7. Juni 2004

Version 1.0
Yuwei Lin (YL107@york.ac.uk)

1 Hybrid Innovation: The Dynamics of Collaboration Between Public and Private in the Free/Libre Open Source Software (FLOSS) Innovation System

This paper analyses the socio-technical construction of GNU/Linux through the co-production of the public and the private sectors, namely the community and corporate. Unlike innovation based on a strong professional culture involving close collaboration between professionals in the academic sector and corporations, FLOSS entails a global knowledge network, which consists of 1) a heterogeneous community of individuals and organisations who do not necessarily have professional backgrounds in computer science but competent skills to understand programming and working in a public domain; 2) corporations. Through analysing how agents in the FLOSS community and the OSS corporations work together and examining their socio-technical practices and relationships, the paper identifies an alternative innovation pattern quite distinct from that found in conventional technological innovation systems.

Commercialisation of OSS denotes a hybrid innovation model, which takes the advantage of acquiring resources both from the community and corporate. The community offers space for experimental projects and informal communications, while corporate stabilises and standardises the development of these community projects by incorporating them together and putting into markets. Unlike working in an informal innovation ambience where shared interests are the main concern for volunteer developers, after joining a firm one has to engage in the operation of a smaller subgroup, working on specific projects, with certain colleagues. However, such a formalised/institutionalised working partnership does not mean that firm-based developers have terminated their connections with the community. By contrast, previous (informal) cooperation on parallel community projects remains of significance in these firm-based developers' daily practices.

The paper draws on data from my fieldwork undertaken in academia, firms and the FLOSS community to show the dynamics of collaboration between public and private in the FLOSS innovation system. The findings will shed light on interactions between OSS firms and community members.

2 Prologue: Beyond Game Theory

This paper raises some qualitative questions about the meanings and values of the collaboration between corporate and the community in the free/libre open source software (FLOSS) social world. It will contribute to the mutual understandings between private and the public sectors, namely corporate and the community, and enhance the trust inter- and intra- users, developers and firms. It aims at creating new concepts in the organisation studies and e-commerce model.

Unlike the ordinary claim of game theory, which is paid a high esteem in the field of political economy, this paper emphasises the everyday practices of and the interactions between actors in software engineering. Game theory, an approach to the study of decision-making that models human interaction in terms of competition, cooperation, and conflict within predetermined sets of rules, strategies, and actor's interests, assumes that actors are rational interest-maximisers. Accordingly, corporation between corporate and the community in developing open source software are based on this premise. However, I would like to argue that this approach has its limits of linearity and simplicity, particularly of its basic assumption that all actors are rational interest-maximisers, and thus has its limitation of assessing the heterogeneity and contingency in the innovation process. Whereas the causality and structure are the focal points in game theory, the innovation pattern I propose is not a linear cause-effect process. Because "the creative capabilities of a firm to do something new, and to take risks with unknown possibility of success, rests fundamentally on recognition of a problem as an opportunity (Jackson, Mandeville & Potts 2002), the diverse choices and recognitions of problems will shape the innovation process. Since actors in innovation processes are not given fixed label/identity, their cognitions of problems and solutions vary along with their socio-technical positions. Consequently, as the history of the IT industry shows, the evolution of its knowledge set is catalysed by the actions of a diverse set of agents, who are driven by a diverse set of motives, all resulting in an explosion of economic activity and an avalanche of creation and destruction. "The heart of this process of creative destruction is the epistemic cycle of uncertainty, imagination and innovation. (ibid.: 329).

This highly fluid, fliexible and contingent innovation pattern I suggest, concentrates on the socio-technical heterogeneity and association in scientific culture and practice. Boundary objects such as source codes, programmes and projects are actively crafted in a process of mutual enrollment'. "[T]he production of successful boundary objects reacts back upon the social worlds thus linked and upon the larger whole they make up, reconstituting the very objects of study, as well as the material, conceptual, and social practices that surround them. (Fujimura 1992). The social worlds theory, which Fujimura employs in her studies, "offers us an image of the interactive stabilisation of a plethora of cultural elements, while enriching our understanding of that process in her focus on the new patterns of intersection and circulation that come into sight when one recognizes the social heterogeneity of practices. (Pickering 1992: forward). This line of reasoning will be in the heart of the paper.

2.1 A Community of Practice

In the FLOSS innovation system, the socio-technical relationships between actors are built on a range of everyday practices, such as talking with other developers or users on IRC, sending messages onto mailing lists, writing one's to-do lists, choosing licences for one's programmes. A set of material mechanisms and daily interactions construct and shape the FLOSS social world. The social world is like our late modern society: diverse and complex. Boundaries emerge when actors group or are grouped according to their identities, roles and affiliations. Allowing boundaries of diverse social groups to exist is crucial in the FLOSS innovation system, insomuch as it sustains the heterogeneity and preserves diverse innovation resources. Boundaries, however, shall not stop members from travelling between groups or communicating with each other. In the FLOSS social world, boundaries softly connect different innovation networks. This phenomenon, on the one hand, allows each group to make their claims about the collective practices and norms across demarcations, but on the other hand, the boundaries are maintained to preserve individualities. Individuals are mobile within such an innovation system, and the artefacts, tangible or intangible, are exchanged fluidly across boundaries among members from different socio-technical groups. It tells why the FLOSS innovation system is both dynamic and energetic.

Such an innovation system is no longer about one person or a small team optimising decision in a controlled environment. It has to do with the whole system, "includes issues of social justices, multiple interpretations, and adjudication of conflict across social boundaries." (Star, Bowker and Neumann 2003: 242). As a case study I've done on EMACS shows, the innovation of the editor epitomises not merely the value of technical innovation (i.e. a powerful editor that can be extended and integrated with different programming languages for different purposes), more importantly it symbolises the social innovation - a social contract initiated by Stallman in the late 1970s when he released EMACS, later on formalised as General Public License (GPL), regulating the liberal re-distribution of source codes of software products and feedbacks of further modification to authors/maintainers (Lin 2004). The former has facilitated the transmission of information without locking software source codes in the black box, while the later secures the innovation resources (mainly the manpower and ideas) for software projects.

In encouraging the transmission of information, lessons of software innovation can be learned easily. If a project stops, say, due to the death of the author, whoever interested in and capable of rendering it can carry on the work. Or, the connected works built on the software product can be improved or accelerated the speed of edifice. This open source practice (for brevity, I refer to the open source practice as *the OSP*), commonly found on FLOSS developers, also corresponds with other practices on the Internet and related ICTs, such as sharing information via E-mails or by mobile phones, contributing knowledge onto wiki sites, making weblogs to document experiences, and so on. These behaviours embody a vision that computer-mediated communication, data-archiving and resource sharing make the construction of knowledge more efficient and more effective.

In stipulating derived works and their further licencing issues, one is able to engage the shared interests and manpower in the existing project. Although diverges of the project might take place, this constriction has reduced the probability of the discontinuity of the project. Keeping consistent contributions to and interests in EMACSen is one of the reasons for its up to date survival amongst a variety of rival editors. The tie to the GPL enhances the relationship between EMACSen and its contributors and therefore its innovation network is maintained and even expanded.

There has been a growing recognition of the OSP in the wider computer world, which is regarded as of help to software innovation (see the UN report). However, there are arguments about to which extent should the source codes and information be open. For example, the materialisation of the OSP (e.g. licences) embodies these diverse views and reinforces the practice in software production. While the institutionalisation of the OSP normalises and standardises the OSP in the software innovation process, diverse actors give different interpretations to their manoeuvres. As a result, their views on the software problems are addressed differently. The collective practice, the OSP, is employed proactively to solve their problems at hands. The practice appears to be the boundary practice that enables cross-boundaries interactions. Based on this collective practice, collaboration and negotiation co-exist between actors.

While the OSP appears to engage diverse actors in the FLOSS social world, issues such as licences, concepts of freedom or open-source, and the degree of commercialisation, stir the harmony. In the case of licencing, some actors contend that the rule that the GPL stipulates makes potential adopters hesitate, and the others doubt the negative influence of this rather political conduct on innovation. Whatever the reason is, various licences are proposed in different terms and conditions to demonstrate their dissimilar views. Individuals or organisations also express their views by adopting different licences for their products. While some software developers claim that they have no specific preference to any licence but just choose one randomly, the others do show their concerns over practical or ideological issues. Companies especially do not want to get too much involved in the *holy war* or political issues. These diverse views, however, do not discount the importance of the GPL in the development of FLOSS. Instead, they confirm the heterogeneity embraced in the FLOSS social world. The plurality of expressions, the freedom of information and participatory democracy are fortified and are becoming one of the positive consequences of the OSP seen in the ongoing history of GPL.

The FLOSS social world as a whole epitomises a community of the OSP where diverse actors gather in this arena and share common conventions, language, practices, and technologies (Lave and Wenger 1992; Star, Bowker, and Neumann 2003). There are a range of peripheral practices derived from the OSP, including reading source codes, reporting bugs, configuring users' own systems. These practices cannot be done easily without the availability of source codes.

While the OSP seems to connect different social groups in the social world, a contradiction seems to emerge owing to the opposite incentives. Whereas the institutionalised OSP is based on a good will of gift culture found in the community, the idea of commercialising open source software is derived from economic self-interest. Hence, the innovation system is hybrid and some of the tensions between these two different ways of looking at values of knowledge and forms of accountability arise. While the tensions seem to be insuperable, I propose to anchor the exploration on the innovation system in dialogues and inter-subjective scrutiny between people who disagree with one another. In examining the practices of FLOSS developers in corporate and the community, the sociotechnical dynamics will provide alternative perspective on actors' decisions to collaborate. This is different from ordinary forms of social coordination built on prices and economic self-interest. In taking social and cultural factors into account, the research in the end explains the conventional (software engineering) and unconventional (the extensible social network centring on the artefacts created) characteristics of the FLOSS innovation. The FLOSS innovation system is framed as a social world where individuals and organisations inhabit, including media agency. As the Linux-Magazine claims, "As a user of FLOSS, one is joining an information network that is dedicated to distributing knowledge and technical expertise. The Linux Magazine is not simply reporting on the Linux and Open Source movement, it's part of it. (Linux-Magazine). Given this statement from the Linux-Magazine, one could see that the notion of the FLOSS social world is very broad and even magazines (conventional media) play an active and important role in it.

2.2 Working Practices in OSS Firms

As a result of dropping hardware prices and more reliable, compact, and standardised computer systems (Ceruzzi 2003), FLOSS join in the trend of commercialising software in the 90s. At some point, the commercialisation of FLOSS institutionalises the OSP and popularises FLOSS. In this process, OSS firms operate as a combination of conventional companies bearing business strategies in mind and unconventional entities holding up the OSP. As one could observe quite easily, the ubiquitous usage of ICTs and virtual communication in the FLOSS community remains the major channel for developers at commercial OSS companies to communicate and to exchange innovation resources (tools, ideas) within and outside companies. Albeit this attribute facilitates higher information flow than that in conventional firms, the managerial business of corporations yet demands developers within the system to do some bureaucratic works, such as attending regular meetings, dealing with paper work, or splitting iterative but necessary jobs such as debugging. Though these unavoidable routines are distracting, however, working in an OSS company appears to be more liberal than in any other commercial sectors while having incomes at the same time.

But the practices of developers in the OSS commercial companies somehow are shaped by this logic, which is a mixture of the conventional capitalist perspective on markets and customers, and the unconventional FLOSS innovation concept. Their conducts are aimed at pursuing profits but deeply connected with the FLOSS community where a strong gift culture resides. The concern on profits becomes a main factor influencing a company's decision-makings towards innovation. Most interviewees have mentioned time pressure, influence of the investors/clients and available financial resources (bank loan) as crucial factors for firms to evaluate innovation policies. Their acts can no longer be "Just For Fun, as often described by many community members. Unlike working in an informal innovation ambience where the shared interests are the main concern for developers volunteering to work on FLOSS projects, after joining a firm, a FLOSS developer has to engage her/himself in the operation of a smaller social group, working on specific projects, with certain colleagues. Regardless of that, many work contents appear to be extensions of previous (informal) community projects parallel to firms' current ambitions. One developer often is hired through the link of the other developer because they all work on the same project and know each other well. Joining a firm for developers, therefore, signifies a formalised/institutionalised working partnership. The trust and the tacit understanding between developers may be enhanced. As many interviewees working in firms have said, if they have problems at hand, colleagues will be the first resources they go after for resolutions, rather than through other means such as posting questions on newsgroups or discussion lists. This is probably because the shared goals are better perceived among colleagues working in the same team. However, it also entails that their behaviours might be presupposed by dominant corporation rationalism. The steps that developers have to follow in order for a transaction of organisational tasks to be properly executed are also proactively stipulated. Hence, developers' daily programming practice is shaped by the institutional rules. Klaus, a desktop developer for a leading OSS company, describes the internal operation in the company:

Because of the people on the internal mailing list are so brilliant that almost every time they would know at least the point to point me to, to find the answer. It's very rare that I [post] question[s] on the Internet. I sign [up] for my local Linux community at home, we have a mailing list where you ask questions. But usually I can find the answer on the mailing list something to do with what's in our distribution and that's almost everything.

(LT060201)

In other words, in setting up such an internal network of the employed developers, the expertise in the firm has been centralised. The expertise of developers is concentrated in a bounded group and their interaction and relationship is fortified formally, in so doing to enhance the trust between colleagues and solve problems more efficiently. With such a strong developing team (expert-oriented), a firm is also able to convince their customers of the quality of their products/services.

It is also argued that clients and markets are the driving forces for firms to innovate. For firms distributing desktop systems, incorporating stable applications together for end-users is the main task. As Klaus says,

[Un]like KDE developers just do it because they want to do it, we have clients, the potential users out there. [T]he application probably needs to be able to work for a lot of different people. And obviously it must be pretty stable.

(LT060202)

As most users would not like to change their usage habit radically and tend to stay with existing interfaces, the key element to make a successful software package for users is to make them "feel at home. With a concern about the lack of good FLOSS applications for end-users/customers, Klaus comments further the current development in application software,

It has been a few years since we last saw a radically new application. Everything else we do is just a little variation on what we did a couple of years ago. So when you do word processor it has to feel like most of the word processors. If you do something it's very very differently, then you have to find something that really it's a much better way of doing it. Otherwise people would say no I can't figure this out; I'll use something else'. That's the same if it's user level graphical tool, or it is command line tool, doesn't really matter. People have to feel some point at home. Which is right it's difficult to change from one operating system to the other because you don't feel at home suddenly. So you have to have a reason for going somewhere else.

(LT060203)

2.3 Linux Conferences: A Bridge for Corporate and the Community

Given the rule of thumb in software engineering of understanding users' requirements, communicating with clients and knowing what they want is vital. And if clients can understand the concept of open source, they would appreciate the product/service more and have better cooperation. The relationship between clients and developers also improves when they have reciprocal understandings on the concept of open source. As James, a CEO in a FLOSS SME describes one of his clients:

He is a quite clever customer, the kind of customer probably I think would be less than ten such customers in the country eletron, such an open-minded and clever person. [Unlike]CEOs of small companies usually keep everything secret, we understand [each other] perfectly and we know why we do that. [T]he guy is clever enough to understand that it's better to share. And I think it's better to share something which at least you can adopt, than to become customers of something completely proprietary you cannot adopt and which is controlled by a third person. Like you have a choice between the proprietary ERP and the open source free ERP, proprietary ERP means you lose complete control on your information system and give to someone else; free software ERP means that you share many things with others that at least you can keep control on your information system. I think the second case is much perfectly.

(LT060204)

Where to find such a nice customer? Linux-related conferences or virtual platforms are good venues. Particularly at conference sites, developers or firms can spot their potential clients/co-workers easily, either through direct face-to-face interaction, or through snowballing contacts. The social function of more informal conferences (e.g. Linux Tag) with less commercial/business atmosphere turns out to be even more useful for developers or firms to build up good relationships. (Who could say no when s/he is drinking or having fun?) In a relaxed environment, participants can establish informal personal contacts with people, among them some will become customers and the others will become colleagues. These conferences provide a platform for diverse actors to meet and interact in person. They exchange information, share experiences and opinions. Since some participants have known each other or worked on joint projects virtually, the trust between them is reinforced when they meet up in person. Conferences have become a conspicuous means for establishing social networks in the FLOSS innovation system. Social networking is seen to be crucial because with these contacts one is able to find whom s/he can turn to, whom s/he can cooperate with when problems come up. Attending conferences serves as an informal and alternative way of seeking for prospect collaborators (i.e. employees or clients).

OSS conferences and meetings secure ventures for collaboration between firms and the community. Apart from that, conferences also act as a key mechanism within the FLOSS social world to reconcile various practices and assimilate diverse actors and organisations. In this sense, conferences acts like a boundary object that shapes and is also shaped by its participants.

2.4 The Relationship Between the FLOSS Community and Corporate

The commercial sector co-exists with the FLOSS community. Their interdependent relationship is endorsed and built on in a number of socio-technical mechanisms. Here, I will illustrate a few of them to explain the mutual-help model.

2.5 Bug-Reports & Patch-Contributions

It is estimated that around three billion dollars worth of hours have gone into this constant upgrade process for the heart of the Linux operating system. And this figure doesn't even take into account the millions of hours that have gone into the thousands of applications that run on Linux. Working closely with the community thus can save firms time and cost in detecting bugs and writing patches (though they still have to do this as routines to ensure qualities of their products). Unlike proprietary software firms that deal with bugs and patches in internal environments where only insiders get involved, OSS firms do welcome and rely heavily on bugs-reports from the community. However, OSS firms formalise this bug-reporting and patching approach in their practice while keeping channels to the innovation system open. It is even more standardised for firms dealing with patches with a series of guidelines, which simplify and standardise the procedure of contributing patches (but this can be seen in big community projects as well). In so doing, it eliminates the risk of potential discontinuity owing to volunteer drop-outs or incompatible submissions, and keeps software products in a consistent condition. Nevertheless, this also implies that prospect contributors have to be wary of the rules to be included. It is a process of technological socialisation. And it also implies that the process of innovation is stipulated at some point.

2.6 Social Capital

The interdependent relationship between OSS firms and the FLOSS community is something that makes FLOSS innovation pattern different from other commercial software companies, particularly those deliver proprietary software. The fundamental software engineering techniques are not particularly distinctive in the FLOSS innovation, but

It's different in the way that [in the community] you certainly have people starting to contribute to the project that you never ask to do it, and that's interesting because they have the probability of giving you ideas you haven't thought of.

(LT060205)

It is recognised that brainstorming and experimental practice in the community (mostly taken place on the Web) stimulate and cultivate the FLOSS development.

2.7 Public Ethics

Corporate social responsibility (CSR) is highlighted particularly in OSS firms, and working with the community kind of sets an ethical tone and strategic agenda of corporations. CSR, together with financial performance and environmental practices, have become the top three goals for big IT companies. Working with the free software community seems to be a perfect technology management tactics that achieves both competitive performance and social responsibility. This "strategic responsibility integrating the fulfillment of corporate social responsibilities with technology-driven strategies for keeping products competitive, provides the basis for new products, and changing oeprational conventions (Brennan & Johnson 2004). Investing in open source software or working with the community gives corporate an ethical image, which might appeal more buyers/users to their products. Additionally, the community does give them surplus technical support in software production.

2.8 Shared Goals and Split Goals

Though the interdependent working relationships between OSS firms and the FLOSS community illustrated is vivid, once the shared goals conceived among the community and firms disappear or asymmetric consensus emerges, their collaboration will come to an end.

Normally, firms receive manpower and ideas from the community, and the members of the community get financial support from the firms, especially for big projects. For example,

A lot of people want to replace X the graphical layer in Unix. But doing that is an enormous task and that's difficult if you don't have a group of people that is able to focus on doing it completely. And that usually means that someone pays them, which means that someone would have to have incentives to do so. And that is where I found it difficult to find that incentive in the free software world.

(LT060206)

However, the economic reason might also endanger the existence of the community. When firms are dealing with business, they no longer abide the OSP (but perhaps the OSP is never their first priority). The financial incentive will possibly force firms to move away from conducting the OSP and lock up their source codes or concede with the proprietary software. For example, based on a pragmatic rationale that the majority of Japanese web sites are using Windows Media format, Turbolinux, a Japanese seller of the Linux operating system, has made an add-on package consisting of the Windows Media format and several proprietary software components in support of their distribution (27 April 2004 CNET News.com). Their act of combining products from the ideologically opposed open source and proprietary software camps has been claimed to help interoperatbility, though most desktop Linux developers would appreciate open standards more.

Richard Stallman is one of the community people who hold a sceptical view on commercialisation. He criticises the commercialisation in light of GNU Emacs that,

I don't think that anything like EMACS could have been developed commercially. Businesses have the wrong attitudes. The primary axiom of the commercial world toward users is that they are incompetent, and that if they have any control over their system they will mess it up. The primary goal is to give them nothing specific to complain about, not to give them a means of helping themselves. The secondary goal is to give managers power over users, because it's the managers who decide which system to buy, not the users. If a corporate editor has any means for extensibility, they will probably let your manager decide things for you and give you no control at all. For both of these reasons, a company would never have designed an editor with which users could experiment as MIT users did, and they would not have been able to build on the results of the experiments to produce an EMACS.

(Stallman)

Though proprietary software firms may not be moral enough to release full source codes, not all open-source developers would agree on a full-scale open source policy either. As William, a CTO at an OSS SME says,

When you talk about open source, not every software can be open source. It would be Who will pay these people doing something very special? These people who do something very special need to communicate with their users and with people doing competitive stuffs. They need to exchange their ideas. But that doesn't necessary mean this thing has to be open source, why should it?

(LT060207)

When being asked does his conversation suggest that he does not support FLOSS, he says, "It's an interesting idea, and works well for some projects, doesn't work for other projects. And it doesn't work well for software company which would have a single product. (LT060208). How is the interdependent relationship between the community and the commercial sector being balanced will be a crucial factor for the further development of FLOSS. FLOSS is of public interests:

For consumers it's always better to have open source or free source. Always. Because it gives you a power over the vendor that you can never have when it's close source. When it's close source then the vendor can lock you in as a user. The small Ebooks is the best example.

(LT060209)

But when public meets private, conflicts and tensions arise. For most developers, however, choose to face the conflicts with a pragmatic attitude:

Before embarking on a full-scale Open Source deployment, you should heed some warnings. Because Open Source software is mostly developed by nonprofit organizations or individuals, there is usually no official support or guarantee provided. But this may not be a problem. Often, commercial support falls short or is too expensive. With Open Source, it is relatively easy to find a mailing list for the software you're using. Although there are no guarantees, you can usually find solutions to most problems online.

(AP010401)

2.9 The Hybrid Identity of OSS Developers

As seen above, developers in the commercial OSS companies mostly remain working closely with the community, or deploy/employ the tools (e.g. Emacs-like editor, GCC compiler etc.) developed in the community. As the manoeuvre in the community is different from the one in the commercial sector, developers actually play double roles in their daily practices, which would give them hybrid identity in the FLOSS social world. Their practices at some point have to qualify the standard of the commercial sector for making profits and fulfil the requirements of the customers, and at the other time their contributions made to the community projects are obligated (morally, socially and technically). Having the hybrid identity in fact gives the developers flexibility to play around. They can acquire resources from both the commercial sector (e.g. money) and the community (e.g. friendship, technical support). They can use the community as a ground to test out their new idea and make some experiments, but later on release their work formally on the market for further incorporation with other applications. For example, a programmer at a German-based OSS SME, who mainly develops software for networking and system administration, still commits his spare time to write scripts of detecting software vulnerability. The scripts he was writing later on would be applied to finding out security holes of the systems he develops for clients. The script did not work out fully as the way he expected, but after releasing it on the Web, a few bug reports directed him to modify the scripts and makes it better. The script now can detect security holes more efficiently. If he did not write this script to try out his idea, he would never find out "if certain things are possible or not (LT060210). The experiment he has done after sharing with the community benefits from the feedback from other members in the community, and later on turns out to be useful for developing commercial software. This experimental manner works well in the common scheme of the community and firms, if no other pressures apply.

The commercial sector and the community however, represent two different manoeuvres and approaches in producing FLOSS. People who work for firms also acknowledge this difference. Their hybrid identity coerces them to take a less political-based view and to be closer to entrepreneurship. As Brian mentions,

In the corporate world you would need the one that I can make money on this because someone else needs it. That's been driving a lot of applications, a lot of good software. It still does. I mean that's the way we distributions compete against each other. Writing added value to the basic Linux part. So that's at least one other thing. But that usually not works interest people or motive people on personal level, but does attract some software developers.

(LT060211)

To support this pragmatic view, another interviewee also explains his feeling on working with commercial companies:

It's a good feeling to have a project out and have people using it. That is something that drives a lot of the people, including me, doing user level application at least. Then it changes from being interested in solving a problem, to being able to give people something that people find useful, that is something I know that a lot of people find interesting and need it.

(LT060212)

James explains his motivation to establish a commercial company as demonstrating his ability to have "the freedom of innovation (LT060213). The incentive to have his owned company is to be economically independent. Having a company he will not being exploited by big firms, and this also prevents his creativity being choked off from big firms' policy.

[Big firms] don't want to create new products for new markets. And I know many people they decided to put down a great job as an engineer in a great large corporation to create their own small company doing free software because they are tired of having ideas, which they could never put into practice and share with others. And so that's one of the reasons also why I am doing my company and the ERP, because if I have an idea, then I can write it as a software then it becomes quickly a product which I can share with others. And even in that new idea can be dangerous for a very old product, I can still do it. So it's like I have the freedom to innovate, put my innovation on the market, and share it with others. And that freedom does not exist in many large companies. Because there are so much innovation actually in free software, and there are so many people who go that way because that's probably the only way whenever you have an idea to see in practice used by many people and to improve it after it has been used.

(LT060214)

2.10 The Community Strikes Back

Though the firm-based innovation appears to provide a standardised and stablised innovation practice, the community-based innovation seems to afford more dynamics than the firm-based innovation. As Stallman remarks on the innovation of EMACS,

[T]he standard EMACS command language was the result of years of experimentation by many user-maintainers on their own editors On the fateful day when I gave users the power to redefine their own editor, I didn't know that it would lead to an earthshaking new editor.

(Stallman)

In order not to be jeopardised by the strategies of big companies and to keep a close relationship with the community, many SMEs developing OSS, as illustrated above by James, are established. The existence of OSS SMEs seems to be a balanced solution for developers who would like to hold a hybrid identity.

Otherwise, non-profit organisations works, too. The Debian distribution of GNU/Linux, "the only significant distributor of Linux that is not a commercial entity (or more correctly speaking, a non-profit entity), strategically sets up a company for fund-raising, but still strongly clings to the community-based innovation [1] . However, as an objection to the common sense that the community-based innovation cares less for the end-users, Debian distribution of GNU/Linux has started a variety of customised distribution, named Custom Debian Distribution (CDD), to challenge this myth. The idea of CDD is to fit different requirements of users with various professional needs and backgrounds. Without yielding to the commercial force, Debian distribution provides a range of distributions for specific user groups while adhering to the devoted Debian manifesto of having a non-commercial distribution of GNU/Linux operating system.

Apart from the organisational actions, some individuals express different opinions towards innovation issues by choosing licences. As said earlier, different licences are proposed to give different levels of openness of source code, modifications for further distributions, and appliance with proprietary software. Adopting licences is also a means of demonstrating one's view. In Sourceforge.net, the most famous FLOSS software collaboration platform, one could see diverse FLOSS released in a number of licences. For instance, ERP is released under Mozzila Public Licence (MPL) while Gaim is released under GNU General Public Licence (GPL).

2.11 The Negotiated Innovation

In a diverse field such as the FLOSS social world, it is common for members to hold different opinions and have hybrid/multiple identities. This phenomenon shows both the complexity as well as the flexibility of the system. It is heterogeneous, however, there are democratic apparatuses for members to express themselves. Therefore, while a collision of cultures or ideas occurs, conflicts can be solved through negotiations, and compromises can be reached. Hence, divergent views do not kill innovation, but on the contrary, they afford constructive dynamics to the innovation system. While developers, users and the industry all bear a rosy picture of Linux in mind (see http://www.ibm.com), it is important to notice the diversity in the social world. The model of hybrid innovation resembles an innovation model mirroring collaboration between the public (the community) and the private (corporate) sectors. The model may work well when firms and the community share the same goals of tackling the perceived problems, for instance solutions for business and end-users. It might endure when trust is built up between corporate and the community. To do so, ongoing dialogues are necessary. In the dialogues, firms and the community have to understand each other's comparative advantages. For instance, apart from employing developers from the community, firms can devote themselves in marketing and socio-technical studies (e.g. conducting customer surveys to understand users' needs or advocating the concept of open-source), popularising FLOSS and extending the innovation networks. Information in the two sectors shall flow more fluidly to encourage collaboration. In this case, various Linux-related conferences serve for this function (just like the subtitle of LinuxTag suggests: when .com meets .org). But in order not to disappoint either side too much, understanding each other's differences is crucial. Apart from more intense and informed communication, the involvement of a wider range of actors (e.g. governments, educational organisations, non-profit organisations etc.) in the FLOSS innovation should be welcomed to open up more possibilities of collaboration. Commercialisation should not be the only way of developing and implementing FLOSS.

The community and corporate are simply two of the many stakeholders in the heterogeneous social world. However, given the assorted episodes described above, we have seen ongoing negotiations between actors in their complex relationships. "The actors negotiate their own identities and interest as well as the existence, nature and volume of overflows. (Callon 1998: 264). Their negotiations denote the dynamics of interactions. These controversies and negotiations might cause positive or negative externality to the development of FLOSS. But no matter what, in Callon's words, "it never ceases to emerge or re-emerge in the course of long and stormy negotiations in which the social sciences have no choice but to participate (ibid.:266).

2.12 References:

Brennan, L. & Johnson V. (2004) "Technology Management for Corporate Social Responsibility. *IEEE Technology and Society Magazine* , Spring 2004.

[1] Nontheless, SPI (Software in the Public Interest) Inc., a New York-based non-profit organisation was founded to help Debian and other similar organisations develop and distribute open hardware and software. Among other things, SPI provides a mechanism by which The Debian Project may accept contributions that are tax deductable in the United States.

Callon, M. (1998) ’An essay on framing and overflowing: economic externalities revisited by sociology’, in Michel Callon (ed.) *The Laws of the Markets* , Oxford: Blackwell.

Ceruzzi, P. (2003) *A History of Modern Computing* . 2nd edition. The MIT press.

CNET News.com. April 27 (2004) “Linux seller licenses Windows Media technology. URL (consulted on 020504): *http://news.com.com/2100-7344-5201352.html* <http://news.com.com/2100-7344_3-5201352.html?tag=prntfr>

Fujimura, J.H. (1992) “Crafting Science: Standardized Packages, Boundary Objects, and Translation, in Andrew Pickering (ed.) *Science as practice and culture,* Chicago: The university of Chicago Press.

2.12.1 Jackson, M. & Mandeville, T. & Potts, J. (2002) The evolution of the digital computation industry. *Prometheus,* Vol. 20 No. 4, p. 323-336.

2.12.2 Lave, J. & Wenger, E. (1992) Situated Learning: Legitimate Peripheral Participation. Cambridge University Press.

2.12.3 Lin, Y. (2004) Epistemologically Multiple Actor-Centered Systems: or, EMACS at work!. Ubiquity, vol. 5: 1. URL (consulted on 020504): http://www.acm.org/ubiquity/views/v5i1_lin.html <http://www.acm.org/ubiquity/views/v5i1_lin.html>

2.12.4 Pickering, A. (1992) “From science as knowledge to science as practice, in Andrew Pickering (ed.) *Science as Practice and Culture,* Chicago: The University of Chicago Press.

2.12.5 Stallman, R. “The EMACS Full-Screen Editor. URL (consulted on 020504): http://www.lysator.liu.se/history/garb/t 1-emacs.txt <http://www.lysator.liu.se/history/garb/txt/87-1-emacs.txt>

2.12.6 Star, S. L. & Bowker, G. C. & Neumann, L. J. (2003) “Transparency beyond the Individual Level of Scale: Convergence between Information Artifacts and Communities of Practice, in Ann Peterson Bishop, Nancy A. Van House, and Barbara P. Buttenfield (eds.) *Digital Library Use: Social Practice in Design and Evaluation* . The MIT Press.

2.12.7 UN Press Release TAD/1967 (2003) “UNCTAD says open-source software could boost information technology sector in developing countries. URL (consulted on 020504): http://www.un.org/News/Press/docs/2003/ <http://www.un.org/News/Press/docs/2003/tad1967.doc.htm>

2.12.8 Abschnitt