Open Source Software: A New Mertonian Ethos?

Paul B. de Laat

Faculty of Philosophy, University of Groningen, A-weg 30, 9718 CW Groningen, THE NETHERLANDS

tel. +31 50 3015902

fax +31 50 3636160

e-mail p.de.laat@philos.rug.nl

Abstract

Hacker communities of the 1970s and 1980s developed a quite characteristic work ethos. Its norms are explored and shown to be quite similar to those which Robert Merton suggested govern academic life: communism, universalism, disinterestedness, and organized scepticism. In the 1990s the Internet multiplied the scale of these communities, allowing them to create successful software programs like Linux and Apache. After renaming themselves the `open source software' movement, with an emphasis on software quality, they succeeded in gaining corporate interest. As one of the main results, their `open' practices have entered industrial software production. The resulting clash of cultures, between the more academic CUDOS norms and their corporate counterparts, is discussed and assessed. In all, the article shows that software practices are a fascinating seedbed for the genesis of work ethics of various kinds, depending on their societal context.

Keywords: academia, CUDOS norms, intellectual property rights, open source software, practices

1. INTRODUCTION

From the 1970s onwards, communities of software developers evolved that freely exchanged source code between them in a *discussion permanente*. These were self-steering collections of volunteers, outside all governmental and industrial circles, writing software just for fun. These are usually referred to as hacker communities. Then, in the 1990s, the Internet multiplied the scale of this movement. On the one hand, it facilitated the free and instantaneous sharing of source code, by creating the download option. Before, `free' distribution had to use the postal system, and always involved some cost and some time delay. Similarly, the Internet has turned the exchange of comments into an instantaneous event. On the other hand, the hacker

movement has returned the favour, and created many programs that are now essential to the running of the Internet, like Apache, BIND, Perl and Sendmail. A symbiosis has developed.

As a result, more and more free software programs were delivered. Linux and Apache are the most famous examples. These are being used nowadays all over the globe by millions of people. Corporations thereupon took an interest in the phenomenon as well, and invented ways to turn open source to their advantage. This entry into the corporate world was actually triggered by a conscious campaign that started in 1998. The hacker community changed its name for the occasion: it was no longer a movement for `free software', but for `open source software' (OSS). As described more fully elsewhere,[1] several phases of ever deepening involvement unfolded.

At first companies started to offer services tied to the free product that facilitate its use: installation, customizing, training of personnel, and so on (cf. RedHat as the main Linux distributor). Next, OSS became more integrated into mainstream ICT-commerce. Suppliers attached it as an extra to their hardware or software (cf. IBM installing Linux on their computers, and integrating Apache into their web-server software). Also, companies started to develop closed application software on top of free OSS (cf. IBM porting Lotus to Linux). In a final phase, firms proceeded to conduct some of their own software development efforts according to the OSS model (cf. Netscape opening up the source code of its new browser, Communicator 5.0). That implies, that they open up their own projects to the outside world. Hackers from everywhere are invited to download the source code, and send in their comments.

It is, of course, this final phase which is most interesting of all. Instead of free riding upon OSS products, firms adopt the model as their own. The tenets of the movement are taken seriously, and taken to their logical conclusion. This represents an enormous challenge. The `original' hacker communities were self-steering, usually led by those who posted the first ideas for a project in cyberspace (or by those who later took over the baton). This last phase implies that the more complicated situation of a `mixed' project obtains: company managers allow source code proposals to be put on a publicly accessible website, and from then on paid developers and volunteer ones have to work together one way or another. As a rule, company employees will steer the project. Volunteers are woven into industrial software project management. So next to `free' OSS development (hacker communities) we now witness `corporate' OSS development.

In this article I will explore the quite distinctive work ethos that has developed within the free hacker

---

[1] PAUL B. DE LAAT, *Protection of Intellectual Property in Software: Towards Property Right or Property Left?*, paper presented at the 15th Colloquium of EGOS (European Group of Organizational Studies), Warwick, UK, 1999.

communities during the last three decades. A quite characteristic set of norms has evolved that regulates their attitudes towards work. It will be shown that these norms closely resemble those regulating work in academia; in particular, they are similar to the CUDOS ethos as originally formulated by Robert Merton. However, a rift runs through the OSS community, between more radical and more liberal Mertonians. It is the latter faction that paved the way for the entry into the industrial world, by stressing the superior qualities of OSS. The clash of work cultures that `corporate' OSS development implies, is assessed. Finally, the analysis is interpreted as vindicating the importance of practices as generating emergent systems of virtues.

2. OPEN SOURCE

An open source community can be defined as a collection of individuals that exchange source code of programs among each other for free. Every hacker is invited to `read' the code and detect bugs, suggest bug fixes and improvements, and so on. If some people take it upon them to exercise some leadership and incorporate all suggestions that are considered useful into the existing code and release new, improved versions regularly, a continuous cycle of comments and updated programs is generated.

What rules of conduct can be said to regulate relationships between these volunteer hackers? A first characteristic is that it is considered to be vital that shared code should (also) be *source* code (in computer language), not (only) object code (in machine language). In the early days of computing only machine language, which is difficult to `read', existed. Programming was a nasty job. To facilitate the process, higher order computer languages were invented that can be handled much easier. A compiler then translates source code into machine code. The creation of computer languages, however, also created possibilities for `black boxing' software. If a party wants to share software, it will readily publish source code (in order to allow others to easily interpret and change the program). If, on the contrary, a party wants to keep a hold on programs it has produced, considering them as its property, it will only reveal the object code to its users. Such a program can hardly be read at all. So the vital function of the source code requirement is to enable fellow programmers to effectively read, use and modify the code as they please.

In order to ensure the continuous flow of source, the OSS movement created several licenses, ranging from the more liberal Berkeley Software Distribution (BSD) style license to the more restrictive General Public License (GPL).[2] Typically, copyright on the program is asserted; on that basis, specific license terms are

[2]For the sequel, cf. DE LAAT, *Protection of Intellectual Property in Software*, n. 1 above.

attached to the source code. Anyone downloading that source code implicitly agrees to these terms. The core of all of these licenses is, that they *allow* (and encourage) fellow hackers to correct, modify and improve the source code, and publish their findings openly and for free. Legal room for continuation of the open and free OSS discussion is thereby guaranteed. Nobody is prevented from freely sharing and distributing the code, with the argument of trespassing upon intellectual private property. Note that OSS is not simply put in the public domain. Although that would achieve the same public function, OSS adepts chose the licensing option, because they typically want to attach some *conditions* to the release of their software. For example, and I will come back to this below, `closing up' the software (i.e. releasing it in object code) and selling it for a fee is sometimes explicitly forbidden (GPL).

Another characteristic of hacker communities is the invitation to all hackers around the world to participate. The core of OSS building, at least from the expansion of the Internet onwards, is *massive participation*. Compare the thousands of hackers that contributed to Linux, and the hundreds that contributed to Apache. The rationale is, that such massive scrutiny guarantees increasing stability and reliability. Software is notoriously difficult to write bugfree; unreliability has been a problem from the beginning. It is one of the central tenets of the OSS movement, that the mass of co-developers is extremely useful in identifying bugs and suggesting fixes for them. As Raymond put it in his famous 1997 essay `The Cathedral and the Bazaar': `Given enough eyeballs, all bugs are shallow'.[3] This contention has drawn many comments. Observers have pointed out that it is the quality not the quantity of testers that matters, that putting so many people on bug identification might be inefficient, that other project tasks (like architecture) also need attention, and so on. What matters here is that, whether the practice is wholly efficient or not (it probably *isn't*), these masses of volunteers do seem to contribute to higher quality indeed.

Although public discussion takes place on a massive scale, contributions are not anonymous. On the contrary; in the hacker community every contribution is personal, and is to receive the *credit* that it deserves. In fact this is, apart from the fun, what motivates volunteers to contribute: in exchange for their suggestions they receive credit, and their status rises accordingly. A continuous cycle of ever rising status may ensue.[4] And higher

---

[3]ERIC S. RAYMOND, The Cathedral and the Bazaar, in: Eric S. Raymond, *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, Sebastopol: O'Reilly, 1999, pp. 27-78: p. 41. Originally appeared in 1997. Also available at <http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar.html>, accessed 23 March 1999, 18:50.

[4]Cf. JOSH LERNER and JEAN TIROLE, *The Simple Economics of Open Source*, Harvard Business School Working Paper # 00-059, 2000. Available at <http://www.people.hbs.edu/jlerner/simple.pdf>, accessed 20 September 2000, 17:45.

status may be translated into many things. Starting down below as simple contributor, one may rise to the more restricted ranks of committed contributors that take decisions about modifying official software versions (like with Apache). It may also translate into job offers and assignments, since the OSS movement has taken root in commercial quarters.

Several measures guarantee that achievements are kept track of. While a project carries on, credits are mentioned in project histories, case files and the like. With Apache, for example, all contributors, whether big or small, are mentioned on its website. Those who contribute most enter the `upper circle' of decision-makers, and are prominently displayed. With Linux, its thousands of contributors may all be traced individually. Similarly, while open source code gets modified over and over again, the copyright notices allow to keep track of individual contributors. Hackers feel strongly about this. As Raymond has put it: `Surreptitiously filing someone's name off as project is (...) one of the ultimate crimes'.[5]

As a more complicated example of the status sensitivities involved, consider the dialectics of patches. Whenever hackers write patches, they could of course easily modify the software involved and release it. However, if many commentators start doing this, many patched up versions would be flying around in cyberspace. It would become difficult to tell the `official' version from the patched up versions. Then, if any of the latter would show serious deficiencies, the reputation of the `owner(s)' would be at stake. Therefore, according to the `official' open source definition - produced after intense debate in hacker circles -, OSS licensors *may* require that modifications only be distributed as patch files; these allow the program to be adapted upon `building'.[6] Maintainers of the OSS involved may then pick and choose the patches they deem appropriate, and incorporate them in their next official version. This option has been created to protect leading developers' integrity. In this respect note also the term `unfriendly patches' which is sometimes used.[7] These are hacker comments on central parts of a program, which are so critical that they may ultimately `destroy' the whole project as it is. Telling whether any patch is friendly or unfriendly is, of course, a matter of interpretation. That the term exists, however, is another sign of the concern about reputation. Not all developers, though, are that sensitive to

---

[5]ERIC S. RAYMOND, Homesteading the Noosphere, in: Eric S. Raymond, *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, Sebastopol: O'Reilly, 1999, pp. 79-135: p. 105. Originally appeared in 1998. Also available at <http://www.tuxedo.org/~esr/writings/homesteading/homesteading.html>, accessed 26 March 1999, 13:58.

[6]OPEN SOURCE INITIATIVE, *Open Source Definition*: clause 4. Available at <http://www.opensource.org/osd.html>. Cf. also BRUCE PERENS, The Open Source Definition, in: Chris DiBona, Sam Ockman, and Mark Stone (eds), *Open Sources: Voices from the Open Source Revolution*, Sebastopol: O'Reilly, 1999, pp. 171-188: p. 178.

[7]Cf. RAYMOND, Homesteading the Noosphere, n. 5 above: pp. 89-90, and p. 263, n. 9.

this issue. Take the GPL: it *does* allow hackers to modify files and redistribute them. On one condition only: these should `carry prominent notices stating that you changed the files and the date of change'.[8] Similar clauses to this effect can be found in the Mozilla Public License (as created by Netscape). Anyway, whatever the policy towards patches, they all allow proper attribution of software `texts' to their respective authors.

3. MERTON

The OSS community can usefully be compared to the academic community. The parallel has not gone unnoticed; observers as diverse as Eric Raymond[9] and Nikolai Bezroukov[10] have drawn attention to it. In their analyses they remark that OSS communities are characterized by much noise and conflict between participants, by problems of leadership and burn out, by discussions about the merits of autocratic versus democratic leadership, and the like - much like in academia. However, I want to leave these issues aside, and instead draw attention to the set of work mores that characterize academia.

In 1942, in the middle of World War II, Robert Merton wrote a short essay about the moral norms that may be said to govern science.[11] This ethos of (institutionalized) science consists of four norms: communism, universalism, disinterestedness and organized scepticism. Together these are abbreviated as CUDOS norms (which, not by coincidence, means glory and honour in Greek). The first one, *communism*, refers to `common ownership of goods': the fruits of science belong to the community. Every scientist should rush to publish his findings as soon as possible. Open and complete communication is the norm, as opposed to secrecy. In this quest for knowledge we all stand on the shoulders of those before us. Science is a common heritage to be expanded ever further. Therefore, any property rights for inventors are out of the question. Similarly, the industrial practice of patenting is quite incompatible with the scientific ethos. The only claim to `property' that remains is that of recognition and esteem for the creator of scientific achievements. As a consequence, fights over scientific priority have become a normal phenomenon in science. Esteem is a valuable good to be distributed fairly.

---

[8] FREE SOFTWARE FOUNDATION, *GNU General Public License*, 1989, 1991. Available at <http://www.gnu.org/copyleft/gpl.html>, accessed 7 May 1999, 13:48.

[9] Cf. RAYMOND, Homesteading the Noosphere, n. 5 above: pp. 131-132.

[10] NIKOLAI BEZROUKOV, Open Source Software Development as a Special Type of Academic Research (Critique of Vulgar Raymondism), in: *First Monday*, October 1999, vol. 4, no. 10. Available at <http://www.firstmonday.dk/issues/issue4_10/bezroukov/index.html>, accessed 27 January 2000, 17:14.

[11] ROBERT K. MERTON, Science and Democratic Social Structure, in: Robert K. Merton, *Social Theory and Social Structure*, enlarged edition, New York/London: Free Press, 1968, pp. 604-615. Originally published in 1942, entitled `Science and Technology in a Democratic Order'.

As another norm *universalism* is to be respected: knowledge claims are to be assessed by impersonal criteria. Achievement, not status should be decisive. That is, judgements of scientific quality should not be made to depend on personal or social attributes like nationality, race, religion or class. `Objectivity precludes particularism'. It is here that we notice most clearly the times in which the essay was written (first composed in 1942, then updated in 1948). Merton explicitly dwells upon the battle between `German' science and `French' science during World War I.[12] He also discusses the spectre of an `Aryan' science[13] and of a `Russian' science in conformity with Marxism-Leninism,[14] which both loomed large at the time.

The last two CUDOS norms are *disinterestedness* and *organized scepticism*. Merton describes them rather succinctly. As to the first one, scientists are to pursue the quest for knowledge in a disinterested fashion. No matter whether the scientist is motivated by scientific curiosity or personal gain, never is (s)he allowed to let personal interests interfere with the quest for truth as such. Fraud and plagiarism, for example, are utterly taboo. Notice, that `peer review' is one such mechanism to guarantee scientific quality. Organized scepticism, finally, closely connected to the other norms, refers to a critical and careful questioning of all spheres of life. No institution, profane nor sacred, is to be excluded from close scrutiny. It is this attitude, of course, that in the past has often brought science into conflict with church and state authorities.

4. OSS PRACTICES AND ACADEMIA COMPARED

It is these norms that Merton proposed primarily as a normative ideal for academia. According to him, however, it was also a *description* of academic mores, at least in democratic societies. Thereupon, much ink has been spilt on the latter claim: do universities actually observe CUDOS norms? Leaving these matters aside, I want to focus here on the relevance of this ethos for regulating *software* communities. Referring back to the description of OSS practices above, these can easily be seen to have much in common with the four Mertonian norms. To be sure, hacker communities are hardly institutionalized, so their characteristic norms are not institutional imperatives as such. Yet, their informality notwithstanding, they bear a close resemblance to CUDOS-type norms.

For one thing, the stress on publication of source code (not only object code) implements a conception of (semi-)publicity of software. Private property as such is not considered legitimate within the hacker community.

---

[12] *Idem*: p. 608.

[13] *Idem*: p. 609.

[14] *Idem*: p. 607, n. 4a.

This, of course, comes close to the norm of communism. In a sense it may be said that OSS has conformed to the norm of communism even more than science, at least until recently. Due to the Internet where such products are posted upon ftp-sites or websites, accessing public knowledge has become easier than ever. No longer, like in science, one has to buy a scientific journal, or go to a library to consult it. Almost all obstacles have been removed; for the digital literati, that is.

Their `communist' convictions notwithstanding, OSS developers do vie for mutual recognition. Their only property rights as inventors are to obtain esteem and prestige. As I have shown above, that mutual competition is closely controlled. The same state of affairs is characteristic of science. The way in which the competition for esteem is regulated, however, differs slightly between the software arena on the one hand and the scientific arena on the other. In OSS, progress is realized through massive participation of volunteers. They send in their comments, big or small, and are rewarded accordingly. Checking upon their quality is usually performed by `inner circles' of hackers that have distinguished themselves before. Comments that are found to be useful, are incorporated by these high status hackers into the next experimental or even `official' version. In science, we have congresses and journals instead. These fulfil analogue functions: the various software proposals and comments may be said to represent the equivalent of papers for scientific congresses (at least if no entry restrictions apply), while the various software versions may be regarded as the equivalents of articles that, after having been reviewed and accepted for publication, appear in scientific journals.

The OSS practice of massive participation has been dubbed `massive peer review' by Eric Raymond. In an interview he stated: `With open source code, everybody can see the blueprints of the software (...). You can have massive peer review, just like in any scientific discipline'.[15] This comparison betrays a slight misunderstanding of what peer review is all about. In science, peers carry out quality control, particularly when contributions to scientific journals, selective work groups and the like are to be judged. This control is carefully shaped. Ideally, it is double blind: neither submitter nor reviewer are to know anything about one another, neither name, nor any of the characteristics of sex, nationality, affiliation, and so on. This requirement, for one thing, protects the submitter from particularist tendencies on the part of the reviewer(s). On the other hand, it protects the reviewer(s) from resentment and protests towards them personally. Their judgements have to be controlled from above, not from below. So parties to a review procedure acquire *mutual* protection from each other. In sum, peer review in science is fashioned to be a check on disinterestedness on the part of those that are under review, while at the same time the norm of universalism on the part of reviewers is guaranteed.

[15]Quoted in DALE DOUGHERTY, What's New in Free and Open Source Software, *Webreview*, Open Source Special Issue, 1998. Available at <http://webreview.com/wr/pub/freeware/whatsnew.html>, accessed 26 March 1999, 15:29.

In OSS, massive participation by volunteers doesn't quite qualify as `peer review' in that sense. To me, it is more like a *discussion* between peers, than a *review* by peers. And this discussion, I would argue, *precedes* actual quality control as performed by the `inner circles'. The OSS quality control procedure, however, is nowhere near `blind'. Contributors are not anonymized one way or another; nor do `inner circles' (let alone single leaders like Torvalds) ever hide their identities, as far as I know. So while disinterestedness is checked upon, guarantees for the proper application of universalist criteria are lacking.

This disregard for the norm of universalism notwithstanding, as yet there are few actual signs of particularist tendencies. It is notable that accepted patches and files for OSS literally have come from all over the globe. Completely unknown Hungarians (cf. Linux) and Australians (cf. Netscape Communicator) seem to face no hurdles to get their innovations accepted. Quarrels and splits between different factions (for example, concerning the development of BSD-Unix software) are reported upon, but these do not seem to revolve around status differences. I would also argue, that *technical* features of the Internet may simply curb particularist tendencies. For one thing, e-mail messages do not display features that are readily apparent in face-to-face communication, like sex, age, and nationality. Only one's institutional affiliation is usually conveyed. For another, if someone has the feeling of being excluded upon particularist grounds, (s)he may easily adopt another identity (or hide behind the veil of anonymity) and start participating all over again. This option of multiple identities represents an extra safety valve.

Disinterestedness, the third CUDOS norm, can also be seen at work in OSS circles. Actually, it has already been mentioned above. Compare the quality control by `inner circles', and the fact that `stealing' of other people's code is severely frowned upon. At the same time, it has to be remarked that the *scope* for fraud in software production seems smaller than in (pure) science. Dreaming up scientific results occasionally happens; if cleverly disguised, fraud may be hard to detect. In software that is hardly possible, as a program simply has to work within reasonable limits. Such a check upon function can be performed easily.

Organized scepticism, finally, is a matter of degree. The OSS community can surely be said to incorporate more of this spirit than say the Microsoft community. If only, because the eyes of thousands of volunteers are allowed to have a critical look at the software in progress. Inside Microsoft only the corporate inner circle assesses performance. And of course, this kind of scepticism brings the OSS community in head-on conflict with that institution, which sees the sale of its closed products threatened. To Microsoft, closed machine code is sacred and not to be questioned.

In all, the hacker/OSS community may be said to conform rather closely to the CUDOS norms, though to varying degrees. Especially communism seems strictly adhered to. Until now, though, the norm of universalism

has only implicitly received attention. Considering the fact that to my knowledge none of their members ever studied Robert Merton's works, the OSS community can be said to have properly *reinvented* this ethos, and to have adapted it to the particular practice of OSS. Informal as the community is, these norms have of course not yet been institutionalized. In a sense, this parallel is not altogether surprising. As many commentators have remarked, the first Unix-BSD hackers had strong connections to Berkeley (University of California), while Linus Torvalds similarly started his `career' as Linux originator while still a university student in Helsinki. The values of that environment may be supposed to unwittingly have been transferred and adapted to the respective hacker communities.

5. SPLITS

Before discussing how free OSS communities penetrated the corporate world, a small excursus is necessary. Until now, the hacker movement has been presented as a unified movement. This is a caricature. It is composed of currents and factions, that differ on many points of substance. For the sequel it is important to describe the main rift that runs through this community. From its beginnings the issues of intellectual property and corporate interests have divided the minds. In the 1970s, a hacker community developed around the creation of the Unix operating system. It was all open source, although the term had not yet been coined. Then, in 1984, AT&T was split up, and gradually started to enforce its intellectual property rights (copyright) on parts of that code. Licenses became obligatory, and the code was no longer free nor freely to be distributed. Hackers were dismayed, and started a countermovement to `free' Unix.

The first to move forward was Richard Stallman. This programmer left his job at MIT and started the ambitious project of writing a free operating system from scratch that should ultimately replace Unix (project name: GNU). In 1984 he set up the Free Software Foundation (FSF). This stable actually produced many valuable pieces of GNU software (which later on could fruitfully be combined with Linux). What matters here is the prevalent hacker ideology behind the FSF. It is expressed in the so-called *GNU Manifesto* written by Stallman.[16] GNU is being developed, as software should be free and freely distributable. According to the manifesto `the fundamental act of friendship among programmers is the sharing of programs'; `copying (...) a program is as natural to a programmer as breathing'. What are the arguments brought forth for this position? Specific property rights, the manifesto argues, were granted in the past for specific purposes. Patents were

[16] RICHARD STALLMAN, *The GNU Manifesto*, 1985, 1993. Available at <http://www.gnu.org/gnu/manifesto.html>, accessed 7 May 1999, 13:42.

awarded to inventors, in order to stimulate more inventions. Copyright was created to encourage authorship. All of these intellectual property rights ultimately were created for the good of society, *not* for rewarding individuals. The case of software today, the manifesto continues, is very different. Both source and object code can be copied in an instant, and readily be used. Any enforcement of copyright (let alone, I am tempted to add, of patent rights) will only restrict the flow and spread of software among users. Moreover, programmers will go on programming even without monetary incentives, because it has an irresistible fascination for them. So, if I may summarize this FSF position in my own words: any system of intellectual property rights in software today is to be condemned, as it will not serve any useful societal purpose. It will not stimulate the creation of more (or better) software, but instead only restrict its spread and use.

In accordance with this anti-private-property stance, the FSF created the so-called General Public License (GPL), to go with all software that the foundation released.[17] As explained above, it allows hackers to download the source for free, and freely modify and redistribute pieces of code as they please. However, restrictions are added in order to *keep* the code `free'. First, taking pieces of GPL-ed code, embedding these in another program, and selling the whole as a commercial product is not allowed. Any program that contains GPL-ed code, may only be released publicly on GPL-conditions. Therefore, the license has been compared with a virus: whenever a program is infected with GPL-ed code, it automatically becomes GPL-ed as a whole. Moreover, as a second condition of the license, any recipient of GPL-ed code who, in turn, starts to redistribute modified code and continues the cycle, should tie the *same* GPL-conditions to his release. Any member of the chain of future hackers is automatically bound to GPL-conditions (this obligatory passing on of license conditions, by the way, goes for all other open source licenses as well). Both restrictions taken together imply that once pieces of code are published on GPL-terms, these will continue to remain free and unappropriable, whatever varieties and recombinations occur, and will never end up in closed and/or commercial software programs of any kind. Once GPL-ed, forever GPL-ed. Once a public good, forever a public good.

Next to this more radical position, there have always been hackers with a more moderate stance on intellectual property rights. From the early 1980s onwards, many hackers from Berkeley took up the challenge of keeping software free in another fashion than the FSF. Instead of writing a whole new operating system which should replace Unix, they proceeded to `liberate' existing Unix networking files, utilities and libraries, one after another. Each of these was rewritten meticulously, in order to get rid of any copyright claim. They were, in fact, mainly motivated by the desire to regain their software tools; they felt that these were stolen from them. This

[17] FREE SOFTWARE FOUNDATION, *GNU General Public License*, n. 8 above.

more moderate rewrite went with a more moderate and pragmatic attitude towards property rights. To them, there are two circles for writing software: hackers and corporate departments. Each is allowed its own place. Therefore, these Berkeley hackers didn't mind whether code came to be distributed in free or commercial form; *any* form of diffusion was welcomed. Compare the kind of license they attached to their Unix rewrites (from 1989 onwards): the so-called Berkeley Software Distribution (BSD) style license.[18] Essentially, anyone is free to hack and modify the source code, and redistribute this on any terms (s)he likes. Incorporating pieces of code into a program and sell the package on the commercial market is not forbidden. So they themselves preferred to operate in the public sphere; they didn't mind, however, if anyone else appropriated their source code in order to make money. Such a move, they argued, would only increase the spread of their source code. In that specific sense, they supported conceptions of software as *both* a public and a private good. This message was not wasted on the corporate world. For example, the X Window System for workstation graphics, originally released as free software on BSD-terms, has been appropriated and incorporated in many commercial Unix versions afterwards.

In all, two camps can be distinguished. These both agree that the hacker community *itself* should adhere to CUDOS-like norms. However, they differ as to their attitude towards the phenomenon of corporately produced software. The radicals essentially condemn this as a `hoarding' activity, and are destined to keep as much software as possible (OSS in particular) in the public domain. They can be said to be Mertonians in the full sense of the term. The moderates, on the other hand, simply accept corporate software as a fact of life. Moreover, they do not object to anyone `hijacking' OSS from the (semi-)public domain and putting it on a commercial course. So they can be said to favour the norm of common ownership, but for utilitarian reasons allowances are made for those who want to take a free ride, turn source code into private property and sell it. Therefore, they will be denominated as `moderate' Mertonians.

6. CAMPAIGN

From the 1990s onwards, at the advent of the Internet, the ranks of the hacker communities swell enormously. Not hundreds but thousands of helping eyes and hands presented themselves. This massiveness seemed only overwhelming at first. Then, gradually, it dawned upon some leading hackers that the potential benefits were enormous: quality could be enhanced by the scrupulous gaze of a multitude of participants. Unreliability of software could be minimized by the open source process. This pro-quality argument was then picked up by the

[18]BSD, *The BSD License*, 1998. Available at <http://www.opensource.org/bsd-license.html>, accessed 7 May 1999, 13:50.

moderate current, and turned into the banner under which they tried to get a foothold in the corporate world (1998). Open source software, it was claimed, is not only the opposite of closed (and non gratis) software, it is also *better* software. Additionally it was argued, that OSS is more attractive to users as its maintenance is no longer dependent on the single company that once wrote it, but upon a whole developer community. Moreover, as the proponents phrased it, OSS has `eternal life', because as a rule it becomes so widespread that there are always people to be found that can maintain it. Of course, profits can no longer be made from OSS proper. Instead, the argument went, companies will have to focus on the sale of services and applications tied to the OSS, which will have a vastly expanded base of users.

With these pro-quality arguments the moderate current took it upon them to `liberate' corporate software production from its self-imposed narrow-mindedness. Obviously, the more radical current was dismayed by this move towards industry. To them, corporate departments were not to be educated and enlightened, and turned into more academic units opened up to the hacker community. If anything, these should more properly be turned into public institutions.

These internal frictions notwithstanding, the campaign proved successful. The doors of the corporate world swung open. Compare the four phases as mentioned in the introduction: firms started to tie services to OSS, to attach OSS to their own hard- and software, and to develop applications on top of it. Finally, some companies adopted the OSS-model proper for their own software efforts. This last development implies, that the Mertonian CUDOS norms, so characteristic for free OSS communities, enter the corporate world. A clash of cultures is bound to take place. For it should be noticed, that corporate software production as a rule is governed by institutional imperatives quite the *opposite* of Merton's CUDOS norms. Instead of open source code, for free and freely redistributable, we have closed machine code, non-free, with a restrictive license that forbids all further distribution. As opposed to `communism', property rights are fully asserted. In the same vein, obtaining public credit for produced code is not usually allowed; it should suffice that programmers are paid for their work. Next, the public eye of fellow hackers is avoided; comments from fellow company programmers are deemed to be sufficient. So achievement is not judged by universal standards, but by local ones. Furthermore, software is not produced for its own sake, but for corporate profit. Ultimately, corporate interests and corporate hierarchy decide upon goals and standards of production. Taken together, these imperatives of private property, localism and hierarchy are the very antithesis of the academic CUDOS norms.[19]

---

[19]For a similar, though more general analysis of these opposites, cf. JOHN ZIMAN, Research as a Career, in: Susan E. Cozzens, Peter Healey, Arie Rip and John Ziman (eds), *The Research System in Transition*, Dordrecht: Kluwer, 1990, pp. 345-359.

How will this clash of opposites in work ethos work out? How can corporate interests and those of the hacker community be reconciled? Corporate opening up to OSS is a very recent phenomenon, and only few firms have taken up the challenge (in particular Netscape, IBM, Apple and HP). So few results are available as yet. Nevertheless, some interesting observations can already be made.

One issue for firms to solve immediately was the kind of license to be attached to software produced in this `open' fashion. It is worthwhile to dwell for a moment on Netscape that offered a free download of its experimental Communicator 5.0 browser.[20] Its company lawyers soon found out that the GPL would harm their interests. Because of the viral character of the GPL Netscape would also have to put in the public domain their (quite similar) branded versions of the browser which they sold to their customers. In those, proprietary pieces of code (licensed from other firms) were present, that Netscape could not just open up without impunity. Moreover, by such a manoeuvre the firm would force itself in the position that it also had to open up its (commercial) server software, having much code in common with Communicator client software. Finally, the company of course wanted to be able to incorporate code modified by hackers into their commercial products; a plain GPL would bar that option.

For these reasons Netscape created a special license. Essentially, it is like the GPL, with one notable exception: the firm obtains special rights to keep their commercial browser products closed, and to incorporate future OSS browser modifications into them. So, essentially, Netscape may do what it fancies, while the rest of the world is tied to the GPL. For the sake of commerce Netscape may open up or close code as it likes, while all others are bound to keep to the open source path. When this proposal was publicly tested through a special website (March 1998), many hackers were outraged. Now, as I would argue, the firm had two options. One was to abandon the special rights granted to Netscape, the other was to grant these to anyone else *as well*. Since the first option amounts to commercial suicide, Netscape is not to blame for choosing the second one. It modified the license proposal, to the effect that source code modifications can be incorporated into commercial products after all, if their creator so desires. However, this liberal attitude only applies to a `larger work', which might take considerable time to produce; smaller modifications are to remain a public affair. So after first having seriously considered to use the restrictive GPL, Netscape, in order to protect its commercial interests (and those of many a hacker) felt forced to retrace its footsteps and attach a more liberal kind of license to its Communicator source code instead. Apart from the restrictive clause about small modifications, the Netscape Public License is

[20]The following analysis is taken from DE LAAT, *Protection of Intellectual Property in Software*, n. 1 above, and relies mainly on JIM HAMERLY and TOM PAQUIN, with SUSAN WALTON, Freeing the Source: The Story of Mozilla, in: Chris DiBona, Sam Ockman, and Mark Stone (eds), *Open Sources: Voices from the Open Source Revolution*, Sebastopol: O'Reilly, 1999, pp. 197-206.

basically similar to a BSD-style license (as described above).

All established firms that followed suit after Netscape (IBM, Apple and HP) wrote down similar licenses. In the main, the incorporation of open source practices into the corporate domain forces firms to adopt a liberal style of licensing. And the models were already there for the taking (BSD-style license). So the conception of `communism', as maintained by the moderate current within the hacker community, has carried the day. The public domain is to be preferred, but room remains for proprietary releases. This is obviously an exciting development, given the fact that as a rule restrictive licensing of closed code is practised. Company property rights come under severe strain.

As to the other CUDOS norms, to my knowledge data are lacking as yet. It would be interesting to see to what extent universalism, disinterestedness and scepticism come to apply. Consider for example the question, whether public credits will be awarded to corporate software developers. And who will finally decide upon matters such as software specifications and testing: a committee consisting of both company managers and hackers? All of these issues remain to be reported upon.

The OSS movement itself knows full well, that integration of hackers and corporate employees poses serious challenges. The gulf between the two worlds might be too big. Therefore some leading figures of the movement have initiated the founding of *intermediary* organizations that could serve as bridges. Collab.Net, founded in July 1999, is one of them. It organizes OSS projects for firms that want to develop some software in the open source manner. It brings together companies opening up and hackers from outside. The firm also fulfils other essential intermediary functions like selecting developers, preparing contracts, monitoring development and settling disputes.[21] Note, that Collab.Net also assigns a so-called `Peer Reviewer' to every project, who is a kind of third-party quality controller.[22] This is an attempt, I presume, to bridge the gap between the company hierarchy on the one hand, and egalitarian hacker circles on the other.

Open source practices with their attendant CUDOS ethos have begun to enter the corporate labs. These are opening up to the world outside. But what the result will be of this clash of opposite cultures, what possibly hybrid work ethos may emerge, is as yet to be determined. The roots of the friction, of course, are the clash between public principles and vested interests. A clash that is none too real. Remarkably, some companies *without* vested interests, i.e. starters, seem to have much less trouble confronting a proper CUDOS regime, and adapting to it. They run their software efforts completely along open source lines, and even subscribe to the

---

[21]LERNER and TIROLE, *The Simple Economics of Open Source*, n. 4 above.

[22]Source: <http://www.collab.net>.

radical conception that software is to belong *fully* to the (semi-)public domain. Take for example Cygnus. From its founding in 1989, the company has focused upon marketing and customizing free GNU-tools.[23] In the process, it has also taken to developing software that supplements the GNU-project. These software efforts are completely open, and the resulting files are released with a GPL. Vested interests being absent, the firm is happy with these terms, as the GPL forces competitors that touch upon the code, to open up their own efforts as well. The competition remains transparent. As Cygnus is confident to have the best development department around, it prospers particularly in such an open environment. A Mertonian ethos is fully embraced, not for ideological reasons this time, but for reasons of corporate profit.

Note, that the preceding analysis is *not* to imply that the future for software *in general* is open source all over. For one thing, specialized applications running on top of open source platforms may continue to be sold in closed form. It is precisely the OSS model itself that creates a user base broad enough for profitable development and sale of such applications. For another, national and corporate interests will almost certainly preclude specific programs from appearing in the public domain in source code form. Think for example of encryption software, software for filtering purposes, and circumvention tools.

7. IN CONCLUSION

Over the last three decades a distinctive Mertonian work ethos has crystallized within the free hacker community. It was found to be necessary to rely on CUDOS-like norms. Then, at the end of the 1990s, open source practices and the attendant ethos started to enter the corporate world. This opening up of corporate software culture towards (some of) the CUDOS norms is a remarkable development. Observers have noticed, that these norms are not always completely disregarded in *industrial science* generally, as the usual stereotype would have it. On the contrary, some of its elements are, for example, observed in pharmaceutical laboratories. Industrial scientists are given some leeway to move around in academic circles as well: they publish scientific articles, attend scientific congresses, and so on. The reasons for this deviation from the usual model are straightforward. Industry uses these incentives to be able to continue to attract top scientists. Moreover, these excursions keep industry in touch with the academic world, and thus with new developments and upcoming talent. So for pragmatic reasons the applied scientists are allowed to function as `pure' scientists as well. As far

[23]For the sequel cf. MICHAEL TIEMANN, Future of Cygnus Solutions: An Entrepreneur's Account, in: Chris DiBona, Sam Ockman, and Mark Stone (eds), *Open Sources: Voices from the Open Source Revolution*, Sebastopol: O'Reilly, 1999, pp. 71-89.

as *industrial software* production in particular is concerned, sporadically traces of CUDOS norms can also be found. In the early days of its dominance, IBM used to share source code with clients.[24] As of today, some firms allow their developers to gain public credit for their efforts.[25]

I would argue, however, that the opening up of corporate software production that is taking place now is a much more drastic change towards academic standards. The concept of private property (and the stream of profits associated with it) comes under severe strain, participants from the outside world are drawn in, quality assessments perforce become a more public matter - such changes towards a Mertonian work ethos are more radical than ever witnessed before. Some analysts have alleged that as a matter of fact a company like Microsoft had already internalized the OSS model *before* that term even had been coined. It introduced the practices of frequent and early releases, modular architecture, using clients as testers (through beta releases), and the like.[26] Being that as it may be, the open source practice once introduced into industry amounts to much more. It implies the challenge that software development is to be practised largely as an academic endeavour, with users in control; it is more than just a marketing gimmick to catch and retain users' attention.

This corporate opening up is also remarkable in another way. In the whole area of knowledge and information products a general trend can be discerned of putting ever more restrictions on initial access to such products as well on their actual use. Doctrines of intellectual property rights are being more narrowly interpreted, and contractual and technological means employed for the purpose.[27] OSS practices, embedded in a corporate environment, can be seen to be the exception to this rule. They deliver software as common property, with (almost) unrestricted access and use for all.

In a more general sense, this narrative of the OSS movement may be related to some of Alasdair MacIntyre's observations. In his seminal *After Virtue*[28] he stresses the importance of cooperative human activities: `practices'. Examples are the arts, sciences, games, and politics. Taking part in such practices may of course yield `external goods' (like status and prestige). But such practices also have an overarching wider

---

[24]Cf. EBEN MOGLEN, Anarchism Triumphant: Free Software and the Death of Copyright, in: *First Monday*, August 1999, vol. 4, no. 8: part II. Available at <http://www.firstmonday.dk/issues/issue4_8/moglen/index.html>, accessed 27 January 2000, 15:00.

[25]Cf. LERNER and TIROLE, *The Simple Economics of Open Source*, n. 4 above.

[26]NIKOLAI BEZROUKOV, A Second Look at the Cathedral and the Bazaar, in: *First Monday*, December 1999, vol. 4, no. 12. Available at <http://www.firstmonday.dk/issues/issue4_12/bezroukov/index.html>, accessed 27 January 2000, 17:17.

[27]Cf. JULIE COHEN's contribution to this volume.

[28]ALASDAIR MACINTYRE, *After Virtue: A Study in Moral Theory*, 2nd edition, London: Duckworth, 1985.

purpose, going beyond individual contributions. In that wider sense, also `internal goods' or `excellences' (like an excellent scientific achievement) may be produced. The state of the art is pushed forward. In order to be able to procure such excellences, MacIntyre argues, practices will generate systems of virtues that are binding upon its practitioners: human qualities that enable them to achieve those internal goods. Moreover, means and ends in practices are constantly being redefined; the `quest' for where the practice is heading is ever open and moving. It is precisely this emergent quality that makes practices an interesting object of study.

And indeed, the observations made above about OSS vindicate MacIntyre on this point. Members of the hacker community have always been convinced that CUDOS-like norms should be observed in order to guarantee high quality and widely distributed software. In corporate software production, however, until recently quite opposite `virtues' were cherished. Since then, some of its practitioners have come in touch with hacker circles and discovered that the overarching `excellences' of reliability and user control require a turnaround of existing virtues. Only a turn towards a Mertonian type of ethos may readily produce these internal goods.

Acknowledgements

I thank all members of the unit `Praktische Filosofie', Faculty of Philosophy, University of Groningen, the Netherlands, for their useful criticisms in an earlier discussion about the central concepts contained in this paper. In particular René Boomkens, Kor Grit, Hans Harbers and Martin van Hees should be mentioned here. The authors of this volume provided valuable comments as well.

Biography

Paul B. de Laat is Assistant Professor of Science Studies at the Faculty of Philosophy of the University of Groningen, the Netherlands. He received his Masters Degree in theoretical physics from the University of Utrecht, and his PhD in organizational sociology from the University of Amsterdam. He published a book on matrix structures in R&D. His current research interests focus on the problem of trust in strategic technology alliances and in cyberspace, and on intellectual property rights in software.