behavior from the meritocrats, which usually means claims for greater transparency, accountability, and accessibility. In the recent DPL debate, for example, one developer argued with an ftpmaster over what could be done to increase the project's transparency and equalize the conditions for access. The developer invoked the specter of the cabal:

> I see Debian as a meritocracy, and the way to receive privileges is to contribute and be pro-active. However, it cannot be the goal to expect from willing users to figure out everything about a job all by themselves prior to being able to gain recognition for the

> contributions they make -- if they are lucky enough to be considered
> useful by current holders of the position strived for. If this is
> actually intended, then it is highly inefficient. If it is not
> intended, then maybe Debian wants to do something about it, and if
> not only to stop cold those rumours about an alleged cabal.

This developer felt that Debian developers could do more to increase transparency in order to facilitate and encourage participation from new members. Many of the developers he was arguing with (those in positions of power), however, disagreed, saying that there was enough transparency and that it is incumbent for interested members to take responsibility for *their own self-education,* independent of the help of others. This is conveyed in the e-mail below, where one ftpmaster responded to the claim that Debian policy and organization is too obscure:

> > What you fail to see is that there is something daunting about
> > a project of this size and complexity to those who are trying to
> > understand it top-down, rather than having been part of building it
> > bottom-up.
>
> What you fail to see is that the bits are available and that you
> "only" have to build the large picture. If you're too lazy to do so,
> it's not the job of the people working on essential corners of the
> project to educate every random Johnny Sixpack for the sake of it.

Even when there is a pressure to equalize conditions for access, which is manifested in jokes about the cabal, equalization within a meritocracy, as I discussed in chapter 4, must proceed by

very specific methods. Many (though not all) developers feel that if too much help is given to newcomers, it will undercut their ability to prove their worth and intelligence within a group that values precisely this sort of performance of self-reliance. The line between the equalization of conditions and too much help is constantly being negotiated in Debian, perhaps more so than in other projects because of its populist bent.

Debian's meritocratic guardians find themselves in a paradoxical position with respect to hackers who accord tremendous weight to liberal individualism, in particular to constant acts of technical self-fashioning and the open-ended process of non-partisan technical debate. Granted the authority to act without the prior consent of the community, the guardians rarely can do so without displaying good and pure intentions. In this way, these developers, such as the early scientist-philosophers of Britain's Royal Society, studied by Steven Shapin (1994), must constantly garner the trust of peers through the performance of character virtues and other related acts.

If the natural philosophers of the Royal Society revealed good faith through a combination of humility, detachment, generosity, and civility, how do the meritocratic guardians of Debian perform their good intentions and navigate this paradox? Delegates and teams manifest their pure technical intentions through a wide range of practices (and humility is not always one of those). Many can display their intentions simply through ongoing technical work, a form of labor that speaks to their unwavering commitment to the project. They are supposed to communicate openly with developers, and in recent times, some developers voiced their dissatisfaction clamoring for more transparency and accountability from a few of the delegates. These arguments follow a predictable arc: some developers complain about an improper lack of transparency among the guardians; on the other hand, the guardians feel suffocated under the weight of their obligations, so that the work necessary to communicate and increase the

transparency of the role becomes an impossible and unnecessary additional burden; then, the developers will offer to help share the work-load; and typically the guardians' exasperated response is that to integrate and train new people would require more work that can now be taken on or to ask the developers to take the initiative themselves.

Nonetheless, there are some established routines for increasing the visibility and transparency of different working groups, including e-mails sent out to all developers on the debian-devel-announce mailing list (a required subscription for developers) that summarize the most recent activities of the different technical teams. These informal updates are sent periodically to members of the project and forge a connection with the body politic of Debian. One ex-ftpmaster, James Troup, known more for his incredible technical prowess and dedication than for his communicative access, once sent such a status report entitled "Bits from the ftpmaster team" after there had been several months of arguments on the mailing lists about the opacity of the ftpmaster's exact role and complaints that the ftpmasters were becoming roadblocks to the project in failing to fulfill their duties. His e-mail, drafted by a number of team members, announced the addition of new members to the team and provided some clarification over the exact responsibilities of each member. Following this information, James Troup and the other ftpmasters ended the email with a humorous, bitingly sarcastic remark that asserted their good while discounting the complaints as overblown: "We hope this has made your day more pleasant, and your nights less filled with the keening wails of the soulless undead." Its ironic and sardonic subtext is clear: he has heard and registered the complaints, is humored that people thought that the situation was so hellishly torturous, and is revealing that there is nothing to worry about, he acts in good faith and this unnecessary email serves as his merciful act of grace releasing souls from the unbearable suffering that they were experiencing.

Along with active communication, delegate technical proposals must be carefully framed

to reflect project and not personal goals. In many instances, it is imperative to let certain decisions be made through an ad hoc, consensual process in which the merit of the outcome emerges via a processes of collective debate rather than as a mandate from those with vested authority (even if the outcome falls entirely within their purview).

An interesting site to examine is the committee invested with the greatest technical power: the Technical Committee. Its role is defined in the Constitution as "the body which makes the final decision on technical disputes in the Debian project,"[10] and its members perform their good intentions largely by way of inaction. For example, in the period during which I followed Debian (2000-2005), this committee rarely exercised its authority. At the time of this writing, the last decision it made was in June 2004, clarifying the name of a technical architecture.

A hands-off approach is thus how committee members performatively establish their "good intentions" toward the very process that afforded them the right to become part of the TC. It reflects the general ideal that those in authority should first defer to the developer community so that differences of opinions can be solved through debate and consensus. The Technical Committee website enunciates this idea clearly under the heading "Some caveats about contacting the committee:"[11]

> A sound and vigorous debate is important to ensure that all the aspects of an issue are fully explored. When discussing technical questions with other developers you should be ready to be challenged. You should also be prepared to be convinced! There is no shame in seeing the merit of good arguments.

10 http://www.debian.org/devel/tech-ctte.

11 See http://www.debian.org/devel/tech-ctte.

The argumentative consensus advocated by the TC is the third mode of governance in Debian, a mode that is understood as a form of self-governance because it stems from the debate, contributions, and actions of independent-minded, consenting individuals. A tremendous faith is placed in the power of what we may call "technical rhetoric" to convince others of the logic of decisions that have been made. Technical rhetoric is about technical work and frequently includes a presentation of the code, a corollary written statement, or a justification as to why no change should be made.

These debates happen on IRC, bug tracking software, and mailing lists; on IRC, the process of argumentation is informal. Developers usually seek the advice of others and move on to do the work. Many times such advice seeking produces robust debate, and when there are especially pronounced differences of opinion, this transforms civil heartiness into vibrant, sometimes vicious flamewars—outbursts of dissent that are characterized by inflammatory language or direct accusations of incompetence. The Debian Bug Tracking System is another site were technical jousting happens, and since there is a formal system that allows developers to rate bugs according to a spectrum of severity from wishlist to severe, these debates can be tracked with more systematicity. The attention a given bug received can be easily tracked by the length of debate contained therein, the multiple reassignments of different levels of severity, closing, and reopening. Some bugs debates have reached legendary status because of multiple reassignments of severity, their length, and their lack of closure. It is often during such debates over technical questions that developers most explicitly raise issues of authority and renegotiate the lines between democracy, consensus, and meritocracy that define their system of governance.[12]

12 To contain the sprawling size of this chapter, I am not providing an example of one of these legendary bugs or how debate over technical issues becomes a place where questions of authority are raised. This must be left for another time. However, for two such legendary Debian bugs see http://bugs.debian.org/cgi-

Part Two: Two Moments of Ethical Cultivation

In terms of its governance alone, Debian is clearly an extraordinarily complex moral and technical environment. It should come as no surprise that the way in which new members integrate themselves within this community, and learn the proper codes of conduct and procedures by which to contribute effectively to the project, and gain the trust of other developers, is not a simple one. Although they prefer coding over organizational building, Debian Developers have, nonetheless, concocted an interesting social solution to this problem of integration and trust-building, called the New Maintainer Process. This process is a social solution for the scalability problem: how to build trust and encourage accountability in the space composed solely of bits and bytes and a growing number of participants.

The Debian New Maintainer Process: Trust and Cohesion in Virtuality

Theoretical Issues

The Debian New Maintainer process is a procedure designed to serve as a bridge between the textually codified moral vision of the Debian project, and the individuals who come to work on Debian. It is meant to ensure that a baseline level of ethical commitment and technical

---

bin/bugreport.cgi?bug=97671 or http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=143825. The first one was over an obscure file that violated the FHS (filesystem hierarchy standard a policy detailing where pieces of programs should be placed in the filesystem) and whether it needed to be fixed in time for the impending release. This turned into a very lengthy, acrimonious debate between a famous package maintainer and the release manager, which had as much to do with whose decisions should stand, the maintainer or the RM. As the maintainer wrote at one part of the voluminous writing dedicated to this one bug, the crux of the issue boiled down to: “The issue is for me is twofold: about the package maintainer being empowered to manage his own bug list for triage purposes, and about the limits of the Release Manager's or another developer's power to make decisions for another developer.” The Technical Committee refused to hear the case, the RM did not budge on his recommendation, and finally the maintainer proceeded to ask for help to fix the problem.

proficiency is shared among a globally dispersed and culturally diverse group of developers.

Essentially a gateway for new members, the New Maintainer Process defines what is morally and technically expected of them. As such, it works powerfully as a centripetal force of ethical enculturation. It is the framework within which new members first confront the socio-political and organizational milieu of Debian. This process represents the first time that many new members meet another Debian developer in person or “ethically voice” their commitment to free software through the prolific writing that is required of them.

Already explicitly committed to a vision of free software, Debian is to some degree a self-selecting organization, unlikely to attract programmers with a staunch commitment to upholding the current status of intellectual property law. But the NM process is unique insofar as it requires prospective Debian developers not only to study detailed texts on the ethics of free software (such as the GPL, the Social Contract, and the Debian Free Software Guidelines) but to also produce their own texts on the subject. Through the NM process, developers become producers of ethically relevant discourse. The extensive narrative work of the NM makes Debian's codified values personally relevant, and this in turn breeds social commitment to the project.

New Maintainer Process

When the Debian project was young, it was a relatively close-knit group. Most project members knew one another and interacted on a frequent basis with most of other active members. Prospective members needed only to informally demonstrate their technical competence and to claim knowledge of and adherence to the project’s Social Contract and policies before being admitted to the project.

As Debian became a much larger organization, the project found itself in a crisis that peaked between 1998 and 1999. New members were being admitted at rates faster than the project's ad hoc social systems could integrate them. Some long-time developers grew skeptical of the quality of incoming developers, complaining that they introduced more bugs into the system than helpful contributions. The populism of open membership began to come under attack. Some developers suggested that Debian had reached its saturation point.[13]

In response to these problems, the DAM (Debian Account Manager who creates accounts for new members) waged a silent revolt by halting the processing of new maintainer accounts—essentially preventing any new members from being able to join the project. This move eventually led the DAM to officially stop accepting members under the informal procedures. Instead, he proposed formalized procedures that could systematically ensure the trustworthiness of new members as members of the community.

The procedure developed was Debian's New Maintainer (NM) process. First presented in October 17, 1999, as a proposal on a mailing list, its preamble (quoted below), indicates that the "growing pains" Debian had been experiencing were not merely technical but also ethical. A small group of developers had been clamoring to loosen the commitment to free software and integrate non-free software in order to be competitive with commercial distributions whose ethical commitments were less stringent. Even though the constitutional charter was already established, the spirit of free software was seemingly losing its potency with the addition of each new wave of developers. The DPL listed the following criteria by which to select new members, and note that the first line is intentionally repeated:

13 Many of the conversations over this dis-ease occurred over the private-mailing list and thus I am unable to quote any of the exact conversations. The mailing list is only supposed to be used for sensitive material, for example, for announcing when a developer will be on vacation. From time to time, some of the more interesting discussions unfold there and then someone gets the bright idea to move it to a public list. I have been told about many such conversations and some individuals have allowed me to see their own personal posts to get some clarity over the issues.

- needs to have a *strong* oppinion (*sic)* for free software
- needs to have a *strong* oppinion (*sic*) for free software
- he needs to be able+willing to make long distance phone calls [for an interview]
- He needs to know what he's doing, that new people need some guidance, we have to prevent ourselves from trojans etc.
- we need to trust him - more than we trust *any* other active person
- He *has to* understand that new-maintainer is *more* than just creating dumb accounts on n [n being a numerical variable] machines

Out of this initial proposal and the establishment of a NM team, the NM process created a standard that all developers must meet. NM is structured not only as a test but as a process for learning, mentoring, and integrating prospective developers into the project by making them work on packaging a piece of software closely with at least one older, “trusted,” project member.

Before prospective developers formally enter the New Maintainer process, they are first asked to identify the contributions they plan to make to Debian. They are encouraged to demonstrate their commitment to Debian, to express why they want to join, and to display some level of technical proficiency. For most developers, this involves making a software package and—because only existing developers can integrate a piece of software into the larger GNU/Linux distribution—finding an existing developer to “sponsor” their work. New maintainers work closely with their sponsors, who check their work for common errors and take partial responsibility for the new maintainer. This supervision is important because, in addition to gaining technical skills, the new volunteer begins to participate in the social sphere of the project. Prospective developers are encouraged to join mailing lists and IRC channels that

provide the medium for technical and social communication.

A NM's sponsor often acts as the new applicant's advocate when the maintainer applies for membership in the project. Advocates are existing developers who vouch for new developers and their history of, and potential for, contributions to the community. Early in the NM process, the issue of establishing trust is crucial. After the applicant's advocacy has been approved, they officially become part of the New Maintainer process and an application manager is assigned to guide the new developer through the remainder of the process. It is this manager who handles the rest of the new maintainer process by acting simultaneously as mentor, teacher, examiner, and evaluator. While it is certainly the case, as phrased by one DD, that a "village idiot can't join Debian," this mentorship is the NM process's implicit concession to the limits of a meritocratic imperative that would otherwise require a person to prove her worth entirely on her own.

The NM process consists of three steps, and each may last several months and requires considerable work on the part of applicants. The three stages attempt to ascertain and confirm the new maintainers' identity, knowledge and position on free software philosophy, and proficiency with the established Debian policies and procedures as well as their overall technical expertise and knowledge. In terms of the social cultivation of trust and ethics, the first two steps are of particular importance within the Debian project.[14]

The act of proving one's identity is accomplished by obtaining *the cryptographic signature* of at least one existing Debian developer on their personal GPG key. A GPG key is encrypted with a pass phrase that is known only to the person holding that key. When properly

14 While I emphasize some of the ethical and social elements of the NM process, it is important to note that it is just as much a method of displaying technical proficiency as well as a process of technical mentoring. In the final step of the new maintainer process, applicants usually demonstrate that they have the technical wherewithal to be trusted with the ability to integrate software into the Debian archive and to represent Debian to the world. This test is often filled with the presentation of a clean, policy compliant, and bug-free example of the type of work the applicant aims to put within the Debian distribution (e.g., a package) although this is complemented by a significant series of technical questions.

unlocked with the pass phrase, a signature can be generated which can then be attached to a particular e-mail message, text, or piece of software. With an attached signature, it is proof that it originated from the person possessing the key. When key owners meet in person, they establish their identity to each other by exchanging pieces of government issued picture identification and the key fingerprint, which uniquely identifies the key itself. Having traded and verified this information, developers later place their unique cryptographic “signature”' on each other's keys to confirm to others that they have connected the key being signed with the individual in possession of those identity documents. This is a process of identity verification that can then be used over the Internet to confirm, with certainty, that an individual is who they say they are.

Because nearly every hacker within Debian has a key signed by at least one existing developer, and because many developers have keys signed by numerous others (the stronger the connected set of signatures is, the more trustworthy it is considered) nearly all maintainers are connected by what they call a cryptographic “web of trust.” In the manner of the famous “six degrees of separation” model, Debian can use cryptographic algorithms to prove that most every developer met at least one other developer, who in turn met at least one other developer, etc., until every developer is connected. Debian's administrative software depends heavily on these keys to identify users for the purposes of integrating software into the distribution, for controlling access to machines, for allowing access to a database with sensitive information on developers, and for restricting publication to announce-only email lists.

By forcing new developers to obtain the signature of an existing DD, the NM process integrates them into this web of trust. The importance of meeting in person and acknowledging each other's existence in a manner that is not solely electronic or virtual is illustrated in the following anecdote, which begins with the controversial claim of one DD:

> I have a potentially controversial thesis.

> My thesis is that the "Raul Miller" who is a Debian Developer and sits on the Technical Committee (and was, for a time, its chairman) doesn't actually exist.
>
> You see, Mr. Miller joined the project before we had the current procedures for vetting developers' identities, and even before we had the semi-informal ones under which I myself was admitted to the project in 1998. Interestingly, there are no signatures on Mr. Miller's PGP key other than his own. A remarkable accomplishment for someone who's been with the project this long, but not so surprising for someone whom no other developer has, as far as I can tell, ever claimed to have met in person.

When it became clear that Raul Miller who occupied an important technical position in the project at that time, was outside of the web of trust, there was such an alarmed reaction that within three days, two developers drove to meet the individual in question and succeeded in bringing him into the cryptographic web. The developers' strong reactions demonstrated the *essential nature* of these infrequent face-to-face interactions and the importance of verifying the identity of one of their technical guardians.

Integration into Debian's web of trust is thus a vital first step in new maintainers' integration into the Debian project. This process connects and leads into the second, and often the most rigorous, part of the NM process: Philosophy and Procedures. The first part of the application requires that new maintainers provide a declaration of intention, a proof of some contribution or skill they could bring to the project, and undergo the Philosophy and Procedure

testing procedure which includes a biographical narrative of why and how they became involved in free software and Debian.

During this “philosophy” step of the NM process, application managers ask prospective developers a series of questions regarding Debian and Free Software philosophy. While general knowledge of the definition and philosophy related to F/OSS is essential, the questions revolve around Debian's Social Contract and the DFSG. New maintainers are asked a series of questions—some are culled from a standard template, others vary depending on application managers—to demonstrate their familiarity with these documents, their ability to apply and synthesize the concepts encapsulated within them, and to articulate their commitment and agreement with Debian ideals.

Although each NM must agree to the Social Contract, the point of the philosophy test is not to ensure that developers have a homogeneous viewpoint on free software. Rather it seeks to ensure that all Debian developers are knowledgeable of, interested in, and committed to its basic principles. Open-ended questions often turn into longer e-mail conversations between application managers and NMs. While I have heard some developers complain of the “wait” and “bureaucracy” introduced by the NM, or even the absurdity of some of the technical questions, I have never heard an objection levied against this part of the application. In fact, most developers recall writing the philosophy section as enjoyable and rewarding.

In particular the initial biography allows developers to take an inventory of their technical past, in a way that starts to imbue it with a decidedly ethical dimension. Here, it is worth quoting large blocks of an application to the reader give a sense of the remarkable detail and ethical nuance these documents provide. Below I quote a very short section from a developer who is answering the biographical question:

This is my story about free software:
In the first times I was excited by the idea of something to which everybody could contribute, just like that Internet that I was discovering at the same time. I could also see that it had a future, because of that part that said that all the contributions would remain free. Wow!

At the same time, I was seeing many closed softwares rise and fall (DOS, Windows 3.x, OS/2, compiler environments, BBS software, Office suites, hardware drivers, proprietary format backup suites, whatever), and everytime they got superseded by some other thing, support fell, bugs remained unfixed, data became unreadable and nobody could do anything about it except spending lots of time and resources relearning everything and porting or even restarting their works from scratch.
[...]

I realized that Free Software was and is the only thing that potentially allows you to be free of the risks that (usually silly) external events pose on your know-how and on the software that you depend on.
[...]

This is about me and my time for Debian:

Between five and six years have passed running Debian, and my experience

> with it has grown. I got used to the Debian phylosophy, did some experience with the BTS, read some mailing lists, the DWN, got curious and
> somewhat knowledgeable on how Debian works, read pages, policies, discussions, I even went to the LSM and Debian One.
> [...]
>
> The packages that I used to create, however, were not perfect, and I would have needed to better study the various Debian policies and manuals to do some better job. Willing to do that, I thought it was silly not to become an official maintainer, and start contributing to the project myself, so that others could take advantage of that knowledge I was about to acquire...

The first section usually sticks to a standard technical life history, gesturing toward the ethical uniqueness of F/OSS; however it is told in a mode closely hinged to practical life experiences with technology. The applicant featured above writes, “I realized that Free Software was and is the only thing that potentially allows you to be free of the risks that (usually silly) external events pose on your know-how and on the software that you depend on” and then goes on to provide a sense of how he arrived at Debian. Often told in a confessional tone, the essay is as much a biography about not only one's discovery of this specific project but also how one arrived to the principles upheld in Debian itself.

The philosophy aspect of the Philosophy and Procedures section continues to cover the

Social Contract and DFSG, and it is here where ethical voicing becomes strikingly pronounced. In contrast to the descriptive register of the biographic section, here prospective developers are required to formulate their personal views on free software, moving from personal experience toward reflective generalizations regarding the legal and ethical principles they are committing to in joining this project. First let's visit the text before commenting on its implications. Below is, again, a *partial* excerpt from a different application than the one quoted above. This applicant, responding to the question “Please explain the key points of the Social Contract and the DFSG - in your own words,” writes:

> The Social contract is the commitment the Debian project makes to its members and users. It is about fostering a community so committed to software freedom, so open, and so supportive that no one would have need to go elsewhere.
> Debian's members and users benefit from the fact that Debian is completely free software and that nothing in the Debian process is hidden from them.
> Debian also provides an outlet for new free software and a "channel" for contributing changes back to the original developers. It also takes the realistic view that some users may still be using non-free software and that providing this software actually helps Debian's users and indirectly the free software community.
>
> The Debian Free Software Guidelines is the set of concrete rules that help determine if a piece of software complies with the Social Contract and the

Project's goals. The rules in the DFSG are chosen to make sure software accepted into Debian maintains the user's freedoms to use, distribute, and modify that software now and forever. This is not just for Debian's users but anyone who might take software in Debian and modify it, create CDROMs, or even create a derivative distribution

[...]

> Also, describe what you personally think about these documents.

The Social Contract and the DFSG represent a very unique idea. In this day and age where society (at least in the US and some other first world countries) encourages individualism and tries to divide the people and control them it is very refreshing to read the Debian Social Contract. Proprietary software made by commercial software companies/developers is exactly that, commercial. Those companies/developers are only about profit or advancing their agenda and will do what they need to in order to maximize that. Often this conflicts with doing the right thing for the user and here are some examples,

- - If a company/developer sells a piece of proprietary software that, as all software does, has bugs and they also sell incident based support contracts then what incentive do they have for fixing bugs in the software?

- - If a company/developer's revenue stream is based on selling new

versions of their proprietary software, what incentive do they have for fixing bugs in the old one rather than forcing users to pay for a new upgrade they may not want.
[..]

- - Imagine a company/developer that develops a proprietary application that initially meets the users needs so well and is priced reasonably that they gain a monopoly on the market. With the competition gone they can raise their prices or bundle additional unwanted applications into their software, or do pretty much anything they want.

- - Now imagine a company/developer that uses that monopoly in one market to gain entry and into other markets and attacking users' freedoms in those as well.

In all these examples the company/developer benefits and the expense of the users.
In order to prevent situations like this one of the things the Social Contract/DFSG effectively says is this,

"Our users are so important to us that we are setting these ground rules to protect their freedoms. If you can develop software that meets these rules then not only do we invite you to include it in Debian, but we accept you into the our community and will expend our resources to distribute your

> software, help keep track of bugs it may have and features that could be added, and help you improve and support it."
>
> In addition to that statement several of the DFSG's clauses have the effect of saying,
>
> "We are so committed to doing the right thing and working together with anyone in an open manner to resolve differences and always do the right thing for the users that we're willing to let you have all the work that we've done. You can do whatever you wish with it as long as you obey the original author's license."
> [...]
>
> These are very powerful ideas that can't be taken away or subverted by someone who wants to extort or control users. Personally, I think my interest in getting involved with Debian is an extension of my overall views and this is the only way that I want to use and develop software.

While the content of this narrative certainly matters, here I want to emphasize the type of ethical labor being produced by this text. The developer takes the vision ensconced in the Debian charters and “adds value” to it in numerous personalized ways: he reformulates its key principles in his own words; to hone down his points, he makes a fairly sophisticated contrast between proprietary software development and free software development largely along the ethical lines that matter to him—transparency, openness and accountability; and he poignantly concludes with

a succinct commitment to this style of development.

What we see here with these NM narratives is what Robert Cover, in his discussion of a *nomos*, describes as a simultaneous process of subjective commitment to and objective projection of norms, a bridging that emerges out of a narrative mode. “This objectification of the norms to which one is committed frequently,” Cover writes, “perhaps always entails a narrative—a story of how the law, now object, came to be, and more importantly, *how it came to be one's own*.” (1992: 145, emphasis added). This is precisely what occurs during the NM process. Developers affirm their commitment to the principles enshrined in their key charters largely by way of a specific gravitational pull: the force of their life experiences is brought to bear directly onto these documents, in the process rendering them objectively real but in a way that subjectively matters within one's personal orbit of life experiences. This is followed by a second move, one that betrays subjective personalization. Developers voice the broader importance of these principles and clarify the social implications of their commitments. Neither purely subjective nor objective accounts, they sit between them.

These narratives are at the basis of temporal movement and personal transformation. They take people to new locations, and past, present, and future come together in a moment of ethical assessment. The past is weaved into the present, and the voicing of commitment in the application becomes the path toward a future within the project. It is a step that brings a developer closer to a new social localization within a larger ethical and technical project of developers who have also undergone the same reflective exercise.

Through this reconfiguration of temporality, developers after the NM can be said to share at least three connections: they are technologically connected through the web of trust that actually requires them to meet at least one another developer; they share the experience of a common ritual of entry; and finally, they will have started to learn a Debian-specific vocabulary

with which to situate themselves within this world, formulate the broader implications of freedom, and continue the conversation on freedom, licensing, and their craft, with a wider body of developers.

Although the Philosophy aspect of the New Maintainer process often results in voluminous expository output, it is by no means the bulk of the process; in fact, it is only half of step three of a five step process. The other half of the Philosophy step is known as “Procedures,” in which applicants must demonstrate how and what the general policies are as well as their ability to perform whatever individual responsibilities they wish to take on within the project itself. Once the “Philosophy & Procedures” step is deemed appropriately passed, the applicant moves on to the rigorous “Tasks & Skills” step. This step confirms that the applicant has the necessary skills to carry out the job that they will take on as a Debian developer. These tests vary depending on what the applicant will be doing, but they typically involve many highly technical questions. The overall process will typically result in several dozen pages of exhaustive responses and many back-and-forth discussions and clarifications over months (sometimes up to a year) with the assigned Application Manager.

If accepted in the project, some developers slip into relative obscurity. Some do not actively participate in Debian's very dynamic culture of debate and discussion. Some follow only as spectators, while others could care less about these dramas. But for most developers, in ethical terms, the NM is a highly condensed version of what flows and follows in the social metabolism of the project, though in a slightly altered version. The narrative work that transforms codified norms into meaningful ones continues within the project itself, and the knowledge gained during the process is necessary for newcomers integrate effectively in the project.

Interactions among developers in the ongoing debate tend to be less concerned with the nature of the principles they committed to in the NM than with the *implications* of these

principles. In other words, common principles start to diverge into a multiplicity of newly generated ethical meanings, some of which alter the basic procedures and structures of the project. Even if their interpretations of principles diverge, developers often refer to the charters or shared precepts in debate, and thus these precepts are kept actively relevant. Divergence and disagreement is thus the basis for moral coevalness. Let's now turn to one such moment of group crisis.

The Difficult Labor of Ethics: Crisis and Ethical Renewal

Theoretical Issues: Crisis and Situational Ethics

Punctuated moments of distrust and despair are responsible for a great deal of the existing framework of Debian itself. Crises occur when there are fundamental disagreements over some issue. These can range from governance to legality, but many consistently revolve around a limited set of themes: project transparency, major technical decisions, the meaning and scope of freedom, and the relations between ordinary developers and those with vested power. These grievances are expressed on mailing lists, IRC, and blogs; the writing that unfolds during moments of crisis is both voluminous and markedly passionate.

These punctuated moments are eminently precarious: the *nomos* is under threat, populated by all sorts of pitfalls and dangers. The drama of dis-ease can spread uncontrollably like a virus, channeling the potent energy of dissatisfaction into a pit of destabilizing disgust or despair. Tempers flare, leading to inflammatory remarks that burn bridges; and people sometimes cling too literally to codified norms, blinding them to a unique situation that yearns for its own unique response. The crisis may be of such great magnitude that it overshadows the positive energy that moves the project towards a solution.

Despite their riskiness, however, periods of crisis are also among the most fertile

moments of ethical production, articulation, and transformation; their mere expression is proof that people are ethically "on call." People would not be willing to take sides if they did not feel personally invested in changing what is collectively diagnosed as a problem. Crisis periods are incipient calls for movement and realignment, and thus reveal commitments that, if acted upon, can lead to positive solutions and a profound renewal of the organization.

The formal attributes of crisis—its drama, high pitched emotional nature, and kinetic energy—has an ethical subtext that speaks to the fact that an altered situation or unsatisfactory event has arisen that demands immediate and overt *attention.* It demands a response, one that a charter or code cannot fully provide but must rather be sculpted through a fraught process of voicing, debate, and action.

Here M. M. Bakhtin's discussion of ethical situationalism can help account for the necessity and importance of the crisis as a moment in which pre-existing norms and codes break down and need to be rearticulated. In *Toward a Philosophy of the Act*, Bakhtin offers an ethical theory of action that repudiates the implications of formalistic theories of ethics, particularly Immanuel Kant's categorical imperative. Formalism requires what Bahktin interprets as a suspect allegiance to universally conceived theoretical precepts standing above time and place.[15] Bahktin argues that overallegiance to theoretical precepts misdirects and thus disables responsibility instead of channeling it toward an active confrontation with the living moment in its full blooded complexity. The effect of such "acts of abstraction" Bakthin says is to be "controlled by . . . autonomous laws" in which people are "*no longer present in it as individually and answerable active human beings"* (1993: 7, emphasis added).

While Bakhtin's dismissal of codified norms is somewhat overstated—in fact as I have

15 Bakhtin, as Greg Nielsen argues, does not entirely repudiate Kant's theory of ethics, but does reject his theory of action (1998).

been arguing here, norms are more practical then he suggests; they are necessary guiding abstractions that establish a common ground for action and social cohesion—his critique nonetheless clarifies a number of important points. For Bakhtin the most problematic aspect of formal ethics is that they provide a false sense of security, “an alibi” for actual ethical being that downplays the inherent risk and conflict of making decisions and the necessity of working toward solutions. The hard labor of ethics, its demanding phenomenology, is an outgrowth of taking risks, putting in the effort to engage with others, and choosing to confront the situation at hand in its specificity.

Despite Bakhtin's repudiation of theoretical dogmatism, he is careful to steer away from advocating moral relativism. As Michael Gardiner argues, Bakhtin rejects relativism for its shaky theoretical presumption that “*a priori* the mutual incomprehension of view … renders authentic dialogue superfluous” (2004: 39). Instead, Bakhtin asserts that individuals can potentially achieve some level of consensus because they are situated within a shared world of meaning. Despite clear differences in opinion that are unquestionably made evident during periods of crisis, people participating in a collective endeavor are nonetheless situated in a shared social space and committed to a baseline set of goals. As a result of Debian developers’ common participation in the project, their common participation within the broader hacker public, and their participation in public events like conferences, they can draw on a set of shared experiences to work toward resolving crisis. This is an important condition of possibility that speaks to a potential, though not a guarantee, for consensus. To reach agreement, ethical labor must be performed.

Because the emotional tone of crisis can diverge significantly from the way many developers expect or desire communication to unfold, I run a risk by portraying crisis as a positive force that can contribute to moral cohesion. Many developers adhere to a Habermasian

ideal of communicative interaction that requires participants to shed personal interest and passion in favor of sober rational discussion, where clarity is achieved because “all participants stick to the same reference point” (1987: 198 **James make sure I have this Habermas theory of communicative action**). While communication can certainly occur along those lines, it downplays the inherently risky nature of any communicative act (Butler 1997, Gardiner 2004). This is probably posed most poignantly by Judith Butler in *Excitable Speech*, where she argues that the Habermasian project is self-limiting, possibly undermining its democratic aspirations, because of its insistence on eliminating personal interest and the inherent risk in the act of communication:

> Risk and vulnerability are proper to the democratic process in the sense that one cannot know in advance the meaning that the other will assign to one's utterance what conflicts of interpretation may well arise, and how best to adjudicate the difference. The effort to come to terms is not one that can be resolved in anticipation but only through a concrete struggle of translation, one whose success has no guarantees” (1997: 87-88).

Butler’s statement shares Bakhtin's distrust of universal formulations that require people to speak from a sanitized location rather than confront a situation whose future ontology is unpredictable. Now let's take a look at one such “concrete struggle of translation:” a legendary crisis that hit Debian in 2005.

The Vancouver Prospectus: A Portrait of A Crisis Over the Limits of Meritocratic Rule

The first story of ethics in Debian I presented began with the story of an ending: the New Maintainer process was a solution to a crisis over the integration of new members. In fact, it created a social architecture that, while imperfect, continues to sustain a baseline level of trust and coherence and helps to absorb and lessen the shocks of future crises. Nonetheless, punctuated periods of distrust or malaise invariably recur, and here I focus on a recent one. So as the opening of this section on ethical moments began with the story of an ending, the closing of this section will end with a beginning.

There are a number of events that I could have chosen to illustrate the social metabolism of a crisis in Debian. I have picked one of the most recent of crises because of the rich multiplicity of issues it raises, and because I actually witnessed and closely followed its ebb and flow from the moment it began to its current recession.

Let me provide some background on the project's status at the time in March 2005. Debian was in the process of choosing a new leader. There were several candidates, and the ideas they brought to the table concerned fundamental questions of governance that could alter the nature of the DPL leadership, communications issues, the role of women in the project, transparency, a perceived hostile working climate, growing pains, and the uncertain threat of a new Linux project based on Debian (Ubuntu). The project was gearing up to complete a new release, and thus there was a heightened sense of pressure. It was in this frenetic climate that a single e-mail was sent that began the crisis.

This e-mail was sent to the developer list by the Debian release manager, announcing the final plans for releasing Debian's latest distribution. An in-person meeting in mid-March had

convened in Vancouver, Canada, bringing together the ftpmasters, the release team, and members of the security team to hammer out a plan that offered a very concrete vision for Debian's technical future. In addition to details about the upcoming release, proposals were advanced detailing how to handle the release after that, called “etch.” The participants in the Vancouver meeting had concluded that the era of universal architecture support was over. Debian did not have the technical or human resources to support nearly a dozen architectures:

> [...]
>
> The much larger consequence of this meeting, however, has been the crafting of a prospective release plan for etch. The release team and the ftpmasters are mutually agreed that it is not sustainable to continue making coordinated releases for as many architectures as sarge currently contains, let alone for as many new proposed architectures as are waiting in the wings. The reality is that keeping eleven architectures in a releasable state has been a major source of work for the release team, the d-i team, and the kernel team over the past year; not to mention the time spent by the DSA/buildd admins and the security team. It's also not clear how much benefit there is from doing stable releases for all of these architectures, because they aren't necessarily useful to the communities surrounding those ports.
>
> Therefore, we're planning on not releasing most of the minor architectures starting with etch. They will be released with sarge, with all that implies (including security support until sarge is archived), but they

would no longer be included in testing.

This is a very large step, and while we've discussed it fairly thoroughly and think we've got most of the bugs worked out, we'd appreciate hearing any comments you might have.
[...]

Note that this plan makes no changes to the set of supported release architectures for sarge, but will take effect for testing and unstable immediately after sarge's release with the result that testing will contain a greatly reduced set of architectures, according to the following objective criteria:

- it must first be part of (or at the very least, meet the criteria for) scc.debian.org (see below)

- the release architecture must be publicly available to buy new

- the release architecture must have N+1 builds where N is the number required to keep up with the volume of uploaded packages

- the value of N above must not be > 2

At first glance it may be unclear what in this technical, matter-of-fact e-mail would have led to a crisis. The meetings participants included the technical guardians of Debian and their advice is usually held with respect. But before I explain why such a seemingly benign proposal produced such an event, first let me describe the response, for it was nothing short of monumental—even by Debian standards of crisis. Within the first day there were over 500 e-mail messages in response, by three days, there were over 900 e-mails. This text of mailing lists, if taken together, could probably fill one, possibly two multi-volume dissertations. On IRC, conversation was bubbling and the talk of the town was this Vancouver debacle. Posts analyzing the event and its significance appeared on Planet Debian, the group Debian blog that aggregates individual developer's blogs. I had to spend days reading this material.

I was left in awe by the cascade of responses that had been provoked by one e-mail. It is first worth describing the emotive and social atmosphere of utter paradox that arose, in which synchronicity sat alongside unsettling discordance. The project was in one of the most pronounced moments of synchronicity that I had seen in a long time. Hundreds and hundreds of developers gave the problem their due attention in the form of *voluminous* writings—on mailing lists, IRC, and blogs. For days the project felt like it was riding the same dangerously large and unstable collective wave. Yet, this was also a moment of pronounced discordance, where difference of opinions rang loud and overblown accusations prevailed, as if a furious legion of frenzied and rabid hydra had suddenly appeared on the scene, each individual head screeching, rearing, and rending in all directions.

It felt as if Debian was coming apart at its seams. But it is a duality of centrifugal discordance *and* centripetal synchronicity that defines crisis. Crisis sits as a crossroads, it is a moment of betwixt and between when outcomes are decidedly uncertain. During this period, people were brought together by a swelling to express dissatisfaction, but pulled apart from one

other by markedly different sets of conflicting opinions, with unity under dire threat. There was a sense that this crisis was at once remarkably silly and overblown, a distraction from the work required by the immanent release, yet fully important and serious, as if there some line had been crossed. Why? What was it about this particular e-mail that caused such collective alarm?

The crisis could not be pinpointed to a single source but rested on several factors. Notably the developers were able to suture a very wide range of concerns to this e-mail, but one of the most significant complaints, stated over and again, was about *its tone*: its disharmonious resonance struck the wrong collective chord, working to resurrect the project's perennial discomfort over the corruptibility of meritocratic authority. Other precipitating factors included its timing and content. Last but not least was the e-mail's content, which many developers found shocking. Over the last five years, since the DPL leadership of Bdale Garbee, Debian had built an image of being a "Universal" OS, special among its class because it ran on more architectures than any other Linux distribution. Developers had informally animated the edifice of the DFSG's non-discrimination clause to include architecture support. The announcement that the era of technical universality was perhaps soon to be ended was a huge blow to Debian's sense of collective pride.

Complaints about the e-mail's "tone" centered on the following sentence: "The release team and the ftpmasters are mutually agreed that it is not sustainable to continue making coordinated releases for as many architectures as Sarge currently contains, let alone for as many new proposed architectures as are waiting in the wings." Even though the e-mail was stated as a proposal, below is a short excerpt from an IRC discussion that articulated the shock the e-mail produced:

> <Kivet> mm, I certainly didn't expect the meeting to be quite so wide-ranging; in advance, I rather expected it to be mostly "ok, let's sort out

sarge; ... oh, and in the five minutes we have left, how can we look to avoid this in the future?"

<Yaarr> vapor-b: if you wanted open discussion, you should have stopped in your tracks when it became apparent that other people might be interested in the subject. Don't do shadowy meetings on your part an request open discussion from us.

<stig> Yaarr: but that is what happend: they put out a proposal and now it can be discussed.

<Yaarr> yeah, sure

<Markel> and the announcement, singned off by all the people who do the work, and most future dpl candidates, had an unfortunate ring of finality. If indeed this is a peoposal, that is open to serious discussion (as opposed to "this is the way it is, unless you happen to convince all of us of something else, despite the hours of discussion where we hammered it all out"), than perhapos a follow up is in order

<Markel> stig: I don't know about you, but my comprehension of the English language has been often deemed adequate, and my take of the announcement was a fait accompli.

<Yaarr> stig: as I told you before, the idea is for our statement to be constructive, in that it'll try to suggest some modifications that will make this mess a bit less of a problem to us.

What this discussion demonstrates is that by presenting a fairly significant technical change in a way that *seemed* like an established decision, the delegates violated the norms of acceptable and

appropriate behavior. In this proposal, many developers found it difficult to believe in the “pure technical intentions” of entrusted members. What had failed here was a necessary performance of the goodwill that normally acts to limit anxieties about corruption in meritocracies, especially those with hierarchies like Debian.

What this event revealed is that Debian's implementation of meritocracy, like all meritocracies, is a fragile framework easily overtaken by the threat of corruptibility. In the case of Debian, this threat is particularly onerous, for it can potentially block the conditions for rough technical consensus; this event ostensibly edged too close to such corruption for the project's comfort zone.

Within Debian, the delegates and teams hold a similar form of authority as the mythical Philosopher King and his guardians presented in one of the most favorable accounts of meritocratic rule, Plato's *Republic*. In this imagined world, rulers are granted authority for life by virtue of their talents, their passion for the inherent good of ruling, and a well cultivated character that breeds the proper “intent” for rule. Leaders are those who are “full of zeal to do whatever they believe is for the good of the commonwealth and never willing to act against its interest. They must be capable of possessing this connection, never forgetting it or allowing themselves to be either forced or bewitched into throwing it over” (The Republic). This sentiment is eerily descriptive of the ways Debian developers conceive of proper meritocratic rule. Team members and delegates are entrusted to hold technical authority for as long as they want to (insofar the DPL has never removed someone from these positions) because they display their “zeal” to do good for the “technical commonwealth” of Debian through superior acts of technical production.

In Plato's imaginary republic, rulers were kept in continual check by being subject to a highly public presence and the demands rigorous ascetic life—little property, no domestic

relations. These confirmed and sustained proper intent. But in Debian there are few formal mechanisms to “check” the excesses of power of those who have been granted positions of technical authority. In theory, teams and delegates are fully trusted members who no longer have to perform their intent in order to prove their worth and make decisions. The teams that convened in Vancouver were empowered to make the significant technical decisions that they proposed. However, in practice (as this crisis made clear) such decision would be very difficult to pull off without first consulting and building technical consensus. The guardians are bound by the informal codes discussed in part one by which they must be seen to act not in self-interest but always in the interest of the entire group.

Thus, the overwhelming response to the Vancouver prospectus was a reaction to what was seen as a violation of meritocratic trust and during this period talk of the cabal became prolific. It seemed to some as if the myth, the joke, of “smoky backrooms” in Debian, was perhaps no joke at all. But hackers, as I explained earilier, will truly exploit any opportunity to joke, as these developers did during the peanut gallery discussion during the DPL IRC debate (even while expressing dismay over the event):

> <endo> Big Trouble in Little Vancouver
>
> <jab> "Serious issues with Mr. Vancouver"
>
> <digger> jty: lol
>
> <rickv> travesty: yeah im from vancouver..
>
> <garison> pickleS: No, I knew about it the same way. But like aj said, it was a surprise to all
>
> <heller> eh, a lot of people knew about the meeting beforehand. that's meaningless.
>
> *** phil** finds this hilarious, being in Vancouver right now

<Zizek> "A bad case of Vancouver"

<tm2> "debian does vancouver"

<Zizek> "Vancoup d'etat"

<jab> "pulling a vancouver”

But if the crisis raised the specter of mistrust, it was also the very mechanism by which trust was re-built again. The overt public voicing and re-voicing that the Vancouver meeting “smacks too much of deals in smoky back rooms, where a seat at the table is by explicit invitation” was a moment of collective clarification. The backlash and discussions that called Debian's philosopher-kings to task served as limits to curb what was seen as potentially inappropriate exercises of technical meritocratic authority and an opportunity for Debian guardians to assert that no such thing had ever happened.

Through an overwhelming tide of e-mails, many of these delegates were forced to explain their reasoning behind their recommendation that Debian limit architecture support. In turn, developers contributed their own views on what would have been the proper way to approach the problem, and others contributed discussions and proposals about how to technically proceed. The release manager and another member of the participating teams were remarkably attentive: they wrote e-mails and talked to developers on IRC to appease fears, explained technical details, took into account the recommendations of others, revealed what happened at the meeting, and—especially—reaffirmed that nothing has been written into stone. In essence they conformed to Plato's stipulation that “[the guardians] must be capable of possessing this connection, never forgetting it or allowing themselves to be either forced or bewitched into throwing it over.” As a result of these often passionate outpourings, the proposal was transformed into an unambiguous

proposition waiting to be further discussed. In the end, although it took a blow, trust was reestablished.

One participant and member of the ftpteam posted the following explanation, which gave a window into the organic development of the meeting and affirmed the openness of the proposal:

> As it happened, James and I were staying at Ryan's, and after dinner on Friday night (before the meeting proper started, but after we'd met everyone), we chatted about the topic and came to the opinion that removing a bunch of architectures from being release candidates would be necessary -- for reasons I hope are adequately explained in the announcement, or that will be on -devel as people ask. As it turned out, when we got to the actual meeting the next day, this was more or less exactly what Steve was wanting to propose, and he seemed to be expecting most of the objections to come from James, Ryan and/or me. So instead of that, we then spent a fair while discussing criteria for what support architectures would/should receive.
>
> Hopefully the above provides some useful specifics for people to talk about.
>
> >As a result, the rest of the project had little input into
> > the decision-making process.
>
> That's why it's posted on the lists now -- it never too late to get input into

> something in Debian; even after we've committed to something, *we can almost always change our minds* (italics, my emphasis).

As a result of these outpourings, many of which were wildly passionate, any ambiguity as to the status of the proposal was shredded, transforming it to an unambiguous proposition waiting to be further discussed.

When the acute phase of the Vancouver crisis was over, energy was diverted back to releasing the subsequent version of Debian: Sarge. Certainly, there remains a tremendous amount of work to be done on the technical problem of architecture support in Debian, and there are many broader questions about governance that this crisis raised. Even if was affirmed that entrusted members of Debian should consult the entire project before proposing radical changes, there seems to be a growing unease among many developers over the scalability of technical consensus. The rough consensus that so many developers are proud of seems to have gotten rougher with the addition of each new developer, and eventually Debian developers may have to start thinking about novel social solutions to accommodate these changes.

For now however, the work laid out by the “Vancouver Prospectus” will be entrusted to anyone who has a stake in the process. The field has been momentarily leveled despite the hierarchies of power that emerge from a meritocratic system. If much of the work performed during this crisis served to avert a false sense of corruption, one outcome of the crisis was a collective affirmation that the values developers desire from technology (accountability, openness, and access) are those that they also expect from project governance and members of their project, especially those who hold vetted positions of power. Nonetheless, Debian is a dynamic organization. It changes. And through changes, unforeseen conflicts will always arise that *potentially* open the door to a reflective processes of assessing such changes, allowing

developers to re-voice their commitment to informal norms of governance and begin the design work to reach a solution within these norms. While these values are codified in their charters, these texts do not fully determine their significance within the everyday lives of Debian developers. They must be enacted in various guises, one of which is a passionate outpouring of commitments.

Conclusion

There remains, of course, the important question of the extent to which the case of Debian can be generalized to illuminate the ethical processes of other virtual or F/OSS projects. With such a stark adherence to moral precepts, is Debian the radical black sheep of development projects? Or can we locate some of the social, organizational, and ethical processes I have discussed within other projects? It is worth noting that with over one thousand developers, Debian attracts a huge cadre of members in the F/OSS community of developers. Furthermore, each project has its own peculiar idiosyncrasies, so it is impossible to use any one project to generalize about all of them. If the processes I discussed here exist on a spectrum, Debian undoubtedly resides on the far end by virtue of its articulation of strong moral commitments. But other projects exhibit many of the elements discussed in this chapter. Every project is a dynamic entity and has had to deal with the problems of trust and scalability; most of the large ones have had to routinize, like Debian, by devising formal procedures for entry that require prospective members to undergo mentorship and training. Many medium to large projects have drafted key documents that define their goals and vision, in the case of Debian, they have formalized this into a contract.

In sum, given what I have written in this chapter, I hope it is clear that the praxis of ethics among Debian developers entails many different forms of labor, above all ethical labor. At times, this ethical work occurs as an implicit form of enculturation, while at other times it takes shape

as a reflective voicing through which a series of temporally and personally significant transformations are declared and achieved. For example, the New Maintainer narratives allow developers to re-evaluate their lives and make them into life histories that publicly offer a current and future commitment to a commonweal. Crises represent a moment of limits; charters and even routine narrative discussion are never enough to confront the emergent realities of new situations. In the simplest terms possible, an ethical life demands constant attention, response, reevaluation, and renewal.